# Codage d'un algorithme Tabou pour le jobshop.

*Le but de ce travail est de coder un Tabou et de l'appliquer à problème de type JobShop. L'énoncé orignal de ce travail appartient au professeur Alain Hertz.*

---

On suppose que n pièces $P_1,\ldots,P_n$ doivent être usinées. Chaque pièce $P_i$ doit passer une et une seule fois sur chacune des m machines $M_1,\ldots,M_m$, dans un ordre défini par les gammes opératoires. Le temps de traitement de la pièce $P_i$ sur la machine $M_j$ est noté $t_{ij}$

Pour coder une solution, on considère m permutations des nombres 1,…,n. La $i^{ème}$ permutation correspond à l'ordre de passage des n pièces sur la machine $M_i$. Ainsi, par exemple, pour un job shop à deux machines et 3 pièces, la solution suivante :

$$\begin{matrix} 1 & 3 & 2 \\ 2 & 1 & 3 \end{matrix}$$

signifie que les pièces passent dans l'ordre $P_1$, $P_3$, $P_2$ sur $M_1$, et dans l'ordre $P_2$, $P_1$, $P_3$ sur $M_2$.

(1) Programmer une procédure GRAPHE(s) qui construit le graphe décrit ci-dessous, associé à une solution s du job shop. Vous pouvez par exemple mémoriser un graphe à l'aide d'une matrice d'adjacence A telle que

$$a_{ij} = \begin{cases} 1 & \text{s'il existe un arc du sommet } i \text{ vers le sommet } j \\ 0 & \text{sinon} \end{cases}$$

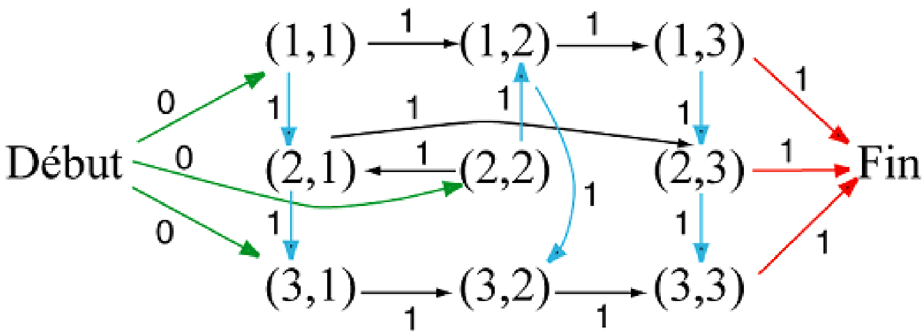
- Les sommets du graphes sont toutes les paires (i,j) avec ($1\leq i\leq n$, $1\leq j\leq m$) ainsi que 2 sommets supplémentaires notés Début et Fin.
- On met un arc de longueur 0 entre Début et chaque sommet (i,j) tel que $M_j$ est la $1^{ère}$ machine sur laquelle $P_i$ doit être usinée (arcs verts).
- On met un arc de longueur $t_{ij}$ de (i,j) vers Fin si $M_j$ est la dernière machine sur laquelle $P_i$ doit être usinée (arcs rouges).
- On met un arc de longueur $t_{ij}$ de (i,j) vers (i,k) si la machine $M_k$ suit immédiatement la machine $M_j$ dans la gamme opératoire de la pièce $P_i$ (arcs noirs).
- On met un arc de longueur $t_{ij}$ de (i,j) vers (k,j) si $P_k$ suit immédiatement $P_i$ sur la machine $M_j$ (seuls ces arcs sont donc dépendants de la solution s) (arcs bleus).

<u>Exemple</u> n=3, m=3,
on suppose que tous les temps d'usinage sont unitaires ($t_{ij}$=1)

Gamme opératoire de $P_1$ : $M_1$ $M_2$ $M_3$
Gamme opératoire de $P_2$ : $M_2$ $M_1$ $M_3$
Gamme opératoire de $P_3$ : $M_1$ $M_2$ $M_3$

Solution s représentée ci dessous :

$$\begin{matrix} 1 & 2 & 3 \\ 2 & 1 & 3 \\ 1 & 2 & 3 \end{matrix}$$




(2) Programmer une procédure PLC(G) qui détermine la longueur du plus long chemin de « Début » à tous les autres sommets du graphe, sachant que G n'a pas de circuit. Pour ce faire, on peut utiliser l'algorithme suivant :

1. Poser S :={Début} plc(Début) :=0 et rang :=0;
2. **tant que** S≠V **faire** (où V est l'ensemble des sommets de G)
   poser rang :=rang+1;
   soit W l'ensemble des sommets du graphe induit par V-S qui n'ont pas de prédécesseur;
   poser r(v)=rang pour tout v dans W et rajouter W à S;
3. **pour** k=1 **à** rang **faire**
   **pour tout** v tel que r(v)=k **faire** plc(v):= $\max\limits_{w\in N^{-}(v)} \{\pi(w)+d_{wv}\}$ où $N^{-}(v)$ est l'ensemble des prédécesseurs de v et $d_{wv}$ est la longueur de l'arc reliant w à v;

La procédure devrait par exemple retourner la valeur plc(2,1)=1 pour le graphe G ci-dessus

(3) Programmer une procédure AUPLUSTARD(s) qui détermine l'heure de démarrage au plus tard de chaque tâche dans la solution s donnée en input. Pour ce faire, il suffit de déterminer la longueur plc(Fin) du plus long chemin de « Début » à « Fin » et de programmer une procédure similaire à celle de la 2ème partie :

1. Poser S :={Fin} auplustard(Fin) :=plc(Fin) et rang :=0;
2. **tant que** S≠V **faire**
   poser rang :=rang+1;
   soit W l'ensemble des sommets du graphe induit par V-S qui n'ont pas de successeur;
   poser r(v)=rang pour tout v dans W et rajouter W à S;
3. **pour** k=1 **à** rang **faire**
   **pour tout** v tel que r(v)=k **faire** auplustard(v):= $\min\limits_{w\in N^{+}(v)} \{\pi(w)-d_{vw}\}$;

La procédure devrait par exemple retourner la valeur auplustard (2,1)=2 pour le graphe G ci-dessus

(4) Programmer une procédure CRITIQUE(s) qui détermine l'ensemble des tâches critiques dans s. Ceci se fait aisément en utilisant les procédures PLC et AUPLUSTARD.

(5) Programmer une procédure CRITIQUESucc(i) qui détermine l'ensemble $N_i$ des paires (x,y) tel que $P_x$ et $P_y$ sont consécutifs sur $M_i$ et l'usinage de $P_x$ et de $P_y$ sur $M_i$ sont deux tâches critiques. Ceci se fait aisément en utilisant la procédure CRITIQUE.

(6) Programmer une procédure PERMUTER(s,x,y) qui construit le graphe associé à la solution obtenue en permutant les tâches x et y dans s.

(7) Programmer une procédure INITIALISATION qui construit une solution initiale dans laquelle les ordres sur chaque machine sont 1,…,n.

(8) Programmer l'algorithme de Recherche Tabou décrit ci-dessous

1. Soit s la solution donnée en output par INITIALISATION;
   calculer le makespan F de s (à l'aide de GRAPHE et PLC);
   poser s* := s; F* :=F; compteur :=0; itération :=0;
   poser tabou(x,y)=0 pour toute paire (x,y) de tâches;

2. **tant que** compteur < 100 **faire**
   compteur :=compteur+1; itération :=itération+1;
   déterminer $N=N_1 \cup \ldots \cup N_m$ (cet ensemble étant obtenu en appliquant m fois la procédure CRITIQUESucc(i) avec i=1,…,m); poser Best := infini;

   **pour toute** paire (x,y) dans N **faire**
   en utilisant la procédure PERMUTER, construire le graphe G' associé à la solution s' obtenue en permutant les tâches x et y dans s;
   calculer le makespan F de s' (en utilisant la procédure PLC dans G');
   **si** F<Best **et** (tabou(x,y)<itération ou F< F*) **alors**
   Best :=F; xBest :=x; yBest :=y;

   modifier s en permutant xBest et yBest;
   poser tabou(xBest,yBest):=itération+10 et tabou(yBest,xBest):=itération+10;
   **si** Best < F* **alors**
   s* := s; F* :=Best et compteur :=0;

(9) Appliquer votre algorithme à l'exemple ci-dessus. Zipper le tout dans une archive « JobShop_votrenom.zip » et déposer la dans le site du cours au plus tard le dimanche 22 novembre 2020, minuit.