Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems

Jack J. Dongarra, Emmanuel Jeannot, **E. Saule** and Zhiao Shi

INRIA & Innovative Computing Laboratory & LIG
dongarra@cs.utk.edu, ejeannot@loria.fr, erik.saule@imag.fr ,shi@cs.utk.edu

Outline of the talk

1. Introduction, related work and modeling
2. The problem
3. Independent unitary tasks
4. Independent tasks
5. General Case
6. Conclusion

Introduction

Problem studied:

- scheduling DAG
- heterogeneous systems
- hardware can fail

Bi-criteria objective:

- given a makespan objective
- optimize reliability

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

- $G = (V, E)$: a DAG.
- $v_i \in V$ is associated a number of operations: o_i.
- $n = |V|$
- $e_i = (i, j) \in E$ is associated l_i the time to send data from task v_i to task v_j (if they are not executed on the processor).
- a set P of m processors
- processor $p_j \in P$ is associated with two values:
  - τ_j the time to perform one operation and
  - λ_j the failure rate.
- v_i executed on p_j will last $o_i \times \tau_j$.

Modeling

- $G = (V, E)$: a DAG.
- $v_i \in V$ is associated a number of operations: o_i.
- $n = |V|$
- $e_i = (i, j) \in E$ is associated l_i the time to send data from task v_i to task v_j (if they are not executed on the processor).
- a set P of m processors
- processor $p_j \in P$ is associated with two values:
  - τ_j the time to perform one operation and
  - λ_j the failure rate.
- v_i executed on p_j will last $o_i \times \tau_j$.

Assumption:

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

⇒ failure model follows an exponential law.

⇒ probability that v_i finishes (correctly) its execution:

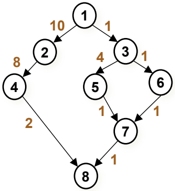
$$e^{-o_i \times \tau_j \times \lambda_j}$$

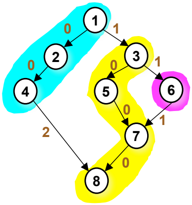
Outline

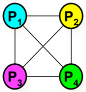
Scheduling problem

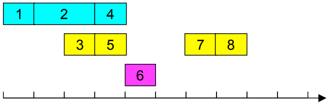
Allocate tasks to processors such that:

- two tasks cannot be allocated to the same processor at the same time,
- dependencies are respected.









Criteria

C_j: termination date of processor j

Two criteria to optimize:

- **Makespan**: minimize

$$M = \max(C_j)$$

- **Reliability**: maximize

$$p_{succ} = \prod_{j=1}^{m} e^{-C_j \lambda_j} = e^{-\sum_{j=1}^{m} C_j \lambda_j}$$

or minimize

$$Rel = \sum_{j=1}^{m} C_j \lambda_j$$

Approximation algorithm and probability

Let p_{succ} (resp. p_{fail}) be the probability of success (resp. of failure) of a schedule S.
Let $\tilde{p}_{\text{succ}}$ (resp. $\tilde{p}_{\text{fail}}$) be the optimal probability of success (resp. of failure) for the same input as S.

Approximation algorithm and probability

Let p_{succ} (resp. p_{fail}) be the probability of success (resp. of failure) of a schedule S.
Let $\tilde{p}_{\text{succ}}$ (resp. $\tilde{p}_{\text{fail}}$) be the optimal probability of success (resp. of failure) for the same input as S.

$$\beta = 5\,, \tilde{p}_{\text{fail}} = 0.3 \Rightarrow p_{\text{fail}} \leq \beta \cdot \tilde{p}_{\text{fail}} = 5 \times 0.3 = 1.5!$$

Approximation algorithm and probability

Let p_{succ} (resp. p_{fail}) be the probability of success (resp. of failure) of a schedule S.
Let $\tilde{p}_{\text{succ}}$ (resp. $\tilde{p}_{\text{fail}}$) be the optimal probability of success (resp. of failure) for the same input as S.

$$\beta = 5\,,\tilde{p}_{\text{fail}} = 0.3 \Rightarrow p_{\text{fail}} \leq \beta \cdot \tilde{p}_{\text{fail}} = 5 \times 0.3 = 1.5!$$

Proposition

$$p_{succ} \geq \tilde{p}_{succ}^{\beta} \Rightarrow p_{fail} \leq \beta \cdot \tilde{p}_{fail}$$

$\Rightarrow$ minimize $Rel = \sum_{j=1}^{m} C_j \lambda_j$

Optimizing rel

Proposition

Let S be a schedule where all the tasks have been assigned, in topological order, to the processor i such that $\lambda_i \tau_i$ **is minimum**. *Then any schedule S' is such that $p'_{succ} \leq p_{succ}$.*

Optimizing rel

Proposition

Let S be a schedule where all the tasks have been assigned, in topological order, to the processor i such that $\lambda_i \tau_i$ **is minimum**. *Then any schedule S' is such that $p'_{succ} \leq p_{succ}$.*

Proof

- s.w.l.o.g $i = 1$ (*i. e.*, $\forall j: \ \tau_1 \lambda_1 \leq \tau_j \lambda_j$).
- $p_{\text{succ}} = e^{-C_1 \lambda_1}$, $p'_{\text{succ}} = e^{-\sum_{j=0}^{m} C'_j \lambda_j}$.
- $T = T_2 \cup \ldots \cup T_m$, sets of the tasks allocated to processors $2, \ldots, m$ by S'.
- $C'_1 \geq C_1 - \tau_1 \sum_{v_i \in T} o_i$.
- $\forall\, 2 \leq j \leq m, \ C'_j \geq \tau_j \sum_{v_i \in T_j} o_i$

Optimizing rel

Proposition

Let S be a schedule where all the tasks have been assigned, in topological order, to the processor i such that $\lambda_i \tau_i$ **is minimum**. *Then any schedule S' is such that $p'_{succ} \leq p_{succ}$.*

Proof

- s.w.l.o.g $i = 1$ (*i. e.*, $\forall j: \ \tau_1 \lambda_1 \leq \tau_j \lambda_j$).
- $p_{\text{succ}} = e^{-C_1 \lambda_1}$, $p'_{\text{succ}} = e^{-\sum_{j=0}^{m} C'_j \lambda_j}$.
- $T = T_2 \cup \ldots \cup T_m$, sets of the tasks allocated to processors $2, \ldots, m$ by S'.
- $C'_1 \geq C_1 - \tau_1 \sum_{v_i \in T} o_i$.
- $\forall \, 2 \leq j \leq m, \ C'_j \geq \tau_j \sum_{v_i \in T_j} o_i$

$$\sum_{j=1}^{m} C'_j \lambda_j - C_1 \lambda_1 \quad \geq \quad \sum_{j=2}^{m} \left((\tau_j \lambda_j - \tau_1 \lambda_1) \sum_{v_i \in T_j} o_i \right) \geq 0$$

$$\Rightarrow \frac{p_{\text{succ}}}{p'_{\text{succ}}} = e^{\sum_{j=1}^{m} C'_j \lambda_j - C_1 \lambda_1} \geq 1$$

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 $\lambda_2 = k^2\lambda_1$ ($k \in \mathbb{R}^{+*}$).
A single task t_1 where $o_1 = 1$.
$C_{\max}(S_1) = \tau_1$ and $C_{\max}(S_2) = \tau_1/k$. This leads to $C_{\max}(S_1)/C_{\max}(S_2) = k$. S_1 is not an approximation on both criteria
$Rel(S_1) = \tau_1\lambda_1$ and $Rel(S_2) = \tau_1\lambda_1 k$. This leads to $Rel(S_1)/Rel(S_2) = k$. S_2 is not an approximation on both criteria.

No solution of this instance approximates both criteria.

Back to Definition

Optimality

In multi-criteria, k functions are optimized $f_1 \ldots f_k$. 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-optimale**.

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-optimale**.

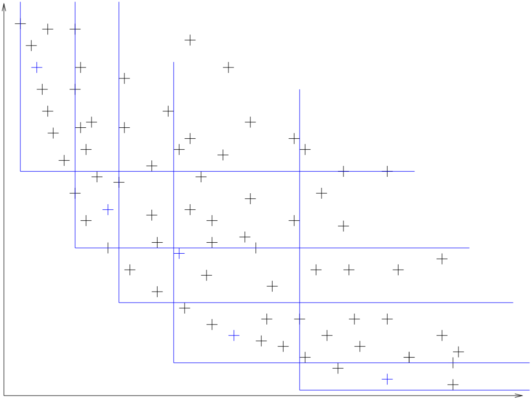
Pareto Curve

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

Approximating the Pareto curve

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, $\forall S^* \in P^*, \exists S \in P, \forall i, f_i(S) \leq \rho_i f_i(S^*)$.



Bi-criteria scheduling

Objective: maximizing the reliability subject to the condition that the makespan is minimized.

- Finding the optimal makespan, is most of the time NP-hard,
- we aim at designing an "α, β"-approximation algorithm.
- "α, β"-approximation algorithm:
  - makespan at most α times larger than the optimal one,
  - probability of failure is at most β times larger than the optimal one (among the schedules that minimize the makespan).

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

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

while $\sum n_i < n$

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

$\quad n_k \leftarrow n_k + 1$

Theorem

Algorithm 1 is optimal for the Makespan

Optimal algorithm for Independent unitary tasks

Algorithm 2 Optimal reliable allocation for independent unitary tasks

Input: $\alpha \in [1, +\infty[$
Compute $M = \alpha M_{\text{opt}}$ using previous algorithm
Sort the processor by increasing $\lambda_i \tau_i$
$X \leftarrow 0$
for i=1 to P
  if $X < N$
    $n_i \leftarrow \min\left(N - X, \left\lfloor \frac{M}{\tau_i} \right\rfloor\right)$
  else
    $n_i \leftarrow 0$
  $X \leftarrow X + n_i$

Optimal algorithm for Independent unitary tasks

Algorithm 2 Optimal reliable allocation for independent unitary tasks

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

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

Sort the processor by increasing $\lambda_i \tau_i$

$X \leftarrow 0$

for i=1 to P

  if $X < N$

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

  else

    $n_i \leftarrow 0$

  $X \leftarrow X + n_i$

Theorem

Algorithm 2 is an "$\alpha, 1$" approximation algorithm

A $(1+\epsilon, 1)$-approximation of the Pareto curve

Algorithm 3 Pareto Curve approximation algorithm

Input: $\epsilon \in [0, +\infty[$

Compute M_{opt} using Algorithm 1.

Compute M_{max} using Proposition 2.

$S \leftarrow \emptyset$

for i=1:$\lceil log_{1+\epsilon} \frac{M_{max}}{M_{\text{opt}}} \rceil$

  let S_i=Algorithm 2 with $\alpha = (1+\epsilon)^i$

  $S \leftarrow S \cup S_i$

return S

A $(1+\epsilon, 1)$-approximation of the Pareto curve

Algorithm 3 Pareto Curve approximation algorithm

Input: $\epsilon \in [0, +\infty[$
Compute M_{opt} using Algorithm 1.
Compute M_{max} using Proposition 2.
$S \leftarrow \emptyset$
for i=1:$\lceil log_{1+\epsilon} \frac{M_{max}}{M_{\text{opt}}} \rceil$
  let S_i=Algorithm 2 with $\alpha = (1+\epsilon)^i$
  $S \leftarrow S \cup S_i$
return S

proof.

Let σ be a Pareto-optimal schedule. Then
$(1+\epsilon)^k M_{\text{opt}} \leq C_{\max}(\sigma) \leq (1+\epsilon)^{k+1} M_{\text{opt}}$.
S_{k+1} is an $(1+\epsilon, 1)$-approximation of σ.

Outline

Independent tasks: the makespan problem

$E = \emptyset$

Independent tasks: the makespan problem

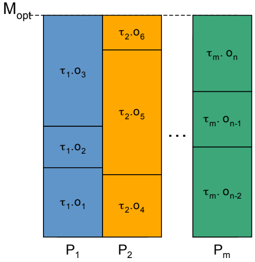
$E = \emptyset$

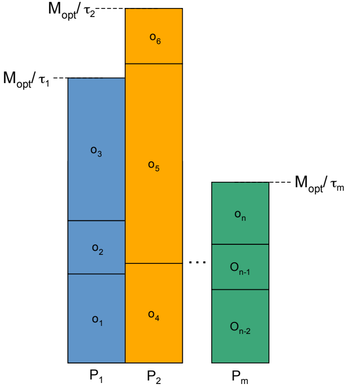
Makespan problem related to the 1-D bin-packing problem with variable bin size.

Independent tasks: the makespan problem

$E = \emptyset$

Makespan problem related to the 1-D bin-packing problem with variable bin size.

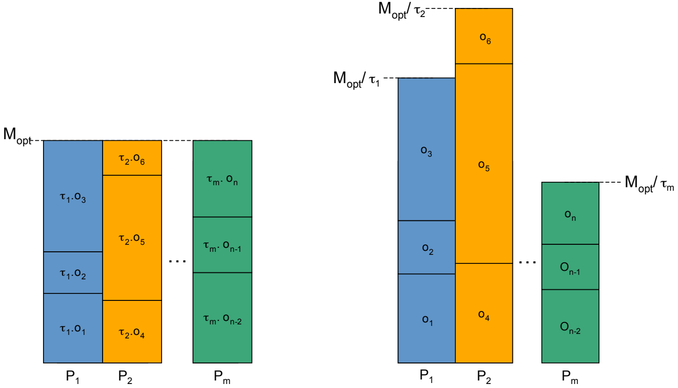




Independent tasks: the makespan problem

$E = \emptyset$

Makespan problem related to the 1-D bin-packing problem with variable bin size.



$$\sum_{j=1}^{m} \frac{M_{\text{opt}}}{\tau_i} = \sum_{i=1}^{n} o_i \Rightarrow M_{\text{opt}} = \frac{\sum_{i=1}^{n} o_i}{\sum_{j=1}^{m} \frac{1}{\tau_i}}$$

Makespan/reliability Trade-off

Recall: scheduling all the tasks on the processors i such that $i = \text{argmin}(\tau_i \lambda_i)$ leads to the best possible reliability.

Makespan/reliability Trade-off

Recall: scheduling all the tasks on the processors i such that $i = \text{argmin}(\tau_i \lambda_i)$ leads to the best possible reliability.

Generalization :

Proposition

The best possible reliability among all the schedule with makespan at most M is achieved when:

Makespan/reliability Trade-off

Recall: scheduling all the tasks on the processors i such that $i = \text{argmin}(\tau_i \lambda_i)$ leads to the best possible reliability.

Generalization :

Proposition

The best possible reliability among all the schedule with makespan at most M is achieved when:

1. *tasks are mapped to $\tilde{m}$ processors in increasing order of $\lambda_i \tau_i$,*
2. *the $\tilde{m} - 1$ first processors execute tasks up to the date M ($C_i = M$),*
3. *the $\tilde{m}$ processor executes the remaining tasks ($C_{\tilde{m}} \leq M$).*

Makespan/reliability Trade-off

Recall: scheduling all the tasks on the processors i such that $i = \text{argmin}(\tau_i \lambda_i)$ leads to the best possible reliability.

Generalization :

Proposition

The best possible reliability among all the schedule with makespan at most M is achieved when:

1. *tasks are mapped to $\tilde{m}$ processors in increasing order of $\lambda_i \tau_i$,*
2. *the $\tilde{m} - 1$ first processors execute tasks up to the date M ($C_i = M$),*
3. *the $\tilde{m}$ processor executes the remaining tasks ($C_{\tilde{m}} \leq M$).*

Remark: such a schedule is not always feasible (it just gives a lower bound).

Outline

Generalizing Scheduling Heuristics: the HEFT case

- HEFT (*Heterogenous Earlisest Finish Time*) to RHEFT (*Reliable Heterogeneous Earlisest Finish Time)*.

Generalizing Scheduling Heuristics: the HEFT case

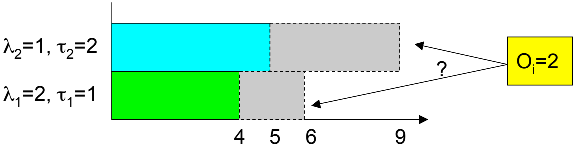
- HEFT (*Heterogenous Earlisest Finish Time*) to RHEFT (*Reliable Heterogeneous Earlisest Finish Time)*.
- HEFT: at each step of the heuristic map the task that finishes the earliest to this processors.

Generalizing Scheduling Heuristics: the HEFT case

- HEFT (*Heterogenous Earlisest Finish Time*) to RHEFT (*Reliable Heterogeneous Earlisest Finish Time)*.
- HEFT: at each step of the heuristic map the task that finishes the earliest to this processors.
- RHEFT: select the task that $T_{\text{end}j} \times \lambda_j$ is minimum.

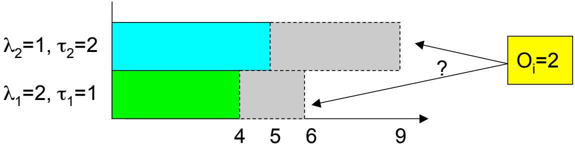
Generalizing Scheduling Heuristics: the HEFT case

- HEFT (*Heterogenous Earlisest Finish Time*) to RHEFT (*Reliable Heterogeneous Earlisest Finish Time)*.
- HEFT: at each step of the heuristic map the task that finishes the earliest to this processors.
- RHEFT: select the task that $T_{\text{end}j} \times \lambda_j$ is minimum.



Generalizing Scheduling Heuristics: the HEFT case

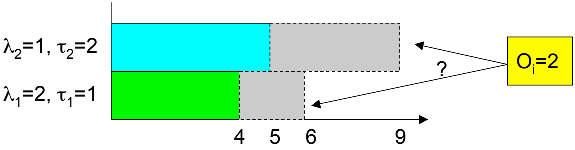
- HEFT (*Heterogenous Earlisest Finish Time*) to RHEFT (*Reliable Heterogeneous Earlisest Finish Time)*.
- HEFT: at each step of the heuristic map the task that finishes the earliest to this processors.
- RHEFT: select the task that $T_{\text{end}j} \times \lambda_j$ is minimum.



- $T_{\text{end}1} = 6,\ T_{\text{end}1} \times \lambda_1 = 12$

Generalizing Scheduling Heuristics: the HEFT case

- HEFT (*Heterogenous Earlisest Finish Time*) to RHEFT (*Reliable Heterogeneous Earlisest Finish Time)*.
- HEFT: at each step of the heuristic map the task that finishes the earliest to this processors.
- RHEFT: select the task that $T_{\text{end}\,j} \times \lambda_j$ is minimum.



- $T_{\text{end}\,1} = 6,\ T_{\text{end}\,1} \times \lambda_1 = 12$
- $T_{\text{end}\,2} = 9,\ T_{\text{end}\,2} \times \lambda_2 = 9$

Reliability/Makespan Trade-off

Two ways top find a good trade-off:

Reliability/Makespan Trade-off

Two ways top find a good trade-off:

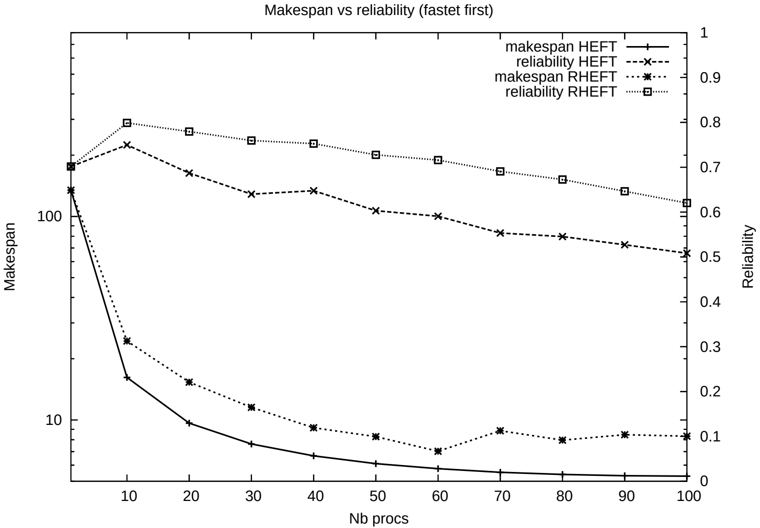
1. Choose a subset of processors; Q: which order?

Reliability/Makespan Trade-off

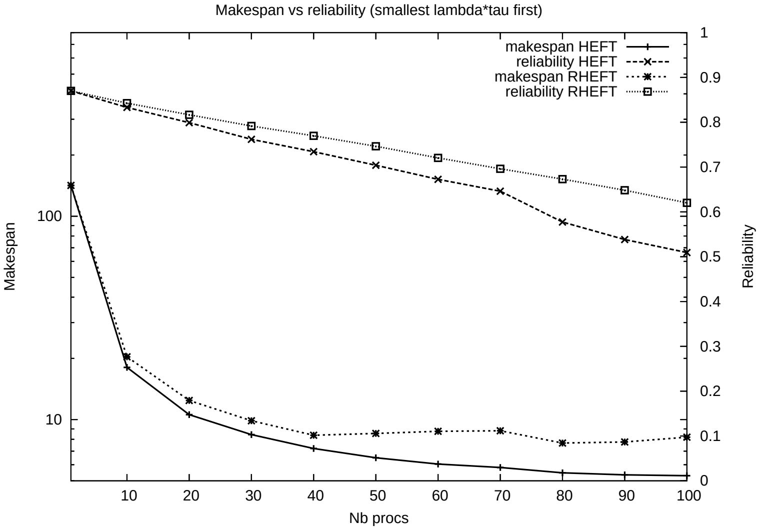
Two ways top find a good trade-off:

1. Choose a subset of processors; Q: which order?
2. Use a trade-off variable α ($\alpha = 1$ switch to HEFT, $\alpha = 0$ switch to RHEFT).

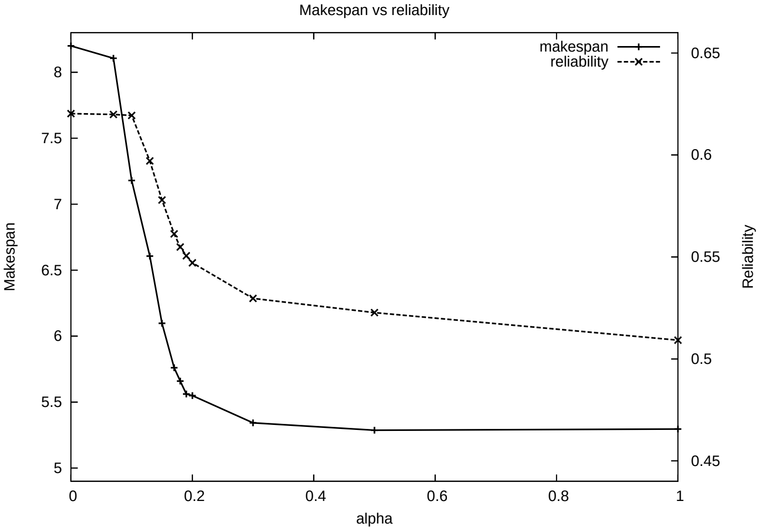
Ordering the processors: fastest first



Ordering the processors: smallest $\lambda\tau$ first



Trade-off variable



Outline

Conclusion

We have studied the problem of scheduling DAGs, with 2 objectives:

Conclusion

We have studied the problem of scheduling DAGs, with 2 objectives:

1. minimize makespan

Conclusion

We have studied the problem of scheduling DAGs, with 2 objectives:

1. minimize makespan
2. maximize reliability

Conclusion

We have studied the problem of scheduling DAGs, with 2 objectives:

1. minimize makespan
2. maximize reliability

Contribution:

Conclusion

We have studied the problem of scheduling DAGs, with 2 objectives:

1. minimize makespan
2. maximize reliability

Contribution:

- optimal algorithms for unitary independent tasks,

Conclusion

We have studied the problem of scheduling DAGs, with 2 objectives:

1. minimize makespan
2. maximize reliability

Contribution:

- optimal algorithms for unitary independent tasks,
- simple way to generalize heuristics to this context,

Conclusion

We have studied the problem of scheduling DAGs, with 2 objectives:

1. minimize makespan
2. maximize reliability

Contribution:

- optimal algorithms for unitary independent tasks,
- simple way to generalize heuristics to this context,
- characterization of the role of the $\lambda\tau$ value.