TD 4: Task graph scheduling

1 Scheduling without communication costs

1.1 Prerequisites of complexity

Definition 1 (Approximation)**.** Let P be an optimisation problem with an integer-valued objective function f. We write $OPT(I)$ for the optimal solution for an instance I of the problem P. We say a polynomial algorithm A is a ρ-approximation of P if for all I we have $f(A(I)) \leq \rho \cdot f(OPT(I))$ (usually $\rho \geq 1$).

Theorem 1 (Impossibility)**.** *Let P be an optimisation problem of integer-valued objective function f. Let c be a positive integer. Suppose the decision problem corresponding to P with the value c is NP-complete. Then for all $\rho < (c+1)/c$ there is no ρ-approximation of P unless $P = NP$.*

Question 1. Show the impossibility theorem.

We recall three common NP-complete problems, that we can use to prove hardness of our scheduling problems.

Definition 2 (2-partition)**.** Given a list of integers $a_1, \ldots, a_n$, find a 2-partition, i.e. I_1 and I_2 such that $I_1 \uplus I_2 = \{1, \ldots, n\}$ and that $\sum_{i \in I_1} a_i = \sum_{i \in I_2} a_i$.

Definition 3 (Clique)**.** Given a graphe (V, E) and an integer k, find a clique of size k, i.e. $C \subseteq V$ such that $|C| = k$ and $\forall u, v \in C\ (u, v) \in E$.

Definition 4 (3-Dimensional-Matching (3DM))**.** Given three sets $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ and $C = \{c_1, \ldots, c_n\}$, and a set $F = \{T_1, \ldots, T_p\}$ of tuples of $A \times B \times C$, find $F' \subseteq F$ such that each element of $A \cup B \cup C$ appears exactly once in a tuple of F'.

1.2 Independent tasks of different durations

If the tasks are identical and independent, the scheduling problem is clearly polynomial. However if the tasks have different durations and independant, the problem is NP-complete (in the weak sense). But there exists a 4/3-approximation, which improves the general result for list algorithms that usually are 2-approximations.

Suppose you have p identical machines and n independent tasks T_i ($1 \leq i \leq n$). We want to defined a scheduling σ assigning to each task T_i a machine $\mu(T_i)$ and a time of start $\tau(T_i)$ given that the duration of T_i is $w(T_i)$. We want to minimise $D(\sigma) = \max_{1 \leq i \leq n}(\tau(T_i) + w(T_i))$.

Question 2. Suppose $D < 3w(T_i)$ for all i. Show that $n \leq 2p$ and give a polynomial algorithm that computes an optimal scheduling.

Question 3. We consider the following list scheduling σ: whenever a machine is available, we assign the task of maximal duration amongst the tasked that are not scheduled yet. Check the inequality:

$$D(\sigma) \leq D_{opt} + \frac{p-1}{p} d$$

where d is the duration of a task finishing at $D(\sigma)$. From this, derive:

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

1.3 Identical tasks with precedente constraints

We want to schedule with p identical machines a set of n tasks T_i of duration 1 and related with a dependency relation $\prec$.

Question 4. Show that deciding if a scheduling of execution time 3 exists is NP-complete. (Hint: use the clique problem.)

Question 5. Using the impossibility theorem, prove that there is no ρ-approximation algorithm for some carefully chosen class of ρ.

2 Scheduling of a FORK graph (with communications)

Definition 5 (FORK with n children)**.** A FORK graph with n children is a task graph with $n+1$ nodes labelled $T_0, \ldots, T_n$. The edges of the graphs are from T_0 to T_i for all i $(1 \leq i \leq n)$. The duration of T_i is $w(T_i)$. Each edge (T_0, T_i) has a weight d_i representing the amount of data transmitted if T_0 and T_i are not on the same process.

We suppose first that we have an infinite amount of identical machines that can send data simultaneously (multi-port).

Definition 6 (FORK-SCHED-$\infty(G)$)**.** Given a FORK graph G with n children and an infinite set of identical machines, what is the duration of the scheduling σ that minimises the execution time?

Question 6. Given a polynomial algorithm solving FORK-SCHED-∞.

Definition 7 (FORK-SCHED-BOUNDED-(G, p))**.** Given a FORK graph G with n children and an p identical machines, what is the duration of the scheduling σ that minimises the execution time?

Question 7. Show FORK-SCHED-BOUNDED is NP-complete.

We get back to the case with an infinite number of identical machines, but we suppose now that machines can only communicate with one other machine at a time (1-port).

Definition 8 (FORK-SCHED-1-PORT-$\infty(G)$)**.** Given a FORK graph G with n children and an infinite set of identical 1-port machines, what is the duration of the scheduling σ that minimises the execution time?

Question 8. Show FORK-SCHED-1-PORT-∞ is NP-complete (hint: we can prove that 2-partition-eq is NP-complete. 2-partition-eq is the variant of 2-partition, where the two sets must be of the same size).