Scheduling

1 Scheduling without communication costs

1.1 Preliminaries

Definition 1 (ρ-approximation)**.** Let $\mathcal{P}$ be an optimisation problem with integer objective function $f_{\mathcal{P}}$. Writing $OPT(I)$ for an optimal solution of the problem $\mathcal{P}$ on instance I, we say a polynomial algorithm A is a ρ-approximation for the problem P if and only if $\forall I : f_{\mathcal{P}}(A(I)) \leqslant \rho f_{\mathcal{P}}(OPT(I))$.

Theorem 1 (Impossibility theorem)**.** Let $\mathcal{P}$ be an optimisation problem with integer objective function $f_{\mathcal{P}}$ and c be a nonnegative integer. If the decision problem associated to $\mathcal{P}$ and value c (namely, "does there exist x such that $f_{\mathcal{P}}(x) \leqslant c$?") is NP-complete, then, the existence of a ρ-approximation of $\mathcal{P}$ for any $\rho < (c+1)/c$ implies P=NP.

▷ **Question 1** *Prove the theorem.*

Let us recall two classical NP-complete problems which we are going to use in the tutorial :

Definition 2 (2-Partition)**.** Given a set $\mathcal{I}$ of n integers $a_1, \dots, a_n$, find a partition of $\mathcal{I} = \mathcal{I}_1 \sqcup \mathcal{I}_2$ such that $\sum_{i \in \mathcal{I}_1} a_i = \sum_{i \in \mathcal{I}_2} a_i$.

Definition 3 (Clique)**.** Given a graph $G = (V, E)$ and an integer k, find a $C \subseteq V$ of size k such that for every $u, v \in C$, $(u, v) \in E$.

1.2 Independent tasks of various durations

If tasks are identical and independents, scheduling can obviously be done in polynomial time. However, if durations of the tasks are allowed to be different, the problem becomes NP-hard. Yet there exists a 4/3-approximation for the scheduling problem, improving on the general result for generic list algorithms (which are always 2-approximations).

Suppose p identical machines and n independent tasks $(T_i)_{1 \leqslant i \leqslant n}$. We seek a schedule σ mapping to each task T_i a machine $\mu(T_i)$ and a starting time $\tau(T_i)$, knowing that T_i takes time $w(T_i)$ to be executed. Ideally, this schedule should minimize $D(\sigma) = \max_{1 \leqslant i \leqslant n}(\tau(T_i) + w(T_i))$.

▷ **Question 2** *Assuming $D_{opt} < 3w(T_i)$ for every i, show that $n \leqslant 2p$ and give a polynomial-time alogorithm to compute an optimal schedule.*

▷ **Question 3** *Let us consider the following list algorithm : whenever a machine is free, we assign it the longest task available. Call σ the induced schedule ; check the following bound*

$$D(\sigma) \leqslant D_{opt} + \left(\frac{p-1}{p}\right) d \, ,$$

where d designates the duration of a(ny) task ending at instant $D(\sigma)$. Then, using the previous question, deduce that :

$$D_{opt} \leqslant D(\sigma) \leqslant \left(\frac{4}{3} - \frac{1}{3p}\right) D_{opt} \, .$$

1.3 Identical tasks with dependencies

Now we want to schedule n tasks $(T_i)_{1 \leqslant i \leqslant n}$ requiring one step of execution while respecting dependancy constraints given by an order $\prec$ with p identical processors.

▷ **Question 4** *Show that deciding the existence of a schedule with makespan 3 is an NP-complete problem (Hint : use clique).*

▷ **Question 5** *What can you deduce on the existence of good approximation algorithms for this problem ?*

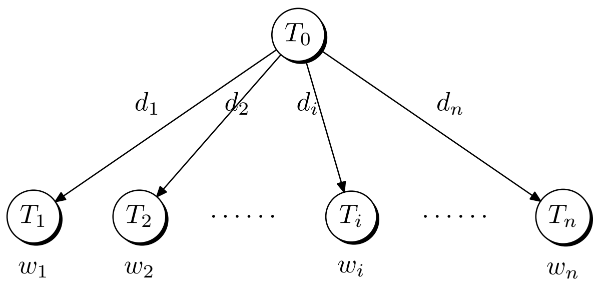

FIGURE 1 – FORK graph with n children

2 Scheduling with communications

2.1 Scheduling of a FORK graph (with communications)

Definition 4 (FORK with n children)**.** A FORK graph with n children is task graph possessing $n+1$ vertices labelled by $T_0, T_1, \cdots, T_n$, as depicted in figure **??**. It has edges between the node T_0 and each of its children T_i, $1 \leqslant i \leqslant n$. Every node has a weight w_i representing the execution time of the task T_i. Each edge (T_0, T_i) also has a weight d_i corresponding to communication costs. Communication costs are incurred only if T_0 and T_i are not run on the same processor.

We first assume that we have infinitely many processors which are multi-port (i.e., can send multiple messages at once). Let us define the following optimization problem :

Definition 5 (FORK-SCHED-∞(G))**.** Given a FORK graph G with n children and infinitely many processors, what is the scheduling σ minimizing the running time ?

▷ **Question 6** *Give a polynomial-time algorithm to solve FORK-SCHED-∞.*

We tackle the same problem with a bounded number of processors :

Definition 6 (FORK-SCHED-BOUNDED(G,p))**.** Given a FORK graph G with n children and p processors, what is the scheduling σ minimizing the running time ?

▷ **Question 7** *Show that the associated decision problem is NP-complete.*

We come back to the problem with infinitely many identical processors, but we no longer suppose them to be multi-port : a processor can only communicate with a single peer at a time.

Definition 7 (FORK-SCHED-1-PORT-∞(G))**.** Given a FORK graph G with n children and infinitely many 1-port processors, what is the scheduling σ minimizing the running time ?

▷ **Question 8** *Show that the associated decision problem is NP-complete (Hint : one can use 2-Partition-Eq, which is a variant of 2-Partition where both subsets are required to be of the same size).*