Planification des Tâches sur Plusieurs Machines : Explication et Implémentation en C++

Votre Nom

16 octobre 2024

Table des matières

1 Introduction

Cet article présente une explication détaillée d'un algorithme de planification des tâches (scheduling) sur plusieurs machines, souvent utilisé dans le contexte du *Job Shop Scheduling*. L'objectif est de déterminer les dates de début (`st[i][j]`) pour chaque tâche sur différentes machines, tout en respectant les contraintes de séquencement et les dépendances entre les tâches.

2 Définition des Variables

Pour comprendre l'algorithme, il est essentiel de définir clairement les variables utilisées :

- **st[i][j]** : Date de début de la j^{ème} tâche sur la i^{ème} machine.
- **pere[i][j] = (k, p)** : Prédecesseur de la j^{ème} tâche sur la i^{ème} machine, représenté sous la forme (k, p), où k est la machine et p est la tâche prédécesseur.
- **njob[j]** : Nombre de tâches déjà planifiées pour la machine j.
- **V[i]** : Tableau représentant l'ordre des tâches à exécuter.
- **m[j][njob[j]]** : Machine assignée à la j^{ème} tâche au n^{ème} ordre.
- **p[j][njob[j]]** : Durée de traitement de la j^{ème} tâche au n^{ème} ordre.
- **nmach[machine]** : Dernière tâche programmée sur une machine spécifique.

3 Étapes de l'Algorithme

L'algorithme se décompose en plusieurs étapes clés :

3.1 Initialisation

```
st[0][0] = 0
Pour tout i et j : st[i][j] = 0
njob[n] = [0]
```

- `st[0][0] = 0` : La première tâche commence à l'instant 0.
- Initialisation de toutes les dates de début `st[i][j]` à 0.
- Initialisation du compteur `njob[j]` pour chaque machine j à 0, indiquant qu'aucune tâche n'est encore planifiée sur aucune machine.

3.2 Boucle Principale

```
Pour i de 1 à n*m faire
  j = V[i]
  njob[j]++
  si njob[j]>1 alors (cas arc horizontal)
    date = st[j][njob[j]-1]
    si date + p[j][njob[j]-1] > st[j][njob[j]]
        st[j][njob[j]] = date + p[j][njob[j]-1]
        pere[j][njob[j]] =  (j, njob[j]-1)
    fsi
```

```
fsi

machine = m[j][njob[j]]
si nmach[machine] != (0,0) alors
  date = st[j][njob[j]-1]
  duree = p[j][njob[j]-1]
  si date + duree > st[j][njob[j]]
      st[j][njob[j]] = date + duree
      pere [j][njob[j]] =  (j, njob[j]-1)
nmach[machine] = (j, njob[j]-1)
```

3.2.1 Description Pas à Pas

1. **Sélection de la Tâche :** Pour chaque étape `i`, la tâche `j` est sélectionnée selon l'ordre défini dans le tableau `V`.
2. **Incrémentation du Compteur :** Le compteur `njob[j]` est incrémenté pour indiquer qu'une nouvelle tâche sur la machine `j` est planifiée.
3. **Gestion des Conflits sur la Même Machine (Arc Horizontal) :**
   — Si `njob[j] > 1`, cela signifie qu'il y a plus d'une tâche à exécuter sur la même machine.
   — La date de début de la tâche actuelle est ajustée pour qu'elle commence après la fin de la tâche précédente sur la même machine.
   — Le prédécesseur de la tâche actuelle est enregistré.
4. **Gestion des Machines :**
   — La machine assignée à la tâche actuelle est déterminée.
   — Si la machine est déjà occupée, la date de début de la tâche actuelle est ajustée pour qu'elle commence après la fin de la tâche en cours sur cette machine.
   — L'état de la machine est mis à jour avec la dernière tâche programmée.

4 Exemple Illustratif

Pour mieux comprendre l'algorithme, considérons un exemple concret avec **2 machines** et **3 tâches**.

4.1 Paramètres de l'Exemple

— **Machines** : M1, M2
— **Tâches** : T1, T2, T3
— **Durée de Traitement** (`p`) :
— **Ordre des Tâches** (`V`) : [T1, T2, T3, T1, T2, T3]

4.2 Étapes de l'Algorithme avec Cet Exemple

Nous allons parcourir chaque étape de la boucle principale et mettre à jour les variables en conséquence.

Tâche	Machine Assignée	Durée
T1	M1	4
T2	M1	3
T3	M2	2
T1	M2	1
T2	M2	2
T3	M1	3

TABLE 1 – Durées des Tâches sur les Machines

4.2.1 Étape 1 : i = 1 (T1)

- **Sélection de la Tâche** : T1
- **Incrémentation** : `njob[T1] = 1`
- **Arc Horizontal** : Pas de conflit car `njob[T1] = 1`.
- **Machine Assignée** : M1
- **Disponibilité de la Machine** : M1 est disponible.
- **Mise à Jour** : `nmach[M1] = (T1, 1)`

4.2.2 Étape 2 : i = 2 (T2)

- **Sélection de la Tâche** : T2
- **Incrémentation** : `njob[T2] = 1`
- **Arc Horizontal** : Pas de conflit car `njob[T2] = 1`.
- **Machine Assignée** : M1
- **Disponibilité de la Machine** : M1 est occupée par T1 jusqu'à 4.
- **Ajustement de la Date** : `st[T2][1] = 4`
- **Mise à Jour** : `nmach[M1] = (T2, 1)`

4.2.3 Étape 3 : i = 3 (T3)

- **Sélection de la Tâche** : T3
- **Incrémentation** : `njob[T3] = 1`
- **Arc Horizontal** : Pas de conflit car `njob[T3] = 1`.
- **Machine Assignée** : M2
- **Disponibilité de la Machine** : M2 est disponible.
- **Mise à Jour** : `nmach[M2] = (T3, 1)`

4.2.4 Étape 4 : i = 4 (T1)

- **Sélection de la Tâche** : T1
- **Incrémentation** : `njob[T1] = 2`
- **Arc Horizontal** :
  - Date précédente : `st[T1][1] = 0`
  - Durée précédente : 4
  - `st[T1][2] = 4`
  - **Prédécesseur** : `pere[T1][2] = (T1, 1)`
- **Machine Assignée** : M2
- **Disponibilité de la Machine** : M2 est occupée par T3 jusqu'à 2.

— **Ajustement de la Date** : `st[T1][2] = 4` (Pas de chevauchement)
— **Mise à Jour** : `nmach[M2] = (T1, 2)`

4.2.5 Étape 5 : i = 5 (T2)

— **Sélection de la Tâche** : T2
— **Incrémentation** : `njob[T2] = 2`
— **Arc Horizontal** :
  — Date précédente : `st[T2][1] = 4`
  — Durée précédente : 3
  — `st[T2][2] = 7`
  — **Prédécesseur** : `pere[T2][2] = (T2, 1)`
— **Machine Assignée** : M2
— **Disponibilité de la Machine** : M2 est occupée par T1 jusqu'à 5.
— **Ajustement de la Date** : `st[T2][2] = 7` (Après la fin de T1)
— **Mise à Jour** : `nmach[M2] = (T2, 2)`

4.2.6 Étape 6 : i = 6 (T3)

— **Sélection de la Tâche** : T3
— **Incrémentation** : `njob[T3] = 2`
— **Arc Horizontal** :
  — Date précédente : `st[T3][1] = 0`
  — Durée précédente : 2
  — `st[T3][2] = 2`
  — **Prédécesseur** : `pere[T3][2] = (T3, 1)`
— **Machine Assignée** : M1
— **Disponibilité de la Machine** : M1 est occupée par T2 jusqu'à 7.
— **Ajustement de la Date** : `st[T3][2] = 7`
— **Mise à Jour** : `nmach[M1] = (T3, 2)`

4.3 Résultat Final

Machine	Tâche	Occurrence	Date de Début
M1	T1	1	0
M1	T2	1	4
M1	T3	2	7
M2	T3	1	0
M2	T1	2	4
M2	T2	2	7

TABLE 2 – Dates de Début des Tâches Planifiées

5 Conclusion

Cet algorithme permet de planifier efficacement des tâches sur plusieurs machines en tenant compte des dépendances et des contraintes de séquencement. L'exemple illustratif

démontre comment les dates de début sont ajustées pour éviter les conflits et optimiser l'utilisation des machines. Pour une application pratique, il est essentiel de valider l'algorithme avec divers scénarios et d'apporter des ajustements si nécessaire.

6 Annexe : Code C++

Le code C++ ci-dessous implémente l'algorithme de planification des tâches décrit précédemment.