Bi-objective approximation scheme for makespan and reliability optimization on uniform parallel machines for independent tasks

Emmanuel Jeannot, Erik Saule, Denis Trystram

ICL & INRIA & LIG

Sheduling in Knoxville, May 13 2009

Outline of the talk

Outline of the talk

Context: large parallel and distributed systems

Difficult to ensure that the resources are always available for a long period of time

- hardware failures
- software faults
- power breakdown
- resources removal

Introduction

Problem studied:

- scheduling independent tasks
- heterogeneous systems (uniform model)
- hardware can fail

Bi-criteria objective:

- given a makespan objective
- optimize reliability

Even if the system have checkpoint restart mechanism, it is important to carefully allocate the tasks.

Related work

A "*new subject*" :

- Dogan & Ozgüner 2002: Model the problem, RDLS bi-criteria heuristic.
- Dogan & Ozgüner 2004: enhancement of previous result (GA).
- Qin & Jiang 2005: first optimize deadline, then maximize reliability.
- Hakem & Butelle 2006: BSA, bi-criteria heuristic that outperforms RDLS.

Modeling

- An application: T a set of n independent tasks.
- Number of operations of tasks i : p_i
- A set Q of m *uniform* processors
- Processor j is associated with two values:
  - τ_j the time to perform one operation and
  - λ_j the failure rate.
- Taks i executed on processor j will last $p_i \times \tau_j = p_{ij}$.

Modeling

- An application: T a set of n independent tasks.
- Number of operations of tasks i : p_i
- A set Q of m *uniform* processors
- Processor j is associated with two values:
  - τ_j the time to perform one operation and
  - λ_j the failure rate.
- Taks i executed on processor j will last $p_i \times \tau_j = p_{ij}$.

Assumption:

- Processors are subject to crash fault only.
- During the execution of the DAG, the failure rate is constant.

$\Rightarrow$ failure model follows an exponential law.

$\Rightarrow$ probability that task i finishes (correctly) its execution:

$$e^{-p_i \times \tau_j \times \lambda_j}$$

Outline of the talk

Scheduling problem

- A schedule: $\pi : T \longrightarrow Q$

Scheduling problem

- A schedule: $\pi : T \rightarrow Q$
- $T(j, \pi) = \{i \mid \pi(i) = j\}$: set of tasks mapped to processor j

Scheduling problem

- A schedule: $\pi : T \rightarrow Q$
- $T(j, \pi) = \{i \mid \pi(i) = j\}$: set of tasks mapped to processor j
- $C_j(\pi) = \sum_{i \in T(j,\pi)} p_i \tau_j$: completion time of a processor j

Scheduling problem

- A schedule: $\pi : T \rightarrow Q$
- $T(j, \pi) = \{i \mid \pi(i) = j\}$: set of tasks mapped to processor j
- $C_j(\pi) = \sum_{i \in T(j,\pi)} p_i \tau_j$: completion time of a processor j
- $C_{max}(\pi) = max_j C_j(\pi)$: makespan of a schedule

Scheduling problem

- A schedule: $\pi : T \rightarrow Q$
- $T(j, \pi) = \{i \mid \pi(i) = j\}$: set of tasks mapped to processor j
- $C_j(\pi) = \sum_{i \in T(j,\pi)} p_i \tau_j$: completion time of a processor j
- $C_{max}(\pi) = max_j C_j(\pi)$: makespan of a schedule
- $p^j_{\text{succ}}(\pi) = e^{-\lambda_j C_j(\pi)}$: probability that processor j executes all its tasks successfully

Scheduling problem

- A schedule: $\pi : T \rightarrow Q$
- $T(j, \pi) = \{i \mid \pi(i) = j\}$: set of tasks mapped to processor j
- $C_j(\pi) = \sum_{i \in T(j,\pi)} p_i \tau_j$: completion time of a processor j
- $C_{max}(\pi) = max_j C_j(\pi)$: makespan of a schedule
- $p^j_{\text{succ}}(\pi) = e^{-\lambda_j C_j(\pi)}$: probability that processor j executes all its tasks successfully
- Assumption: faults are independent.

$$p_{\text{succ}} = \prod_j p^j_{\text{succ}}(\pi) = e^{-\sum_j C_j(\pi)\lambda_j}$$

probability that schedule π finishes correctly.

Criteria

Two criteria to optimize:

- **Makespan**: minimize

$$M = C_{max}(\pi) = max_j C_j(\pi)$$

- **Reliability**: maximize

$$p_{\text{succ}} = \prod_j e^{-C_j(\pi)\lambda_j} = e^{-\sum_j C_j(\pi)\lambda_j}$$

or minimize

$$Rel = \sum_j C_j(\pi)\lambda_j$$

Two antagonistic criteria

Question: Is there an algorithm which approximates both criteria at the same time ?

Two antagonistic criteria

Question: Is there an algorithm which approximates both criteria at the same time ?

Theorem

The problem of minimizing the C_{max} and Rel is unapproximable within a constant factor.

Two antagonistic criteria

Question: Is there an algorithm which approximates both criteria at the same time ?

Theorem

The problem of minimizing the C_{max} and Rel is unapproximable within a constant factor.

Proof Two machines such that $\tau_2 = \tau_1/k$ and $\lambda_2 = k^2\lambda_1$ $(k \in \mathbb{R}^{+*})$.

Two antagonistic criteria

Question: Is there an algorithm which approximates both criteria at the same time ?

Theorem

The problem of minimizing the C_{max} and Rel is unapproximable within a constant factor.

Proof Two machines such that $\tau_2 = \tau_1/k$ and $\lambda_2 = k^2\lambda_1$ ($k \in \mathbb{R}^{+*}$).
A single task t_1 where $p_1 = 1$.

Two antagonistic criteria

Question: Is there an algorithm which approximates both criteria at the same time ?

Theorem

The problem of minimizing the C_{max} and Rel is unapproximable within a constant factor.

Proof Two machines such that $\tau_2 = \tau_1/k$ and $\lambda_2 = k^2\lambda_1$ ($k \in \mathbb{R}^{+*}$).
A single task t_1 where $p_1 = 1$.
$C_{max}(\pi_1) = \tau_1$ and $C_{max}(\pi_2) = \tau_1/k$. This leads to $C_{max}(\pi_1)/C_{max}(\pi_2) = k$. π_1 is not an approximation on both criteria

Two antagonistic criteria

Question: Is there an algorithm which approximates both criteria at the same time ?

Theorem

The problem of minimizing the C_{max} and Rel is unapproximable within a constant factor.

Proof Two machines such that $\tau_2 = \tau_1/k$ and $\lambda_2 = k^2\lambda_1$ ($k \in \mathbb{R}^{+*}$).
A single task t_1 where $p_1 = 1$.
$C_{max}(\pi_1) = \tau_1$ and $C_{max}(\pi_2) = \tau_1/k$. This leads to $C_{max}(\pi_1)/C_{max}(\pi_2) = k$. π_1 is not an approximation on both criteria
$Rel(\pi_1) = \tau_1\lambda_1$ and $Rel(\pi_2) = \tau_1\lambda_1 k$. This leads to $Rel(\pi_2)/Rel(\pi_1) = k$. π_2 is not an approximation on both criteria.

Two antagonistic criteria

Question: Is there an algorithm which approximates both criteria at the same time ?

Theorem

The problem of minimizing the C_{max} and Rel is unapproximable within a constant factor.

Proof Two machines such that $\tau_2 = \tau_1/k$ and $\lambda_2 = k^2\lambda_1$ ($k \in \mathbb{R}^{+*}$).
A single task t_1 where $p_1 = 1$.
$C_{max}(\pi_1) = \tau_1$ and $C_{max}(\pi_2) = \tau_1/k$. This leads to $C_{max}(\pi_1)/C_{max}(\pi_2) = k$. π_1 is not an approximation on both criteria
$Rel(\pi_1) = \tau_1\lambda_1$ and $Rel(\pi_2) = \tau_1\lambda_1 k$. This leads to $Rel(\pi_2)/Rel(\pi_1) = k$. π_2 is not an approximation on both criteria.

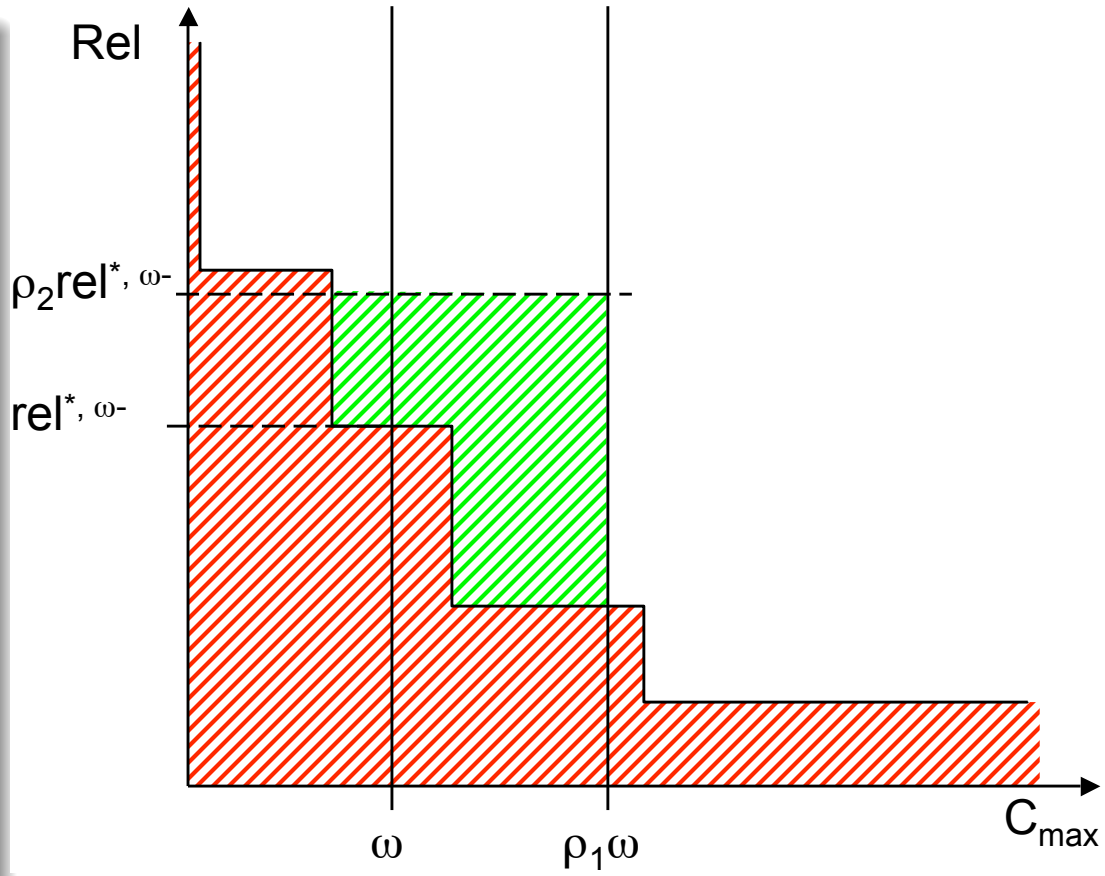
No solution of this instance approximates both criteria.

Studied problem

Minimizing *Rel* when subject to a makespan objective.

Definition

- ω: makespan threshold value
- $\langle \bar{\rho_1}, \rho_2 \rangle$-approximation algorithm.
- $C_{max} \leq \rho_1 \omega$
- $rel \leq \rho_2 rel^{*,\omega-}$
- $rel^{*,\omega-}$ is the best possible value of *rel* in schedules whose makespan is less than ω.



Outline

Independent unitary tasks

$o_i = 1$ and $E = \emptyset$, $n = |V|$.

Independent unitary tasks

$o_i = 1$ and $E = \emptyset$, $n = |V|$.

Algorithm 1 Makespan-optimal allocation for independent unitary tasks

for i=1 to P

$n_i \leftarrow \left\lfloor \frac{1/\tau_i}{\sum 1/\tau_i} \right\rfloor \times n$

while $\sum n_i < n$

$k = \text{argmin}(\tau_k(n_k + 1))$

$n_k \leftarrow n_k + 1$

Independent unitary tasks

$o_i = 1$ and $E = \emptyset$, $n = |V|$.

Algorithm 1 Makespan-optimal allocation for independent unitary tasks

for i=1 to P

$n_i \leftarrow \left\lfloor \frac{1/\tau_i}{\sum 1/\tau_i} \right\rfloor \times n$

while $\sum n_i < n$

$k = \text{argmin}(\tau_k(n_k + 1))$

$n_k \leftarrow n_k + 1$

Above algorithm gives M_{opt} the best achievable makespan.
For the reliability criteria the user gives the value of α that tells how far from the optimal makespan he/she can tolerate to be.
Then we compute a schedule such that:

- $\omega \leq \alpha M_{\text{opt}}$
- it has the best reliability among all the schedules with makespan $\leq \omega$.

An $\langle \bar{\alpha}, 1 \rangle$ algorithm for Independent unitary tasks

Algorithm 2 Optimal reliable allocation for independent unitary tasks

Input: $\alpha \in [1, +\infty[$

Compute $\omega = \alpha M_{\text{opt}}$ using previous algorithm

Sort the processor by increasing $\lambda_i \tau_i$

$X \leftarrow 0$

for i=1 to P

  if $X < N$

    $n_i \leftarrow \min\left(N - X, \left\lfloor \frac{\omega}{\tau_i} \right\rfloor\right)$

  else

    $n_i \leftarrow 0$

  $X \leftarrow X + n_i$

Proof of optimality of the reliability

We need to show that $\sum_{i \in [1,P]} n_i \lambda_i \tau_i$ is minimum.

Proof of optimality of the reliability

We need to show that $\sum_{i \in [1,P]} n_i \lambda_i \tau_i$ is minimum.

- First let us remark that the algorithm fills the processor of task in the increasing order of $\lambda_i \tau_i$.

Proof of optimality of the reliability

We need to show that $\sum_{i \in [1,P]} n_i \lambda_i \tau_i$ is minimum.

- First let us remark that the algorithm fills the processor of task in the increasing order of $\lambda_i \tau_i$.

$\Rightarrow$ any other valid allocation $\{n'_1, \ldots, n'_N\}$ is such that $n'_i < n_i$ and $n'_j > n_j$ for any $i < j$.

Proof of optimality of the reliability

We need to show that $\sum_{i\in[1,P]} n_i\lambda_i\tau_i$ is minimum.

- First let us remark that the algorithm fills the processor of task in the increasing order of $\lambda_i\tau_i$.

$\Rightarrow$ any other valid allocation $\{n'_1, \ldots, n'_N\}$ is such that $n'_i < n_i$ and $n'_j > n_j$ for any $i < j$.

- w.l.o.g. let $n'_1 = n_1 - k$, $n'_i = n_i + k$ and $n'_j = n_j$ for $k \in [1, n_i]$, $j \neq 1$ and $j \neq i$.

Proof of optimality of the reliability

We need to show that $\sum_{i\in[1,P]} n_i\lambda_i\tau_i$ is minimum.

- First let us remark that the algorithm fills the processor of task in the increasing order of $\lambda_i\tau_i$.

$\Rightarrow$ any other valid allocation $\{n'_1, \ldots, n'_N\}$ is such that $n'_i < n_i$ and $n'_j > n_j$ for any $i < j$.

- w.l.o.g. let $n'_1 = n_1 - k$, $n'_i = n_i + k$ and $n'_j = n_j$ for $k \in [1, n_i]$, $j \neq 1$ and $j \neq i$.
- Then the difference between the two objective values is:

Proof of optimality of the reliability

We need to show that $\sum_{i\in[1,P]} n_i\lambda_i\tau_i$ is minimum.

- First let us remark that the algorithm fills the processor of task in the increasing order of $\lambda_i\tau_i$.

$\Rightarrow$ any other valid allocation $\{n'_1, \ldots, n'_N\}$ is such that $n'_i < n_i$ and $n'_j > n_j$ for any $i < j$.

- w.l.o.g. let $n'_1 = n_1 - k$, $n'_i = n_i + k$ and $n'_j = n_j$ for $k \in [1, n_i]$, $j \neq 1$ and $j \neq i$.
- Then the difference between the two objective values is:

$$\begin{aligned} X &= n_1\lambda_1\tau_1 + \ldots + n_i\lambda_i\tau_i + \ldots + n_N\lambda_N\tau_N - n'_1\lambda_1\tau_1 - \ldots - n'_i\lambda_i\tau_i - \ldots + n'_N\lambda_N\tau_N \\ &= \lambda_1\tau_1(n_1 - n'_1) + \lambda_i\tau_i(n_i - n'_i) \\ &= k\lambda_1\tau_1 - k\lambda_i\tau_i \\ &= k(\lambda_1\tau_1 - \lambda_i\tau_i) \\ &\leq 0 \text{ because } \lambda_i\tau_i \geq \lambda_1\tau_1. \end{aligned}$$

Hence, the first allocation has a smaller objective value.

Outline of the talk

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
    sort tasks in non-increasing p_i order
    sort processors in non-decreasing τ_j order
    Let H be an empty heap
    $j = 1$
    for $i = 1$ *to* n **do**
        while $P_j \in M(i)$ **do**
            Add P_j to H with key $\lambda_j \tau_j$
            $j = j + 1$
        if $H.\mathrm{empty}()$ **then**
            Return *no solution*
        schedule i on $j' = H.\min()$
        $C_{j'} = C_{j'} + p_i \tau_{j'}$
        if $C_{j'} > \omega$ **then**
            Remove j' from H
end

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if $H.\mathrm{empty}()$ **then**
      Return *no solution*
    schedule i on $j' = H.\min()$
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if $H.\mathrm{empty}()$ **then**
      Return *no solution*
    schedule i on $j' = H.\min()$
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if $H.\mathrm{empty}()$ **then**
      Return *no solution*
    schedule i on $j' = H.\min()$
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	0	0	(,)	1

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	0	0	(P_1,3)	1

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	0	0	(P_1,3)	2

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	0	(P_1,3)	2

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	0	(P_1,3),(P_2,2)	2

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	0	(P_1,3),(P_2,2)	3

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	10	(P_1,3),(P_2,2)	3

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	10	(P_1,3),(P_2,2)	3
3	7	10	(P_1,3),(P_2,2)	3

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	10	(P_1,3),(P_2,2)	3
3	7	18	(P_1,3),(P_2,2)	3

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	10	(P_1,3),(P_2,2)	3
3	7	18	(P_1,3)	3

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	10	(P_1,3),(P_2,2)	3
3	7	18	(P_1,3)	3
4	7	18	(P_1,3)	3

A Dual Approximation algorithm

CMLT

ω: makespan threshold
$M(i) = \{j \mid p_{ij} \leq \omega\}$
begin
  sort tasks in non-increasing p_i order
  sort processors in non-decreasing τ_j order
  Let H be an empty heap
  $j = 1$
  for $i = 1$ *to* n **do**
    while $P_j \in M(i)$ **do**
      Add P_j to H with key $\lambda_j \tau_j$
      $j = j + 1$
    if H.empty() **then**
      Return *no solution*
    schedule i on $j' = H$.min()
    $C_{j'} = C_{j'} + p_i \tau_{j'}$
    if $C_{j'} > \omega$ **then**
      Remove j' from H
end

$p = \{7, 5, 4, 2\}$
$\lambda_1 = 3,\ \tau_1 = 1,\ p_{i1} = \{7, 5, 4, 2\}$
$\lambda_2 = 1,\ \tau_2 = 2,\ p_{i2} = \{14, 10, 8, 4\}$
$\omega = 10$
$M(1) = \{1\}$
$M(2) = M(3) = M(4) = \{1, 2\}$

i	C_1	C_2	H	j
	0	0	(,)	1
1	7	0	(P_1,3)	2
2	7	10	(P_1,3),(P_2,2)	3
3	7	18	(P_1,3)	3
4	9	18	(P_1,3)	3

Properties of CMLT

$C_{max}(\text{CMLT}) \leq 2\omega$

- Tasks are ranked by non-increasing duration
- Processor j is used until $C_j > \omega$

Properties of CMLT

$C_{max}(\text{CMLT}) \leq 2\omega$

- Tasks are ranked by non-increasing duration
- Processor j is used until $C_j > \omega$

If CMLT does not return a schedule then there is no schedule π with $C_{max}(\pi) < \omega$

- $M(i) = \{j \mid p_{ij} \leq \omega\}$

Properties of CMLT

$C_{max}(\text{CMLT}) \leq 2\omega$

- Tasks are ranked by non-increasing duration
- Processor j is used until $C_j > \omega$

If CMLT does not return a schedule then there is no schedule π with $C_{max}(\pi) < \omega$

- $M(i) = \{j \mid p_{ij} \leq \omega\}$
- If task i cannot be executed on any processors of $M(i) \Longrightarrow \forall j \in M(i), C_j > \omega$

Properties of CMLT

$C_{max}(\text{CMLT}) \leq 2\omega$

- Tasks are ranked by non-increasing duration
- Processor j is used until $C_j > \omega$

If CMLT does not return a schedule then there is no schedule π with $C_{max}(\pi) < \omega$

- $M(i) = \{j \mid p_{ij} \leq \omega\}$
- If task i cannot be executed on any processors of $M(i) \Longrightarrow \forall j \in M(i), C_j > \omega$
- $\forall i, i' \in T$ such that $p_i \leq p_{i'}$, $M(i') \subseteq M(i)$

Properties of CMLT

$C_{max}(\text{CMLT}) \leq 2\omega$

- Tasks are ranked by non-increasing duration
- Processor j is used until $C_j > \omega$

If CMLT does not return a schedule then there is no schedule π with $C_{max}(\pi) < \omega$

- $M(i) = \{j \mid p_{ij} \leq \omega\}$
- If task i cannot be executed on any processors of $M(i) \Longrightarrow \forall j \in M(i), C_j > \omega$
- $\forall i, i' \in T$ such that $p_i \leq p_{i'}$, $M(i') \subseteq M(i)$
- $\forall i' \leq i$ such that $p_{i'} > p_i$ must have been schedule to a processor of $M(i)$.

Properties of CMLT

$C_{max}(\text{CMLT}) \leq 2\omega$

- Tasks are ranked by non-increasing duration
- Processor j is used until $C_j > \omega$

If CMLT does not return a schedule then there is no schedule π with $C_{max}(\pi) < \omega$

- $M(i) = \{j \mid p_{ij} \leq \omega\}$
- If task i cannot be executed on any processors of $M(i) \Longrightarrow \forall j \in M(i), C_j > \omega$
- $\forall i, i' \in T$ such that $p_i \leq p_{i'}$, $M(i') \subseteq M(i)$
- $\forall i' \leq i$ such that $p_{i'} > p_i$ must have been schedule to a processor of $M(i)$.
- There is more operations in the set of tasks $\{i' \leq i\}$ than processors in $M(i)$ can execute in ω units of time.

Properties of CMLT

CMLT generates a schedule such that $rel \leq rel^{*,\omega-}$

Idea: tasks are scheduled on the processor that have the minimum $\lambda\tau$ product which is known to be optimal for unitary tasks.

CMLT a $\langle \bar{2}, 1 \rangle$-approximation algorithm

Properties of CMLT

CMLT generates a schedule such that $rel \leq rel^{*,\omega-}$

Idea: tasks are scheduled on the processor that have the minimum $\lambda\tau$ product which is known to be optimal for unitary tasks.

CMLT a $\langle \bar{2}, 1 \rangle$-approximation algorithm

The time complexity of CMLT is in $O(n \log n + m \log m)$

- cost of sorting tasks: $O(n \log n)$
- cost of sorting processors: $O(m \log m)$
- Adding elements to heap: $O(m \log m)$
- Getting and removing elements from heap $0(1 * n)$

Outline of the talk

Back to Definition

Optimality

In multi-criteria, k functions are optimized $f_1 \dots f_k$. Solution S' is **Pareto dominated** by S if $\forall i, f_i(S) \leq f_i(S')$. A solution which is not dominated is **Pareto-optimal**.

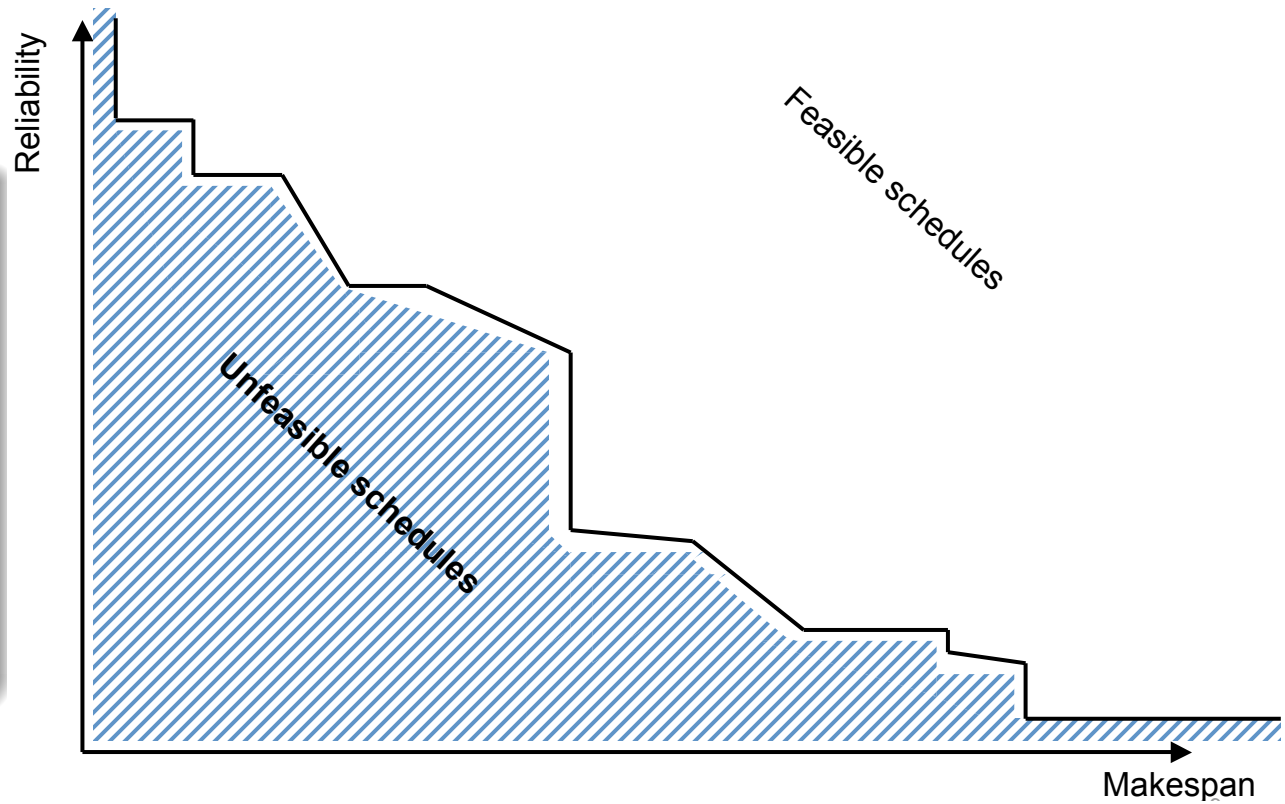
Back to Definition

Optimality

In multi-criteria, k functions are optimized $f_1 \dots f_k$. Solution S' is **Pareto dominated** by S if $\forall i, f_i(S) \leq f_i(S')$. A solution which is not dominated is **Pareto-optimal**.

Pareto Front

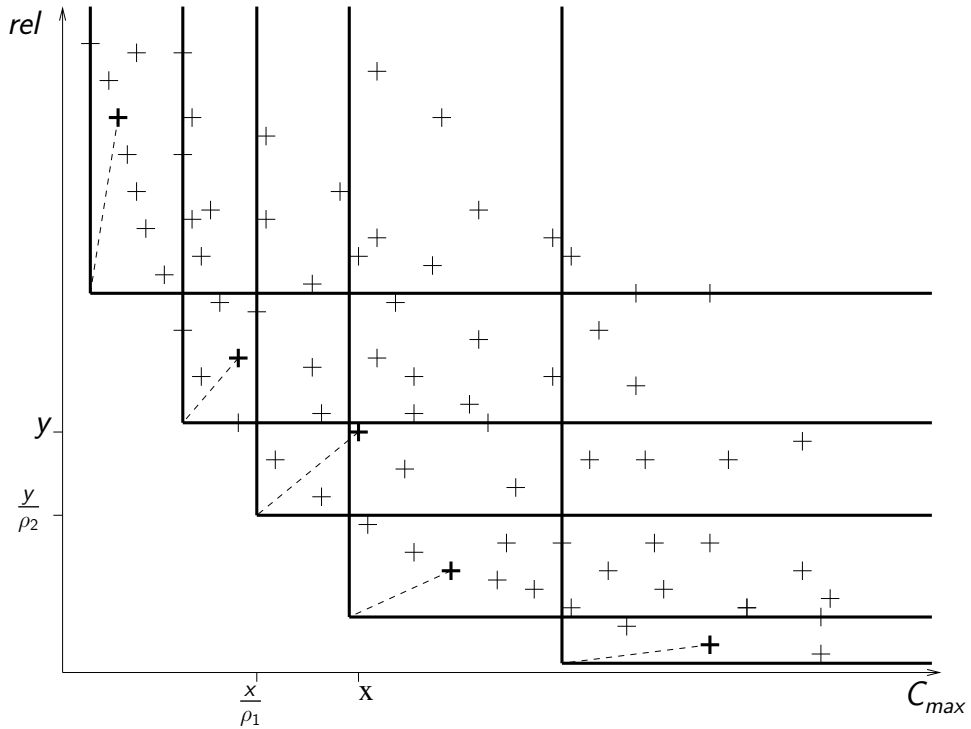
The **Pareto front** is the set P of all Pareto optimal solution. We should remark that the size of P can be exponential.



Approximating the Pareto front

Definition

Informally, P is a $\rho = (\rho_1, \rho_2, \ldots, \rho_k)$ approximation of P^* if each solution $S^* \in P^*$ is ρ approximated by a solution $S \in P$. Formally, $\forall S^* \in P^*, \exists S \in P, \forall i, f_i(S) \leq \rho_i f_i(S^*)$.



Bold crosses are a (ρ_1, ρ_2)-approximation of the Pareto set.

Pareto front approximation algorithm

Papadimitriou and Yannakakis approximation method of the Pareto front

Data: ϵ a positive real number

Result: S a set of solutions

$C_{max}^{min} = \frac{\sum_i p_i}{\sum_j \frac{1}{\tau_j}}$

$C_{max}^{max} = \sum_i p_i \max_j \tau_j$

begin

  $i = 0$

  $S = \emptyset$

  while $i \leq \left\lceil \log_{1+\epsilon/2} \left(\frac{C_{max}^{max}}{C_{max}^{min}} \right) \right\rceil$ **do**

    $\omega_i = (1 + \frac{\epsilon}{2})^i C_{max}^{min}$

    $\pi_i = CMLT(\omega_i)$

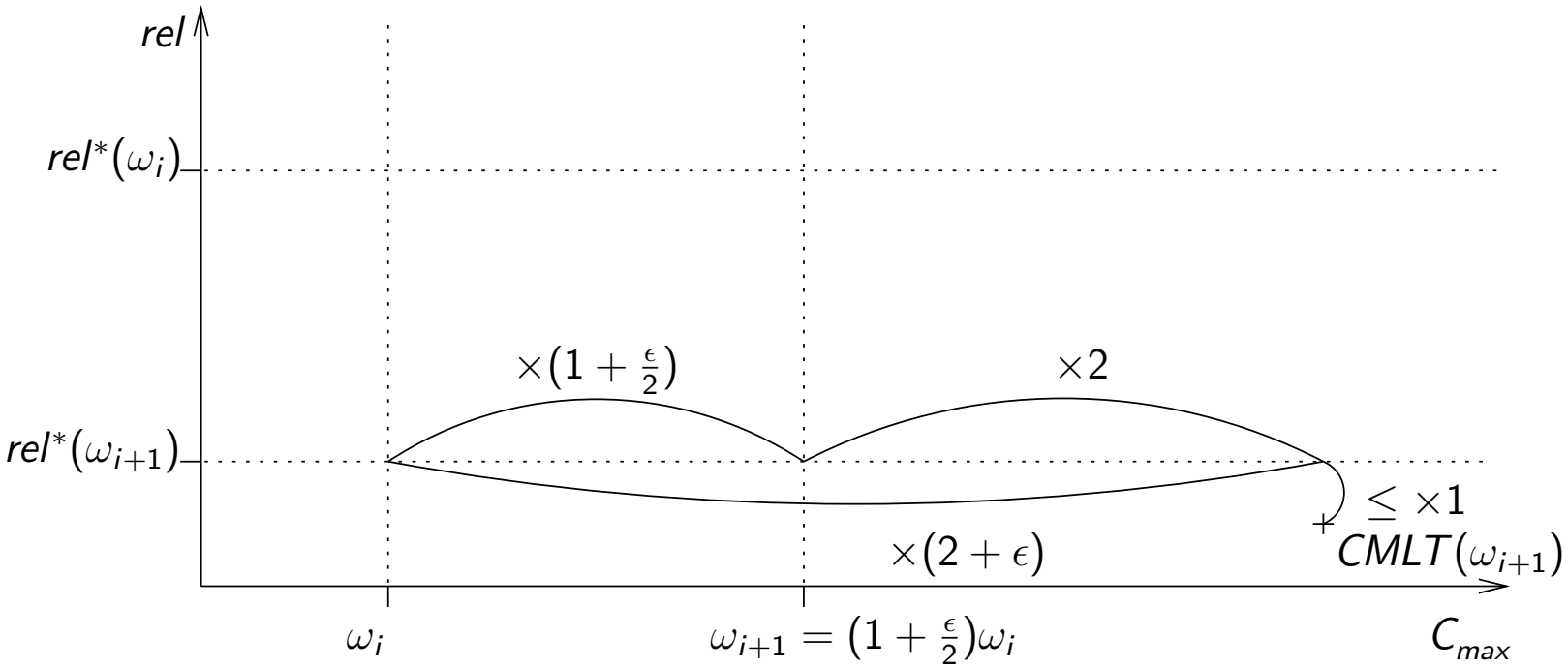
    $S = S \cup \pi_i$

    $i = i + 1$

  return S

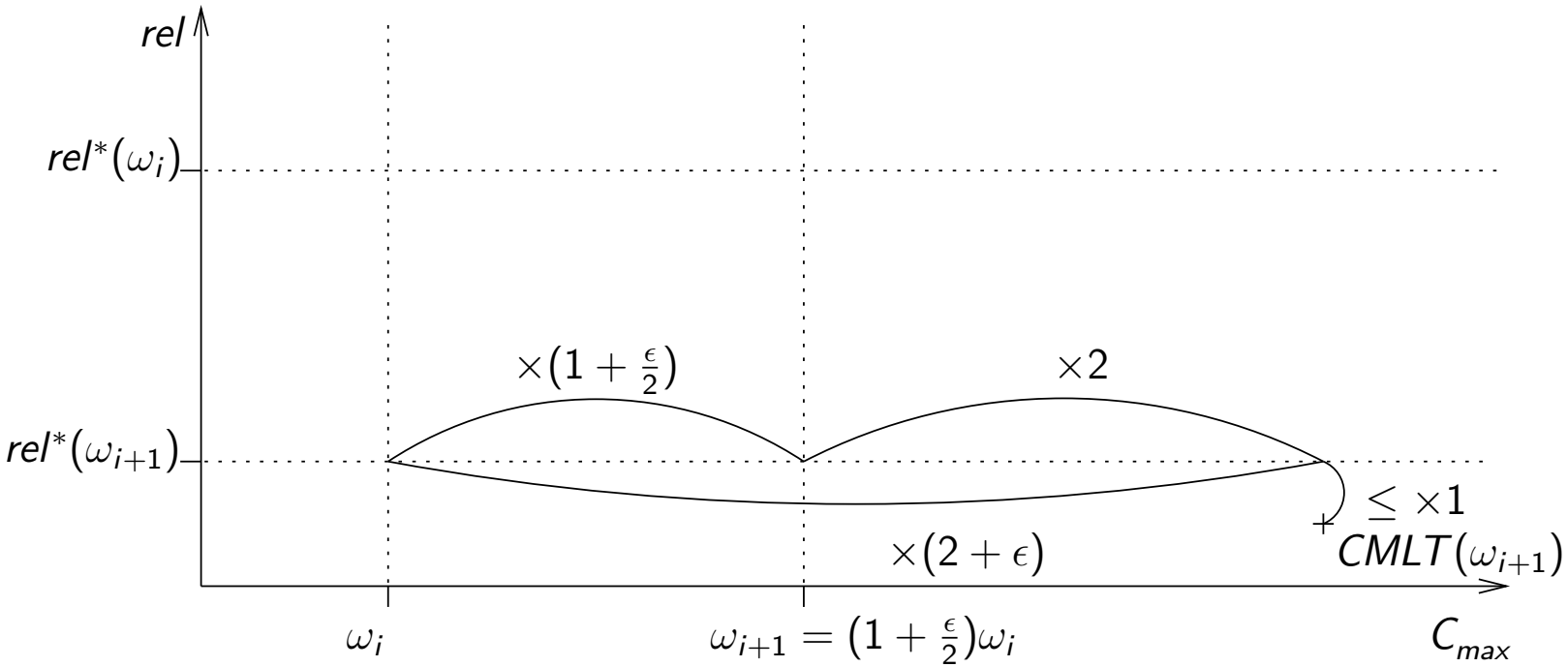
end

A $(2+\epsilon, 1)$ approximation algorithm of the Pareto front



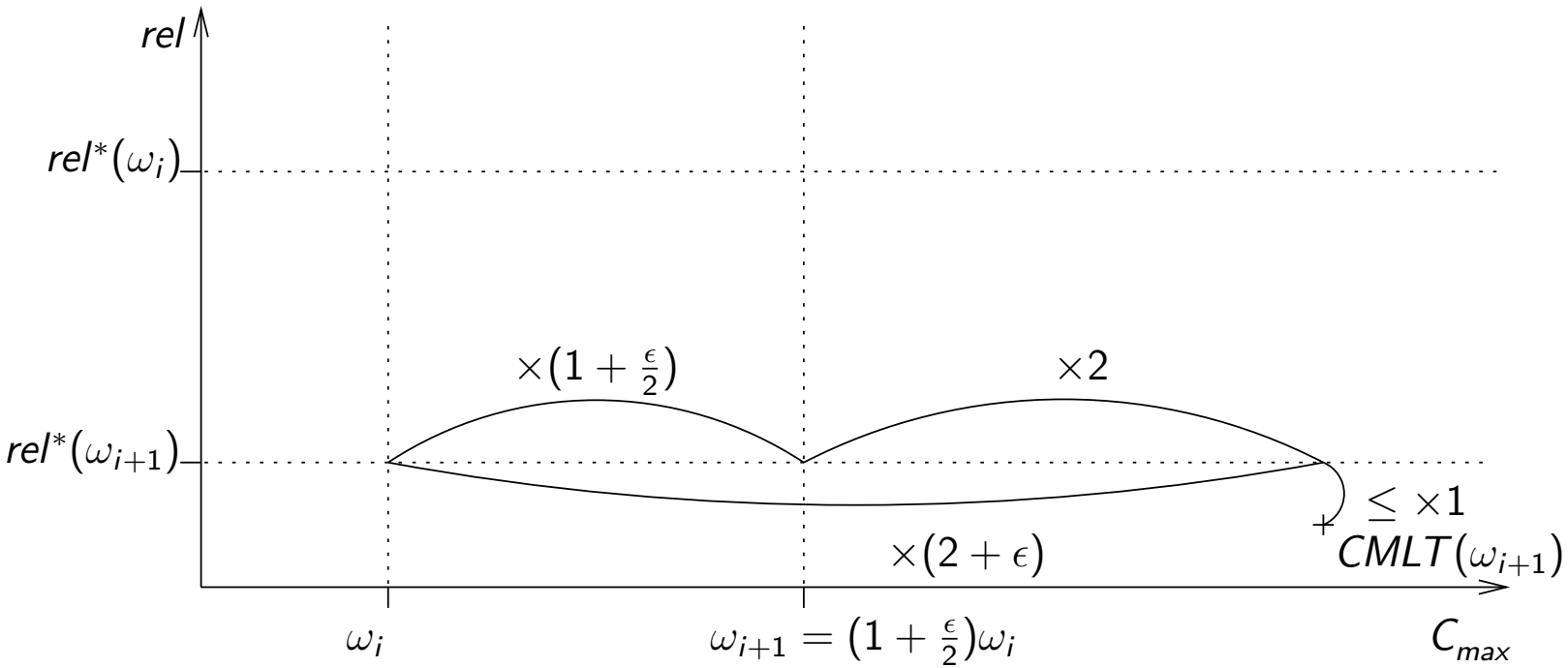
- $C_{max}(CMLT(\omega_{i+1})) \leq 2.\omega_{i+1}$

A $(2+\epsilon, 1)$ approximation algorithm of the Pareto front



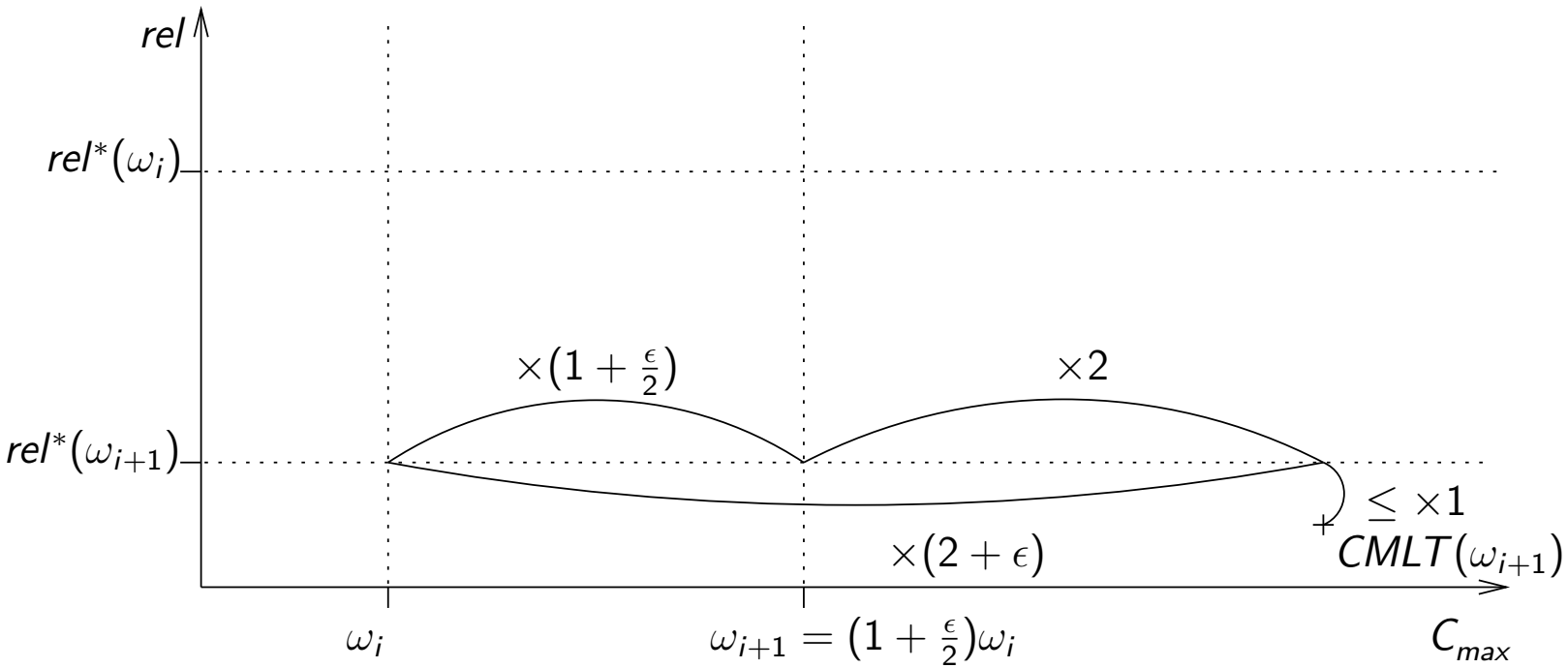
- $C_{max}(CMLT(\omega_{i+1})) \leq 2.\omega_{i+1}$
- $\omega_{i+1} = (1+\frac{\epsilon}{2})\omega_i \Longrightarrow C_{max}(CMLT(\omega_{i+1})) \leq (2+\epsilon)\omega_i$

A $(2+\epsilon, 1)$ approximation algorithm of the Pareto front



- $C_{max}(CMLT(\omega_{i+1})) \leq 2.\omega_{i+1}$
- $\omega_{i+1} = (1+\frac{\epsilon}{2})\omega_i \Longrightarrow C_{max}(CMLT(\omega_{i+1})) \leq (2+\epsilon)\omega_i$
- $rel(CMLT(\omega_{i+1})) \leq rel^*(\omega_{i+1})$

A $(2+\epsilon, 1)$ approximation algorithm of the Pareto front



- $C_{max}(CMLT(\omega_{i+1})) \leq 2.\omega_{i+1}$
- $\omega_{i+1} = (1+\frac{\epsilon}{2})\omega_i \Longrightarrow C_{max}(CMLT(\omega_{i+1})) \leq (2+\epsilon)\omega_i$
- $rel(CMLT(\omega_{i+1})) \leq rel^*(\omega_{i+1})$
- $CMLT(\omega_{i+1})$ is a $(2+\epsilon, 1)$-approximation of $(\omega_i, rel^*(\omega_{i+1}))$

Cardinality and complexity

Cardinality

- Number of solutions less than:
  $\left\lceil \log_{1+\frac{\epsilon}{2}} \frac{C_{max}^{max}}{C_{max}^{min}} \right\rceil \leq \left\lceil \log_{1+\frac{\epsilon}{2}} max_i \tau_i \sum_j 1/\tau_j \right\rceil \leq \left\lceil \log_{1+\frac{\epsilon}{2}} m \frac{max_i \tau_i}{min_i \tau_i} \right\rceil$
- polynomial in $1/\epsilon$ and in m

Cardinality and complexity

Cardinality

- Number of solutions less than:
  $\left\lceil \log_{1+\frac{\epsilon}{2}} \frac{C_{max}^{max}}{C_{max}^{min}} \right\rceil \leq \left\lceil \log_{1+\frac{\epsilon}{2}} max_i \tau_i \sum_j 1/\tau_j \right\rceil \leq \left\lceil \log_{1+\frac{\epsilon}{2}} m \frac{max_i \tau_i}{min_i \tau_i} \right\rceil$
- polynomial in $1/\epsilon$ and in m

Complexity

- Sorting the tasks:independent of ω (can be done once for all)
- Complexity of the Pareto front approximation algorithm:
  $O(n \log n + \left\lceil \log_{1+\epsilon/2}(\frac{C_{max}^{max}}{C_{max}^{min}}) \right\rceil (n + m \log m))$

Outline of the talk

Conclusion

Problem

- Reliability is a crucial issue
- Scheduling independent tasks on related processors
- Optimizing makespan and reliability
- These criteria are conflicting

Conclusion

Problem

- Reliability is a crucial issue
- Scheduling independent tasks on related processors
- Optimizing makespan and reliability
- These criteria are conflicting

Contribution

- a $\langle \bar{\alpha}, 1 \rangle$-approximation with $\alpha \in [1, +\infty[$ for unitary independant tasks
- CMLT: a $\langle \bar{2}, 1 \rangle$-approximation for non unitary independant tasks algorithm
- A $(2 + \epsilon, 1)$-approximation of the Pareto front
- Exemplify the role of the $\lambda\tau$ product