Multi-Objective Optimization/Approximation in Scheduling

Erik Saule

The Ohio State University
BioMedical Informatics

ASTEC 09

Going to ASTEC

Through a complex road network, what should you do ?

- national roads ?
- highways ?

Trade-off

- national roads are cheaper but slower
- highways are faster but expensive (**toll and oil**)

No best solution, only different trade-offs.

Complex problem

- several roads
- variable speed (but no more than the limit)

Driving at low speed on highway is inefficient.

Going to ASTEC

Through a complex road network, what should you do ?

- national roads ?
- highways ?

Trade-off

- national roads are cheaper but slower
- highways are faster but expensive (**toll and oil**)

No best solution, only different trade-offs.

Complex problem

- several roads
- variable speed (but no more than the limit)

Driving at low speed on highway is inefficient.

Multi-Objective in Computing Systems

Several trade-offs in modern computing systems:

- Computation time
- Power consumption
- Real-time constraints
- Reliability
- Memory consumption
- Image quality/Refresh rate
- Latency/Bandwidth

Scheduling

Allocate a set of tasks onto machines (processors) respecting a set of constraints to optimize a performance index.
Broad literature on **single** objective optimization.

Outline

Definitions - Problem

Multi objective scheduling problem

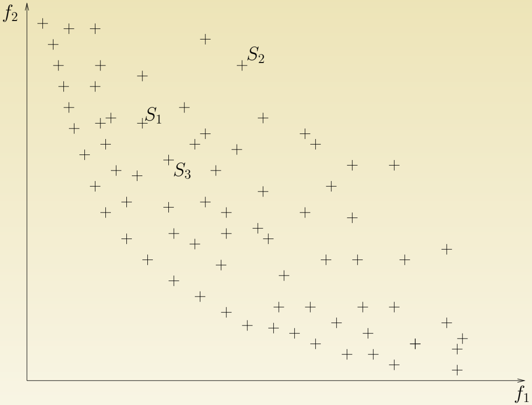
Let m be the number of processors, denoted by $P_1, \dots, P_m$. Let n be the number of task, denoted by $t_1, \dots, t_n$, with processing time $p_{i,j}$ for task t_i on processor P_j.

The multi-objective optimization problem consists of finding starting times $\sigma(i)$ for all tasks and a function π that maps tasks to processors ($\pi(i) = j$ if t_i is scheduled on P_j), such that the processors compute jobs one at a time:

$$\forall i, i' \text{ if } \pi(t_i) = \pi(t_{i'}) \text{ then } C_i \leq \sigma(i') \text{ or } C_{i'} \leq \sigma(i)$$

and the objective functions are minimized:

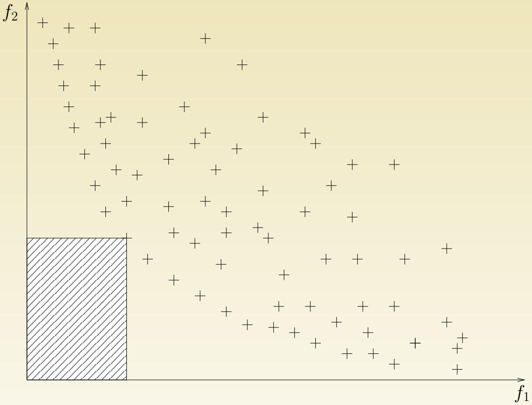
$$\min \Big(f_1(\pi, \sigma(1), \dots, \sigma(n)), \dots, f_k(\pi, \sigma(1), \dots, \sigma(n)) \Big)$$



Pareto dominance (Partial Order)

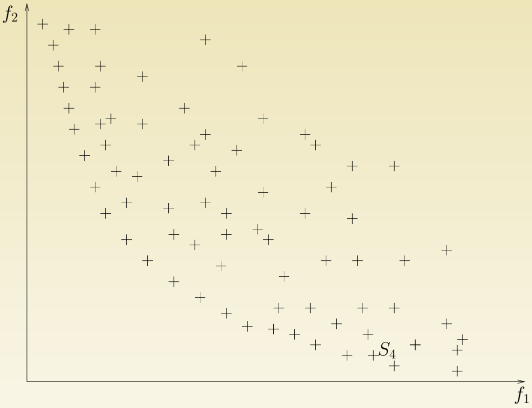
S_1 Pareto dominates S_2 if S_1 is not worse than S_2 on all dimensions and better on at least one.
Otherwise they are Pareto independent, such as S_1 and S_3.

Definitions - Optimality



Pareto optimal solution

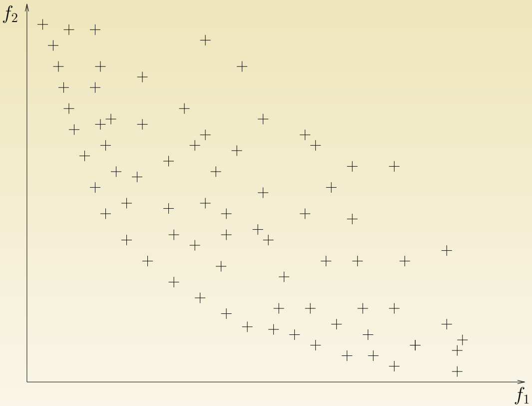
A Pareto non-dominated solution.



Weak Pareto optimality

S_4 is a Weak Pareto optimal solution if no solution is strictly better than S_4 on all the dimensions.

Definitions - Optimality



Pareto set

The set of Pareto optimal solutions.

How to Solve a Multi Objective Problem ?

Three main methods :

Lexicographical Ordering

Objective functions are totally ordered: $Lex(f_1, f_2, \ldots, f_k)$.

$$S_1 < S_2 \Leftrightarrow (f_1(S_1) < f_1(S_2)) \vee (f_1(S_1) = f_1(S_2) \wedge f_2(S_1) < f_2(S_2)) \vee \ldots$$

$$\vee\, (f_1(S_1) = f_1(S_2) \wedge \cdots \wedge f_{k-1}(S_1) = f_{k-1}(S_2) \wedge f_k(S_1) < f_k(S_2))$$

Aggregation

Optimizes an aggregation function (usually linear):
$f(S) = \alpha_1 f_1(S) + \alpha_2 f_2(S) + \cdots + \alpha_k f_k(S)$

ϵ-Constraint

$\epsilon(f_1, \ldots, f_{k-1} \backslash f_k)$: Given parameters $\omega_1, \ldots, \omega_{k-1}$, find the solution S that minimizes f_k such that $f_1(S) \leq \omega_1, \ldots, f_{k-1}(S) \leq \omega_{k-1}$.

Three main methods :

Lexicographical Ordering

Objective functions are totally ordered: $Lex(f_1, f_2, \ldots, f_k)$.

$$S_1 < S_2 \Leftrightarrow (f_1(S_1) < f_1(S_2)) \vee (f_1(S_1) = f_1(S_2) \wedge f_2(S_1) < f_2(S_2)) \vee \ldots$$

$$\vee (f_1(S_1) = f_1(S_2) \wedge \cdots \wedge f_{k-1}(S_1) = f_{k-1}(S_2) \wedge f_k(S_1) < f_k(S_2))$$

Aggregation

Optimizes an aggregation function (usually linear):
$f(S) = \alpha_1 f_1(S) + \alpha_2 f_2(S) + \cdots + \alpha_k f_k(S)$

ϵ-Constraint

$\epsilon(f_1, \ldots, f_{k-1} \backslash f_k)$: Given parameters $\omega_1, \ldots, \omega_{k-1}$, find the solution S that minimizes f_k such that $f_1(S) \leq \omega_1, \ldots, f_{k-1}(S) \leq \omega_{k-1}$.

How to Solve a Multi Objective Problem ?

Three main methods :

Lexicographical Ordering

Objective functions are totally ordered: $Lex(f_1, f_2, \dots, f_k)$.

$$S_1 < S_2 \Leftrightarrow (f_1(S_1) < f_1(S_2)) \vee (f_1(S_1) = f_1(S_2) \wedge f_2(S_1) < f_2(S_2)) \vee \dots$$

$$\vee (f_1(S_1) = f_1(S_2) \wedge \dots \wedge f_{k-1}(S_1) = f_{k-1}(S_2) \wedge f_k(S_1) < f_k(S_2))$$

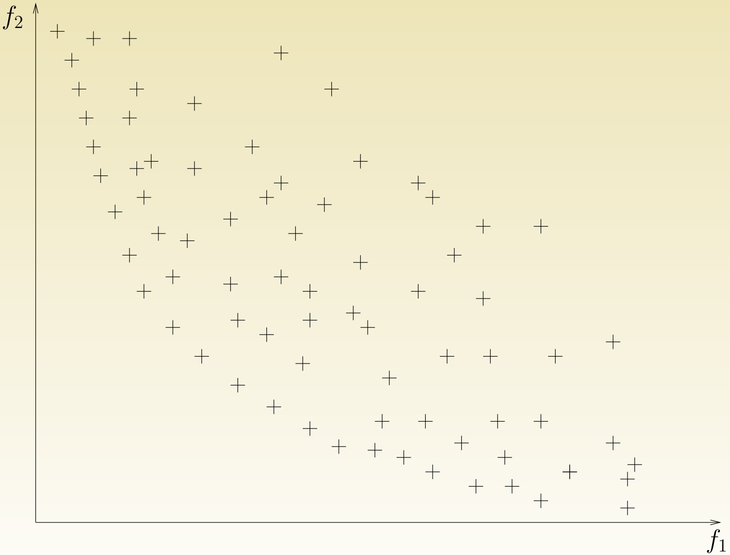
Aggregation

Optimizes an aggregation function (usually linear):
$f(S) = \alpha_1 f_1(S) + \alpha_2 f_2(S) + \dots + \alpha_k f_k(S)$

ϵ-Constraint

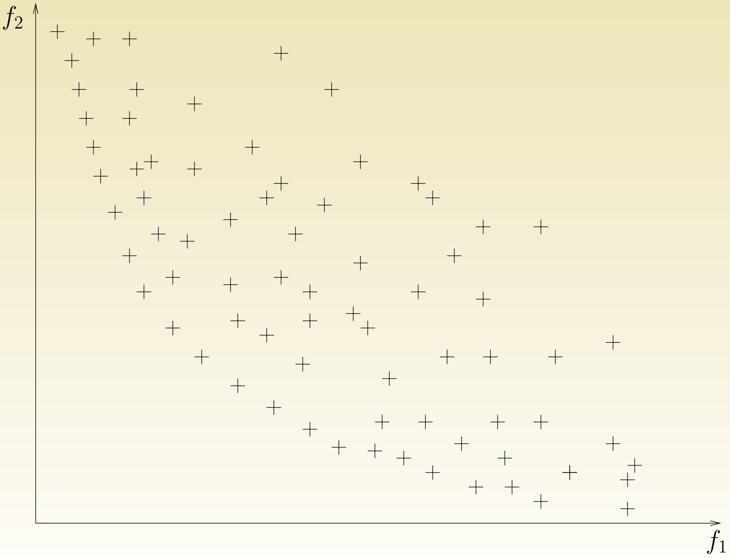
$\epsilon(f_1, \dots, f_{k-1} \backslash f_k)$: Given parameters $\omega_1, \dots, \omega_{k-1}$, find the solution S that minimizes f_k such that $f_1(S) \leq \omega_1, \dots, f_{k-1}(S) \leq \omega_{k-1}$.

Classical Method: Lexicographical Ordering

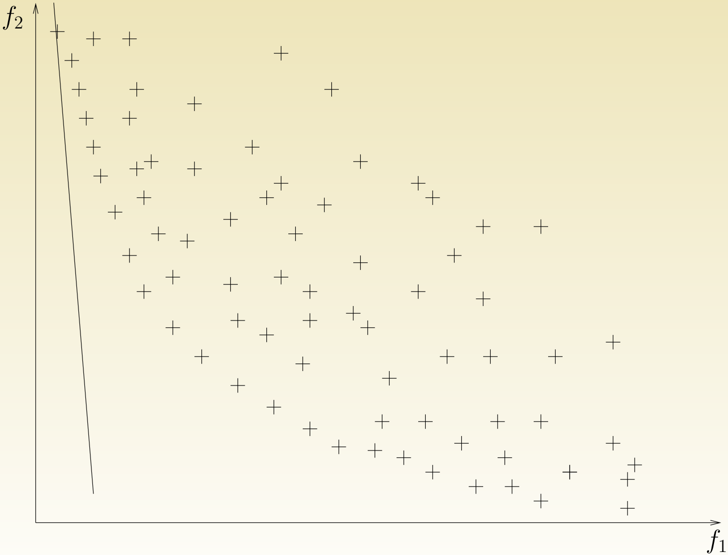


Only a few solutions are reachable (and no tradeoff solutions).

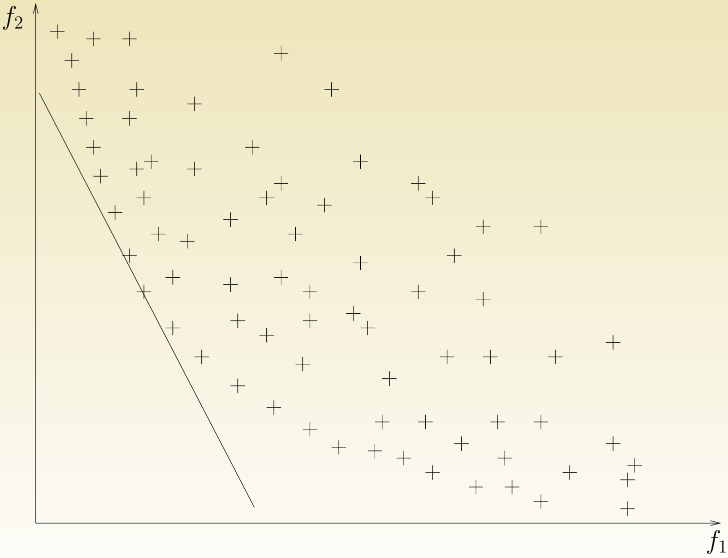
Classical Method: Linear Aggregation



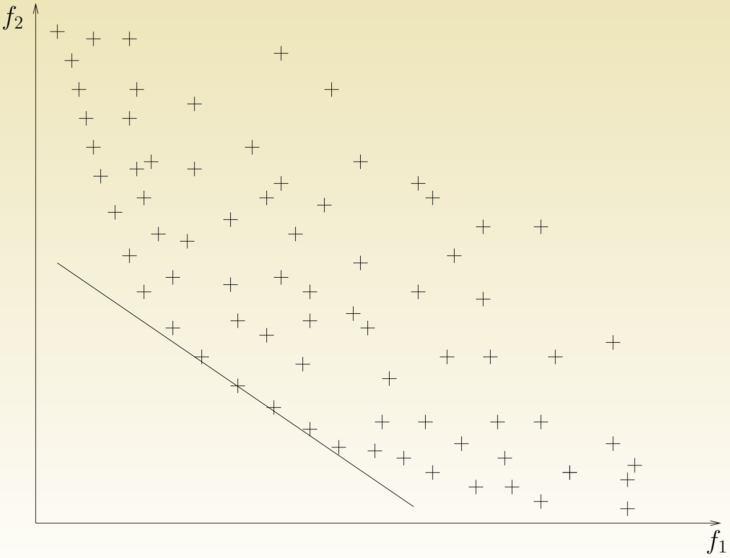
Classical Method: Linear Aggregation



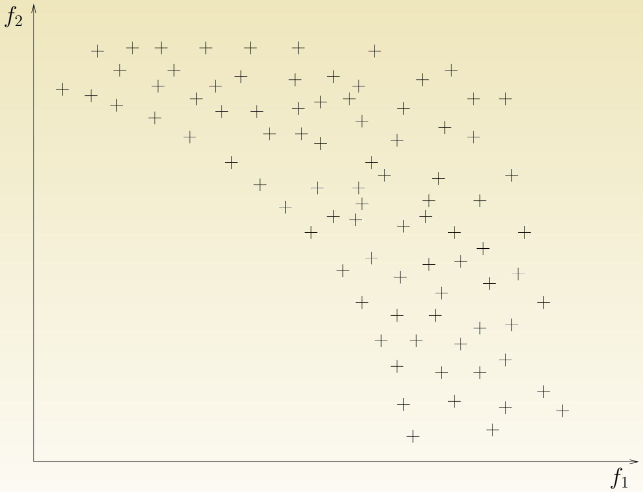
Classical Method: Linear Aggregation



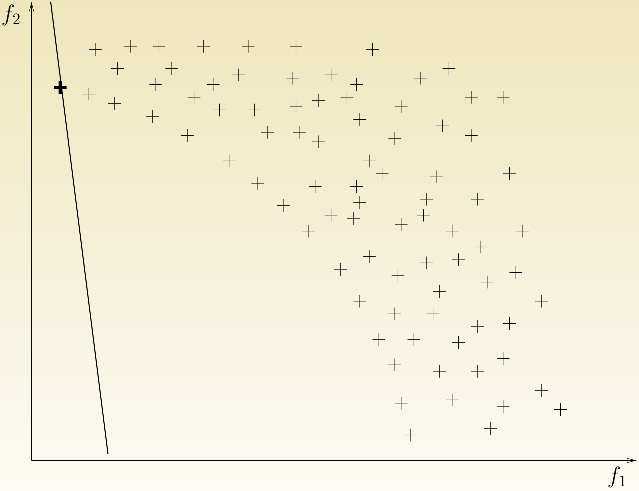
Classical Method: Linear Aggregation



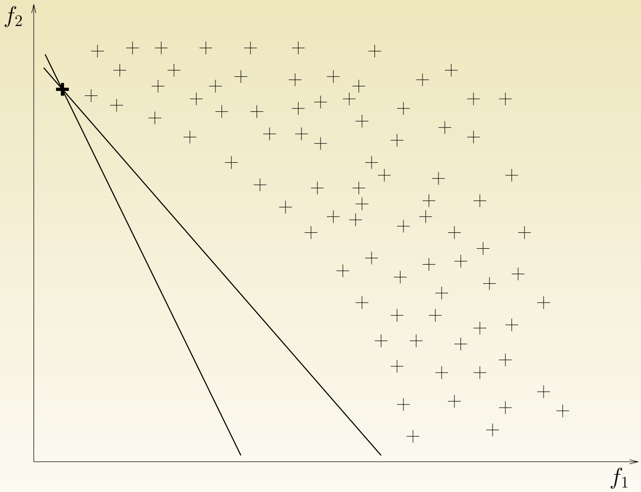
Classical Method: Linear Aggregation, drawbacks



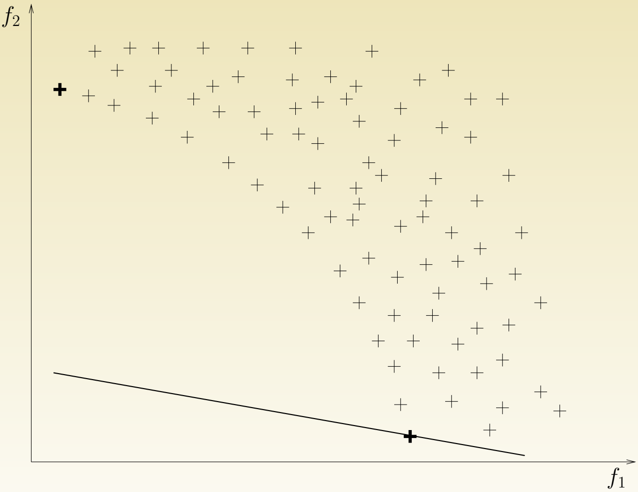
Classical Method: Linear Aggregation, drawbacks



Classical Method: Linear Aggregation, drawbacks

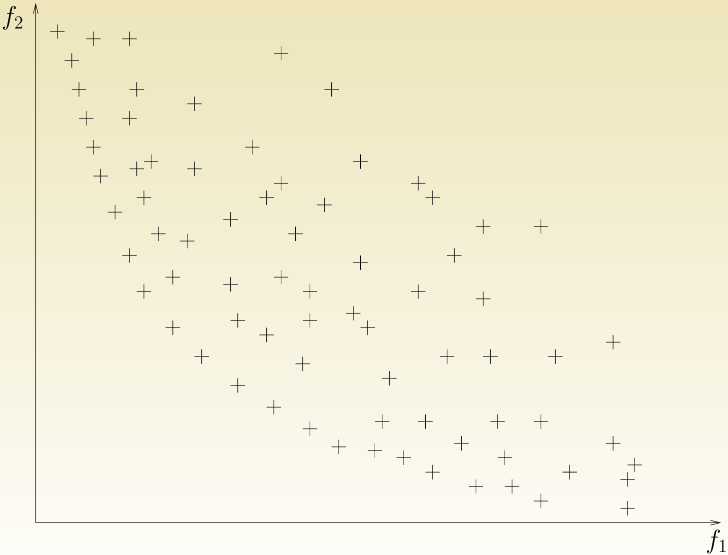


Classical Method: Linear Aggregation, drawbacks

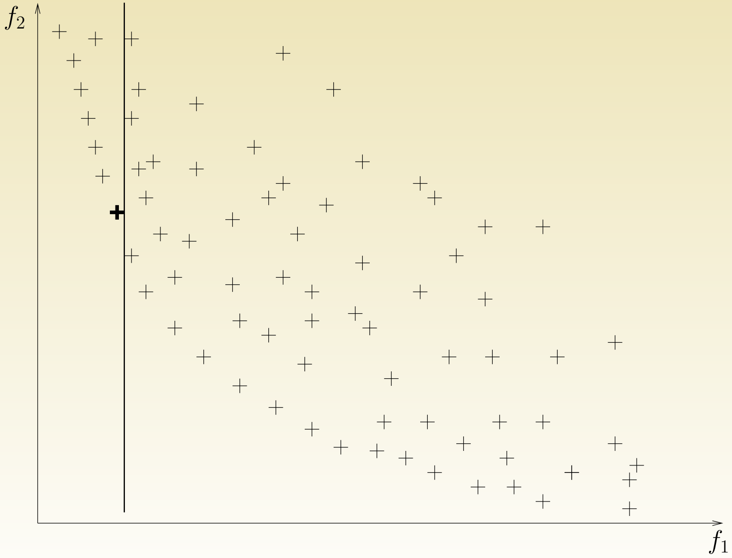


Reachable solutions are said to be **supported** (or extreme).

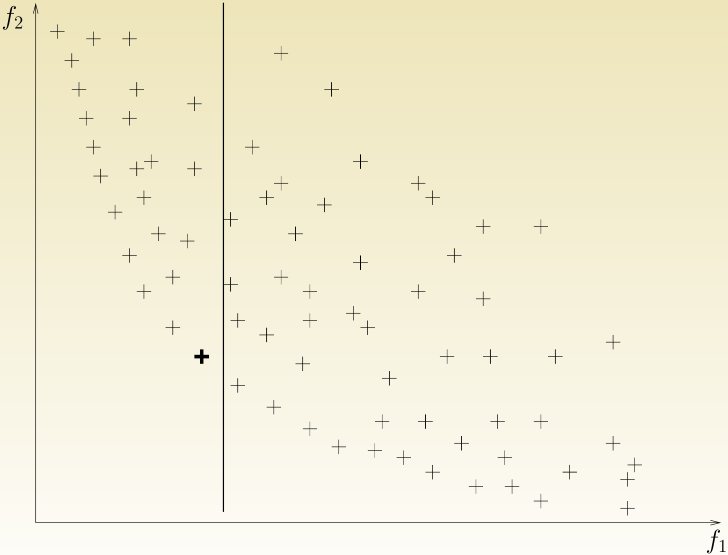
Classical Method: ϵ-constraint



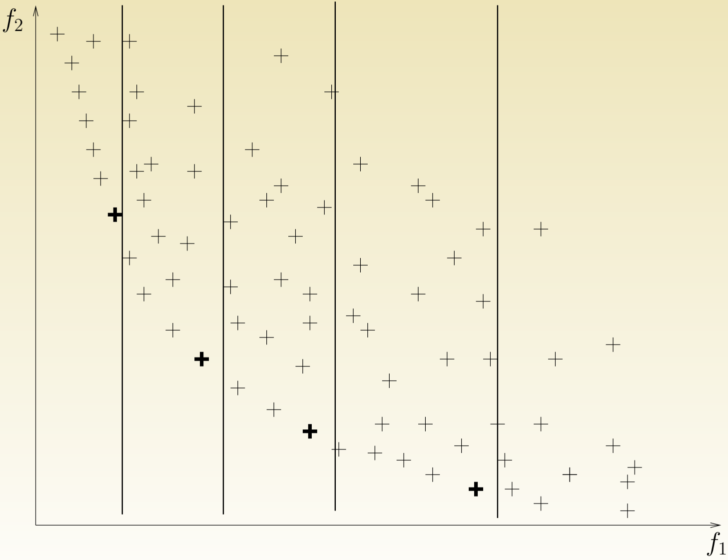
Classical Method: ϵ-constraint



Classical Method: ϵ-constraint



Classical Method: ϵ-constraint



Classical Method: ϵ-constraint

Good points

- Allows the enumeration of the Pareto set.
- If the ϵ-constraint problem is polynomial and the cardinality of the Pareto set is polynomial, enumerating the Pareto set can be done in polynomial time.
- In fact, ϵ-constraint is even equivalent to the enumeration.

Bad points

- Solving each subproblem is generally an NP-Hard problem.
- The cardinality of the set is generally exponential.

Need for approximation!

Classical Method: ϵ-constraint

Good points

- Allows the enumeration of the Pareto set.
- If the ϵ-constraint problem is polynomial and the cardinality of the Pareto set is polynomial, enumerating the Pareto set can be done in polynomial time.
- In fact, ϵ-constraint is even equivalent to the enumeration.

Bad points

- Solving each subproblem is generally an NP-Hard problem.
- The cardinality of the set is generally exponential.

Need for approximation!

Classical Method: ϵ-constraint

Good points

- Allows the enumeration of the Pareto set.
- If the ϵ-constraint problem is polynomial and the cardinality of the Pareto set is polynomial, enumerating the Pareto set can be done in polynomial time.
- In fact, ϵ-constraint is even equivalent to the enumeration.

Bad points

- Solving each subproblem is generally an NP-Hard problem.
- The cardinality of the set is generally exponential.

Need for approximation!

Approximating

Mono-objective approximation

Well defined: S is a ρ-approximation if $f(S) \leq \rho f^*$

Multi-objective approximation

definition needed!

Approximating

Mono-objective approximation

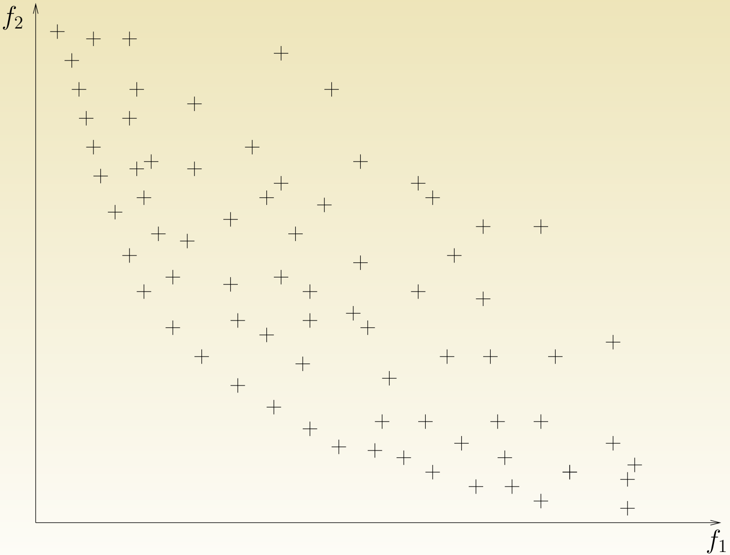
Well defined: S is a ρ-approximation if $f(S) \leq \rho f^*$

Multi-objective approximation

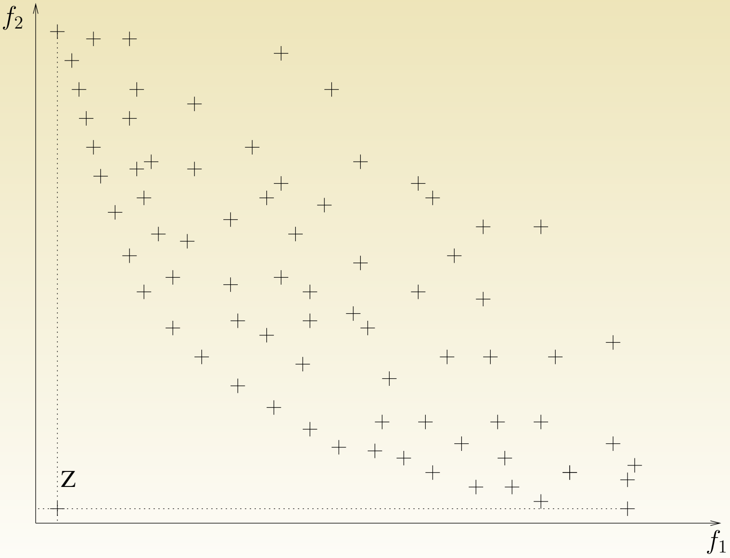
S is a $\rho = (\rho_1, \ldots, \rho_k)$-**approximation of the Zenith** if for all objective o, $f_o(S) \leq \rho_o f_o^*$ (sometimes called simultaneous approximation).

P is a $\rho = (\rho_1, \ldots, \rho_k)$-**approximation of the Pareto set** P^* if $\forall S^* \in P^*, \exists S \in P$, for all objective o, $f_o(S) \leq \rho_o S^*$ [PY00]

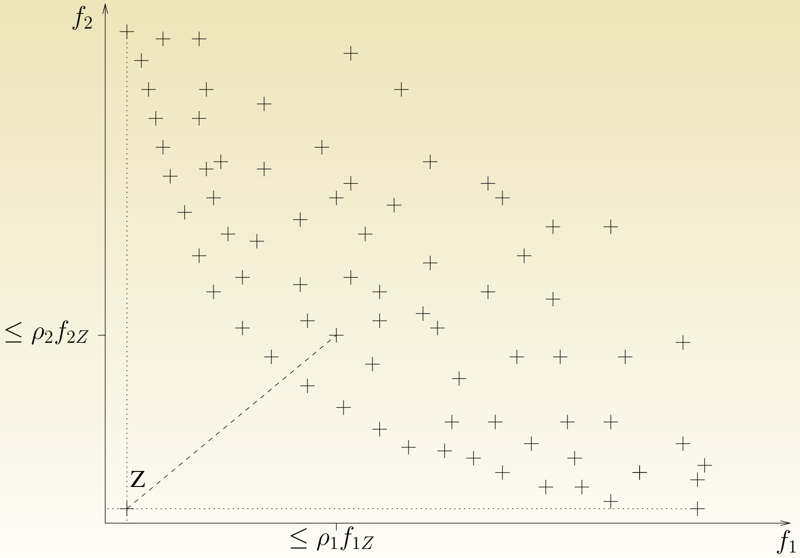
Multi-Objective Approximation



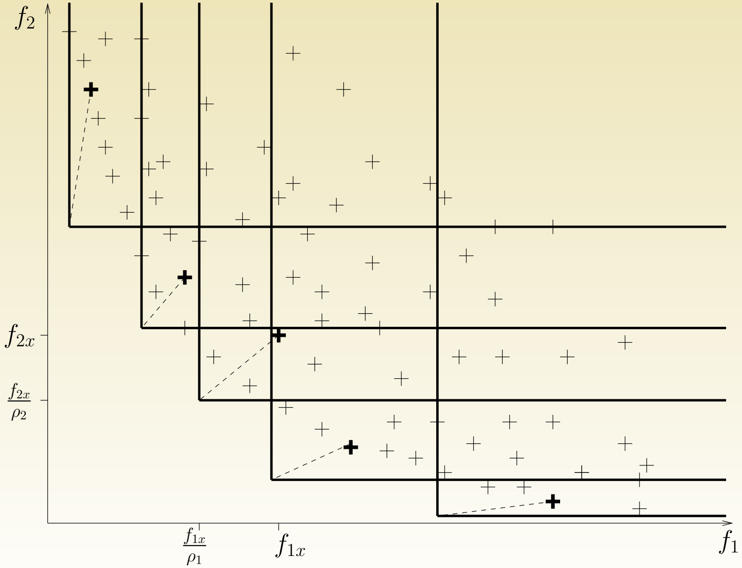
Multi-Objective Approximation



Multi-Objective Approximation



Multi-Objective Approximation



How to obtain a Zenith approximation ?

- *ad hoc* methods
- degradation control
- combining solutions [SW97]
- parametric algorithm $(1 + \Delta, 1 + \frac{1}{\Delta})$-approximation

How to obtain a Pareto set approximation ?

$\langle \overline{\rho_1}, \rho_2 \rangle$-approximation algorithm

Given a parameter ω, a $\langle \overline{\rho_1}, \rho_2 \rangle$-approximation algorithm $Algo$ returns a solution S such that $f_1(S) \leq \rho_1 \omega$ and $f_2(S) \leq \rho_2 f_2^{\omega-,*}$ where $f_2^{\omega-,*}$ is the best value of f_2 in solution such that $f_1 \leq \omega$.

$(\rho_1 + \epsilon, \rho_2)$-approximation of the Pareto set [PY00]

$\omega_i = (1 + \frac{\epsilon}{\rho_1})^i f_1^{min}$
$S_i = Algo(\omega_i)$
$i_{max} = \log_{1+\frac{\epsilon}{\rho_1}} \frac{f_1^{max}}{f_1^{min}}$
return $\{S_1, \ldots, S_{i_{max}}\}$

How to obtain a Pareto set approximation ?

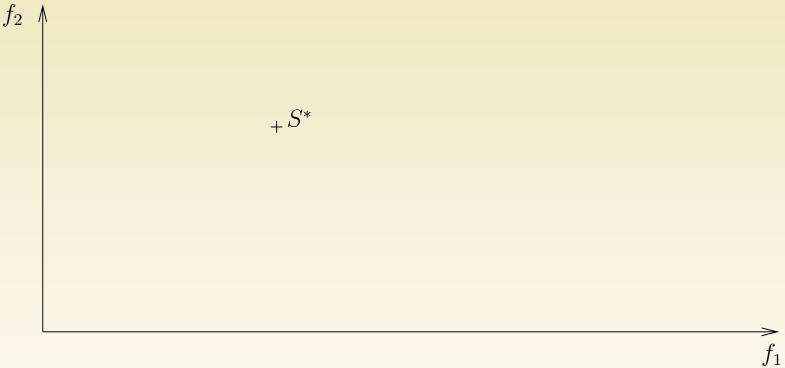
$\langle\overline{\rho_1}, \rho_2\rangle$-approximation algorithm

Given a parameter ω, a $\langle\overline{\rho_1}, \rho_2\rangle$-approximation algorithm $Algo$ returns a solution S such that $f_1(S) \leq \rho_1\omega$ and $f_2(S) \leq \rho_2 f_2^{\omega-,*}$ where $f_2^{\omega-,*}$ is the best value of f_2 in solution such that $f_1 \leq \omega$.

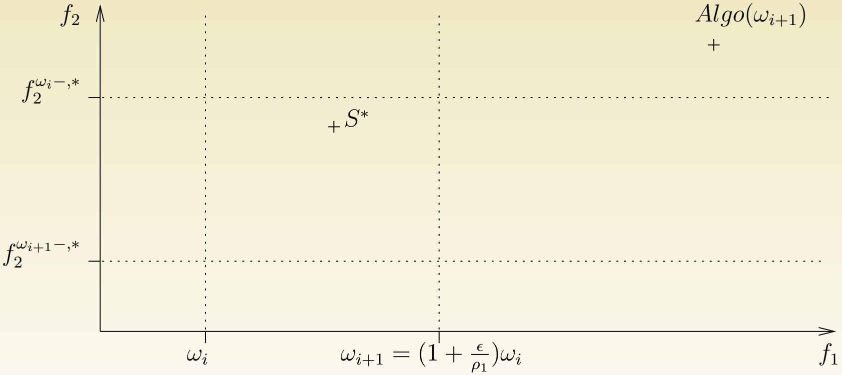
$(\rho_1 + \epsilon, \rho_2)$-approximation of the Pareto set [PY00]

$\omega_i = (1 + \frac{\epsilon}{\rho_1})^i f_1^{min}$
$S_i = Algo(\omega_i)$
$i_{max} = \log_{1+\frac{\epsilon}{\rho_1}} \frac{f_1^{max}}{f_1^{min}}$
return $\{S_1, \ldots, S_{i_{max}}\}$

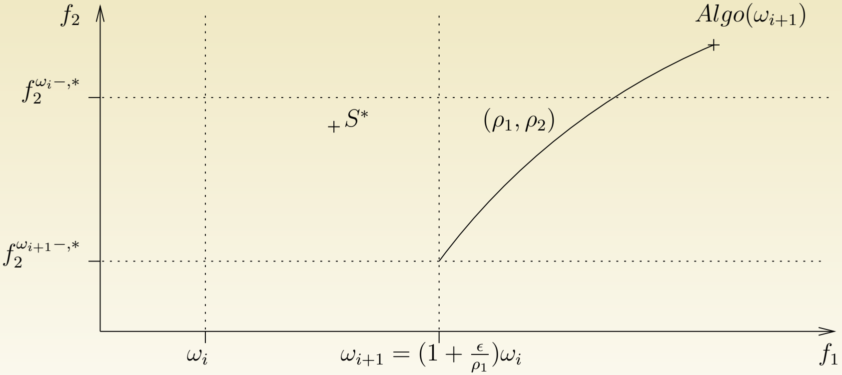
Pareto set approximation : [PY00]



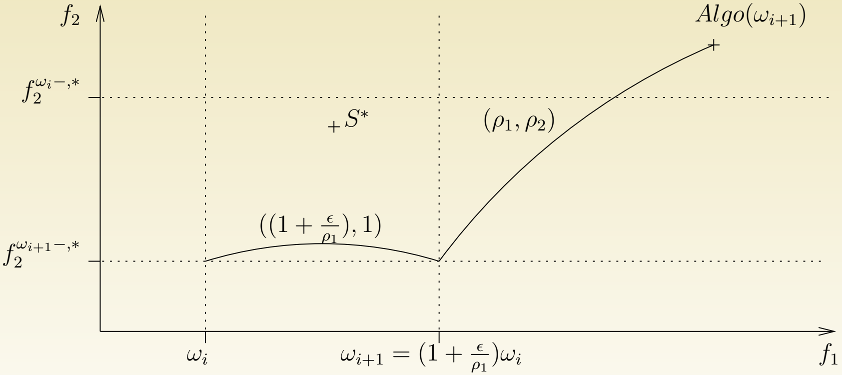
Pareto set approximation : [PY00]



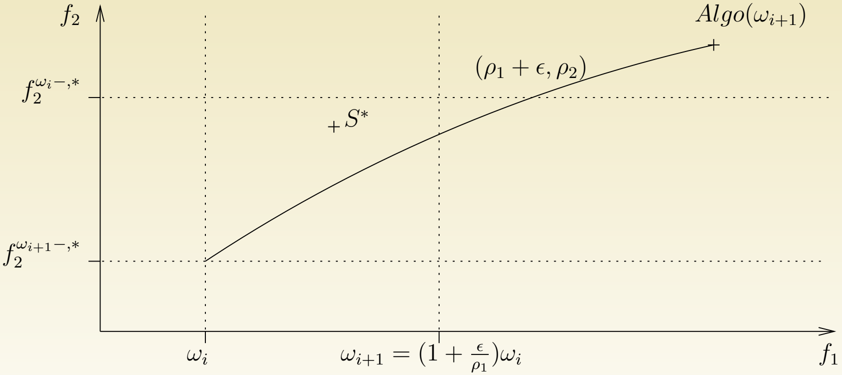
Pareto set approximation : [PY00]



Pareto set approximation : [PY00]



Pareto set approximation : [PY00]



Outline

Generalities on $1 \mid\mid L_{max}, \sum C_i$

The problem

One processor, n tasks having deadline d_i to optimize both $\sum C_i$ and $max_i C_i - d_i$.

On $\sum C_i$

$1 \mid\mid \sum C_i$ is solved by the Shortest Processing Time (Smith's rule).

On L_{max}

$1 \mid\mid L_{max}$ reduces to $1 \mid d_i \mid \emptyset$ using a binary search (the same is true for $1 \mid\mid L_{max} = k$).

$1 \mid d_i \mid \emptyset$ is solved by Earliest Deadline First (Jackson's rule).

The ϵ constraint problem: $\epsilon(L_{max} \backslash \sum C_i)$

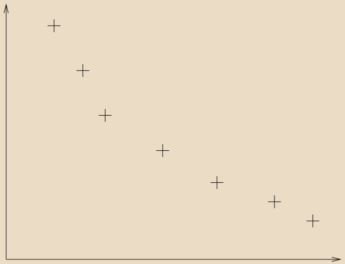
$1 \mid d_i \mid \sum C_i$

Can be solved using the backward Smith's rule: From the latest deadline to the first one, schedule the largest job available

Bounding the number of Pareto optimal solutions

Pareto optimal solutions can be reached by local improvement: there is less than n^2 of them.[Hoo04]

Enumeration can be done in polynomial time



The ϵ constraint problem: $\epsilon(L_{max} \backslash \sum C_i)$

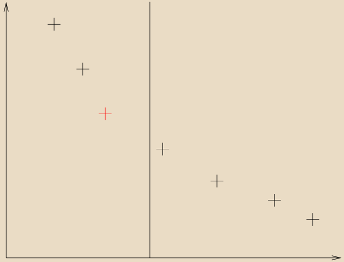
$1 \mid d_i \mid \sum C_i$

Can be solved using the backward Smith's rule: From the latest deadline to the first one, schedule the largest job available

Bounding the number of Pareto optimal solutions

Pareto optimal solutions can be reached by local improvement: there is less than n^2 of them.[Hoo04]

Enumeration can be done in polynomial time



The ϵ constraint problem: $\epsilon(L_{max} \backslash \sum C_i)$

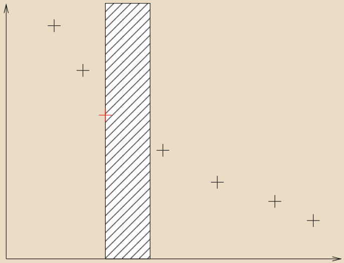
$1 \mid d_i \mid \sum C_i$

Can be solved using the backward Smith's rule: From the latest deadline to the first one, schedule the largest job available

Bounding the number of Pareto optimal solutions

Pareto optimal solutions can be reached by local improvement: there is less than n^2 of them.[Hoo04]

Enumeration can be done in polynomial time



The ϵ constraint problem: $\epsilon(L_{max} \backslash \sum C_i)$

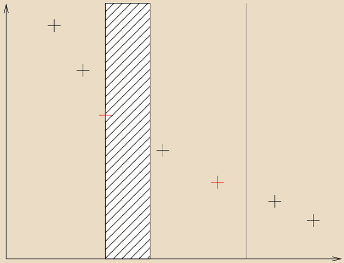
$1 \mid d_i \mid \sum C_i$

Can be solved using the backward Smith's rule: From the latest deadline to the first one, schedule the largest job available

Bounding the number of Pareto optimal solutions

Pareto optimal solutions can be reached by local improvement: there is less than n^2 of them.[Hoo04]

Enumeration can be done in polynomial time



The ϵ constraint problem: $\epsilon(L_{max} \backslash \sum C_i)$

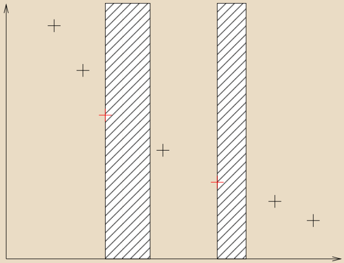
$1 \mid d_i \mid \sum C_i$

Can be solved using the backward Smith's rule: From the latest deadline to the first one, schedule the largest job available

Bounding the number of Pareto optimal solutions

Pareto optimal solutions can be reached by local improvement: there is less than n^2 of them.[Hoo04]

Enumeration can be done in polynomial time



Outline

Motivation

Physic applications

In the LHC, simulation tasks generate huge amount of data. Storage is a key issue as changing/cleaning hard-drives takes a long time. [CBB+05]

Embedded Systems

Application graphs are scheduled onto a MPSoC. Processing times are worst-case evaluations. The makespan is optimized at run-time by changing the processor executing the task. Code size on a processor in a MPSoC is limited. [CKC07]

Model

Instance

- A set of tasks $T = \{t_1, \ldots, t_n\}$
- m processors
- Processing time p_i
- Memory consumption s_i
- A memory limit M_{max}
- (A precedence constraint graph G)

Solution

- A function π allocating tasks to processors.
- A function σ allocating tasks to times.

Memory constraint: $\max_j \sum_{\pi(i)=j} s_i \leq M_{max}$
Optimize C_{max}, the date when the last task finishes.

Model

Instance

- A set of tasks $T = \{t_1, \ldots, t_n\}$
- m processors
- Processing time p_i
- Memory consumption s_i
- A memory limit M_{max}
- (A precedence constraint graph G)

Solution

- A function π allocating tasks to processors.
- A function σ allocating tasks to times.

Memory constraint: $\max_j \sum_{\pi(i)=j} s_i \leq M_{max}$
Optimize C_{max}, the date when the last task finishes.

Complexity

No memory constraint

Optimizing the makespan is NP-hard. But there exists approximation algorithms: LPT is a $\frac{4}{3}$-approximation algorithm and there exists a PTAS.

Our case

Deciding whether there is a solution or not is NP-complete. Thus, no polynomial approximation algorithm could be derived (unless $P = NP$).

$\Rightarrow$ What could we do ?

Complexity

No memory constraint

Optimizing the makespan is NP-hard. But there exists approximation algorithms: LPT is a $\frac{4}{3}$-approximation algorithm and there exists a PTAS.

Our case

Deciding whether there is a solution or not is NP-complete. Thus, no polynomial approximation algorithm could be derived (unless $P = NP$).

$\Rightarrow$ What could we do ?

Complexity

No memory constraint

Optimizing the makespan is NP-hard. But there exists approximation algorithms: LPT is a $\frac{4}{3}$-approximation algorithm and there exists a PTAS.

Our case

Deciding whether there is a solution or not is NP-complete. Thus, no polynomial approximation algorithm could be derived (unless $P = NP$).

$\Rightarrow$ What could we do ?

A Classical Solution

Dealing with NP-Complete Decision Problems

Two techniques:

- deriving structural properties
- consider the optimization problem and derive approximation properties

A Bi-objective Optimization Problem

Transform the memory constraint into an objective

$M_{max} = \max_j \sum_{\pi(i)=j} s_i$

Minimize C_{max} and M_{max}

Notice: the problem is still NP-Complete. Approximation is needed.

A Classical Solution

Dealing with NP-Complete Decision Problems

Two techniques:

- deriving structural properties
- consider the optimization problem and derive approximation properties

A Bi-objective Optimization Problem

Transform the memory constraint into an objective

$M_{max} = \max_j \sum_{\pi(i)=j} s_i$

Minimize C_{max} and M_{max}

Notice: the problem is still NP-Complete. Approximation is needed.

A Classical Solution

Dealing with NP-Complete Decision Problems

Two techniques:

- deriving structural properties
- consider the optimization problem and derive approximation properties

A Bi-objective Optimization Problem

Transform the memory constraint into an objective

$M_{max} = \max_j \sum_{\pi(i)=j} s_i$

Minimize C_{max} and M_{max}

Notice: the problem is still NP-Complete. Approximation is needed.

A Classical Solution

Dealing with NP-Complete Decision Problems

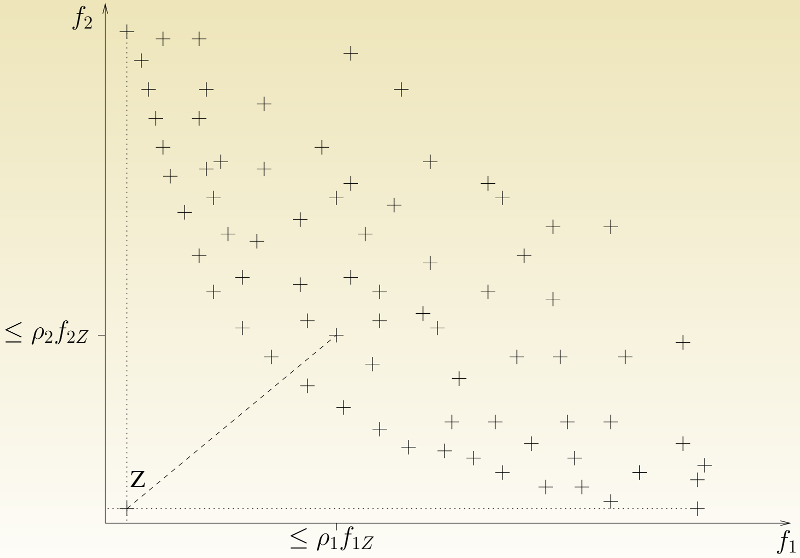
Two techniques:

- deriving structural properties
- consider the optimization problem and derive approximation properties

A Bi-objective Optimization Problem

Transform the memory constraint into an objective
$M_{max} = \max_j \sum_{\pi(i)=j} s_i$
Minimize C_{max} and M_{max}
Notice: the problem is still NP-Complete. Approximation is needed.

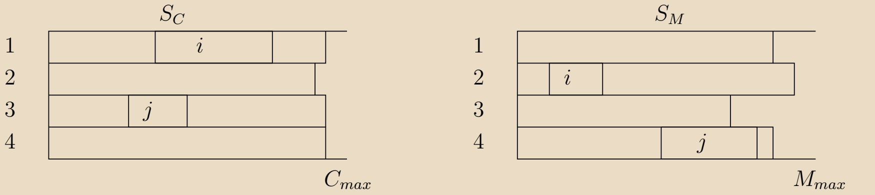
Zenith approximation



Mixing Two Schedules

The SBO algorithm

Let us have 2 schedules S_M and S_C, each one optimal on one objective. For all i, if $\frac{s_i}{M^*_{max}} \leq \frac{p_i}{C^*_{max}}$ schedule i according to S_C. Or schedule i according to S_M otherwise.



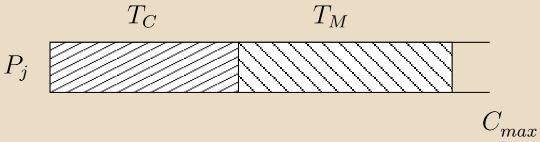
i is scheduled on processor 1. j is scheduled on processor 4.

SBO's property

Property

$C_{max}^{(SBO)} \leq 2C_{max}^{*}$ (and $M_{max}^{(SBO)} \leq 2M_{max}^{*}$)

Proof:



- If $\frac{s_i}{M_{max}^*} \leq \frac{p_i}{C_{max}^*}$, then i is scheduled according to S_c, S_M otherwise.
- $\sum_{i \in T_C} p_i \leq C_{max}^*$
- $\sum_{i \in T_M} p_i \leq \sum_{i \in T_M} \frac{s_i C_{max}^*}{M_{max}^*} \leq \frac{C_{max}^*}{M_{max}^*} \sum_{i \in T_m} s_i \leq C_{max}^*$
- $\sum_{i \in P_j} p_i \leq \sum_{i \in T_C} p_i + \sum_{i \in T_M} p_i \leq 2C_{max}^*$

$\square$

Enhanced Version

Corolary

SBO is a $(2,2)$-approximation of the Zenith

Adding a parameter

Add a Δ parameter. The comparison becomes "if $\frac{s_i}{M^*_{max}} \leq \Delta \frac{p_i}{C^*_{max}}$".

Using non-optimal schedules

When S_M (resp. S_C) is a ρ_M-approximation (resp ρ_C-approximation). The comparison becomes "if $\frac{s_i}{M_{max}(S_m)} \leq \Delta \frac{p_i}{C_{max}(S_C)}$".

Properties

SBO_Δ is a $((1+\frac{1}{\Delta})\rho_C, (1+\Delta)\rho_M)$-approximation algorithm of the Zenith.

using a PTAS : $(1+\frac{1}{\Delta}+\epsilon, 1+\Delta+\epsilon)$

using LPT: $(\frac{4}{3}(1+\frac{1}{\Delta}), \frac{4}{3}(1+\Delta))$.

Enhanced Version

Corolary

SBO is a $(2, 2)$-approximation of the Zenith

Adding a parameter

Add a Δ parameter. The comparison becomes "if $\frac{s_i}{M^*_{max}} \leq \Delta \frac{p_i}{C^*_{max}}$".

Using non-optimal schedules

When S_M (resp. S_C) is a ρ_M-approximation (resp ρ_C-approximation). The comparison becomes "if $\frac{s_i}{M_{max}(S_m)} \leq \Delta \frac{p_i}{C_{max}(S_C)}$".

Properties

SBO_Δ is a $((1 + \frac{1}{\Delta})\rho_C, (1 + \Delta)\rho_M)$-approximation algorithm of the Zenith.

using a PTAS : $(1 + \frac{1}{\Delta} + \epsilon, 1 + \Delta + \epsilon)$

using LPT: $(\frac{4}{3}(1 + \frac{1}{\Delta}), \frac{4}{3}(1 + \Delta))$.

Enhanced Version

Corolary

SBO is a $(2,2)$-approximation of the Zenith

Adding a parameter

Add a Δ parameter. The comparison becomes "if $\frac{s_i}{M^*_{max}} \leq \Delta \frac{p_i}{C^*_{max}}$".

Using non-optimal schedules

When S_M (resp. S_C) is a ρ_M-approximation (resp ρ_C-approximation).
The comparison becomes "if $\frac{s_i}{M_{max}(S_m)} \leq \Delta \frac{p_i}{C_{max}(S_C)}$".

Properties

SBO_Δ is a $((1+\frac{1}{\Delta})\rho_C, (1+\Delta)\rho_M)$-approximation algorithm of the Zenith.
using a PTAS : $(1+\frac{1}{\Delta}+\epsilon, 1+\Delta+\epsilon)$
using LPT: $(\frac{4}{3}(1+\frac{1}{\Delta}), \frac{4}{3}(1+\Delta))$.

Enhanced Version

Corolary

SBO is a $(2, 2)$-approximation of the Zenith

Adding a parameter

Add a Δ parameter. The comparison becomes "if $\frac{s_i}{M^*_{max}} \leq \Delta \frac{p_i}{C^*_{max}}$".

Using non-optimal schedules

When S_M (resp. S_C) is a ρ_M-approximation (resp ρ_C-approximation). The comparison becomes "if $\frac{s_i}{M_{max}(S_m)} \leq \Delta \frac{p_i}{C_{max}(S_C)}$".

Properties

SBO_Δ is a $((1 + \frac{1}{\Delta})\rho_C, (1 + \Delta)\rho_M)$-approximation algorithm of the Zenith.

using a PTAS : $(1 + \frac{1}{\Delta} + \epsilon, 1 + \Delta + \epsilon)$

using LPT: $(\frac{4}{3}(1 + \frac{1}{\Delta}), \frac{4}{3}(1 + \Delta))$.

Enhanced Version

Corolary

SBO is a $(2, 2)$-approximation of the Zenith

Adding a parameter

Add a Δ parameter. The comparison becomes "if $\frac{s_i}{M^*_{max}} \leq \Delta \frac{p_i}{C^*_{max}}$".

Using non-optimal schedules

When S_M (resp. S_C) is a ρ_M-approximation (resp ρ_C-approximation). The comparison becomes "if $\frac{s_i}{M_{max}(S_m)} \leq \Delta \frac{p_i}{C_{max}(S_C)}$".

Properties

SBO_Δ is a $((1 + \frac{1}{\Delta})\rho_C, (1 + \Delta)\rho_M)$-approximation algorithm of the Zenith.

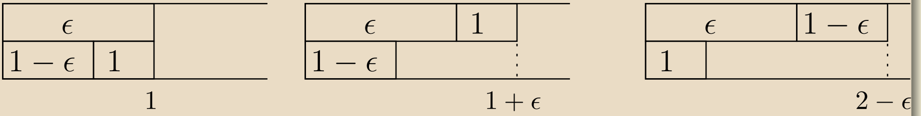
using a PTAS : $(1 + \frac{1}{\Delta} + \epsilon, 1 + \Delta + \epsilon)$

using LPT: $(\frac{4}{3}(1 + \frac{1}{\Delta}), \frac{4}{3}(1 + \Delta))$.

The $(\frac{3}{2}, \frac{3}{2})$ Impossibility

A particular instance

3 tasks. $(1, \epsilon), (\epsilon, 1), (1-\epsilon, 1-\epsilon)$. Only 3 Pareto optimal solutions:



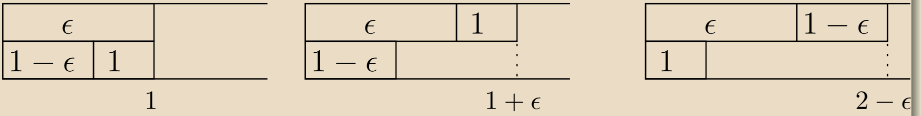
Inapproximability

When ϵ goes to $\frac{1}{2}$, values of Pareto optimal solutions goes to $(1, \frac{3}{2}), (\frac{3}{2}, \frac{3}{2}), (\frac{3}{2}, 1)$. There is no algorithm better than $(\frac{3}{2}, \frac{3}{2})$.
It does not rely on $P \neq NP$ assumption

The $(\frac{3}{2}, \frac{3}{2})$ Impossibility

A particular instance

3 tasks. $(1, \epsilon), (\epsilon, 1), (1-\epsilon, 1-\epsilon)$. Only 3 Pareto optimal solutions:



Inapproximability

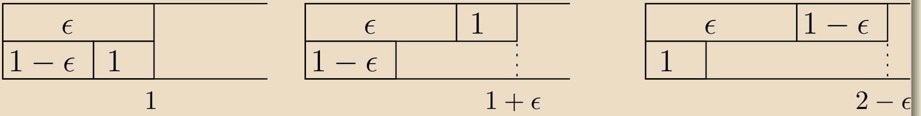
When ϵ goes to $\frac{1}{2}$, values of Pareto optimal solutions goes to $(1, \frac{3}{2}), (\frac{3}{2}, \frac{3}{2}), (\frac{3}{2}, 1)$. There is no algorithm better than $(\frac{3}{2}, \frac{3}{2})$.

It does not rely on $P \neq NP$ assumption

The $(\frac{3}{2}, \frac{3}{2})$ Impossibility

A particular instance

3 tasks. $(1, \epsilon), (\epsilon, 1), (1-\epsilon, 1-\epsilon)$. Only 3 Pareto optimal solutions:



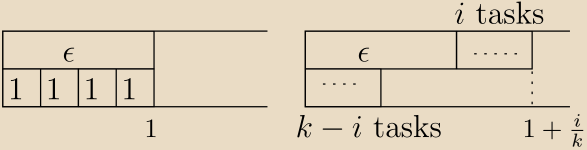
Inapproximability

When ϵ goes to $\frac{1}{2}$, values of Pareto optimal solutions goes to $(1, \frac{3}{2}), (\frac{3}{2}, \frac{3}{2}), (\frac{3}{2}, 1)$. There is no algorithm better than $(\frac{3}{2}, \frac{3}{2})$.
It does not rely on $P \neq NP$ assumption

The $(1+\frac{i}{km}, 1+(m-1)(1-\frac{i}{k}))$ Impossibility

The instance

$k+1$ tasks. $(1,\epsilon)$ and k tasks: $(\epsilon, \frac{1}{k})$ Only $k+1$ Pareto optimal solutions:



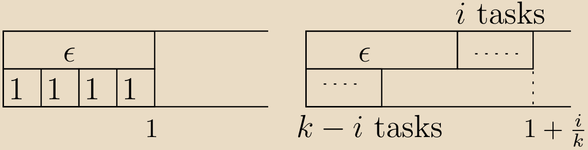
Inapproximability

The idea extends to m processors:
There is no algorithm better than
$(1+\frac{i}{km}, 1+(m-1)(1-\frac{i}{k})), \forall k \geq 2, 0 \leq i \leq k$

The $(1+\frac{i}{km}, 1+(m-1)(1-\frac{i}{k}))$ Impossibility

The instance

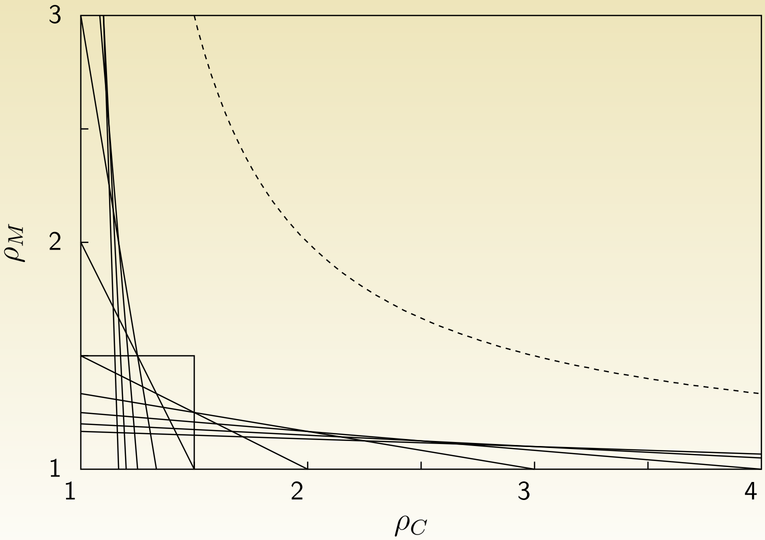
$k+1$ tasks. $(1,\epsilon)$ and k tasks: $(\epsilon, \frac{1}{k})$ Only $k+1$ Pareto optimal solutions:



Inapproximability

The idea extends to m processors:
There is no algorithm better than
$(1+\frac{i}{km}, 1+(m-1)(1-\frac{i}{k})), \forall k \geq 2, 0 \leq i \leq k$

Summing Up



Restricted List Scheduling

Precedence graph G

RLS_Δ

- Compute a lower bound LB on memory.
- Select a ready task i.
- Mark processors with memory usage greater than $\Delta LB - s_i$
- Schedule i on the unmarked processor that will complete it the soonest
- Loop until the end of the DAG.

RLS's properties

Property

There is less than $\left\lfloor \frac{m}{\Delta-1} \right\rfloor$ marked processors for $\Delta \geq 2$.

Proof: argument based on the area

Theorem

RLS_Δ is a $(2 + \frac{1}{\Delta-2} - \frac{\Delta-1}{m(\Delta-2)}, \Delta)$-approximation algorithm of the Zenith.

Proof: Graham analysis on the unmarked processors

RLS's properties

Property

There is less than $\left\lfloor \frac{m}{\Delta-1} \right\rfloor$ marked processors for $\Delta \geq 2$.

Proof: argument based on the area

Theorem

RLS_Δ is a $(2 + \frac{1}{\Delta-2} - \frac{\Delta-1}{m(\Delta-2)}, \Delta)$-approximation algorithm of the Zenith.

Proof: Graham analysis on the unmarked processors

Outline

Motivation

Non-safe systems

- Hardware breakdown
- Cosmic rays (ECC memory)

In Grid computing

- more processors, more failures
- maintenance operations
- power outage

Critical Embedded Systems

Car braking systems, Avionic, Aeronautic, ...

- Real-time Constraints
- No error allowed

Motivation

Non-safe systems

- Hardware breakdown
- Cosmic rays (ECC memory)

In Grid computing

- more processors, more failures
- maintenance operations
- power outage

Critical Embedded Systems

Car braking systems, Avionic, Aeronautic, ...

- Real-time Constraints
- No error allowed

A Bunch of Models

Architecture

- Heterogeneous processors
- (Fully connected network of processors)
- (Communication according to the delay model)
- (An application DAG)

Failures

- transient
- fail-silent
- statistically independent
- (only affect computations)
- occur following a Poisson's process

A Bunch of Models

Architecture

- Heterogeneous processors
- (Fully connected network of processors)
- (Communication according to the delay model)
- (An application DAG)

Failures

- transient
- fail-silent
- statistically independent
- (only affect computations)
- occur following a Poisson's process

Formal definitions

The Scheduling Problem

- A set T of n tasks (with dependencies)
- A set of m processors
- Task i on processor j is computed in p_{ij} time units
- Failures on j follow a Poisson's process of parameter λ_j: $P(i,j) = e^{-\lambda_j p_{ij}}$

A solution is composed of two functions:

- π, a spatial allocation ($\pi(i)$ is the set of processors scheduling i)
- σ, temporal allocation ($\sigma(i,j)$ is the starting time of i on j)

Objective functions

- The makespan C_{max}
- The reliability rel, the probability of success of the application

Formal definitions

The Scheduling Problem

- A set T of n tasks (with dependencies)
- A set of m processors
- Task i on processor j is computed in p_{ij} time units
- Failures on j follow a Poisson's process of parameter λ_j: $P(i,j) = e^{-\lambda_j p_{ij}}$

A solution is composed of two functions:

- π, a spatial allocation ($\pi(i)$ is the set of processors scheduling i)
- σ, temporal allocation ($\sigma(i,j)$ is the starting time of i on j)

Objective functions

- The makespan C_{max}
- The reliability rel, the probability of success of the application

Computing the reliability

In general

$rel(\pi, \sigma) = \prod_i (P(i \text{ is ok}))$.

Without duplication

$rel(\pi, \sigma) = \prod_i \left(\exp^{-\lambda_{\pi(i)} p_{i,\pi i}} \right)$.

With duplication

$rel(\pi, \sigma) = \prod_{i \in T} \left(1 - (\prod_{j \in \pi(i)} 1 - P(i,j)) \right) =$
$\prod_{i \in T} \left(1 - (\prod_{j \in \pi(i)} 1 - \exp^{-\lambda_{\pi(i)} p_{i,j}}) \right)$

Remark: Does not depend on starting time

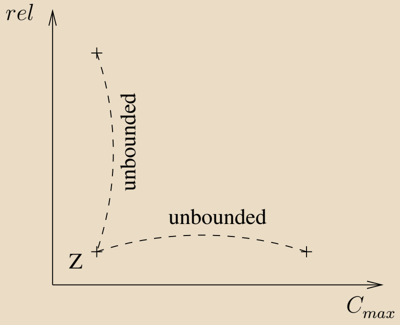
Inapproximability of the Zenith

Theorem

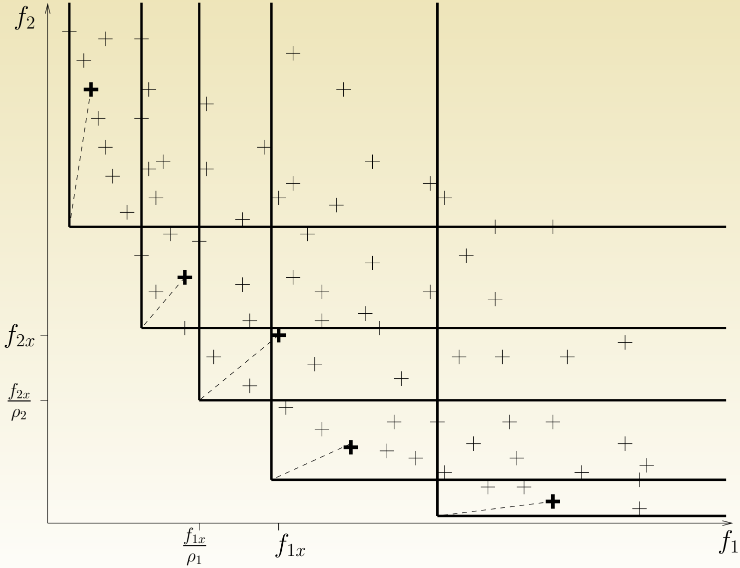
The zenith solution can not be approximated within a constant factor

Idea of the proof

One task, two processors.
The first is fast but unreliable.
The second is slow but very reliable.
No existing tradeoff



Pareto Set Approximation



Independent Task no Replication on Uniform Machines

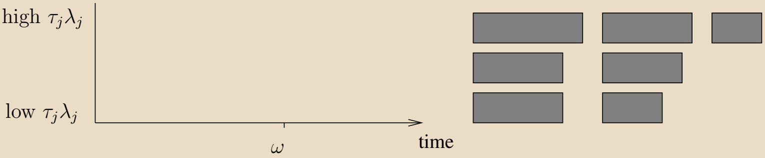
Properties

Processor capabilities are linked by their speed. $p_{ij} = p_i \tau_j$.

The reliability becomes : $rel = e^{-\sum_{i \in T} \lambda_{\pi(i)} \tau_{\pi(i)} p_i}$

CMLT

- Dual approximation: looking for a schedule of $C_{max} \leq \omega$
- Sort task in non increasing order of p_i
- Schedule greedily each task on the processor j that minimizes $\lambda_j \tau_j$ under two constrains $p_{i,j} \leq \omega$ and $C^{(j)} \leq \omega$
- If no such processor exist, reject ω.



Independent Task no Replication on Uniform Machines

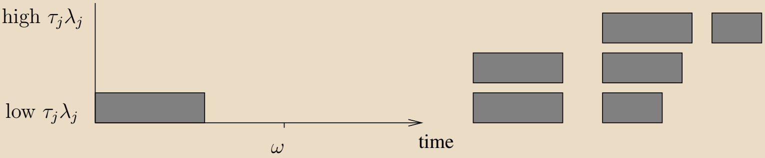
Properties

Processor capabilities are linked by their speed. $p_{ij} = p_i \tau_j$.

The reliability becomes : $rel = e^{-\sum_{i \in T} \lambda_{\pi(i)} \tau_{\pi(i)} p_i}$

CMLT

- Dual approximation: looking for a schedule of $C_{max} \leq \omega$
- Sort task in non increasing order of p_i
- Schedule greedily each task on the processor j that minimizes $\lambda_j \tau_j$ under two constrains $p_{i,j} \leq \omega$ and $C^{(j)} \leq \omega$
- If no such processor exist, reject ω.



Independent Task no Replication on Uniform Machines

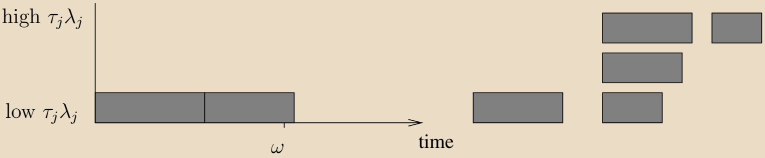
Properties

Processor capabilities are linked by their speed. $p_{ij} = p_i \tau_j$.

The reliability becomes : $rel = e^{-\sum_{i \in T} \lambda_{\pi(i)} \tau_{\pi(i)} p_i}$

CMLT

- Dual approximation: looking for a schedule of $C_{max} \leq \omega$
- Sort task in non increasing order of p_i
- Schedule greedily each task on the processor j that minimizes $\lambda_j \tau_j$ under two constrains $p_{i,j} \leq \omega$ and $C^{(j)} \leq \omega$
- If no such processor exist, reject ω.



Independent Task no Replication on Uniform Machines

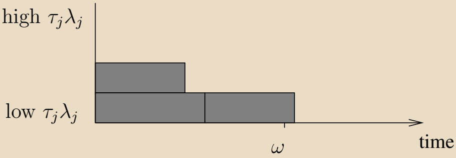
Properties

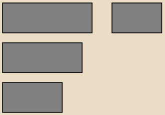
Processor capabilities are linked by their speed. $p_{ij} = p_i \tau_j$.

The reliability becomes : $rel = e^{-\sum_{i \in T} \lambda_{\pi(i)} \tau_{\pi(i)} p_i}$

CMLT

- Dual approximation: looking for a schedule of $C_{max} \leq \omega$
- Sort task in non increasing order of p_i
- Schedule greedily each task on the processor j that minimizes $\lambda_j \tau_j$ under two constrains $p_{i,j} \leq \omega$ and $C^{(j)} \leq \omega$
- If no such processor exist, reject ω.





Independent Task no Replication on Uniform Machines

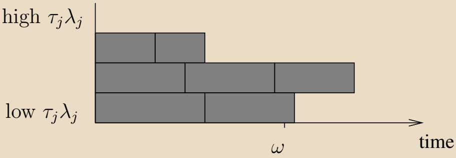
Properties

Processor capabilities are linked by their speed. $p_{ij} = p_i \tau_j$.

The reliability becomes : $rel = e^{-\sum_{i \in T} \lambda_{\pi(i)} \tau_{\pi(i)} p_i}$

CMLT

- Dual approximation: looking for a schedule of $C_{max} \leq \omega$
- Sort task in non increasing order of p_i
- Schedule greedily each task on the processor j that minimizes $\lambda_j \tau_j$ under two constrains $p_{i,j} \leq \omega$ and $C^{(j)} \leq \omega$
- If no such processor exist, reject ω.



Rejection

If $CMLT$ rejects the makespan then no such solution exists.

The proof shows a set of task that can not be scheduled on the processor using a area argument.

Lemmas

$C_{max} \leq 2\omega$
rel is optimal (among schedule with $C_{max} \leq \omega$)

The makespan bound is direct from the algorithm.
The reliability optimality comes from the overloading of reliable processors.

Theorems

CMLT is a $\langle\overline{2}, 1\rangle$-approximation algorithm.
Using CMLT, one can construct a $(2 + \epsilon, 1)$-approximation of the Pareto set.

CMLT

Rejection

If $CMLT$ rejects the makespan then no such solution exists.

The proof shows a set of task that can not be scheduled on the processor using a area argument.

Lemmas

$C_{max} \leq 2\omega$
rel is optimal (among schedule with $C_{max} \leq \omega$)

The makespan bound is direct from the algorithm.
The reliability optimality comes from the overloading of reliable processors.

Theorems

CMLT is a $\langle \overline{2}, 1 \rangle$-approximation algorithm.
Using CMLT, one can construct a $(2+\epsilon, 1)$-approximation of the Pareto set.

CMLT

Rejection

If $CMLT$ rejects the makespan then no such solution exists.

The proof shows a set of task that can not be scheduled on the processor using a area argument.

Lemmas

$C_{max} \leq 2\omega$
rel is optimal (among schedule with $C_{max} \leq \omega$)

The makespan bound is direct from the algorithm.
The reliability optimality comes from the overloading of reliable processors.

Theorems

CMLT is a $\langle \overline{2}, 1 \rangle$-approximation algorithm.
Using CMLT, one can construct a $(2 + \epsilon, 1)$-approximation of the Pareto set.

Tasks on Homogeneous Processors with Duplication

Property

$rel(\pi) = \prod_{i \in T} \left(1 - (1 - \exp^{-\lambda_{\pi(i)} p_i})^{|\pi(i)|}\right)$

Remarks

Reliability does not depend on the actual schedule but only on the number of copies of each task are scheduled.

Reliability is difficult to analyze. So let's compute it!

A Dynamic Programming Formulation

$R(C,n) = \max_{j \in M} \left(R(C - jp_n, n-1)\left(1 - (1 - \exp^{-\lambda_{\pi(i)} p_n})^j)\right)\right)$.

$R(C,0) = 1$ if $C \geq 0$ and 0 otherwise.

Tasks on Homogeneous Processors with Duplication

Property

$rel(\pi) = \prod_{i \in T} \left(1 - (1 - \exp^{-\lambda_{\pi(i)} p_i})^{|\pi(i)|}\right)$

Remarks

Reliability does not depend on the actual schedule but only on the number of copies of each task are scheduled.

Reliability is difficult to analyze. So let's compute it!

A Dynamic Programming Formulation

$R(C, n) = \max_{j \in M} \left(R(C - jp_n, n-1) \left(1 - (1 - \exp^{-\lambda_{\pi(i)} p_n})^j)\right)\right)$.
$R(C, 0) = 1$ if $C \geq 0$ and 0 otherwise.

To Approximation Algorithm

Scaling technique

By scaling the processing time, it is possible to keep the computation volume below $(1+\epsilon)C$ and obtaining optimal reliability.

Algorithm

- Dual approximation: looking for a schedule of $C_{max} \leq \omega$.
- Let $C = \omega m$.
- Use the scaled DP to get the number of copies r_i of each task.
- Schedule the r_i copies of each task on different processors using List Scheduling.

Theorem

This algorithm is a $\langle \overline{2+\epsilon}, 1 \rangle$-approximation algorithm.

To Approximation Algorithm

Scaling technique

By scaling the processing time, it is possible to keep the computation volume below $(1+\epsilon)C$ and obtaining optimal reliability.

Algorithm

- Dual approximation: looking for a schedule of $C_{max} \leq \omega$.
- Let $C = \omega m$.
- Use the scaled DP to get the number of copies r_i of each task.
- Schedule the r_i copies of each task on different processors using List Scheduling.

Theorem

This algorithm is a $\langle \overline{2+\epsilon}, 1 \rangle$-approximation algorithm.

Outline

Why use multi objective approximation ?

It allows to study/tackle more complex problem.

With such properties :

- Really multi-objective
- Impossible to approximate due to strong constraints
- Complex objective function that can be expressed as an increasing aggregation

How to study a multi objective problem ?

The important questions :

- What about the mono objective sub problems ?
- What is the complexity of the multi objective decision problem ?
- What is the shape of the Pareto set ?

How to study a multi objective problem ?

The important questions :

- What about the mono objective sub problems ?
  - Complexity, approximation algorithms, negative approximation result.
  - Helps understanding the problem.
  - Most of the time solve the next question.
- What is the complexity of the multi objective decision problem ?
- What is the shape of the Pareto set ?

How to study a multi objective problem ?

The important questions :

- What about the mono objective sub problems ?
  - Complexity, approximation algorithms, negative approximation result.
  - Helps understanding the problem.
  - Most of the time solve the next question.
- What is the complexity of the multi objective decision problem ?
  - If NP-Complete, approximation is needed.
  - Caution: all mono objective versions may be polynomial, but the multi objective one still can be NP-Complete.
- What is the shape of the Pareto set ?

How to study a multi objective problem ?

The important questions :

- What about the mono objective sub problems ?
  - Complexity, approximation algorithms, negative approximation result.
  - Helps understanding the problem.
  - Most of the time solve the next question.
- What is the complexity of the multi objective decision problem ?
  - If NP-Complete, approximation is needed.
  - Caution: all mono objective versions may be polynomial, but the multi objective one still can be NP-Complete.
- What is the shape of the Pareto set ?
  - Cardinality ? Maximum values Convex ? Concave ?
  - If its size is exponential, approximation is needed.
  - If the interesting objective values are unbounded, Pareto set approximation is not polynomial.
  - If it is not convex, linear aggregation is a bad idea.

How to solve a multi-objective optimization problem ?

Optimally in polynomial time

If the cardinality of the Pareto set is bounded and if the ϵ constraint problem can be solved in polynomial time.

Approximating the Zenith

- look for inapproximability bound (not complexity results)
- mixing several solutions
- most of the time it reuses the mono objective arguments

Approximating the Pareto set

- there could be no Zenith approximation possible
- if the size of the objective values are polynomial
- when the cardinality of the Pareto set is exponential.
- close to dual approximation techniques

What's next ?

Multi-objective as a field to study

- Methods and concept come from the study of various problems
- Studying other applicative problems could lead to new results
- link between Zenith and Pareto set approximation ?

What about problems where the information is incomplete ?

- Most algorithms use a global knowledge of the instance.
- How to deal with Online or Distributed problems ?
- How to derive a Pareto set approximation ?

What about links with other theory ?

Mainly, with Game Theory.

- Truthfulness, Equity, Jealousy, ...
- If players rewards are not unidimensional

What's next ?

Multi-objective as a field to study

- Methods and concept come from the study of various problems
- Studying other applicative problems could lead to new results
- link between Zenith and Pareto set approximation ?

What about problems where the information is incomplete ?

- Most algorithms use a global knowledge of the instance.
- How to deal with Online or Distributed problems ?
- How to derive a Pareto set approximation ?

What about links with other theory ?

Mainly, with Game Theory.

- Truthfulness, Equity, Jealousy, ...
- If players rewards are not unidimensional

What's next ?

Multi-objective as a field to study

- Methods and concept come from the study of various problems
- Studying other applicative problems could lead to new results
- link between Zenith and Pareto set approximation ?

What about problems where the information is incomplete ?

- Most algorithms use a global knowledge of the instance.
- How to deal with Online or Distributed problems ?
- How to derive a Pareto set approximation ?

What about links with other theory ?

Mainly, with Game Theory.

- Truthfulness, Equity, Jealousy, ...
- If players rewards are not unidimensional

Going Further

Presented Things

- General: [Hoo04, TB07, DRST09]
- Memory constraint: [SDM08]
- Fault Tolerant: [JST08, DJSS07, GST09, ST09]

Other Things

- Polynomial Problems: [Hoo04, TBE07]
- Zenith Approximation:
  [SW97, BFM06, RSTU02, ARSY99, CMNS97, BBL04, ABF07]
- Pareto Approximation: [PY00, ST93, ABG05, ABK01]
- Power Aware: [GM02, Bun06, AF06]
- Pipelined execution: Ask Anne ! :)

E. Angel, E. Bampis, and A. V. Fishkin.
A note on scheduling to meet two min-sum objectives.
Operation Research Letters, 35(1):69–73, 2007.

E. Angel, E. Bampis, and L. Gourvès.
Approximation results for a bicriteria job scheduling problem on a single machine without preemption.
Information Processing Letters, 94(1):19–27, April 2005.

E. Angel, E. Bampis, and A. Kononov.
A FPTAS for approximating the unrelated parallel machines scheduling problem with costs.
In Proceeding of European Symposium on Algorithms (ESA), pages 194–205, 2001.

S. Albers and H. Fujiwara.
Energy-efficient algorithms for flow time minimization.
In Springer LNCS, editor, *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science*, volume 3884, pages 621–633, 2006.

J. Aslam, A. Rasala, C. Stein, and N. Young.
Improved bicriteria existence theorems for scheduling.
In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 846–847, 1999.

F. Baille, E. Bampis, and C. Laforest.
A note on bicriteria schedules with optimal approximations ratios.
Parallel processing letters, 14(2):315–323, 2004.

Vittorio Bil, Michele Flammini, and Luca Moscardelli.
Pareto approximations for the bicriteria scheduling problem.
JPDC, 66(3):393–402, 2006.

D. Bunde.
Power-aware scheduling for makespan and flow.
In *Proceedings of the 18th Symposium of Parallelim in Algorithms and Architecture*, pages 190–196, 2006.

S. Campana, D. Barberis, F. Brochu, A. De Salvo, F. Donno, L. Goossens, S. Gonzalez de la Hoz, T. Lari, D. Liko, J. Lozano,

G. Negri, L. Perini, G. Poulard, S. Resconi, D. Rebatto, and L. Vaccarossa.
Analysis of the atlas rome production experience on the lhc computing grid.
In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pages 82–89, Washington, DC, USA, 2005. IEEE Computer Society.

P. Choudhury, R. Kumar, and P. P. Chakrabarti.
Hybrid scheduling of dynamic task graphs with selective duplication for multiprocessors under memory and time constraints.
IEEE Transactions on Parallel and Distributed Systems, (preprint), 2007.

C. Chekuri, R. Motwani, B. Natarajan, and C. Stein.
Approximation techniques for average completion time scheduling.
In *SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 609–618, 1997.

J. J. Dongarra, E. Jeannot, E. Saule, and Z. Shi.

Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems.
In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 280–288. ACM press, June 2007.

P-F Dutot, K. Rzadca, E. Saule, and D. Trystram.
Introduction to Scheduling, chapter Multi-objective approximation.
2009.

R. GrayBill and R. Melhem, editors.
Power Aware Computing.
Series in Computer Science. Kluwer Academic/Plenum Publishers, New York, May 2002.

Alain Girault, Erik Saule, and Denis Trystram.
Reliability versus performance for critical applications.
JPDC, 69(3):326–336, March 2009.

H. Hoogeveen.
Multicriteria scheduling.

European Journal of Operational Research, 167(3):592–623, December 2004.

E. Jeannot, E. Saule, and D. Trystram.
Bi-objective approximation scheme for makespan and reliability optimization on uniform parallel machines.
In *Euro-Par 2008*. LNCS, August 2008.

C. H. Papadimitriou and M. Yannakakis.
On the approximability of trade-offs and optimal access of web sources.
In FOCS, editor, *41st Annual Symposium on Foundations of Computer Science*, pages 86–92, 2000.

A. Rasala, C. Stein, E. Torng, and P. Uthaisombut.
Existence theorems, lower bounds and algorithms for scheduling to meet two objectives.
In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 723–731, 2002.

E. Saule, P.-F. Dutot, and G. Mounié.

Scheduling With Storage Constraints.
In *Electronic proceedings of IPDPS 2008*, April 2008.

D. B. Shmoys and E. Tardos.
Scheduling unrelated machines with costs.
In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pages 448–454, 1993.

Erik Saule and Denis Trystram.
Analyzing scheduling with transient failures.
IPL, 109(11):539–542, May 2009.

C. Stein and J. Wein.
On the existence of schedules that are near-optimal for both makespan and total weighted completion time.
Operational research letters, 21(3):115–122, October 1997.

V. T'kindt and J.-C. Billaut.
Multicriteria Scheduling.
Springer, 2007.

V. T'kindt, K. Bouibede-Hocine, and C. Esswein.

Counting and enumeration complexity with application to multicriteria scheduling.
Annals of Operations Research, April 2007.