Projet RP : Problème d'équilibage de charge : glouton et recherche locale

A faire en binôme.

Date limite pour la remise du rapport : 9 mai 2023.

Considérons le problème d'équilibrage de charge où nous avons n tâches et m machines $M_1, M_2, \dots M_m$. Nous devons affecter chaque tâche à une machine. On s'intéresse donc à une partition de l'ensemble des tâches spécifiant pour chaque tâche sur quelle machine elle est affectée. La charge de la i-ième tâche est notée par t_i. La charge d'une machine est la somme des charges de tâches qui sont affectées sur cette machine. Le "*makespan*" d'une affectation des tâches aux machines est la charge maximale :

$$\max_{M_i} \sum_{j\text{: tâche affectée à } M_i} t_j.$$

Notre objectif est de déterminer une affectation des tâches aux machines minimisant le makespan. Ce problème étant un problème NP-difficile, on va s'intéresser à des méthodes approchées retournant des solutions sous-optimales. On va étudier, à l'aide de simulations la qualité de la solution retournée par différentes méthodes ainsi que le temps de calcul pour déterminer cette solution.

1 Algorithmes gloutons

Une première méthode est l'algorithme glouton suivant :

Algorithme de Liste

1: Former une liste de priorité en ordonnant les tâches arbitrairement.
2: A chaque fois qu'une machine devient disponible, on lui affecte la première tâche de la liste de priorité non encore traitée.

L'algorithme LPT (Largest Processing Time first) est une spécification de l'algorithme de Liste qui utilise comme liste de priorité la liste de tâches dans l'ordre non-croissant de leurs charges.

2 Recherche locale

Nous considérons par la suite un algorithme de recherche locale qui fonctionne de la manière suivante : on commence par une affectation arbitraire des tâches aux machines, puis on essaie de façon répétée d'appliquer un mouvement de recherche locale en considérant trois types de voisinage différents :

Insertion : Dans le voisinage "insertion", à partir d'un ordonnancement S, on obtient une solution voisine en déplaçant une tâche de la machine à laquelle elle est affectée, M_i, dans S à une autre machine M_j avec $i \neq j$.

Echange : Dans le voisinage "échange", des voisins d'une solution sont générés en échangeant deux tâches affectées sur des machines différentes.

Permutation : Dans le voisinage "permutation", on passe d'une solution S à une solution voisine S' de la manière suivante : Soit $A(i)$ et $A(j)$ les tâches affectées aux machines M_i et M_j, respectivement. Pour effectuer une permutation sur M_i et M_j, on choisit des sous-ensembles de travaux $B(i) \subseteq A(i)$ et $B(j) \subseteq A(j)$, et on "échange" ces tâches entre les deux machines. En d'autres termes, on met à jour $A(i)$ pour qu'il devienne $A(i) \cup B(j) - B(i)$, et $A(j)$ pour qu'il devienne $A(j) \cup B(i) - B(j)$. (On est autorisé à avoir $B(i) = A(i)$, ou à avoir $B(i)$ comme ensemble vide ; et de façon analogue pour $B(j)$).

Pour décider si un mouvement est améliorant, nous allons utilisé trois critères:

Cmax : Un mouvement est améliorant si le makespan est amélioré.

Lexico : Pour déterminer si un mouvement est améliorant on introduit un ordre lexicographique pour comparer deux solutions voisines. On appele une *machine critique*, une machine dont la charge n'est pas plus petite que le makespan. On utilise une fonction coût qui retourne deux valeurs, le makespan et le nombre de machines critiques. On dit qu'un ordonnancement S a un coût plus faible qu'un ordonnancement S' si le coût de S est plus faible lexicographiquement que S' où le makespan est le premier objectif et le nombre de machines critiques le deuxième.

Pair : Considérons un mouvement impliquant les machines M_i et M_j. Supposons que les charges sur M_i et M_j avant la permutation soient T_i et T_j, respectivement, et que les charges après la permutation soient T'_i et T'_j. Nous disons que le mouvement de permutation est améliorant si $\max(T'_i, T'_j) < \max(T_i, T_j)$ - en d'autres termes, la plus grande des deux charges impliquées a strictement diminué.

En combinant le type de voisinage et le critère utilisé pour décider si un mouvement est améliorant, on arrive à 9 variantes de l'heuristique de recherche locale.

Nous dirons qu'une affectation de tâches aux machines est stable s'il n'existe pas de mouvement améliorant, à partir de l'affectation courante. Ainsi, l'heuristique de recherche locale continue simplement à exécuter des mouvements améliorants jusqu'à ce qu'une affectation stable soit atteinte ; à ce stade, l'affectation stable résultante est renvoyée comme solution.

3 Travail à effectuer

1. Coder l'algorithme de liste.
2. Coder l'algorithme LPT.
2. Coder l'heuristique de recherche locale pour toutes les combinaisons de types de voisinage et de critère.
3. Générer des instances aléatoires pour le problème d'équilibrage de charge.
4. Comparer de manière systématique les différentes méthodes en terme de qualité de solutions retournées et de durée d'exécution de l'algorithmes.
5. Commenter les résultats obtenus.