New Pruning Rules for Optimal Task Scheduling on Identical Parallel Machines

SPAA 2024, Nantes

Matthew Akram, Dominik Schreiber | June 18, 2024

The NP-complete Scheduling Problem $P||C_{\max}$

Karlsruhe Institute of Technology

$m = 4$ **processors**

$n = 10$ **jobs** $j_1, j_2, \ldots, j_n$

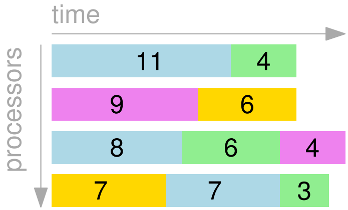
n **job durations** $W = \{11, 9, 8, 7, 7, 6, 6, 4, 4, 3\}$

The NP-complete Scheduling Problem $P||C_{\max}$

$m = 4$ **processors**

$n = 10$ **jobs** $j_1, j_2, \ldots, j_n$

n **job durations** $W = \{11, 9, 8, 7, 7, 6, 6, 4, 4, 3\}$

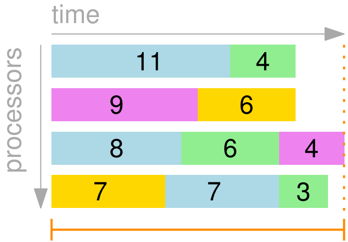


The NP-complete Scheduling Problem $P||C_{\max}$

$m = 4$ **processors**

$n = 10$ **jobs** $j_1, j_2, \ldots, j_n$

n **job durations** $W = \{11, 9, 8, 7, 7, 6, 6, 4, 4, 3\}$



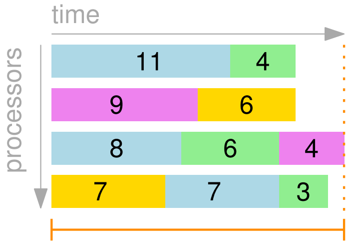
Minimize $C_{max} = 18$

The NP-complete Scheduling Problem $P||C_{\max}$

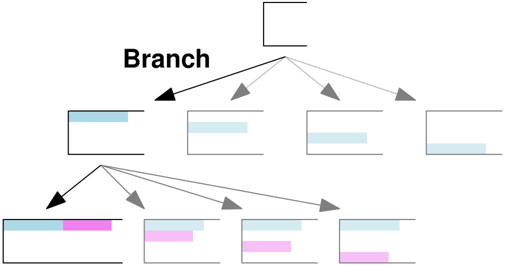
$m = 4$ **processors**

$n = 10$ **jobs** $j_1, j_2, \ldots, j_n$

n **job durations** $W = \{11, 9, 8, 7, 7, 6, 6, 4, 4, 3\}$



Minimize $C_{max} = 18$

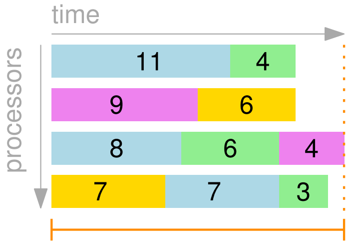


The NP-complete Scheduling Problem $P||C_{\max}$

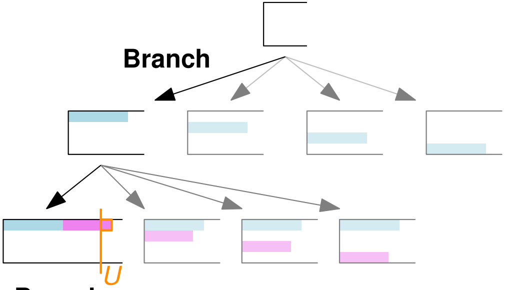
$m = 4$ **processors**

$n = 10$ **jobs** $j_1, j_2, \ldots, j_n$

n **job durations** $W = \{11, 9, 8, 7, 7, 6, 6, 4, 4, 3\}$



Minimize $C_{max} = 18$

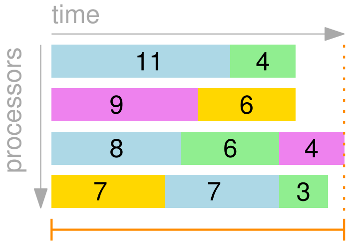


The NP-complete Scheduling Problem $P||C_{\max}$

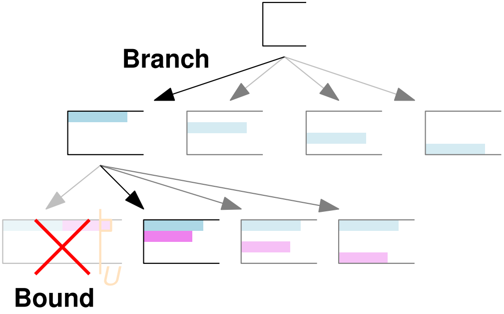
$m = 4$ **processors**

$n = 10$ **jobs** $j_1, j_2, \ldots, j_n$

n **job durations** $W = \{11, 9, 8, 7, 7, 6, 6, 4, 4, 3\}$



Minimize $C_{max} = 18$

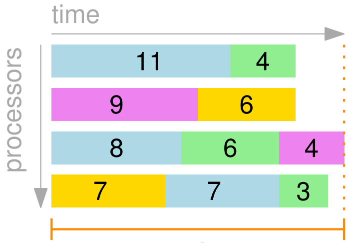


The NP-complete Scheduling Problem $P||C_{\max}$

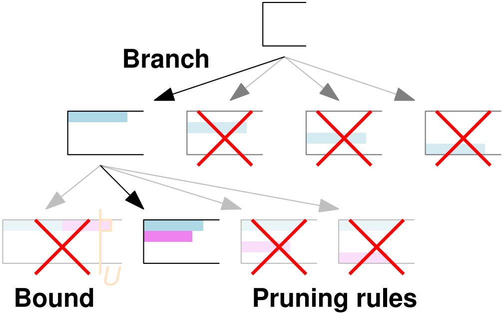
$m = 4$ **processors**

$n = 10$ **jobs** $j_1, j_2, \ldots, j_n$

n **job durations** $W = \{11, 9, 8, 7, 7, 6, 6, 4, 4, 3\}$



Minimize $C_{max} = 18$

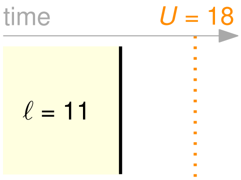


= *Dominance criteria*

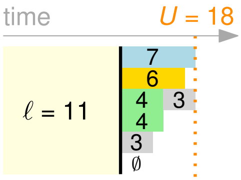
Contributions

The function ϕ

$\phi(j_i, \ell)$ denotes all subsets of $j_i, \ldots, j_n$ which still fit onto a processor with load ℓ.



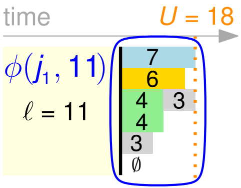
Contributions



The function ϕ

$\phi(j_i, \ell)$ denotes all subsets of $j_i, \ldots, j_n$ which still fit onto a processor with load ℓ.

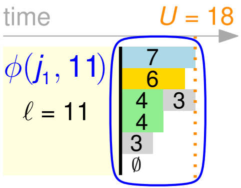
Contributions



The function ϕ

$\phi(j_i, \ell)$ denotes all subsets of $j_i, \ldots, j_n$ which still fit onto a processor with load ℓ.

Contributions



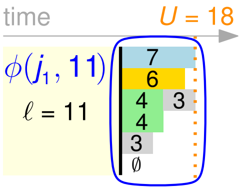
The function ϕ

$\phi(j_i, \ell)$ denotes all subsets of $j_i, \ldots, j_n$ which still fit onto a processor with load ℓ.

Pruning Rule 4

Consider two processors with loads ℓ, ℓ'.
If $\phi(j_i, \ell) = \phi(j_i, \ell')$, then only one processor needs to be considered.

Contributions



The function ϕ

$\phi(j_i, \ell)$ denotes all subsets of $j_i, \ldots, j_n$ which still fit onto a processor with load ℓ.

Pruning Rule 4

Consider two processors with loads ℓ, ℓ'.
If $\phi(j_i, \ell) = \phi(j_i, \ell')$, then only one processor needs to be considered.

Pruning Rule 5 (The Fill-Up-Rule, *FUR*)

Consider processor x with load ℓ and the largest job j_i which still fits onto x.
If the duration of j_i dominates the duration of any job set in $\phi(j_i, \ell)$, we can always just assign j_i to x.

Results

Implementation

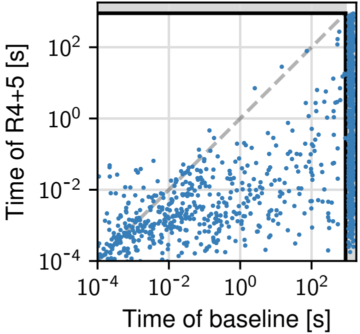
Branch-and-bound algorithm maintaining a $\mathcal{O}(U \cdot n)$ space lookup table for checking ϕ set equivalencies

Results

Karlsruhe Institute of Technology

Implementation

Branch-and-bound algorithm maintaining a $\mathcal{O}(U \cdot n)$ space lookup table for checking ϕ set equivalencies



Evaluation

- 3500 instances by Mrad & Souayah, $n/m \in [2, 3]$
- Baseline $\rightarrow$ R4: +13% solved, -44% explored nodes
- R4 $\rightarrow$ R4+5: +99% solved, -97% explored nodes
- Outperforms state-of-the-art ILP approach for large makespans