Coupled task scheduling

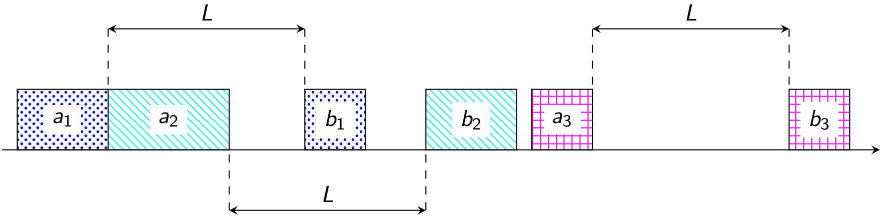
Anna Markó

Supervisor: Péter Györgyi

May 29, 2024

Introduction

- Input: $(a_j, L, b_j)^n$
- Output: $(s_1, \ldots, s_n)$
- Objective function $\sum_{j=1}^{n} C_j$



Goals

- Handle inputs consisting of as many jobs as possible
- Improve the 3-approximation algorithm

Approximation algorithm

Algorithm 1:

```
Input  : (a_j, b_j)   j = 1...n, L
Output: (s_j)_{j=1}^n   j = 1...n
Sort the jobs in non-decreasing order of a_j + b_j;
s_1 := 0;
for j = 2...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;
```

Improvement by rearranging blocks

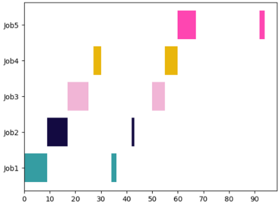
- Blocks are formed by jobs whose starting times immediately follow each other

$$\sum_{i=1}^{k} C_j = \sum_{i=1}^{k} \left(B_i^w + B_i^n \sum_{j=1}^{i-1} B_j^l \right).$$

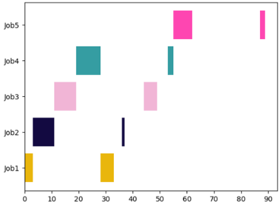
Lemma

Let $B = (B_1, \ldots, B_m)$ denote a set of blocks. Schedule them in non-decreasing order of $\frac{B_j^n}{B_j^l}$. This schedule is optimal.

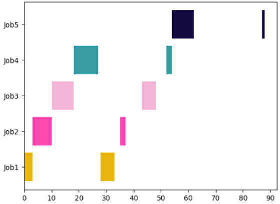
Local search



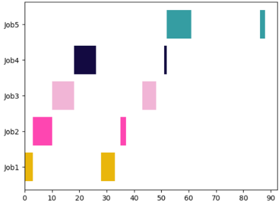
(a) Initial schedule



(b) 1. step



(c) 2. step



(d) 3. step

Figure 1: Local search with interchange operator

Local search

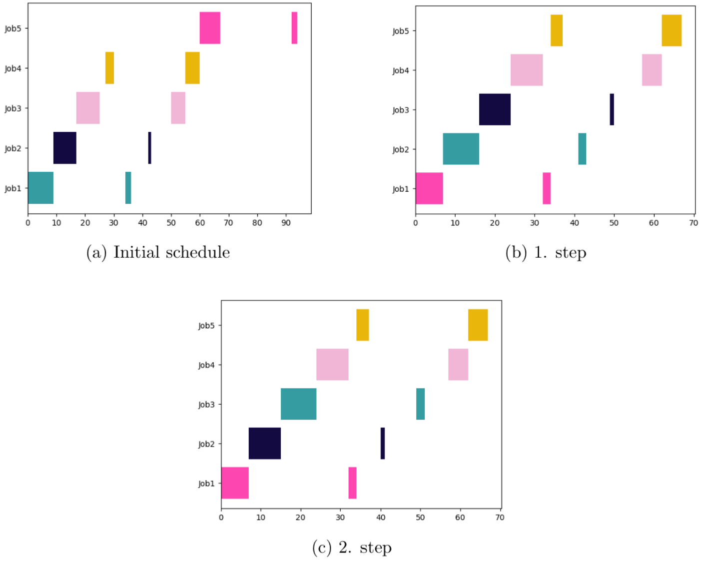
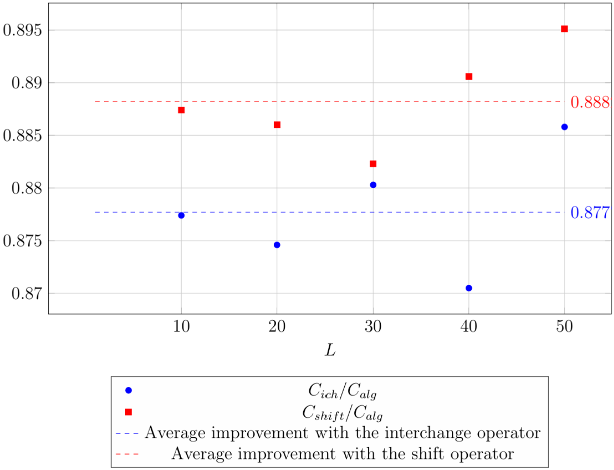
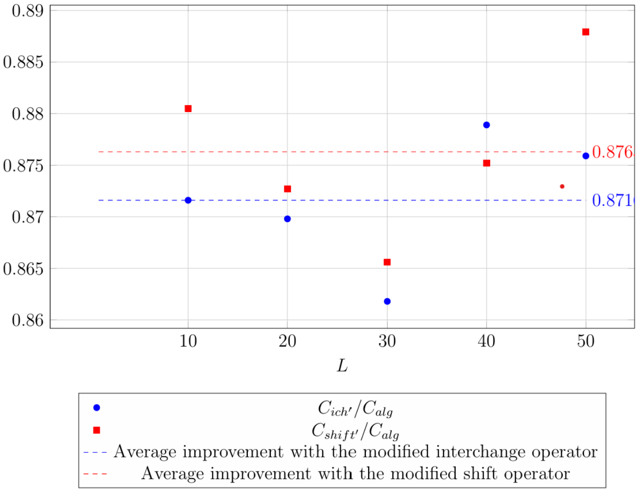


Figure 2: Local search with shift operator

Improvement using local search with a single operator at a time



Improvement using local search with a single modified operator at a time



Local search with alternating use of operators

Algorithm 2: Local search with alternating use of operators

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

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

```
while Schedule can be improved do
    while The interchange operator improves the schedule do
        Apply the interchange operator;
    while The shift operator improves the schedule do
        Apply the shift operator;
return best_solution
```

L	50	40	30	20	10
C	0.865	0.868	0.861	0.867	0.850

Table:

Local search with tabu search

Algorithm 3: Local search with tabu search

Input : $(a_j, b_j) \quad j = 1 \ldots n$, L, tabu_size, max_iter
Output: $(s_j)_{j=1}^{n} \quad j = 1 \ldots n$

```
Initialize current solution, task list, and order;
Initialize tabu list as an empty queue with a maximum size of tabu_size;
iter_count ← 0;
while iter_count < max_iter do
    iter_count ← iter_count + 1;
    Initialize best solution, task list, and order to current solution, task list,
     and order;
    for operator ∈ {interchange, shift} do
        Apply operator to obtain new solution, task list, and order;
        if improvement found and move not in tabu list then
            Update best solution, task list, and order;
    if improvement found then
        Update current solution, task list, and order with best move;
        Add best move to tabu list;
        if tabu list exceeds tabu_size then
            Remove the oldest move from tabu list;
    else
        Apply a random operator to get a new solution, task list, and order;
        Add the new move to tabu list;
        if tabu list exceeds tabu_size then
            Remove the oldest move from tabu list;
return best_solution
```

Summary

- Local search with alternating use of operators proved to be the most effective approach
- My assumption is that further improvement would require a new evaluation algorithm