Leçon 13 : Algorithmes d'ordonnancement de tâches et de gestion de ressources

Motivation : beaucoup d'applications (gestion de projets, ordonnancement de processus ...)

I- Définitions et notations :

Def : Un problème d'ordonnancement est un problème d'optimisation sous contraintes mettant en relation des tâches à exécuter, des machines permettant l'exécution et le temps.

On note $\mathcal{J} = \{J_1, \dots, J_n\}$ l'ensemble des tâches et $\mathcal{M} = \{M_1, \dots, M_m\}$ l'ensemble des machines

Contraintes du problème :
- à un instant donné, au plus une tâche en cours d'exécution sur une machine

Selon les problèmes, on peut avoir (entre autres) :
- préemption : possibilité d'interrompre l'exécution d'une tâche
- précédence : contrainte d'ordre sur l'exécution des tâches.
- date de début au plus tôt (notée r_i) ou de fin au plus tard (notée d_i).

On note S_i et C_i les dates de début et de fin de la tâche J_i, ainsi que p_i son temps d'exécution

Critères de minimisation (exemples) :
- Temps total : $C_{max} = \max_{1 \le i \le n} C_i$
- Retard maximal : $L_{max} = \max_{1 \le i \le n} (C_i - d_i)$
- Somme (pondérée) des retards, date de fin, ...
- Nombre de tâches en retard

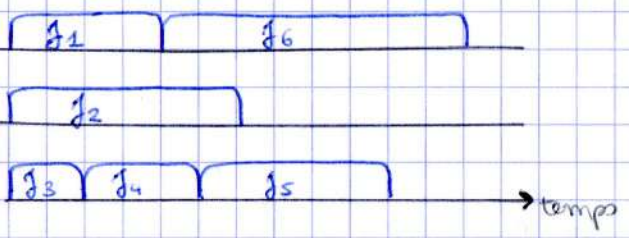
etc.

Notation de classe de problème $\alpha | \beta | \gamma$ avec :

α : type, éventuellement nombre, des machines parmi
- P : machines parallèles identiques
- Q : machines parallèles avec vitesse propre (uniforme) : une machine M_j de vitesse s_j exécute J_i en temps $\frac{p_i}{s_j}$
- R : machines parallèles sans relation : chaque machine M_j a une vitesse pour chaque tâche J_i : s_{ij}.

Représentation d'un ordonnancement : diagramme de Gantt avec en abscisse le temps, en ordonnée les différentes machines

ex :



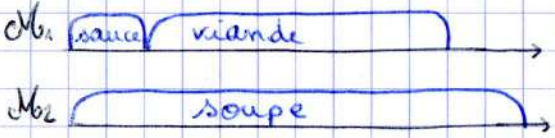
Ex de modélisation d'une situation concrète

Objectif : Cuisiner en temps minimal une soupe (1h cuisson), une ~~soupe~~ sauce (10 min de cuisson) et une viande (40 min).

On a besoin de la sauce pour cuire la viande.
On dispose de 2 casseroles identiques.

Classe du problème : $P2 | prec | C_{max}$

Solution possible



II - Algorithmes d'ordonnancements (Exemples) théoriques

A - Cas sans limite de ressources :

Problème central d'ordonnancement :
- suppose ressources non limitantes (infini machines)
- seule contrainte : relations de précédence

Def : Graphe de précédence $G = (J, A)$
Sommet : tâches
Arêtes : $(J_i, J_j) \in A$ ssi relation de précédence de J_i à J_j

Def : Graphe potentiel tâches
- 2 sommets fictifs (*)
- arcs valués par durée des tâches

Ex : Minimiser C_{max}
- Début au + tôt d'une tâche : calcul d'un chemin le + long
- Tps min : Chemin le + long du début à fin

Algo : calcul de + long chemin ds un graphe.

(*) un qui précède toutes les tâches sans prédécesseurs et un qui succède ttes les tâches sans successeurs

B - Ressources limitées :

1 - Une machine :

Ex : 1er cas :
- Hyp : - Non préemptif
  - Minimiser $\sum \omega_i C_i$ avec $\omega_i \geq 0$ poids associé à chaque tâche
- Règle de Smith : Ordonner par $\frac{P_i}{\omega_i}$ (weighted Shortest Processing Time).
- 2e cas :
  - préemptif
  - date de début au plus tôt

DEV1 : Règle de Smith optimale pour 1er cas
2-approximation ds 2e.
(non préparé...)

2 - Plusieurs machines :

Hyp et : · tâches durée unitaire
· relation précédence forme anti-arborescence
· non préemptif

Def : Niveau : nombre de descendants d'un sommet, lui-même inclus

Def/Algo : Ordonnancement de niveau
L : liste ordonnée par niveau décroissant

```
Tant que L ≠ ∅
    Tant qu'il existe Mi libre et tâche faisable
        (tous les successeurs terminés)
        J ← 1ère tâche faisable de L
        Enlever J de L
        Exécuter J sur Mi
    temps += 1
```

DEV2 : Ordonnancement de niveau optimal.

III - Cas particuliers : ordonnancement d'un processeur

A- Spécificité :

On se place dans le cas d'un système interactif à un processeur

Contraintes :
- préemptif
- maximiser l'utilisation du processeurs (éviter les attentes d'entrée/sortie (E/S))
- Equité : Tous les processus (tâches) doivent avoir accès au processeur

Critères d'optimisation :
- temps de réponse : vitesse de réaction aux interventions extérieurs.
- prévisibilité : une tâche perçue comme rapide devrait être exécutée rapidement.

Premier algorithme : Tourniquet ou algorithme de Round-Robin.

F : file des processus
q : quantum (petite durée)

Boucle infinie :
J ← défiler F
Executer J pendant un temps q
Si J non fini :
Enfiler J dans F

Avantages : - Equité respectée
- Simple

Inconvénients : Pas de garantie de temps de réponse (si beaucoup de processus en cours) ou de maximisation de l'utilisation du CPU (si beaucoup de processus nécessitant des E/S).

Réglages de la valeur q : petite → perte de temps pendant la commutation
grande → augmentation du temps d'attente des processus.

B- Ordonnancement par priorité

1- Priorité statique

Principe : A chaque processus est attribué une priorité fixe.

Les processus sont exécutés par ordre de priorité décroissante (par ex : tourniquet dans une même classe de priorité).

- Temps de réponse amélioré pour les processus de haute priorité
- Risque de famine : Processus de faibles priorités peuvent ne jamais accéder au CPU (pas d'équité).

2- Priorité dynamique :

Principe : Avantager les processus de temps court entre deux blocages.

La priorité est actualisée après chaque exécution partielle selon la fraction du quantum utilisé

ex : Un processus utilisant tout le quantum sera de priorité 1, un autre utilisant la moitié aura une priorité 2.