Distanciel Min-Makespan

Exercice

1. Commençons par ordonner les tâches dans l'ordre décroissant de leur durée :

d1'=7 ; d2'=6; d3'=6 ; d4'=5 ; d5'=3 ; d6'=3 ; d7'=2 ; d8'=2 ; d9'=2 ; d10'=2 ; d11'=1

M3	6		3		2		2	
M2	6			5			2	
M1		7		3		2		1

Le temps de fin de la dernière tâche est 13.

2. Dans le cas présent l'optimal est de 13. En effet la somme des durées des tâches est égale à 39 et nous possédons 3 machines. Dans le meilleur cas, ces 3 machines sont toutes occupées à chaque instant, la durée totale est donc égale à 39/3 = 13.

Ici, nous obtenons le même résultat avec la méthode LPT, le ratio est donc de 1.

3. Plaçons nous dans le cas où il ne reste qu'une tâche dj' à effectuer. Il existe alors 2 scénarios :

- Les machines finissent toutes leur tâche en même temps, soit au temps $\frac{\sum_{k\neq j} d'_k}{m}$. La tâche dj' sera donc affectée à la machine d'indice le plus faible, et le temps total d'exécution sera donc égal à $\frac{\sum_{k\neq j} d'_k}{m}+d'_j$.
- Au moins une des machines finit plus tôt que les autres. Dans ce cas, elle est libre à un instant $t<\frac{\sum_{k\neq j} d'_k}{m}$, sinon la somme des durées des tâches ne pourrait pas être respectée. Ainsi, c'est à cette machine que va être attribuée la tâche dj', le temps total d'éxecution sera donc inférieur à $\frac{\sum_{k\neq j} d'_k}{m}+d'_j$.

4. Si n≤m, alors chaque tâche est affectée à une machine unique au début de l'exécution. Le temps d'exécution total sera donc égal à la durée de la tâche la plus longue, autrement dit d1'.

5. Si $n\geq m+1$, alors on affecte 1 tâche à chaque machine, après quoi il reste au moins 1 tâche. Ainsi on est obligé d'attribuer la ou les taches restantes dans la machine avec la plus petite valeur. Donc dans le meilleur des cas on aura $d'_m+d'_{(m+1)}$ avec $d'_m\geq d'_{(m+1)}$ donc $d'_m+d'_{(m+1)}\geq 2*d'_{(m+1)}$, donc ici égal si et seulement si $d'_m=d'_{(m+1)}$. On obtient donc ici $T_{opt}(I)\geq 2d'_{(m+1)}$.

6. Si $j \geq m+1$ on est dans le cas des deux questions précédentes, c'est-à-dire $n \geq m+1$ avec $T_{opt}(I) \geq 2d'_{(m+1)}$ ainsi $T_{LPT}(I) \geq T_{opt}(I)$. Avec $T_{LPT}(I) = T_{opt}(I)$ on doit s'assurer que soit $n \leq m$ ou bien que $n = m+1$ et $d'_m = d'_{(m+1)}$.

7.

8.

9. On peut trouver pour chacune de ces instances une affectation telle que toutes les machines finissent au même instant, ce qui est le meilleur des cas. Pour cela, il suffit d'affecter à une machine les 3 tâches de durée m. Pour les machines restantes, on peut trouver un couple de tâche dont la somme est égale à 3m. Ainsi, $T_{opt}(I_m) = 3*m$.

Instance I_5 :

M5	5	5	5
M4	8	7	
M3	8	7	
M2	9	6	
M1	9	6	

10. Prenons une exécution de LPT. Dans un premier temps, les m premières tâches vont être affectées au machine, de telle sorte que les machines d'indice plus petit auront les tâches les plus longues. Une fois cela fait, les premières machines à terminer vont donc être celles d'indice le plus élevé. De cette manière, les m tâches suivantes seront réparties en allant de l'indice le plus élevé au moins élevé. Ici, chaque machine possédera un couple de tâche dont la somme des durées sera égale à la moyenne de la durée des tâches effectuées jusqu'ici, de par la construction de l'instance. Enfin, il ne restera que la dernière tâche à affecter, ce qui nous donne $T_{LPT}(I_m) = \frac{\left(\sum_{i=m}^{(m+m)-1} i*2\right)}{m} + m$.

Instance I_5 :

M5	7	7	
M4	8	6	
M3	8	6	
M2	9	5	
M1	9	5	5

11. Le ratio tend vers 2 quand m devient très grand. La borne inférieure est donc de 2.

12. Dans le cas de cette instance, l'algorithme LSA est exécuté sur un tableau de durées trié dans l'ordre décroissant. Pour commencer, les m premières tâches vont être affectées aux machines, de telle sorte que les machines d'indice le plus faible vont hériter des tâches les plus courtes. Pour les m tâches suivantes, le même processus est répété. Enfin la dernière tâche sera affectée) la machine

M1, qui aura eu la tâche la plus courte à chaque affectation. Ainsi, la durée totale d'exécution sera équivalente au temps que met M1 à finir, soit la somme de la durée des tâches 1, m+1 et 2m+1. On obtient ainsi $T_{LSA}(I_m)=m+\lfloor\frac{(m+m-1)}{2}\rfloor+(m+m-1)$.

Instance I_5 :

M5	6	9	
M4	6	8	
M3	5	8	
M2	5	7	
M1	5	7	9

13.

$$\frac{T_{LSA}(I_m)}{T_{opt}(I_m)}=\frac{\frac{2m+(m+m-1)+2(m+m-1)}{2}}{3*m}$$

$$\frac{T_{LSA}(I_m)}{T_{opt}(I_m)}=\frac{\frac{2m+3(m+m-1)}{2}}{3*m}$$

$$\frac{T_{LSA}(I_m)}{T_{opt}(I_m)}=\frac{\frac{2m+3m+3m-3}{2}}{3*m}$$

$$\frac{T_{LSA}(I_m)}{T_{opt}(I_m)}=\frac{\frac{8m-3}{2}}{3*m}$$

$$\frac{T_{LSA}(I_m)}{T_{opt}(I_m)}=\frac{4m-\frac{3}{2}}{3m}$$

Cela tend donc vers 4/3 en + l'infini.

Programmation

Organisation :

Nous avons choisi de programmer cette application en Java car il s'agit du langage que nous avons le plus utilisé au cours de notre cursus. De plus, un projet de L3 nous a permis de maîtriser JavaFX, un framework permettant de créer aisément une interface, ce qui se révèle tout à fait pertinent ici.

Nous avons donc représenté une Instance comme expliqué dans l'énoncé, à l'aide d'un tableau D stockant la durée de chaque tâche et un tableau M de Machine. Une Machine contient les durée des tâches qu'elle a effectuée auparavant et la somme de celles-ci.

Dans le cas de l'utilisation de la dernière méthode de saisie d'instance, l'application gère alors plusieurs Instances simultanément et propose un affichage des résultats adéquat.

Il existe 2 fichiers .txt permettant de tester l'importation d'une instance depuis un fichier.

Complexité :

En ce qui concerne la génération d'instances avec I_m, cela s'effectue en O(m), il suffit de remplir un tableau de 2m+1 valeurs.

Pour ce qui est du mode I_R, la génération se fait en O(n*k), il faut remplir un tableau de n valeurs aléatoires et répéter ce processus k fois.

La copie des instances à l'aide des deux modes restants s'effectue en O(n), il faut parcourir toute la chaîne de caractère pour pouvoir obtenir les durées correctes des tâches.

MyAlgo est un algorithme qui calcule tout d'abord la borne inférieure moyenne et s'en sert pour remplir les machines. En effet, MyAlgo va essayer de remplir complètement une machine jusqu'à cette borne inférieure moyenne et cela pour toutes les machines. A la fin, s'il reste des tâches, celles-ci sont attribuées comme dans les deux algorithmes précédents. De plus, il est important de préciser que les tâches sont ordonnées de façon décroissante comme dans LPT. Une autre idée aurait été de réaliser des combinaisons des tâches pour arriver à cette borne mais cette solution n'est plus un algorithme d'approximation mais bien un algorithme coûteux permettant d'obtenir à coup sûr la meilleure solution.

LSA sera au mieux aussi bien que les autres algorithmes si les tâches sont de base dans l'ordre décroissant.

LPT sera en moyenne plus efficace que les deux autres algorithmes. Pour les instances I_m LPT sera le plus efficace.

MyAlgo sera aussi efficace que LPT dans les cas où son ratio est de 1. Cela grâce à sa similitude avec celui-ci dans le sens où il va remplir les machines comme indiqué ci-dessus puis finir par exécuter LPT s'il lui reste des tâches.