Parallel and Distributed Algorithms and Programs
TD n°4 - Scheduling (2)

Hadrien Croubois
hadrien.croubois@ens-lyon.fr

Aurélien Cavelan
aurelien.cavelan@ens-lyon.fr

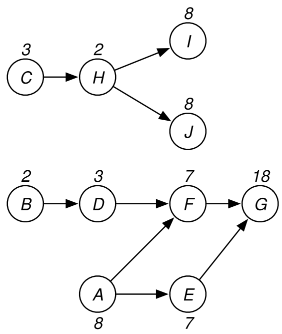
2/12/2015

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

Part 1

Anomalies with list scheduling

Consider the following graph, where each task is represented by a letter and has a number to indicate its weight.



Question 1

a) What is the makespan obtained with a list scheduling based on the critical path, with 2 processors? Is it optimal?

b) Suppose that the weight of each task is now decreased by one unit (A now has a weight of 7, B has a weight of 1, ...). Show that the makespan obtained with a list scheduling based on the critical path is increasing. Show that the makespan obtained with any list heuristic is increasing.

c) Back to the initial weights. Suppose that we now have 3 processors. Show that the makespan obtained with a list scheduling based on the critical path is increasing. Show that the makespan obtained with any list heuristic is increasing.

Part 2

Scheduling on a set of heterogenous processors (without communications)

Consider a set n independant tasks $T_1, ..., T_n$ to be scheduled on p processors. We denote by p_{ij} the time to compute the task T_j on the processor P_i. In the case where all processors are simply going at different speeds (i.e. when $p_{ij} = p_j/s_i$, where s_i represents the speed of the processor i and p_j the amount of work needed for task T_j), the problem is more simple and we have the same result as in the homogeneous case. If not, the current best approximation is a 2-approximation.

Question 2

a) Show that deciding of the existence of a schedule whose execution time is 3 for a set of independant tasks $T_1, ..., T_n$ on processors $P_1, ..., P_p$ is an NP-complete problem. (*You may consider a reduction to 3DM.*)

Definition 1 (3-Dimensional-Matching (3DM)). Given $A = a_1, ..., a_n, B = b_1, ..., b_n, C = c_1, ..., c_n$ be three finite, disjoint sets, and $F = T_1, ..., T_n$ a subset of triples (a, b, c) such that $a \in A, b \in B$, and $c \in C$, find a subset F' of F such as for any two distinct triples $(a1, b1, c1) \in M$ and $(a2, b2, c2) \in F'$, we have $a1 \neq a2$, $b1 \neq b2$, and $c1 \neq c2$ (i.e. any element of $A \cup B \cup C$ appears in exactly one triple of F').

Part 3

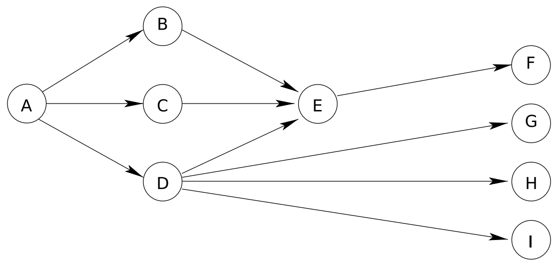
Coffman and Graham scheduling

Consider 2 identical machines, n tasks $(T_i), i = 1 \ldots n$ with same length and $\prec$ a strict partial order on the tasks. We denote by $\sigma = (\mu, \tau)$ a schedule where $\mu(i)$ is the machine executing T_i and $\tau(i)$ is the start date of the execution of T_i.

When $T_i \prec T_j$, we say that T_j is a successor of T_i. In addition, when there is no task T_k such that $T_i \prec T_k \prec T_j$, we say T_j a direct successor of T_i. We define in the same way the notion of direct predecessor.

Question 3

a) Give an optimal schedule for the following graph.



Given a priority function p (assumed injective) on the tasks, we consider a list schedule $\sigma_p = (\mu_p, \tau_p)$ defined as follows: we choose among all free tasks the highest priority one and we execute it on machine 1. Similarly, the second highest priority task is executed on machine 2.

Question 4

a) Which condition p must verify if we want the tasks to be executed in a compatible order with the precedance constraints?

b) Show that the machine 1 is always active, and that if T_i is executed on machine 1, all the tasks T_j executed after (or at the same time) are of lower priority than T_i.

To simplify, we assume that in σ_p, when there is no free task to be executed on machine 2, we execute a "ghost" task without predecessor, and of lower priority than the initial tasks. We define a sequence of pair of tasks (D_k, J_k), executed at the same time, respectively on machine 1 and 2. D_0 is the last task to be executed on machine 1, and similarly J_0 is the last task to be executed on machine 2. J_k (if it exists) is the latest task task to be executed before D_{k-1} on machine 2, which is of lower priority than D_{k-1}. We note F_k the set of tasks that are executed strictly after D_{k+1} and strictly before Dk, plus the task D_k, and we note E_k the tasks of F_k without predecesoors in F_k.

Question 5

a) Give for the last example, the schedule σ_p the tasks D_k and J_k and the set F_k and E_k, assuming that the priority p follows alphabetical order (increasing) on the name of the tasks.

b) Same question with a priority compatible with the precedance constraints (to specify) leading to an optimal result.

c) Show that any task of F_k is of higher priority than D_k and that any task T_i of F_{k-1} is successor of D_k. Deduce that the tasks of $E_{k-1}$are the direct successors of D_k and that any other direct successor of D_k has a lower priority.

We consider $\prec_l$ the lexicographical order and we assume that the priority p verifies (in addition) the following property: $p(T_j) < p(T_i)$ if and only if $l(T_j) \prec_l l(T_j)$ (resp. $l(T_i)$) is the priority list of the direct successor of T_j (resp. T_i) ordered by decreasing order.

Question 6

a) Show that all the tasks of F_k (in articular those without a successor in F_k) are predecessor of all the tasks of F_{k-1}. Conlcude by giving an algorithm that can build such a priority list and an optimal schedule.