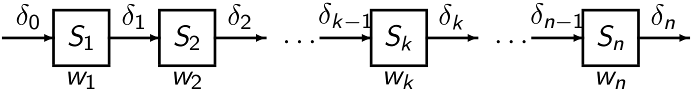
Scheduling algorithms for workflow optimization

Loïc Magnan

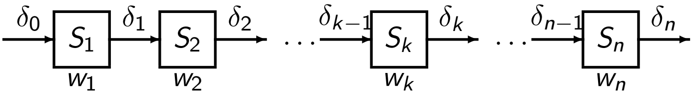
in collaboration with Kunal Agrawal, Anne Benoit & Yves Robert

École Normale Supérieure de Lyon

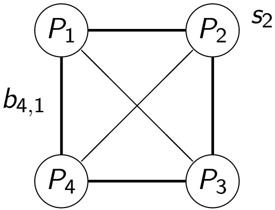
We represent a program by a linear graph:



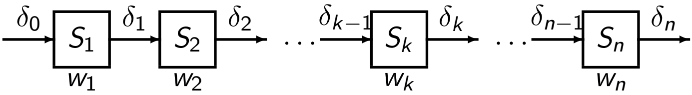
We represent a program by a linear graph:



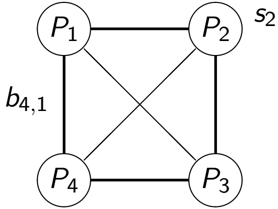
The platform consists of p processors:



We represent a program by a linear graph:



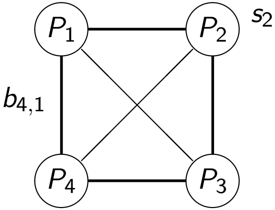
The platform consists of p processors:



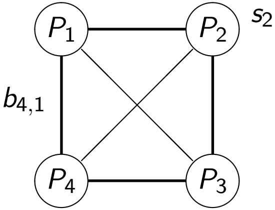
Goal:

- minimize period
- minimize latency

The platform consists of p processors:



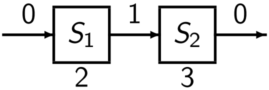
The platform consists of p processors:

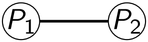


There are two platform models:

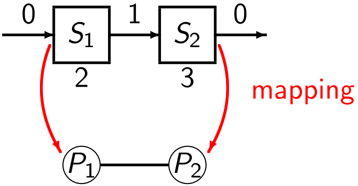
- one-port model (one processor can either compute or receive or send)
- multi-port model (one processor can compute, receive and send at the same time)

Let's take an example in the one-port model:

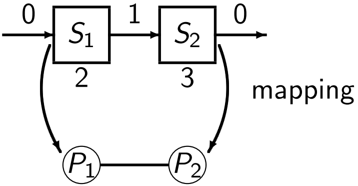




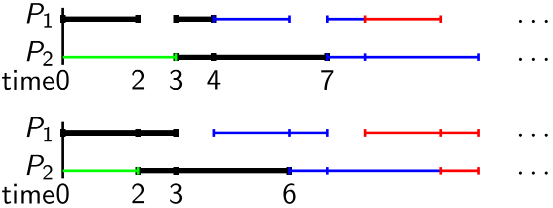
Let's take an example in the one-port model:



Let's take an example in the one-port model:



Two schedules of period 4:



Goal: minimize period and/or latency

More precisely:

- When the mapping is not given: most problems are NP-hard (related work)

Goal: minimize period and/or latency

More precisely:

- When the mapping is not given: most problems are NP-hard (related work)

- When the mapping is given: we search for a schedule which
  - minimizes period
  - minimizes latency
  - respects a period and a latency (bi-criteria)

  for the
  - one-port model
  - multi-port model

We will prove that for a given mapping, finding a schedule that minimizes the period is NP-hard.

We will prove that for a given mapping, finding a schedule that minimizes the period is NP-hard.

The 2-PARTITION problem

Given a set S of n integers $S = \{a_1, a_2, \ldots, a_n\}$ such that

$$\sum_{a_i \in S} a_i = P$$

Decide if it is possible to partition S into two subsets S_1 and S_2 such that

$$\sum_{a_i \in S_1} a_i = \sum_{a_i \in S_2} a_i = P/2$$

is NP-hard in the weak sense.

We want to prove that for some linear graphs and mappings, it's equivalent to say:

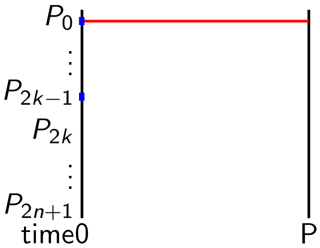
- There exists a schedule of period P.
- We can 2-PARTITION the set $\{a_1, a_2, \ldots, a_n\}$ into two subsets of sum $P/2$.

We want to prove that for some linear graphs and mappings, it's equivalent to say:

- There exists a schedule of period P.
- We can 2-PARTITION the set $\{a_1, a_2, \ldots, a_n\}$ into two subsets of sum $P/2$.

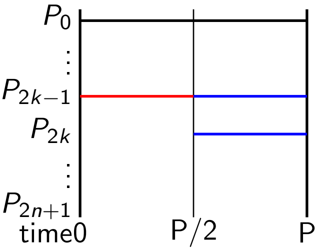
We explain this by constructing a schedule of period P.

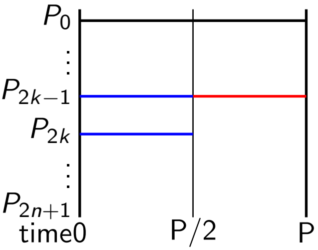
We first add a computation of size P on processor P_0 and communications of size 0 between P_0 and P_{2k-1}



Then we add on P_{2k-1} a computation of size $P/2$ and a communication of size $P/2$ with P_{2k}.

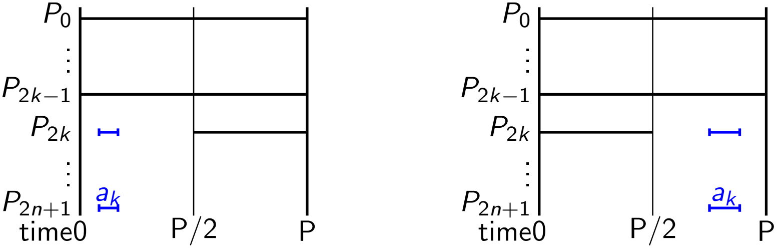
Then we add on P_{2k-1} a computation of size $P/2$ and a communication of size $P/2$ with P_{2k}.



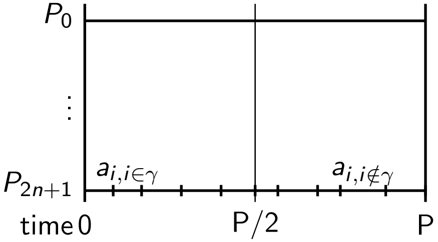


Finally we add a communication of size a_k between P_{2k} and P_{2n+1}.

Finally we add a communication of size a_k between P_{2k} and P_{2n+1}.



Repeating the previous steps n times leads to:



and is equivalent to the 2-PARTITION problem:

$$\sum_{i\in\gamma} a_i = \sum_{i\notin\gamma} a_i = P/2$$

Assuming that $P \neq NP$, there is no way to compute a schedule with optimal period in the one-port model in polynomial time.

Assuming that $P \neq NP$, there is no way to compute a schedule with optimal period in the one-port model in polynomial time.

Preliminary remark: in the one-port model for a given mapping,

- a communication between stages S_k and S_{k+1} mapped on P_u and P_v lasts $\frac{\delta_k}{\min\{b_{u,v}, B^i_v, B^o_u\}}$ time-units.
- a computation on stage S_k mapped on P_u lasts $\frac{w_k}{s_u}$ time-units.

Assuming that $P \neq NP$, there is no way to compute a schedule with optimal period in the one-port model in polynomial time.

Preliminary remark: in the one-port model for a given mapping,

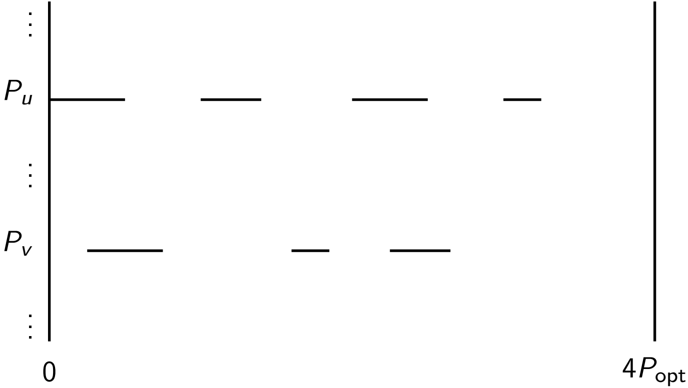
- a communication between stages S_k and S_{k+1} mapped on P_u and P_v lasts $\frac{\delta_k}{\min\{b_{u,v}, B_v^i, B_u^o\}}$ time-units.
- a computation on stage S_k mapped on P_u lasts $\frac{w_k}{s_u}$ time-units.

Longest First algorithm:

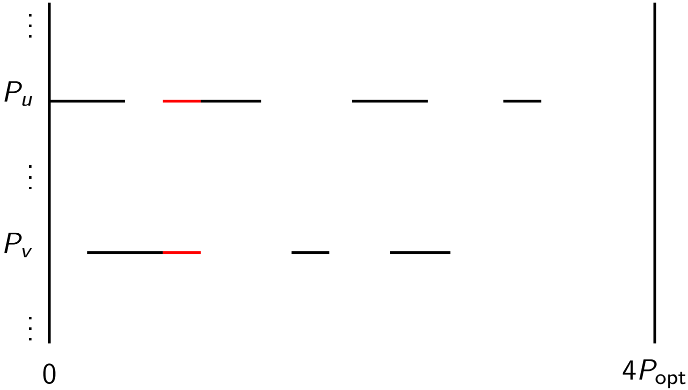
for all tasks, communications and computations, from the longest to the shortest **do**
  add the task as soon as possible in the schedule
end for

The Longest First algorithm constructs a schedule of period P, with $P \leq 4P_{opt}$.

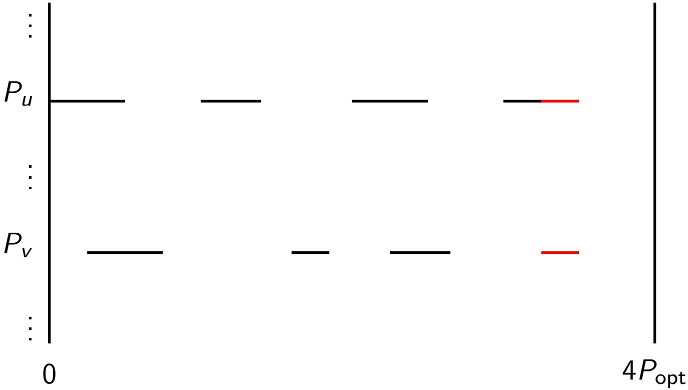
By induction: let's suppose that the result is true for the first k longest tasks.



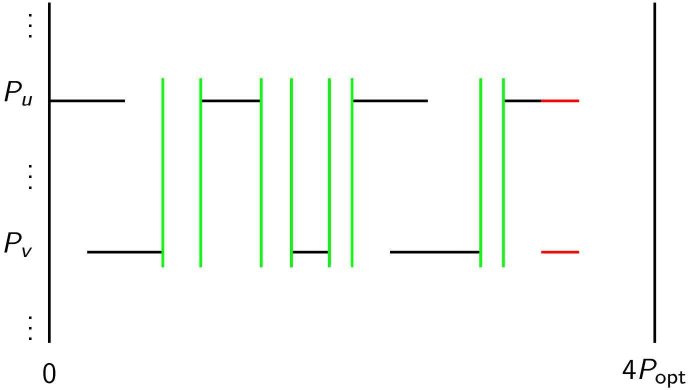
The Longest First algorithm adds a new communication between P_u and P_v:



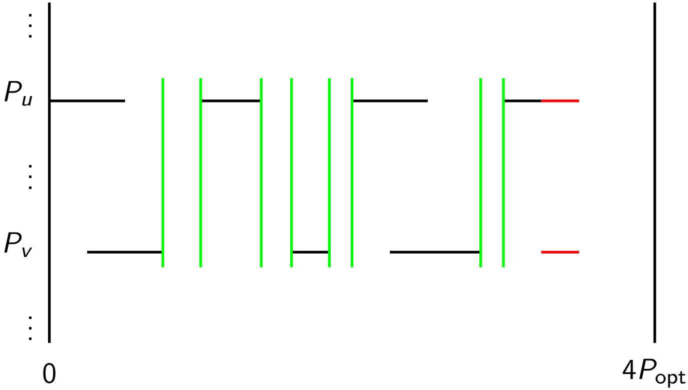
The Longest First algorithm adds a new communication between P_u and P_v:



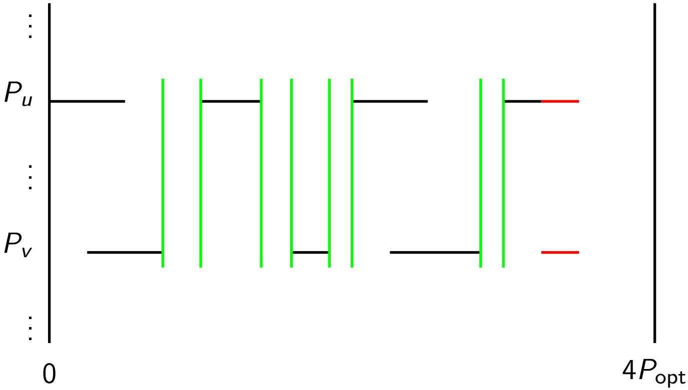
Common gaps between P_u and P_v are smaller than the last communication:



The sum of common gaps sizes is smaller than $2P_{\text{opt}}$.



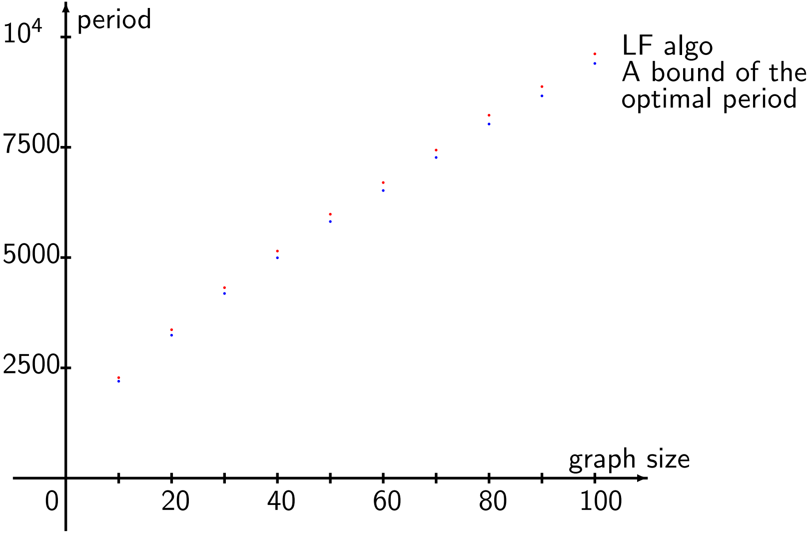
The sum of common gaps sizes is smaller than $2P_{\text{opt}}$.
The size of the schedule minus common gaps is smaller than $2P_{\text{opt}}$.



The Longest First algorithm is a 4-APPROXIMATION for the period.

The Longest First algorithm is a 4-APPROXIMATION for the period.

This algorithm is not a k-APPROXIMATION for any constant $k < 4$.


period
10^4
7500
5000
2500
0
20
40
60
80
100
graph size
LF algo
A bound of the optimal period

Some results:

One-port model

- Latency is easy
- Period is NP-hard (proved)
- Bi-criteria is NP-hard

Multi-port model

- Latency is easy
- Period is polynomial
- Bi-criteria is conjectured NP-hard

Thank you for your attention

Questions ?