Ordonnancement

1 Ordonnancement sans coûts de communications

1.1 Prérequis de complexité

Définition 1 (ρ-approximation)**.** Soit $\mathcal{P}$ un problème d'optimisation combinatoire dont la fonction objectif $f_{\mathcal{P}}$ est à valeurs entières. Si on note $OPT(I)$ une solution optimale du problème $\mathcal{P}$ pour l'instance I, on dira qu'un algorithme polynomial A est une ρ-approximation de $\mathcal{P}$ si et seulement si $\forall I : f_{\mathcal{P}}(A(I)) \leqslant \rho f_{\mathcal{P}}(OPT(I))$.

Théorème 1 (Théorème d'impossibilité)**.** Soit $\mathcal{P}$ un problème d'optimisation combinatoire dont la fonction objectif $f_{\mathcal{P}}$ est à valeurs entières et soit c un entier positif. Si le problème de décision associé à $\mathcal{P}$ et à la valeur c est NP-complet, alors pour tout $\rho < (c+1)/c$ il n'existe pas de ρ-approximation de $\mathcal{P}$ (à moins que P=NP).

▷ **Question 1** *Démontrer le théorème d'impossibilité.*

Nous rappelons trois problèmes NP-complets classiques qui pourront être utilisés pour démontrer la difficulté de nos problèmes d'ordonnancement :

Définition 2 (2-Partition)**.** Étant donné un ensemble $\mathcal{I}$ de n nombres $a_1, \ldots, a_n$, trouver une partition de $\mathcal{I}$ en deux ensembles $\mathcal{I}_1$ et $\mathcal{I}_2$ tels que $\sum_{i \in \mathcal{I}_1} a_i = \sum_{i \in \mathcal{I}_2} a_i$.

Définition 3 (Clique)**.** Étant donné un graphe $G = (V, E)$ et un entier k, trouver un sous-ensemble C de V de taille k tel que pour tout $u, v \in C$, $(u, v) \in E$.

Définition 4 (3-Dimensional-Matching (3DM))**.** Étant donné trois ensembles $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ et $C = \{c_1, \ldots, c_n\}$ ainsi qu'un ensemble $F = \{T_1, \ldots, T_p\}$ de triplets de $A \times B \times C$, trouver un sous-ensemble F' de F tel que tout élément de $A \cup B \cup C$ apparaît dans exactement un triplet de F'.

1.2 Tâches indépendantes de durées différentes

Si les tâches sont identiques et indépendantes, le problème est clairement polynomial. En revanche, si les tâches sont de durées différentes et indépendantes, le problème est NP-complet (au sens faible). Mais il existe une 4/3-approximation pour ce problème, ce qui améliore le résultat général pour les algorithmes de liste qui sont toujours des 2-approximations.

Soient p machines identiques et n tâches $(T_i)_{1 \leqslant i \leqslant n}$ indépendantes. On cherche donc à définir un ordonnancement σ qui attribue à chaque tâche T_i une machine $\mu(T_i)$ et une date de début d'exécution $\tau(T_i)$ sachant que la durée de la tâche T_i est $w(T_i)$. On cherche à minimiser $D(\sigma) = \max_{1 \leqslant i \leqslant n}(\tau(T_i) + w(T_i))$.

▷ **Question 2** *En supposant que $D_{opt} < 3w(T_i)$ pour tout i, montrer que $n \leqslant 2p$ et donner un algorithme polynomial permettant de calculer un ordonnancement de durée minimale.*

▷ **Question 3** *On s'intéresse à l'ordonnancement de liste suivant : dès qu'une machine est libre, on lui affecte la tâche de durée maximale parmi les tâches non encore ordonnancées. Vérifier l'inégalité :*

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

où d est la durée d'une tâche se terminant à l'instant $D(\sigma)$. En déduire l'inégalité :

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

1.3 Tâches identiques avec contraintes de précédence

On cherche à ordonnancer avec p processeurs identiques un ensemble de n tâches $(T_i)_{1 \leqslant i \leqslant n}$ unitaires (de durée 1) et reliées par des contraintes de dépendances $\prec$.

▷ **Question 4** *Montrer que décider de l'existence d'un ordonnancement dont le temps d'exécution est* 3 *est un problème NP-complet. (*Indication : on pourra se ramener au problème de la clique.*)*

▷ **Question 5** *En utilisant le théorème d'impossibilité, donner des résultats d'existence ou d'inexistence d'algorithmes d'approximation pour ce problème d'ordonnancement.*

2 Ordonnancement avec communications

2.1 Ordonnancement d'un graphe FORK (avec communications)

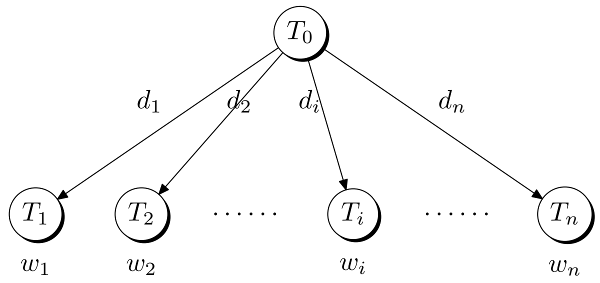

Fig. 1 – Graphe de FORK à n fils.

Définition 5 (FORK à n fils)**.** Un graphe FORK à n fils est un graphe de tâches à $n+1$ nœuds étiquetés par $T_0, T_1, \cdots, T_n$, comme illustré figure 1. Il y a une arête entre le nœud T_0 et chacun de ses fils T_i, $1 \leqslant i \leqslant n$. Chaque nœud possède un poids w_i qui représente le temps de traitement de la tâche T_i. Chaque arête (T_0, T_i) possède aussi un poids correspondant au volume de données échangées d_i si la tâche T_0 et la tâche T_i ne sont pas traitées sur le même processeur.

On suppose d'abord disposer d'une infinité de processeurs identiques et multi-port (qui peuvent initier plusieurs envois simultanés). On définit le problème d'optimisation suivant :

Définition 6 (FORK-SCHED-∞(G))**.** Étant donné un graphe FORK G à n fils et un ensemble infini de processeurs identiques, quelle est la durée de l'ordonnancement σ qui minimise le temps d'exécution ?

▷ **Question 6** *Donner un algorithme polynomial résolvant FORK-SCHED-∞.*

On s'intéresse maintenant au même problème avec un nombre borné de processeurs :

Définition 7 (FORK-SCHED-BOUNDED(G,p))**.** Étant donné un graphe FORK G à n fils et un ensemble de p processeurs identiques, quelle est la durée de l'ordonnancement σ qui minimise le temps d'exécution ?

▷ **Question 7** *Montrer que le problème de décision associé à FORK-SCHED-BOUNDED est NP-complet.*

On revient enfin au problème avec une infinité de processeurs identiques, mais on suppose désormais qu'un processeur ne peut communiquer qu'avec un seul processeur à la fois (modèle 1-port).

Définition 8 (FORK-SCHED-1-PORT-∞(G))**.** Étant donné un graphe FORK G à n fils et un ensemble infini de processeurs 1-port identiques, quelle est la durée de l'ordonnancement σ qui minimise le temps d'exécution ?

▷ **Question 8** *Démontrer que le problème de décision associé à FORK-SCHED-1-PORT-∞ est NP-complet. (*Indication : on pourra se ramener à 2-Partition-Eq, une variante de 2-Partition où les deux partitions doivent avoir le même cardinal.*)*