# Ordonnancement sans communications

**Résumé:** Dans ce TD, nous introduisons la notion d'approximation pour un problème d'optimisation et étudions quelques problèmes d'ordonnancement sans communications. Nous démontrons des résultats de complexité (NP-complétude, inexistence d'heuristiques garanties pour certains facteurs) et exposons une heuristique garantie pour l'un des problèmes.

## 1 Prérequis de complexité

### 1.1 Quelques théorèmes

*Définition* 1 *(ρ-approximation).* Soit $\mathcal{P}$ un problème d'optimisation combinatoire dont la fonction objectif $f_{\mathcal{P}}$ est à valeur entière. Si on note $OPT(I)$ une solution optimale du problème $\mathcal{P}$ pour l'instance $I$, on dira qu'un algorithme polynomal $A$ est une $\rho$-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 valeur entière 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 $\rho$-approximation de $\mathcal{P}$ (à moins que P=NP).

*Définition* 2 *(Passage à l'échelle).* Soit $\mathcal{P}$ un problème d'optimisation combinatoire dont la fonction objectif $f_{\mathcal{P}}$ est à valeur entière. On dira que $\mathcal{P}$ passe bien à l'échelle si pour tout $k$ et pour toute instance $I$, on peut transformer (de façon algorithmique) $I$ en une instance équivalente $I'$ telle que toute solution associée à $I$ correspond à une solution pour $I'$ dont la valeur objectif est $k$ fois plus grande.

*Théorème* 2 *(Théorème de passage à l'échelle).* Soit $\mathcal{P}$ un problème d'optimisation combinatoire dont la fonction objectif $f_{\mathcal{P}}$ est à valeur entière et qui possède la propriété de passage à l'échelle définie précédemment. S'il existe un algorithme polynomial $A$ tel que $\forall I : f_{\mathcal{P}}(A(I)) \leqslant \rho f_{\mathcal{P}}(OPT(I)) + \beta$, alors en appliquant cet algorithme à une instance judicieusement agrandie, on obtient une $\rho$-approximation de $\mathcal{P}$.

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

*Réponse.* Supposons qu'il existe un algorithme polynomial $A$ qui soit une $\rho$-approximation pour $\mathcal{P}$ avec $\rho < (c+1)/c$. Soit $I$ une instance de $\mathcal{P}$. Si $OPT(I) \leqslant I$, alors $f_{\mathcal{P}}(A(I)) \leqslant \rho f_{\mathcal{P}}(OPT(I)) < \frac{c+1}{c} f_{\mathcal{P}}(OPT(I)) \leqslant c$. Ainsi l'algorithme fournit une solution de taille inférieure à $c$ quand l'optimal est de taille inférieure à $c$, et dans le cas contraire fournit nécéssairement une solution dont la taille est supérieure strictement à $c$. Il peut donc être utilisé pour décider s'il existe une solution de taille inférieure ou égale à $c$, ce qui implique que P=NP. □

### 1.2 Quelques problèmes classiques

Dans cette section, nous rappelons un certain nombre de problèmes NP-complets qui pourront être utilisés pour démontrer la difficulté de nos problèmes d'ordonnancement.

*Définition* 3 *(Bin-Packing).* Étant donnés un ensemble de $n$ objets de volume $a_1, \ldots, a_n$ et des boites de taille $b$, trouver le nombre minimal de boites nécessaire pour empaqueter tous les objets.

*Définition* 4 *(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* 5 *(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* 6 *(3-Dimensionnal-Matching).* Étant donnés trois ensembles $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ , $C = \{c_1, \ldots, c_n\}$ et un ensemble $F = \{T_1, \ldots, T_p\}$ d'éléments 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 $F'$.

# 2 Ordonnancement sur un ensemble de processeurs identiques

Dans cette section, on va s'intéresser à des problèmes d'ordonnancement sur un ensemble de machines identiques.

## 2.1 Quelques rappels

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. Enfin, si les tâches sont de durées différentes et qu'il y a des dépendances, le problème est toujours NP-complet (qui peut le plus peut le moins...) et il a été démontré en cours que tout algorithme de liste était une 2-approximation. Nous allons regarder dans quels cas on peut améliorer ce résultat.

## 2.2 Tâches indépendantes de durées différentes

Il existe une 4/3-approximation pour ce problème. C'est un simple algorithme de liste où les tâches sont triées en fonction de leur taille.

Soient $p$ machines identiques et $n$ tâches $(T_i)_{1\leqslant i\leqslant n}$ indépendantes. On cherche donc à définir un ordonnancement $\sigma$ 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.

*Réponse.* Si $D_{opt} < 3w(T_i)$ pour tout $i$, alors chaque machine calcule au plus 2 tâches dans l'ordonnancement optimal et on a donc $n \leqslant 2p$. S'il y a $n = 2p - h$ tâches, un ordonnancement optimal consiste à placer les $h$ tâches les plus longues seules sur les $h$ premières machines et à placer les $2(p-h)$ tâches restantes sur les $p-h$ machines restantes en les groupant deux par deux (la plus longue avec la plus courte, la seconde plus longue avec la seconde plus courte, et ainsi de suite).

Considérons un ordonnancement optimal. On a $2p-h$ tâches à répartir sur $p$ processeurs avec au plus 2 tâches par processeurs. Il y a donc exactement $h$ tâches seules sur un processeur. On peut toujours échanger ces tâches avec les $h$ tâches les plus longues sans augmenter la durée de l'ordonnancement. Considérons maintenant deux processeurs responsables chacun de l'exécution de deux tâches ($\{T_{i_1}, T_{i_2}\}$ et $\{T_{j_1}, T_{j_2}\}$ où $T_{i_1}$ est plus longue que $T_{i_2}$ et $T_{j_1}$ plus longue que $T_{j_2}$). Si $T_{i_1}$ est plus longue que $T_{j_1}$ et que $T_{i_2}$ est plus longue que $T_{j_2}$, alors on peut échanger $T_{i_2}$ et $T_{j_2}$ sans augmenter la durée de l'ordonnancement. Donc si on ordonne les processeurs de façon à ce que les durées des $T_{i_1}$ soient croissantes, alors les durées de $T_{i_2}$ sont décroissantes. □

▷ **Question 3.** On va maintenant s'intéresser 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 suivante :

$$D(\sigma) \leqslant D_{opt} + \left(\frac{p-1}{p}\right) d \quad \text{, où } d \text{ est la durée d'une tâche se terminant à l'instant } D(\sigma))$$

En déduire l'inégalité suivante :

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

*Réponse.* On va supposer que les tâches triées par ordre croissant de durée. Soit $T_j$ la tâche se terminant en dernier. On a $D(\sigma) = \tau(T_j) + w(T_j)$. Aucun des processeurs ne s'arrête donc de travailler avant la date $\tau(T_j)$ sinon on aurait assigné cette tâche à un autre processeur. Tous les processeurs ont donc fini de traiter leurs tâches après la date $\tau(T_j)$, ce qui signifie que $\sum_{i\neq j} w(T_i) \geqslant p\tau(T_j) = p(D(\sigma) - w(T_j))$. On a donc $D(\sigma) \leqslant \frac{1}{p}\left(\sum_{i\neq j} w(T_i)\right) + w(T_j)$, ce qui implique que $D(\sigma) \leqslant \frac{1}{p}\sum_i w(T_i) + \left(\frac{p-1}{p}\right) w(T_j) \leqslant D_{opt} + \left(\frac{p-1}{p}\right) w(T_j)$.

Raisonnons maintenant par l'absurde en supposant que $D(\sigma) > \left(\frac{4}{3} - \frac{1}{3p}\right) D_{opt}$. En combinant cette inégalité avec la précédente, on obtient $D_{opt} + \left(\frac{p-1}{p}\right) w(T_j) > \left(\frac{4}{3} - \frac{1}{3p}\right) D_{opt}$, soit $D_{opt} < 3w(T_j)$.
Si $T_j$ est la tâche la plus courte (c'est à dire si $j = n$) alors en utilisant le résultat de la question précédente on arrive à une absurdité. En effet l'algorithme de la question précédente n'est autre que l'algorithme de liste proposé ici et fournit donc un ordonnancement optimal, ce qui contredit l'inégalité $D(\sigma) > \left(\frac{4}{3} - \frac{1}{3p}\right) D_{opt}$.
Si ce n'est pas le cas on peut faire le même raisonnement en ne considérant que $T_1, \ldots, T_j$. En effet, l'ordonnancement $\sigma'$ défini pour ce sous-ensemble par la procédure de l'énoncé n'est autre que la restriction de $\sigma$ à ce sous-ensemble. Il donc la même durée et le $D_{opt}$ correspondant est plus petit ou égal. L'inégalité n'est donc pas vérifiée non plus pour cet ensemble de tâches. En remarquant enfin que l'ordonnancement fourni par l'algorithme de la question précédente n'est autre que $\sigma'$, on en déduit que $\sigma'$ est donc est optimal pour $T_1, \ldots, T_j$, ce qui est absurde pour la même raison que précédemment. □

## 2.3 Tâches identiques avec contraintes de précédences

On considère une plateforme constituées de $p$ processeurs identiques. On cherche à ordonnancer un ensemble $n$ tâches unitaires $(T_i)_{1 \leqslant i \leqslant n}$ sur ces $p$ processeurs tout en respectant des contraintes de dépendances $\prec$.

▷ **Question 4.** Montrer que décider de l'existence d'un ordonnancement dont le temps d'exécution est 3 pour des tâches unitaires (de durée 1) en respectant des contraintes de dépendances $\prec$ est un problème NP-complet. (*Indication : on pourra se ramener au problème de décision qui consiste à déterminer s'il existe une clique de taille $k$ dans un graphe*)

*Réponse.* Soit $G$ un graphe et $k$ un entier naturel. On doit définir une instance $I_{(G,k)}$ pour notre problème d'ordonnancement tel qu'il existe une clique de taille $k$ dans $G$ si et seulement s'il existe un ordonnancement de $I_{(G,k)}$ de durée 3.

Définissons les quantités suivantes : $\bar{k} = |V| - k$, $l = \frac{1}{2}k(k-1)$ et $\bar{l} = |E| - l$. Notre instance sera composée de $p = \max(k, \bar{k} + l, \bar{l}) + 1$ processeurs et de $n = 3p$ tâches. À tout sommet $v$ de $G$ on associe une tâche $J_v$ et à toute arête $e$ de $V$, on associe une tâche $K_e$. Les relations entre ces tâches sont que si deux sommets $u$ et $v$ sont reliés par une arête $e$ alors $J_u \prec K_e$ et $J_v \prec K_e$. Nous aurons également besoin de tâches factices $(X_x)_{1 \leqslant x \leqslant p-k}$, $(Y_y)_{1 \leqslant y \leqslant p-l-\bar{k}}$ et $(Z_z)_{1 \leqslant z \leqslant p-\bar{l}}$.

Montrons qu'il existe une clique de taille $k$ dans $G$ si et seulement s'il existe un ordonnancement de $I_{(G,k)}$ de durée 3.

[⇒] Supposons qu'il existe une clique de taille $k$ dans $G$. Alors il est possible de positionner les $k$ tâches correspondants aux sommets de la clique ainsi que l'intégralité des $(X_x)_{1 \leqslant x \leqslant p-k}$ sur les $p$ processeurs au temps $t = [0, 1]$. On peut ensuite placer au temps $t = [1, 2]$ les $l$ tâches correspondant aux arêtes de la clique, les $\bar{k}$ tâches correspondants aux sommets restants ainsi que l'intégralité des $(Y_y)_{1 \leqslant y \leqslant p-l-\bar{k}}$ tout en respectant les contraintes de dépendances. Enfin, on peut placer au temps $t = [2, 3]$ toutes les autres tâches, c'est à dire les $\bar{l}$ tâches correspondant aux arêtes n'appartenant pas à la clique et l'intégralité des $(Z_z)_{1 \leqslant z \leqslant p-\bar{l}}$.

[⇐] Supposons qu'il existe un ordonnancement de durée 3. Alors, en raison des dépendances entre les tâches de type $X$, $Y$, et $Z$, l'intégralité des $(X_x)_{1 \leqslant x \leqslant p-k}$ doit être ordonnancée au temps $t = [0, 1]$, les tâches $(Y_y)_{1 \leqslant y \leqslant p-l-\bar{k}}$ au temps $t = [1, 2]$ et les tâches $(Z_z)_{1 \leqslant z \leqslant p-\bar{l}}$ au temps $t = [2, 3]$. Étant donné qu'il y a $p$ processeurs et $3p$ tâches, $k$ tâches correspondant à des sommets doivent être ordonnancées au temps $t = [0, 1]$. Les tâches $Y$ occupant $p - l - \bar{k}$ processeurs, on doit trouver $\bar{k} + l$ tâches parmi les tâches de sommets et d'arêtes ordonnançables au temps $t = [1, 2]$. Il y a donc au moins $l$ tâches associées à des arêtes ordonnancées au temps $t = [1, 2]$, ce qui signifie qu'il y a une clique de taille $k$ dans le graphe $G$.

□

▷ **Question 5.** En utilisant les théorèmes de la section 1, donner des résultats d'existence ou d'inexistence d'algorithmes d'approximation pour ce problème d'ordonnancement.

*Réponse.* En utilisant le théorème 1, on montre qu'il n'existe pas de $\rho$-approximation pour $\rho < 4/3$ si $P \neq NP$. Ce type de résultat n'empêche cependant pas l'existence d'un algorithme polynomial fournissant des ordonnancement qui soient, par exemple, proches à une valeur constante de l'optimum[1]. Par contre, si le problème possédait la propriété de passage à l'échelle, on pourrait renforcer notre résultat d'inexistence. Ce n'est malheureusement pas aussi simple qu'il n'y parait. Les tâches sont unitaires et il est impossible de changer leur durée pour augmenter celle de l'ordonnancement. Il faut donc procéder autrement. Soit $I = (\mathcal{T}, \prec)$ une instance de notre problème d'ordonnancement et $k$ un entier. Construisons une nouvelle instance $I^{(k)}$ constituée de $k$ copies $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ de $\mathcal{T}$ ainsi que de $k-1$ tâches $J_1, \ldots, J_{k-1}$. Si $u$ et $v$ sont dans $T_i$ alors $u \prec^{(k)} v$ si et seulement si $u \prec v$. De plus, $\forall i, \forall v \in \mathcal{T}_i : J_{i-1} \prec^{(k)} v \prec^{(k)} J_i$. Ainsi, s'il existe un ordonnancement de durée $d$ pour $I$, on sait qu'il existe un ordonnancement de durée $kd+k-1$ pour $I^{(k)}$. De même d'un ordonnancement de durée $d$ pour $I^{(k)}$, on peut déduire un ordonnancement de durée inférieure à $(d-(k-1))/k$. Même si la relation entre $I$ et $I^{(k)}$ n'est pas parfaitement linéaire, cette relation est suffisante pour utiliser le théorème 2, ce qui nous permet de renforcer notre résultat d'impossibilité. □

# 3 Ordonnancement sur un ensemble de processeurs hétérogène

On dispose de $p$ processeurs et d'un ensemble de $n$ tâches indépendantes $T_1, \ldots, T_n$. On notera $p_{ij}$ le temps de calcul de la tâche $T_j$ sur le processeur $P_i$. Dans le cas où les processeurs vont simplement à des vitesses différentes (c'est à dire quand $p_{ij} = p_j/s_i$ où $s_i$ représente la vitesse du processeur $i$ et $p_j$ la quantité de travail nécessaire au traitement de la tâche $T_j$), le problème est plus simple et on a le même type de résultat que dans le cas homogène. Dans le cas contraire, la meilleure approximation actuellement connue est une 2-approximation.

▷ **Question 6.** Montrer que décider de l'existence d'un ordonnancement dont le temps d'exécution est 3 pour des tâches indépendantes $T_1, \ldots, T_n$ sur les processeurs $P_1, \ldots, P_p$ est un problème NP-complet. (*Indication : on pourra se ramener à 3DM*)

*Réponse.* Soit $I = (A, B, C, F)$ une instance de 3DM. Notre instance $I'$ sera composée de $p = |F|$ processeurs et de $2n + p$ tâches (où $n = |A| = |B| = |C|$). La processeur $P_i$ est associé au triplet $T_i$ pour $i \in [\![1, p]\!]$. Une tâche $J_e$ est associée à chaque élément $e$ de $A \cup B \cup C$. Enfin, nous aurons également besoin de tâches factices $(X_x)_{1 \leqslant x \leqslant p-n}$ pour arriver au bon nombre de tâches. La durée de la tâche $J_e$ est de 3 sur tout processeur $P_i$ tel que $e \in T_i$ et de 1 autrement. Les tâches $X_x$ sont de durée 3 et ce quelque soit le processeur sur lequel elles sont exécutées.

Montrons qu'il existe un ordonnancement de durée 3 pour $I'$ si et seulement s'il existe un couplage 3-dimensionnel de l'instance $I$.

$\boxed{\Rightarrow}$ S'il existe un ordonnancement de durée 3 alors les $p-n$ tâches $X$ étant de durée 3, les $3n$ tâches restantes (celles qui sont associées aux éléments de $A \cup B \cup C$) doivent être exécutées sur des processeurs où leur temps de calcul est 1. Autrement dit, chaque processeur $P_i$ se voit assigner le traitement de trois tâches $J_a, J_b, J_c$ dont le temps d'exécution est 1, ce qui signifie que $T_i = (a, b, c)$. On peut donc en extraire un couplage 3-dimensionnel solution du problème initial.

$\boxed{\Leftarrow}$ S'il existe un couplage 3-dimensionnel $F'$ de l'instance $I$, alors en en associant pour tout $T_i = (a, b, c) \in F'$ les tâches $J_a, J_b, J_c$ au processeur $P_i$ et les autres $p - n$ tâches $X$ restantes aux $p - n$ processeurs restants, on obtient bien un ordonnancement de durée 3.

□

▷ **Question 7.** En utilisant les théorèmes de la section 1, donner des résultats d'existence ou d'inexistence d'algorithmes d'approximation pour ce problème d'ordonnancement.

*Réponse.* En utilisant la question précédente et le théorème 1, on montre qu'il n'existe pas de $\rho$-approximation pour $\rho < 4/3$ si $P \neq NP$. Comme dans la question 5, on peut renforcer ce résultat en remarquant que ce problème possède la propriété de passage à l'échelle. □

[1] On peut montrer, par exemple, qu'il n'existe pas de $\rho$-approximation pour le problème du bin-packing quand $\rho < 3/2$. Il existe cependant un algorithme polynomial $A$ pour ce problème tel que pour tout instance $I$, on ait $A(I) \leqslant 11/9 \cdot OPT(I) + 1$