Parallel and Distributed Algorithms and Programs
TD n°3 - Scheduling

Hadrien Croubois
hadrien.croubois@ens-lyon.fr

Aurélien Cavelan
aurelien.cavelan@ens-lyon.fr

27/11/2015

All documents are available on my website: `http://hadriencroubois.com/#Teaching`

Part 1

Scheduling without communication cost

1.1

Complexity

Definition 1 (ρ-approximation). Let $\mathcal{P}$ be a combinatorial optimisation problem with an objective function $f_{\mathcal{P}}$ taking integer values. We note $OPT(I)$ an optimal solution to $\mathcal{P}$ of an instance I, and we say that a polynomial algorithm A is a ρ-approximation of $\mathcal{P}$ if and only if $\forall I : f_{\mathcal{P}}(A(I)) \leq \rho f_{\mathcal{P}}(OPT(I))$.

Theorem 1 (Impossibility theorem). Let $\mathcal{P}$ be a combinatorial optimisation problem with an objective function $f_{\mathcal{P}}$ (with positive integer values) and c a positive integer. If the decision problem associated to $\mathcal{P}$ and to the value c is NP-complete, then for all $\rho < \frac{c+1}{c}$ there is no ρ approximation of $\mathcal{P}$ (unless P=NP).

Question 1

a) Prove the impossibility theorem.

Here are three classical NP-complete problems which we can use to demonstrate our scheduling problems difficulty.

Definition 2 (2-partition). Given $\mathcal{I}$ a set of n numbers $a_1, \dots, a_n$, find a partition of $\mathcal{I}$ into two subsets $\mathcal{I}_1$ and $\mathcal{I}_2$ such that

$$\sum_{a_i \in \mathcal{I}_1} a_i = \sum_{a_j \in \mathcal{I}_2} a_j$$

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

Definition 4 (3-Dimensional-Matching — 3DM). Given three sets $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$ and $C = \{c_1, \dots, c_n\}$ as well as a set $F = \{T_1, \dots, T_n\}$ of triplets of $A \times B \times C$, find a subset F' of F such that all elements of $A \cup B \cup C$ appear in exactly one triplet of F'.

1.2

Independent tasks of different lengths

If all tasks are identical and independent, the problem is obviously polynomial. However, when tasks have different lengths, the problem is NP-complete. Still there exists a 4/3-approximation of this problem, which is better than all list algorithms which are 2-approximation.

Let's consider p identical processors and n independent tasks $(T_i)_{1 \leq i \leq n}$. We want to find a scheduling σ that matches each task T_i to a processor $\mu(T_i)$ and a start time $\tau(T_i)$. Considering that task T_i has a duration $w(T_i)$, we want to minimise

$$D(\sigma) = \max_{1 \leq i \leq n} (\tau(T_i) + w(T_i))$$

Question 2

a) Assuming that $D_{opt} < 3w(T_i)$ for all i, show that $n \le 2p$ and give a polynomial algorithm which computes an optimal schedule.

b) We now consider the following algorithm: as soon as a processor is available, we assign the longest remaining task to it. Prove the inequality

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

with d the length of the task finishing at time $D(\sigma)$. From that, prove the following inequality:

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

1.3 Identical tasks with dependencies

We want to schedule n tasks $(T_i)_{1 \le i \le n}$ of length 1, with dependencies constraints $\prec$ on p identical processors.

Question 3

a) Prove that saying whether or not there exists an optimal schedule of size 3 is NP-complete. (You might want to consider the clique problem.)

b) Using the impossibility theorem, find some results about the existence or non-existence of polynomial approximations to this problem.

Part 2 **Fork scheduling with communication**

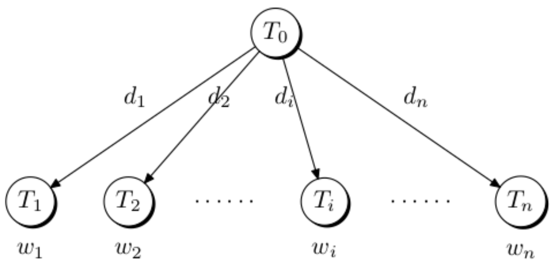

Figure 1: FORK graph with n sons.

Definition 5 (FORK with n sons). A FORK graph with n sons is a graph with $n+1$ vertices $(T_0, \ldots, T_n)$ as illustrated in figure **??**. We have an edge between vertex T_0 and each of its sons $T_i, 1 \le i \le n$. Each vertex has a weight w_i which is the computation time of task T_i. Each edge (T_0, T_i) also has a weight d_i which is the amount of data that has to be exchanged if T_0 and T_1 are running on different processors.

We first assume that we have an infinite number of identical multi-port processors (they can do multiple communication simultaneously).

Definition 6 (FORK-SCHED-$\infty(G)$). Given a FORK graph G with n sons and an infinite number of identical processors, what is the makespan of an optimal schedule σ ?

Question 4

a) Find a polynomial algorithm that solves FORK-SCHED-∞.

Definition 7 (FORK-SCHED-BOUNDED(G, p)). Given a FORK graph G with n sons and p identical processors, what is the makespan of an optimal schedule σ ?

Question 5

a) Show that the decision problem associated to FORK-SCHED-BOUNDED is NP-complete

Finally, we come back to having an infinite number of identical processors but now each processor can only communicate with one other processor at a time (1-port).

Definition 8 (FORK-SCHED-1-PORT-$\infty(G)$). Given a FORK graph G with n sons and an infinite number of identical 1-port processors, what is the makespan of an optimal schedule σ ?

Question 6

a) Show that the decision problem associated to FORK-SCHED-1-PORT is NP-complete (You may want to consider 2-partition-eq which is a variant of 2-partition where both subsets must have the same cardinal.)