

Optimisation

Ordonnancement de tâches indépendantes

Louis-Claude Canon

louis-claude.canon@univ-fcomte.fr

Master 2 Informatique – Semestre 9

Plan

Introduction

Classification des problèmes

Analyse de problèmes classiques

Focus sur $P||C_{\max}$

Conclusion

Plan

Introduction

Classification des problèmes

Analyse de problèmes classiques

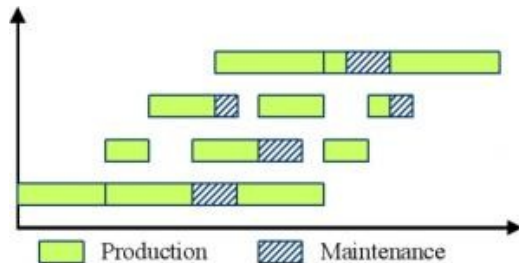
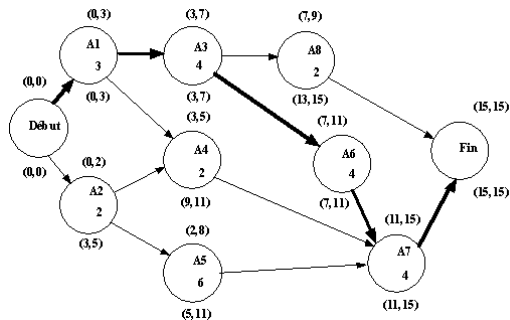
Focus sur $P||C_{\max}$

Conclusion

Définition

- ▶ Un problème d'ordonnancement (*scheduling*) consiste à organiser et optimiser dans le temps la réalisation d'un ensemble de tâches, compte tenu de contraintes.
- ▶ Prise en compte simultanée des contraintes de temps et de ressources pour optimiser l'exécution d'activités.

Représentations : DAG et diagramme de Gantt



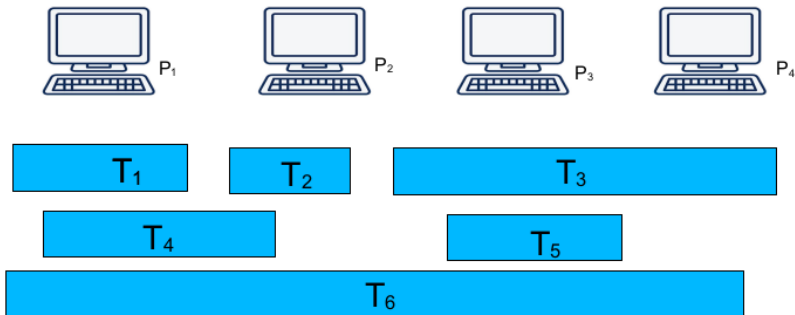
Contextes applicatifs

L'ordonnancement est un problème qui concerne un grand nombre de domaines. Quelques exemples de domaines :

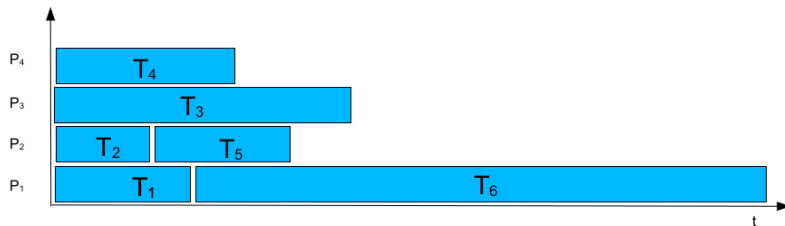
- ▶ Production : ateliers (*shop*) ;
- ▶ Projets : gestion de projets ;
- ▶ Administration : gestion de ressources humaines, emploi du temps ;
- ▶ Informatique : exécution de programmes, optimisation de code.

Problème illustratif (1/2)

Notre problème : trouver un algorithme pour placer un ensemble de tâches sur des machines.



Problème illustratif (2/2)



Types d'approche

- ▶ Solution optimale : tester toutes les solutions (*bruteforce*) ou trouver un algorithme.
- ▶ Étude de la complexité du problème.
- ▶ Technique d'optimisation combinatoire.
- ▶ Bonne heuristique si problème difficile ou algorithme d'approximation (avec une performance garantie).

Plan

Introduction

Classification des problèmes

Analyse de problèmes classiques

Focus sur $P||C_{\max}$

Conclusion

Notation $\alpha|\beta|\gamma$

Classification à trois champs (Graham et al. 1979) :

α Caractéristiques des ressources.

β Caractéristiques des tâches.

γ Critère(s) à optimiser.

Les ressources : α

- ▶ Problème à une machine : $\alpha = 1$
- ▶ Problème à m machines :
 - $\alpha = P$ machines *identiques* : une tâche a le même temps d'exécution quelque soit la machine.
 - $\alpha = Q$ machines *uniformes* : les temps d'exécution sont proportionnels à la performance des machines.
 - $\alpha = R$ machines *indépendantes* : certaines machines peuvent être spécialisées pour certaines tâches.

Les ressources : β

- ▶ Ensemble de n tâches : $T = \{T_1, \dots, T_n\}$.
- ▶ Temps d'exécution d'une tâche : p_j .
- ▶ Date de fin (complétion) d'une tâche : C_j .

Les critères d'optimisation : γ

Sur l'ordonnancement global :

- ▶ Fin de l'ordonnancement (*makespan*) : $\gamma = C_{\max} = \max_j C_j$.
- ▶ Somme des fins d'exécutions : $\gamma = \sum C_j$.
- ▶ Débit (*flow*) : nombre de tâches finies par unité de temps.
- ▶ Avec pondération possible, multi-critères, etc.

Exemples illustratifs

$1||C_{\max}$ Optimisation du temps d'exécution global (makespan) sur une seule machine.

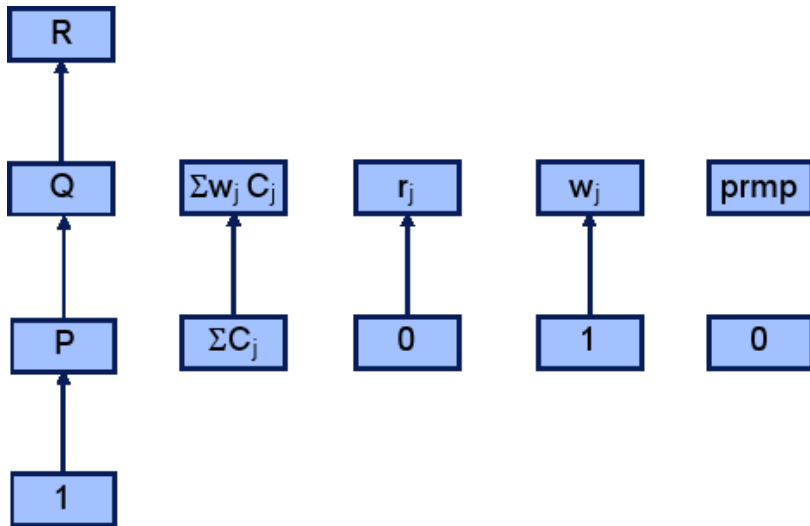
$1||\sum w_j C_j$ Optimisation de la somme des temps d'exécution pondérés par w_j sur une seule machine.

$P||C_{\max}$ Optimisation du makespan sur m machines identiques.

Organisation des problèmes

- ▶ Large gamme de problèmes issus de la combinaison des caractéristiques et d'un critère d'optimisation.
- ▶ Il existe de nombreuses liens entre ces problèmes :
 - ▶ certains sont identiques ;
 - ▶ certains en généralisent d'autres (réduction).
- ▶ Soit \mathcal{P}_1 un problème généralisé par \mathcal{P}_2 ($\mathcal{P}_1 \leq \mathcal{P}_2$) :
 - ▶ Si \mathcal{P}_1 est NP-complet (difficile), \mathcal{P}_2 l'est aussi.
 - ▶ Si un algorithme efficace résout \mathcal{P}_2 , alors il résout aussi \mathcal{P}_1 .

Réductions classiques



Plan

Introduction

Classification des problèmes

Analyse de problèmes classiques

Focus sur $P||C_{\max}$

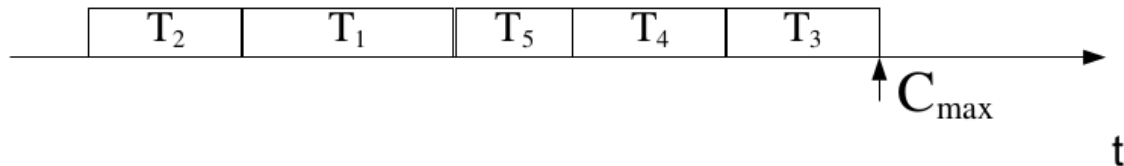
Conclusion

$1||C_{\max} (1/2)$

- ▶ Ensemble de n tâches : $T = \{T_1, \dots, T_n\}$.
- ▶ Temps d'exécution d'une tâche : p_j .
- ▶ Ordre optimal pour optimiser le makespan C_{\max} ?

$1||C_{\max}$ (2/2)

Ordonnancement sans délai (ou compact) :

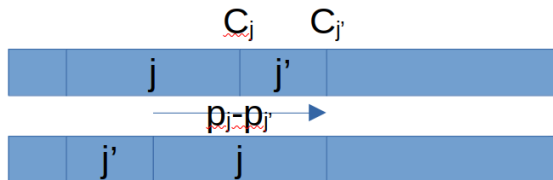


Preuve triviale :

- ▶ borne directe sur le makespan optimal : $OPT \geq \sum_j p_j$;
- ▶ sans délai, le makespan est $C_{\max} = \sum_j p_j \leq OPT$.

$$1 || \sum C_j \quad (1/2)$$

- ▶ Ensemble de n tâches : $T = \{T_1, \dots, T_n\}$.
- ▶ Temps d'exécution d'une tâche : p_j .
- ▶ Ordre optimal pour optimiser la somme des fins d'exécution $\sum C_j$?

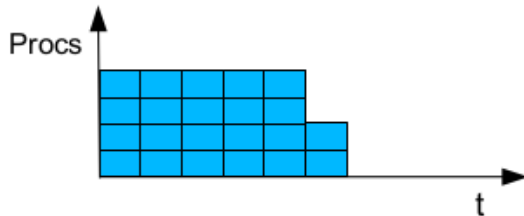
1 || ΣC_j (2/2)

Shortest Processing Time (SPT) :

- ▶ Preuve par contradiction puis argument d'échange.
- ▶ Soit une solution optimale dans laquelle il existe 2 tâches non-ordonnées selon leurs temps d'exécution ($\exists j, j' : C_j < C_{j'}, p_j > p_{j'}$) et dont l'objectif est OPT.
- ▶ Après échange des 2 tâches, toutes les $N \geq 0$ tâches entre les tâches T_j et $T_{j'}$ finissent plus tôt (décalage de $p_j - p_{j'}$).
- ▶ Le nouvel objectif devient : $\text{OPT} - (N + 1) \times (p_j - p_{j'}) < \text{OPT}$.
- ▶ On obtient une solution meilleure ce qui contredit l'optimalité de la solution considérée.

$$P|p_j = 1|C_{\max}$$

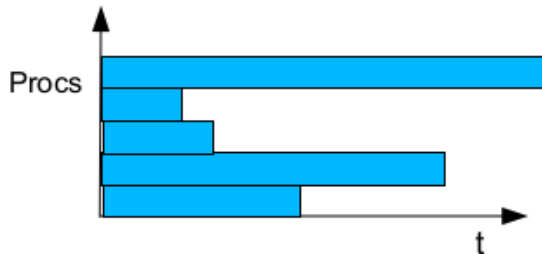
- ▶ Earliest Finish Time (EFT) ou ordonnancement compact au plus tôt.
- ▶ $\text{OPT} = \left\lceil \frac{n}{m} \right\rceil$.



Une borne directe pour le makespan optimal sur m machines : $\text{OPT} \leq \left\lceil \frac{\sum_j p_j}{m} \right\rceil$.

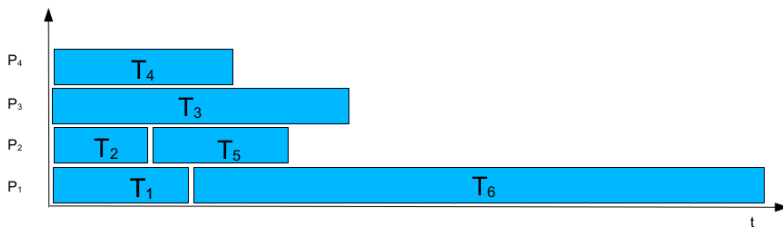
$$P_{\infty} || C_{\max}$$

Une tâche par machine.



$P||C_{\max}$

- ▶ Ensemble de n tâches : $T = \{T_1, \dots, T_n\}$.
- ▶ Temps d'exécution d'une tâche : p_j .
- ▶ Ordre et répartition optimaux pour optimiser le makespan C_{\max} sur m machines identiques ?



- ▶ Problème difficile (NP-complet, Garey et Johnson 1978), pas de solution évidente.
- ▶ Le problème $R||C_{\max}$ est donc aussi difficile (voir “réductions classiques”).

Plan

Introduction

Classification des problèmes

Analyse de problèmes classiques

Focus sur $P||C_{\max}$

Conclusion

Algorithme (ou heuristique) de liste

Définition :

- ▶ Les tâches sont ordonnées en liste en fonction d'un critère :
 - ▶ LPT/SPT : Longest/Shortest Processing Time First.
 - ▶ Aléatoire.
- ▶ Dès qu'une machine est disponible, elle exécute la première tâche de la liste (ordonnancement compact et au plus tôt).
- ▶ Robuste : utilisable même si les temps d'exécution p_j ne sont pas connus et avec un ordre aléatoire.

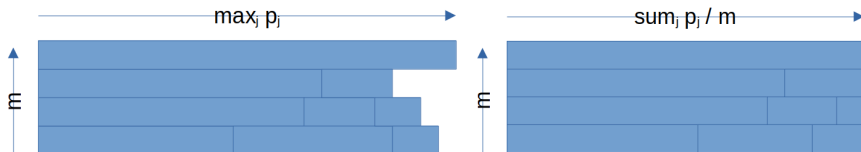
Borne de Graham : résultat (1/3)

Résultat :

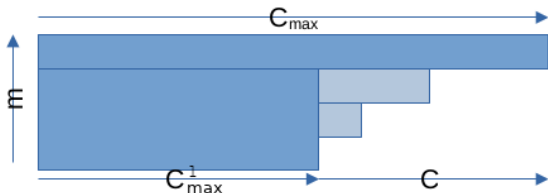
- ▶ Soit LST le makespan obtenu via un algorithme de liste.
- ▶ On peut borner la performance de cet algorithme : $\frac{LST}{OPT} \leq 2 - \frac{1}{m}$.
- ▶ On dit qu'il s'agit d'un algorithme d'approximation $(2 - \frac{1}{m})$.

Propriétés utilisées dans la preuve :

- ▶ Borne sur le temps d'exécution maximum : $OPT \geq \max_j p_j$;
- ▶ Borne sur la charge moyenne : $OPT \geq \frac{1}{m} \sum_j p_j$;
- ▶ Un algorithme de liste commence les tâches au plus tôt.



Borne de Graham : preuve (2/3)



- Soit C_{\max}^1 la première date à laquelle au moins une machine devient disponible.
- Pas de délai avant C_{\max}^1 (ordonnancement compact et au plus tôt) :
 $m \times C_{\max}^1 + C \leq \sum_j p_j$ et donc $C_{\max}^1 \leq \text{OPT} - \frac{C}{m}$.
- Pas de tâche commençant après C_{\max}^1 (sinon, elle aurait été commencée à la date C_{\max}^1 puisqu'au moins une machine devient disponible) : $C \leq \max_j p_j$ et donc $C \leq \text{OPT}$.
- Intégration : $\text{LST} = C_{\max}^1 + C \leq (2 - \frac{1}{m}) \times \text{OPT}$.

Borne de Graham : optimalité (3/3)

Quelle instance (i.e. nombre de tâches et temps d'exécution) permet d'obtenir le plus mauvais facteur d'approximation pour un algorithme de liste pour $m = 2$ machines ?

Borne de Graham : optimalité (3/3)

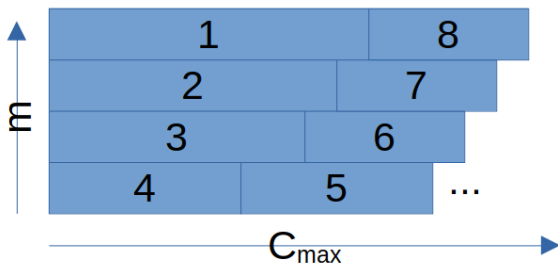
Quelle instance (i.e. nombre de tâches et temps d'exécution) permet d'obtenir le plus mauvais facteur d'approximation pour un algorithme de liste pour $m = 2$ machines ?



- ▶ $\{p_j\} = \{1, 1, 2\}$.
- ▶ Dans ce cas, le facteur d'approximation est bien $2 - \frac{1}{m} = \frac{3}{2}$.
- ▶ Comme c'est aussi le cas pour chaque valeur de m , on dit que la borne de Graham est optimale (ou *tight*).

LPT : résultat (1/2)

- ▶ Longest Processing Time first.
- ▶ Heuristique de liste très répandue et intuitive : priorité aux tâches de poids maximum.
- ▶ Complexité en temps : $O(n \log(n))$.
- ▶ Facteur d'approximation optimal : $\frac{LPT}{OPT} = \frac{4}{3} - \frac{1}{3m}$.



LPT : pire cas (2/2)

Quelle instance permet d'obtenir le plus mauvais ratio pour LPT ?

- ▶ Exercice vu en TD.
- ▶ Pour $m = 2$, le ratio est $\frac{7}{6}$. On peut chercher une instance qui donnerait un makespan de 7 avec LPT alors que la valeur de l'objectif optimale serait à 6.
- ▶ On sait aussi que LPT mettra les grandes tâches en premier alors que ce ne sera pas forcément le cas dans l'ordonnancement optimal.

LPT : pire cas (2/2)

Quelle instance permet d'obtenir le plus mauvais ratio pour LPT ?

- ▶ Exercice vu en TD.
- ▶ Pour $m = 2$, le ratio est $\frac{7}{6}$. On peut chercher une instance qui donnerait un makespan de 7 avec LPT alors que la valeur de l'objectif optimale serait à 6.
- ▶ On sait aussi que LPT mettra les grandes tâches en premier alors que ce ne sera pas forcément le cas dans l'ordonnancement optimal.

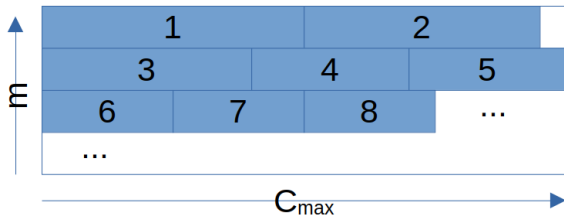
MULTIFIT : résultat (1/2)

- ▶ Recherche dichotomique sur le makespan entre $\left\lceil \max\left(\frac{1}{m} \sum_j p_j, \max_j p_j\right) \right\rceil$ et $\left\lceil \max\left(\frac{2}{m} \sum_j p_j, \max_j p_j\right) \right\rceil$.
- ▶ Pour chaque makespan considéré : application de l'heuristique First-Fit-Decreasing (FFD).
- ▶ Complexité en temps : $O(n \log(n) + n \log(m))$.
- ▶ Facteur d'approximation optimal : $\frac{\text{MULTIFIT}}{\text{OPT}} = \frac{13}{11}$.



MULTIFIT : FFD (2/2)

- ▶ On tri les tâches par temps d'exécution décroissant.
- ▶ On commence par une seule machine.
- ▶ Pour chaque tâche : on l'ordonne sur la première machine qui peut la terminer avant le makespan considéré.
- ▶ Si aucune machine ne le permet, on en rajoute une jusqu'à m .
- ▶ Si toutes les tâches n'ont pas été ordonnancées, FFD échoue.



Plan

Introduction

Classification des problèmes

Analyse de problèmes classiques

Focus sur $P||C_{\max}$

Conclusion

Prochaines échéances

- ▶ Second rendu intermédiaire du projet-tournoi le 20/10.
- ▶ Épreuve sur table le 23/10.
- ▶ Rendu final pour le projet-tournoi le 4/11.
- ▶ Épreuve de TP le 5/11.
- ▶ Restitution du projet-tournoi le 6/11.