Coupled task scheduling

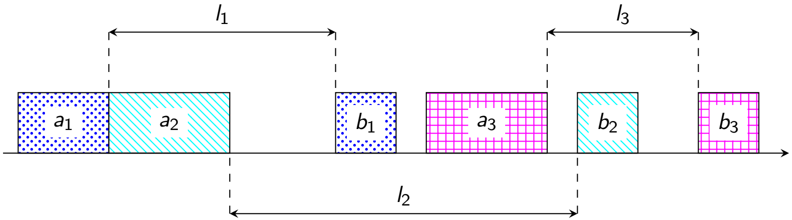
Anna Markó

Supervisor: Péter Györgyi

January 8, 2024

Introduction

- Input: $(a_j, l_j, b_j)^n$
- Output: $(s_1, \ldots, s_n)$



Objective function

- Sum of completion times: $\sum_{j=1}^{n} C_j$

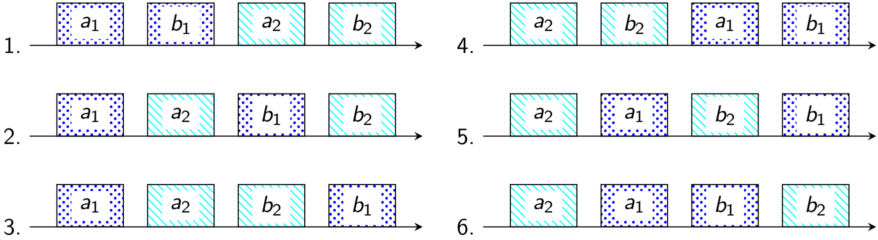
3-approx	2-approx	1.5-approx
a, l_j, b	$a, l_j, b, b \leq a$	$1, l_j, 1$
a_j, L, b_j	a_j, p_j, p_j	p_j, L, p_j
	p_j, p_j, b_j	p_j, p_j, p_j

Goals

- Create an IP solver for $(a_j, L, b_j)^n$
- Investigate how effective a 3-approximation algorithm is in practice
- Search for instances of poor results

IP solver

- Sherali and Smith's model
- Variables: $\forall i, j, k \;\; 1 \leq i, j \leq n, \, 1 \leq k \leq 6 : \; y_{i,j,k} \in \{0, 1\}$



Approximation algorithm

Algorithm 1:

Input : $(a_j, b_j) \quad j = 1 \ldots n,\ L$

Output: $(s_j)_{j=1}^{n} \quad j = 1 \ldots n$

Sort the jobs in non-decreasing order of $a_j + b_j$;

$s_1 := 0$;

for $j = 2 \ldots n$ **do**

  if *a_j can be scheduled immediately after a_{j-1} without overlapping into the processing time of other tasks* **then**

    Schedule it this way;

    $s_j := s_{j-1} + a_{j-1}$

  else

    if *b_j can be scheduled immediately after b_{j-1} without overlapping into the processing time of other tasks* **then**

      Schedule it this way;

      $s_j := s_{j-1} + a_{j-1} + b_{j-1} - a_j$

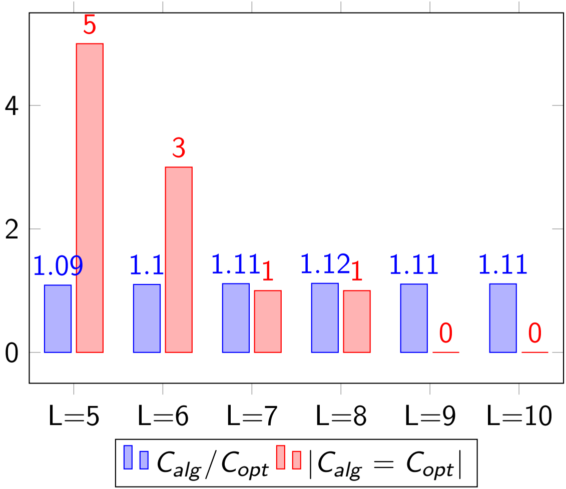
    else

      Start a_j immediately after b_{j-1};

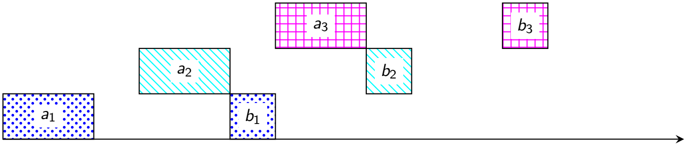
      $s_j := s_{j-1} + a_{j-1} + b_{j-1} + L$;

Results

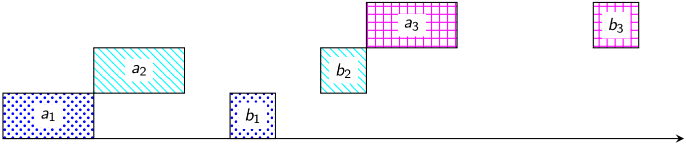
$6 * 50$ input, $n = 7$, $a_j, b_j \in \{1, \ldots, 10\}$



The optimal schedule:



The schedule obtained by the algorithm:



$$\frac{C_{alg}}{C_{opt}} = \frac{2(n+1)(2n+1)}{3n(n+3)} \approx \frac{4}{3}.$$

Further plans

- Develop my IP solver
- Make further progress in identifying instances where the algorithm significantly underperforms

References

[1] D. Fischer, P. Györgyi: *Approximation algorithms for coupled task scheduling minimizing the sum of completion times* Ann. Oper. Res. 328(2): 1387-1408 (2023)

[2] Hanif D. Sheralia, J. Cole Smith. *Interleaving two-phased jobs on a single machine.* Discrete Optimization 2 348 – 361 (2005)