Ordonnancement et conduite de systèmes informatiques

Cours du département d'informatique de l'EPFL

Semestres d'hiver 2001 et d'été 2002
Enseignant : Eddy Mayoraz

Objectifs du cours

Connaître les modèles mathématiques les plus courants qui permettent d'évaluer et d'optimiser les performances de systèmes informatiques complexes et de savoir les utiliser, les modifier et les appliquer à des cas réels.

Table des matières

Chapitre 1

Présentation de quelques problèmes types

1.1 Un problème d'ordonnancement sur deux processeurs

Données

- 2 processeurs
- n tâches à exécuter. Les tâches sont non-préemptives (i.e. d'une fois que l'exécution d'une tâche a débuté sur un processeur elle ne peut être interrompue). Elles doivent toutes passer d'abord sur le processeur R_1, puis sur le processeur R_2. Les temps d'exécution p_{kj} de la tâche T_j sur le processeur R_k sont connus.

Problème

Minimiser le temps total pour exécuter toutes les tâches.

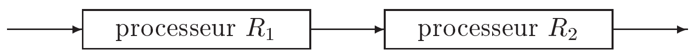
Algorithme de Johnson

1. Trouver $\min_j\{p_{1j},p_{2j}\}$
2. Si le temps d'exécution minimum a lieu sur le processeur R_1, placer la tâche associée à ce temps à la première place disponible dans la séquence. Aller à l'étape 4.
3. Si le temps d'exécution minimum a lieu sur le processeur R_2, placer la tâche associée à ce temps à la dernière disponible dans la séquence.
4. Supprimer de l'ensemble des tâches à prendre en considération la tâche qui vient d'être classée. Retourner à l'étape 1 jusqu'à ce que toutes les tâches soient placées.

Exemple

5 tâches avec temps d'exécution suivants :

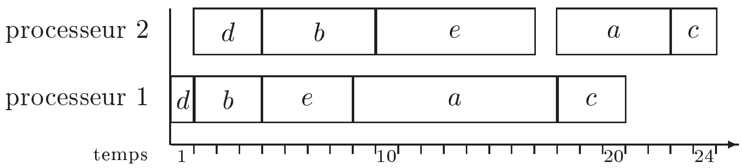
tâche / **processeur**	a	b	c	d	e
p_{1j}	9	3	3	1	4
p_{2j}	5	5	2	3	7



Déroulement de l'algorithme de Johnson :

Étape	**tâche à ordonner**	$\min_j\{p_{1j},p_{2j}\}$	**tâche associée**	**séquence partielle**
1	$\{a,b,c,d,e\}$	1	d	$d\ .\ .\ .\ .$
2	$\{a,b,c,e\}$	2	c	$d\ .\ .\ .\ c$
3	$\{a,b,e\}$	3	b	$db\ .\ .\ c$
4	$\{a,e\}$	4	e	$dbe\ .\ c$
5	$\{a\}$	5	a	$dbeac$

Résultat de l'ordonnancement :



Autres problèmes

- Comment faire si les tâches doivent passer sur $m > 2$ processeurs ?
- Comment faire si les tâches peuvent passer sur les processeurs dans un ordre quelconque ?
- Que faire s'il y a des délais ?

1.2 Un problème de réseaux de Pétri

"Modélisation d'un système temps réel"

Données

Un système informatisé est composé de 6 modules. Le module M_4 commence son travail lorsque les modules M_1, M_2 et M_3 ont terminé le leur. La fin du travail du module M_4 provoque le début des travaux sur les modules M_5 et M_6. Tant que le module M_6 travaille, le module M_3 ne fonctionne pas.

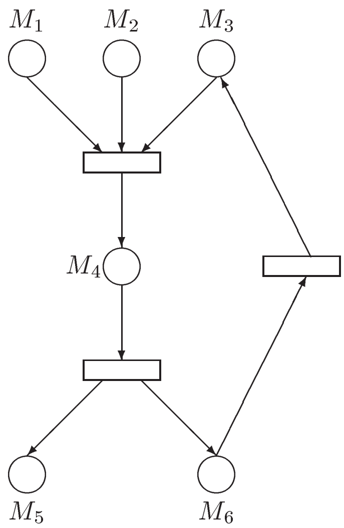
Problème

Modéliser ce système informatisé.

Résolution

Réseaux de Pétri temporisé

graphe biparti



Autres problèmes

- Peut-on résoudre des problèmes d'ordonnancement avec de tels modèles ?
- Que se passe-t-il pour des modèles plus complexes ?

1.3 Un problème de chemin critique

Données

Un logiciel de GPAO (gestion de production assistée par ordinateur) composé de 7 modules (sous-programmes) doit être conçu. Des estimations sur la durée de programmation des modules sont connues. De plus, certains modules doivent être finis avant d'autres (gestion de stock avant graphique, par exemple).

Problème

Au bout de combien de temps le logiciel sera-t-il opérationnel ?

Méthode

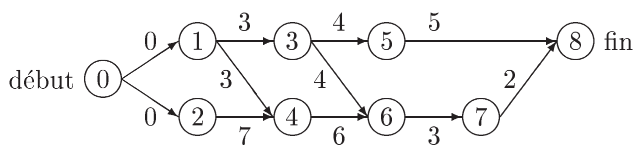
Construire un graphe potentiels-tâches. Chercher le chemin critique (grâce à un algorithme de type Bellman).

Exemple

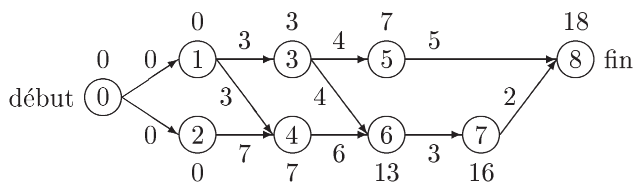
5 modules avec contraintes de précédence.

module	**durée**	**contraintes d'ordre**
1	3	-
2	7	-
3	4	après module 1
4	6	après modules 1 et 2
5	5	après module 3
6	3	après modules 3 et 4
7	2	après module 6

Construction du graphe potentiels-tâches



Recherche du chemin critique par l'algorithme de Bellman 1.1 pour la recherche d'un chemin de longueur maximum.
Le graphe avec les marques obtenues par l'algorithme est le suivant



Autres problèmes

- Que se passe-t-il si des modules peuvent commencer alors que d'autres sont partiellement terminés ?
- Comment répartir le travail si l'on a plus de modules que de programmeurs?

FIG. 1.1 – *Algorithme de recherche de plus long chemin*

début
  $r_0 = 0$;
  marquer le sommet 0 ;
  tant que ($\exists$ des sommets non marqués) **faire**
    si ($\nexists j$ non marqué
      t.q. tous ses prédécesseurs sont marqués) **alors**
      exit; — le graphe a un circuit
    sinon
      soit j un tel sommet;
      $r_j := \max_{(i,j) \in U} (r_j + v_{ij})$
      marquer j;
    fin de si
  fin de tant que
fin

1.4 Un problème d'ordonnancement avec processeurs identiques et tâches indépendantes

Données

- m processeurs identiques en parallèle
- n tâches à exécuter, les tâches sont non préemptives et indépendantes

Problème

Minimiser le temps total pour exécuter toutes les tâches.

Heuristique

Une heuristique simple met efficace pour ce problème est connue sous le nom de LPT (Longest Processing Time).

FIG. 1.2 – *algorithme LPT*

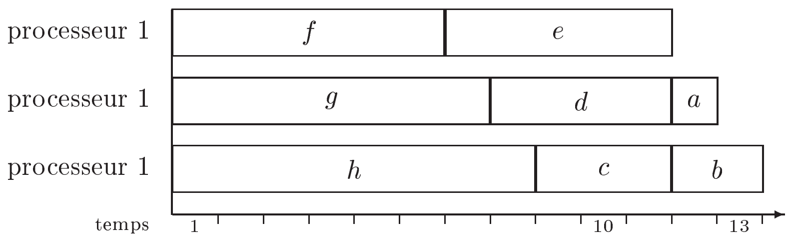
initialisation
  Ordonner les tâches par temps d'exécution décroissant
tant que il reste des tâches non distribuées **faire**
  affecter la première tâche au processeur le moins occupé
  supprimer cette tâche de la liste
fin de tant que

Exemple

3 processeurs et 8 tâches

tâche	a	b	c	d	e	f	g	h
temps d'exécution	1	2	3	4	5	6	7	8

L'heuristique ci-dessus produit l'ordonnancement suivant :



Autres problèmes

- La solution de l'heuristique est-elle optimale ?
- Que se passe-t-il si les tâches sont préemptives ?
- Que se passe-t-il s'il y a des délais ?

1.5 Un problème de file d'attente

"Le temps de réponse d'un ordinateur"

Données

- n terminaux reliés à un ordinateur qui n'a qu'un seul processeur.
- Le temps d'exécution d'une tâche sur le processeur suit une loi exponentielle de moyenne x secondes.
- La durée entre deux demandes d'un même terminal suit une loi exponentielle de moyenne y secondes.

Problème

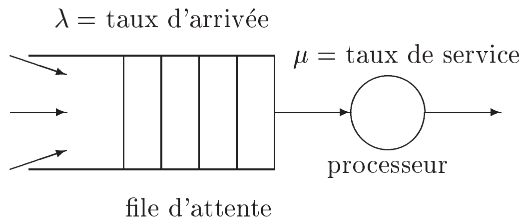
Combien de temps une tâche devra-t-elle attendre en moyenne avant d'être traitée par le processeur ?

Résolution

Utilisation des résultats de la théorie des réseaux de files d'attente.

Exemple

- $n = 36$ terminaux
- $x = 0.5$ seconde
- $y = 20$ secondes



Taux d'arrivée : $\lambda = \frac{n}{y} = \frac{9}{5}$, car la superposition de processus indépendants poissonnien est aussi poissonnienne.
Taux d'exécution : $\mu = \frac{1}{x} = 2$.
Temps de réponse moyen : $E[T] = \frac{1}{\mu - \lambda} = 5$ secondes.

Autres problèmes

- Que se passe-t-il si le nombre de processeurs est doublé ?
- Que se passe-t-il dans les systèmes plus complexes ?

1.6 Un problème d'atelier flexible

"Estimation du taux de production d'un atelier flexible"

Données

Un atelier flexible est composé de trois stations, d'un système de transport, de vingt palettes, dans lequel deux types de produits doivent être fabriqués. Le temps moyen de transport ainsi que les gammes opératoires des produits sont connus.

Problème

Estimer le taux de production horaire de l'atelier.

Résolution

Utilisation de la théorie des réseaux de files d'attente appliquée aux ateliers flexibles.

Exemple

station	nom	nb de serveurs
i	laminoir	2
ii	foreuse	3
iii	inspection	1
iv	convoyeur	2

Le temps moyen de transport est de 4 minutes, et il y a deux gammes opératoires A et B.

gamme opératoire A

op.	stat.	durée	fréq.
1	i	20	1
2	ii	30	1
3	iii	15	0.5
4	i	14	1

gamme opératoire B

op.	stat.	durée	fréq.
1	ii	90	1
2	i	30	1

Mise en équation

Voir théorie des réseaux de files d'attente. Solution : Trois pièces par heure.

Autres problèmes

– Que se passe-t-il si on augmente le nombre de palettes ?
– Quel est le temps moyen passé par les pièces dans le système ?
– Peut-on modéliser les pannes des diverses machines ?

Première partie

Ordonnancement et conduite

Chapitre 1

Introduction

Où rencontre-t-on les problèmes d'ordonnancement ? L'action de trouver un ordre pour exécuter un certain nombre de tâches est omniprésente :

- industrie : organisation du passage de pièces sur différentes machines dans un atelier
- transport : atterrissage des avions sur des pistes libres, gestion des voies d'une gare
- informatique : gestion d'un ordinateur central
- école : horaires de cours ou d'examen
- etc.

Beaucoup de systèmes de la vie quotidienne fonctionnent sur des règles simples et égalitaires telle que

Le premier arrivé est le premier servi (FIFO)

Malheureusement, cette règle ne s'applique que dans les cas où les tâches à exécuter ont un ordre temporel initial, elle ne fournit que très rarement la solution optimale. Dans ce cours, nous allons cataloguer les différents problèmes d'ordonnancement en fonction de leur nature, leurs contraintes et leurs objectifs. Dans chaque cas, nous verrons des règles adaptées pour leur résolution et en étudierons la difficulté.

1.1 Définitions et généralités

Definition 1.1 L'ordonnancement est l'allocation de ressources dans le temps pour exécuter un ensemble de tâches. •

La théorie de l'ordonnancement regroupe :

– Les modèles mathématiques en relation avec la fonction ordonnancement (fonction qui permet de déterminer la séquence dans laquelle un ensemble de tâches va être exécuté),
– les techniques permettant de résoudre ces problèmes.

Une approche quantitative est nécessaire pour traduire les *buts de décision* en *fonctions objectif* et les *restrictions* en *contraintes explicites.*

Les buts de décision les plus souvent rencontrés sont :

– utiliser efficacement les ressources
– répondre rapidement à la demande
– rester proche des délais prévus.

La fonction objectif représente la somme des coûts dépendant de l'ordonnancement. Ces coûts sont souvent difficile à estimer, donc on choisit d'autres critères : temps d'utilisation, durée d'attente, retard, ...).

Une restriction omniprésente est la limite sur le nombre de ressources. Une restriction très fréquente est un ensemble de contraintes techniques sur l'ordre des opérations pour exécuter les tâches.

Une solution du problème d'ordonnancement est une solution admissible (respectant les contraintes) qui permet de répondre aux questions :

– Quelles ressources sont nécessaires pour exécuter chaque tâche ?
– Quand chaque tâche doit-t-elle être exécutée ?

Definition 1.2 En anglais, on parle de sequencing pour désigner un problème d'ordonnancement sur une seule machine, alors que scheduling dénote un ordonnancement simultané et synchronisé sur plusieurs machines. •

1.2 Description du problème de base

Definition 1.3 L'élément de base dans un processus d'ordonnancement est l'opération. C'est l'action élémentaire à effectuer. Une tâche est constituée d'une séquence ordonnée d'opérations appelée sa gamme opératoire. •

Definition 1.4 Une ressource est l'élément nécessaire à l'exécution d'une opération. Une tâche requiert un ensemble de ressources pour son exécution. •

terminologie française	exemples dans un système informatique	exemples dans un atelier de production	terminologie anglaise
opération	compilation, exécution	fraisage, usinage, emballage	task
tâche	processus, script	produit, pièce	job
ressource	processeur, mémoire	machine, fraise, station	ressource
ensemble de ressources	ordinateur central	atelier	set of ressources

Notez la différence entre les terminologies françaises et anglaises (tâche $\neq$ task).

Definition 1.5 Une opération est liée (i) à une tâche, ce qui lui impose des contraintes de précédence, (ii) à une ressource, qui implique des contraintes de disponibilité, et (iii) à un temps d'exécution. On distingue généralement différents types de contraintes de disponibilité. Les contraintes disjonctives apparaissent lorsque deux tâches utilisant la même ressource ne peuvent pas s'exécuter simultanément. Les contraintes cumulatives apparaissent lorsque par exemple trois processeurs sont disponibles pour l'exécution de quatre tâches : on ne pourra exécuter plus de trois de ces tâches simultanément, sans que l'on puisse savoir à l'avance laquelle sera différée. •

Étant donné un ensemble de ressources et un ensemble de tâches, le but du problème d'ordonnancement est donc de trouver le ou les intervalles de temps durant lesquels l'exécution de chaque opération doit être effectuée, tout en respectant les contraintes.

Definition 1.6 Deux cas peuvent se présenter dans la description des tâches : si le nombre de tâches et leurs caractéristiques sont connues, le problème est dit statique, sinon il est dynamique. •

Definition 1.7 Les ressources peuvent être soit parallèles, si elles ont toutes les mêmes fonctionnalités, soit elles sont dédiées. •

Definition 1.8 Des ressources parallèles sont dites identiques si leur vitesse sont les mêmes, ou uniformes si leur vitesse ne sont pas nécessairement les mêmes mais ne dépendent pas des opérations. •

Definition 1.9 On distingue généralement trois types de problèmes à ressources dédiées. Soit $O_{ij},i = 1,\ldots,n_j$, les opérations composant la tâche T_j. Si tous les n_j sont égaux à n et l'opération O_{ij} est effectuée par la même ressource R_i quelle que soit la tâche T_j, on parle alors d'un problème d'open-shop. Si de plus toutes les opérations doivent être effectuées dans le même ordre $O_{1j},O_{2j},\ldots,O_{nj}$ quelle que soit T_j, il s'agit d'un problème de flow-shop (e.g. chaîne de montage). Par contre, si le nombre d'opérations n_j de chaque tâche varie et chaque opération O_{ij} est effectuée par une ressource R_{ij}, mais la séquence des opérations $O_{1j},O_{2j},\ldots,O_{nj}$ de chaque tâche T_j est fixée, il s'agit d'un problème de job-shop.
•

1.3 Modélisation

Passons à présent à un domaine plus mathématique : la modélisation des problèmes d'ordonnancement.

1.3.1 Les hypothèses de base

Tout problème d'ordonnancement se base sur les cinq hypothèses suivantes :

1. un ensemble de n tâches indépendantes sont à exécuter dans l'atelier.
2. les temps de réglage des machines sont indépendants de la séquence dans laquelle les tâches sont exécutées et ces temps sont inclus dans les temps d'exécution des opérations.
3. la gamme opératoire de chaque tâche est connue.
4. les machines sont toujours disponibles (pas de panne) et il n'y a jamais de tâches en attente devant une machine libre.
5. une seule tâche est exécutée à la fois dans une même machine et l'opération se fait de manière continue (pas d'interruption).

Ces hypothèses peuvent paraître très rigides. Nous verrons par la suite des modèles où certaines d'entre elles sont ignorées (prise en compte des temps de réglage, modélisation des pannes et des maintenances, ...).

1.3.2 Les données

Pour chaque produit T_j ($j = 1,...,n$, n étant le nombre de produits) devant être traité dans l'atelier, les données suivantes sont nécessaires.

- n_j nombre d'opérations composant la tâche T_j
- r_j heure d'entrée de la tâche T_j dans l'atelier (heure à partir de laquelle T_j est disponible pour son exécution)
- d_j heure de sortie au plus tard de la tâche T_j de l'atelier (heure à partir de laquelle la tâche T_j doit être complètement exécutée)

p_{ij} le temps d'exécution de l'opération O_{ij} de la tâche $T_j (i = 1, \ldots, n_j)$.

Les temps d'exécution sont naturellement positifs ($p_{ij} > 0$). A partir de ces éléments, la durée maximale a_j que peut passer le produit T_j dans l'atelier est donnée par la formule :

$$a_j = d_j - r_j.$$

De plus, une condition nécessaire pour avoir une solution admissible est fournie par cette durée maximale : le temps total d'exécution p_j de la tâche T_j ne doit pas lui être supérieur !

$$p_j = \sum_{i=1}^{n_j} p_{ij} \leq a_j.$$

Rappelons que le temps de réglage de la machine pour l'opération que doit subir la tâche T_j est contenu dans le temps d'exécution p_{ij}.

1.3.3 Les variables

Analysons à présent les variables qui vont intervenir dans le problème. Très souvent, une pièce sortant d'une machine ne peut pas être immédiatement servie par la machine devant exécuter sa prochaine opération. La pièce est stockée en attendant que la machine se libère. Les variables suivantes sont définies (pour chaque produit T_j, $j = 1, \ldots, n$) :

w_{ij} temps d'attente de la tâche T_j pour subir l'opération $O_{ij} (i = 1, \ldots, n_j)$
w_j temps total d'attente de la tâche T_j

En supposant que les pièces terminées quittent immédiatement l'atelier, l'heure à laquelle la dernière opération de la tâche T_j se termine dépend de l'heure d'entrée, du temps d'attente et du temps d'exécution :

C_j heure de sortie effective de l'atelier est défini par $C_j = r_j + w_j + p_j$

La date de sortie au plus tard nous impose la contrainte :

$$C_j \leq d_j.$$

Le temps effectivement passé dans l'atelier pour le produit T_j est donné par la somme des temps totaux d'exécution et d'attente ou par la différence des heures de sortie et d'entrée :

F_j temps passé dans l'atelier est défini par $F_j = p_j + w_j = C_j - r_j$

La condition nécessaire pour avoir une solution admissible se transforme ainsi

$$F_j \leq a_j.$$

L'expérience de la vie quotidienne a montré que produire pour une date précise est une pure illusion. Une nouvelle variable, le temps de décalage, doit donc être définie :

L_j temps de décalage (lateness) est donné par $L_j = C_j - d_j = F_j - a_j$.

Comme le décalage exprime aussi bien retard qu'une avance, les variables associées sont les suivantes :

T_j retard de la tâche T_j (Tardiness) est donné par $T_j = \max\{0, L_j\}$,

E_j avance de la tâche T_j (Earliness) est donné par $E_j = \max\{0, -L_j\}$.

Ces diverses variables nous sont utiles pour construire la fonction objectif.

1.3.4 Les mesures de performance

Les désirs des chefs d'entreprise peuvent être multiples. Si la volonté de produire davantage à moindres coûts est générale, les moyens pour y parvenir sont divers. Certains veulent diminuer le plus possible les pièces en cours dans l'atelier (agissant sur le temps passé dans l'atelier), d'autres préfèrent minimiser les retards dans les livraisons.

Les critères de décision peuvent être, par exemple :

$\overline{F} = \frac{1}{n}\sum_{j=1}^{n} F_j$ temps moyen passé dans l'atelier

$\overline{T} = \frac{1}{n}\sum_{j=1}^{n} T_j$ retard moyen d'une pièce

$F_{\max} = \max_j F_j$ temps maximum passé dans l'atelier

$T_{\max} = \max_j T_j$ retard maximum d'une pièce

$N = \sum_{j=1}^{n} \Theta(T_j)$ nombre de pièces en retard, où la fonction Θ est définie comme $\Theta(x) = 1$ si $x > 0$ et 0 sinon.

Les mesures de performance présentées ici sont les plus fréquemment utilisées. Les décideurs sont naturellement susceptibles de faire maintes variations avec ces fonctions en introduisant d'autres paramètres tels des coûts de stockage, de pénurie. L'une de ces mesures devient la fonction objectif que l'on cherchera à minimiser.

1.4 Classification des problèmes d'ordonnancement

Pour définir un problème d'ordonnancement, quatre types d'information sont primordiaux (pour fixer les idées, nous nous intéressons aux ateliers de production) :

- les produits (et les opérations) qui doivent être exécutées
- le nombre et le type de machines composant l'atelier
- les restrictions qui limitent la manière de faire les affectations
- les critères qui permettent d'évaluer l'ordonnancement

Il existe une notation à trois paramètres classifiant les différents problèmes d'ordonnancement :

$$\boxed{\text{A / B / C}}$$

avec la signification suivante :

A donne le nombre et le type de machines de l'atelier, ainsi que l'ordre dans lequel sont exécutées les différentes opérations de chacune des tâches :

- **1** une seule machine
- P_m m machines parallèles identiques
- F_m parcours séquentiel des m machines commun à toutes les tâches (flow-shop)
- J_m parcours séquentiel des m machines dépendant des tâches (job-shop)
- O_m m machines, parcours non fixé (open-shop)

B décrit les particularités des tâches :

- **pmtn** préemptions autorisées
- **prec** existence de relations de précédence entre tâches
- r_j existence de dates au plus tôt
- $p_{ij} = 1$ durée d'exécution des opérations égale à 1

C décrit le critère à travers lequel l'ordonnancement doit être évalué (la fonction objectif) :

- $F_{\max}$ le temps maximum passé dans le système
- $C_{\max}$ l'heure de sortie du système de la dernière pièce
- $\overline{F}$ le temps moyen passé dans le système

A titre d'exemple, le problème de n tâches dans un atelier comportant m machines de façon à ce que la dernière tâche accomplie soit terminée le plus tôt possible s'écrit ainsi : J_m // $C_{\max}$. Notons au passage que ce problème n'est pas encore résolu.