

Allocation de tâches

À des machines avec critère égalitarien



Introduction

Centre de données traitant des tâches

Problème fondamental de l'ordonnancement, problème NP-difficile

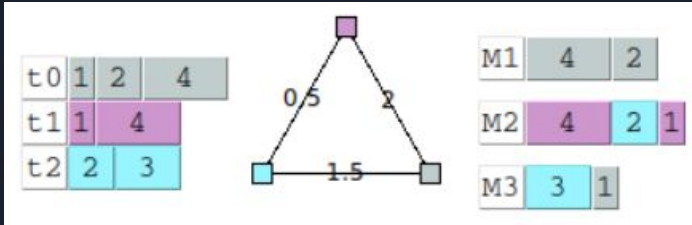
Machines, tâches, types, compatibilité

Critère égalitarien

Notations et Définitions:

- MCoeff: La matrice des coefficients de compatibilité entre les types

$$MCoeff = M\alpha_{i,j}, (i,j) \in types$$



```
MCoeff = np.array([[1, 2, 1.5],  
                  [2, 1, 0.5],  
                  [1.5, 0.5, 1]])
```

- Évaluation de coût

$$makespan = \max_{mach \in LM} \sum_{i \in mach} p_i$$

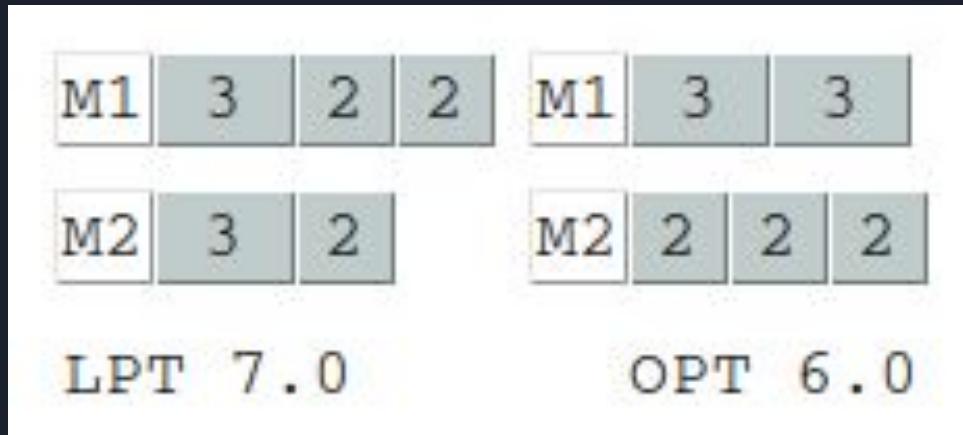
$$coutMax = \max_{mach \in LM} \sum_{i,j \in mach} p_i \times \alpha_{i,j}$$

Max(4+2*0.5+1 , 2+4*0.5+1*0.5)

LPT (Longest Processing Time)

ListeTache triée en ordre décroissant selon la taille des tâches

Alloue LT[0] sur la machine la moins chargée

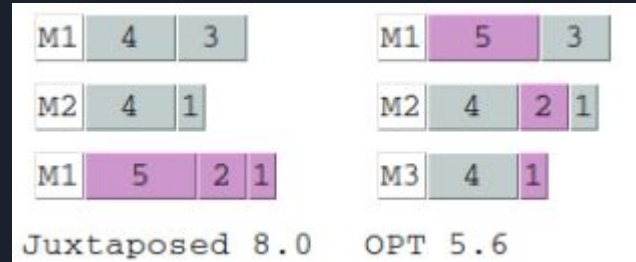
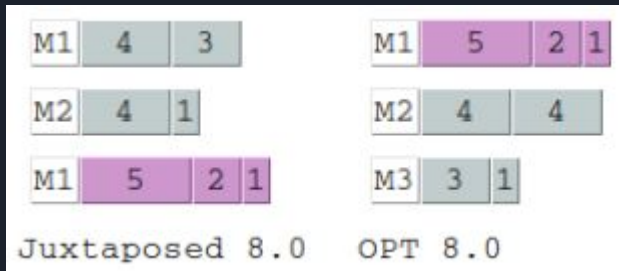



$O(NbM \cdot NbT)$

1 seul type

Juxtaposed

- Lorsqu'il n'y a que deux types
- Séparer les tâches de type 1 et type 2
- Allouer les tâches de type 1 à un nombre de machines i compris entre 1 et m , type 2 à $m-i$
- Fait appel à un algorithme ALLOC
- Complexité: $O(m * \text{ALLOC})$





Lower-Bound

$$LB = \max(p_{max}, \frac{\sum_{i=1,n} p_i}{m})$$

$$LB \leq MakespanOpt \leq 2 * LB$$

$$LB_{type} = \max(p_{max}, \max_{t \in types} \frac{\sum_{task_de_type=t} p_{task}}{m})$$

$$LB_{type} < CoutMaxOpt < 2 * LB * max_{MCoeff}$$



Bin-packing

- Schéma d'approximation
- Chercher à minimiser le makespan t des machines(bins).
- Recherche dichotomique entre l'intervalle, epsilon
- Séparer et arrondir
- Construire une matrice pour obtenir le nombre minimum de bins nécessaires de taille t
- 3 cas
- Complexité: $O(n^k)$ entrées, le calcul de chaque entrée est $O(n^k)$ $\rightarrow O(n^{2k})$

Bin-packing

Exemple: $2 * 60$ $3 * 40$

nb60 \ nb40	0	1	2	3
0	1	1: 40	1: 40	1: 40 40 2: 40
1	1: 60	1: 60 40	1: 40 40 2: 60	1: 40 40 2: 60 40
2	1: 60 2: 60	1: 60 40 2: 60	1: 60 40 2: 60 40	1: 60 40 2: 60 40 3: 40

(nb60, nb40)
(1 , 0)
(0 , 1)
(0 , 2)
(1 , 1)



PTAS

Polynomial-Time Approximation Scheme

Alloue optimalement les k plus grandes tâches

$O(NbM^k)$

Alloue les tâches restantes à l'aide d'un autre algorithme



LPT_typed

Inspiré de LPT

Prend en compte les types

1 allé test

Choisi la meilleure machine et l'alloue

$O(\text{NbTaches}^2 \times \text{NbMachines})$



Greedy_cluster

Regroupement par cluster

Allocation par cluster trié

Limite $t \rightarrow$ passage machine suivante

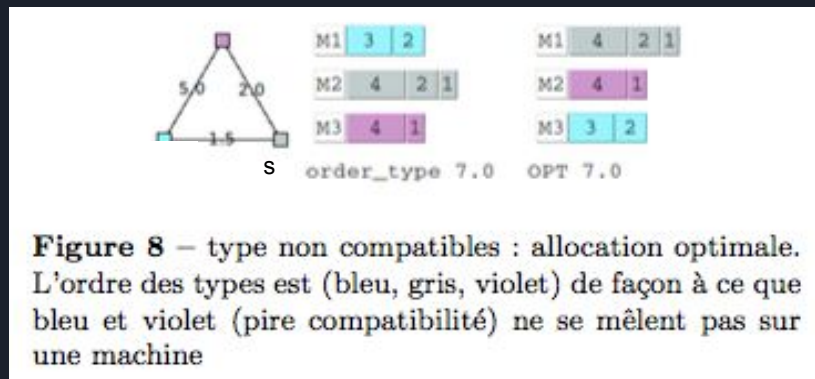
Recherche dichotomique du meilleur t

$O(N_{types} \times NbT^2)$

Variante tâches restantes : `with_lpt_typed`

Order-type

- Établir un ordre de type
- Manière 1 :Optimal. Tester toutes les permutations de types, et choisir celle qui minimise le produit des coefficients partagés par les types côte à côte.
- Manière 2: Glouton. Choisir le type suivant qui partage le coefficient le plus faible avec le type précédent.

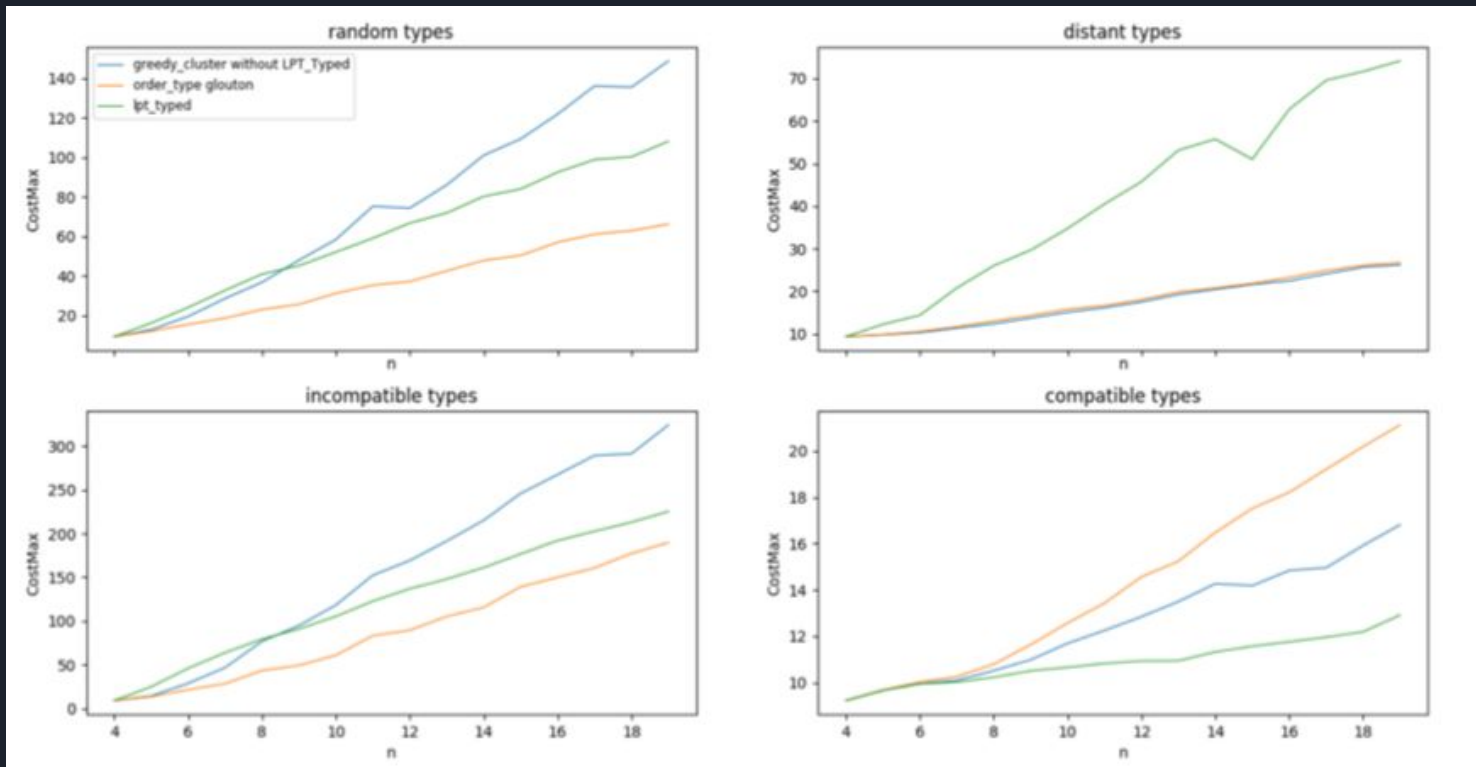


Comparaisons

Comparaison des algorithmes de regroupement des types pour plus de 2 types

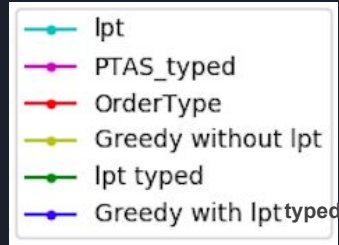
Max cost
VS
nbTasks

10 machines
maxT=10

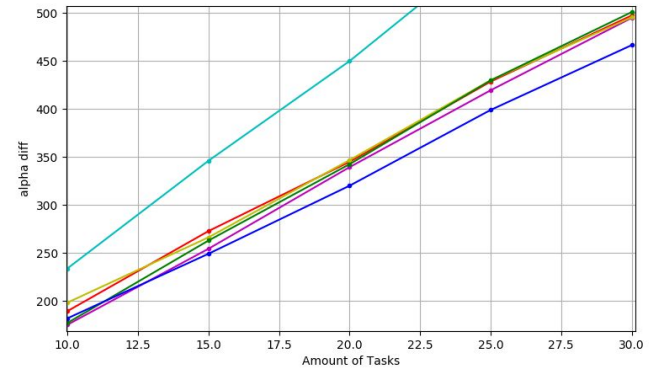
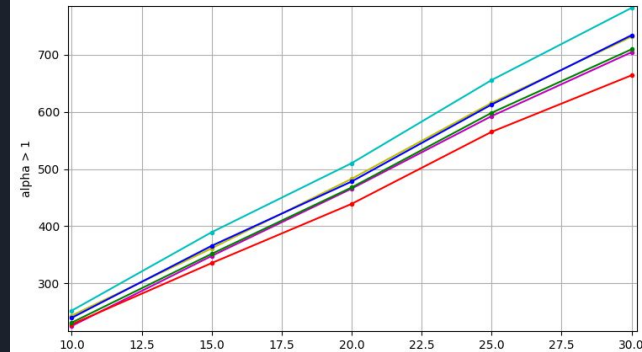
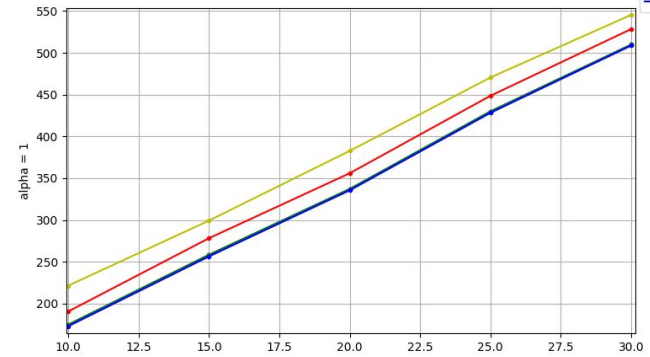
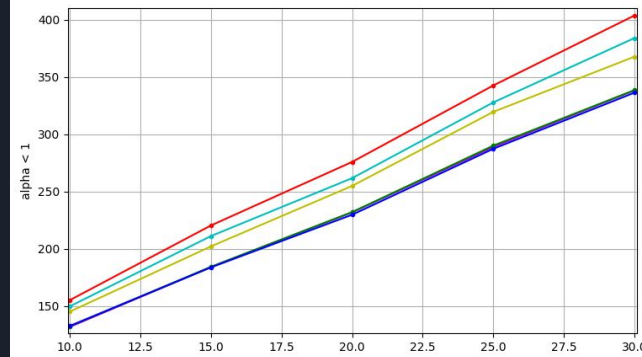


Comparaisons

Comparaison des algorithmes pour plus de 2 types



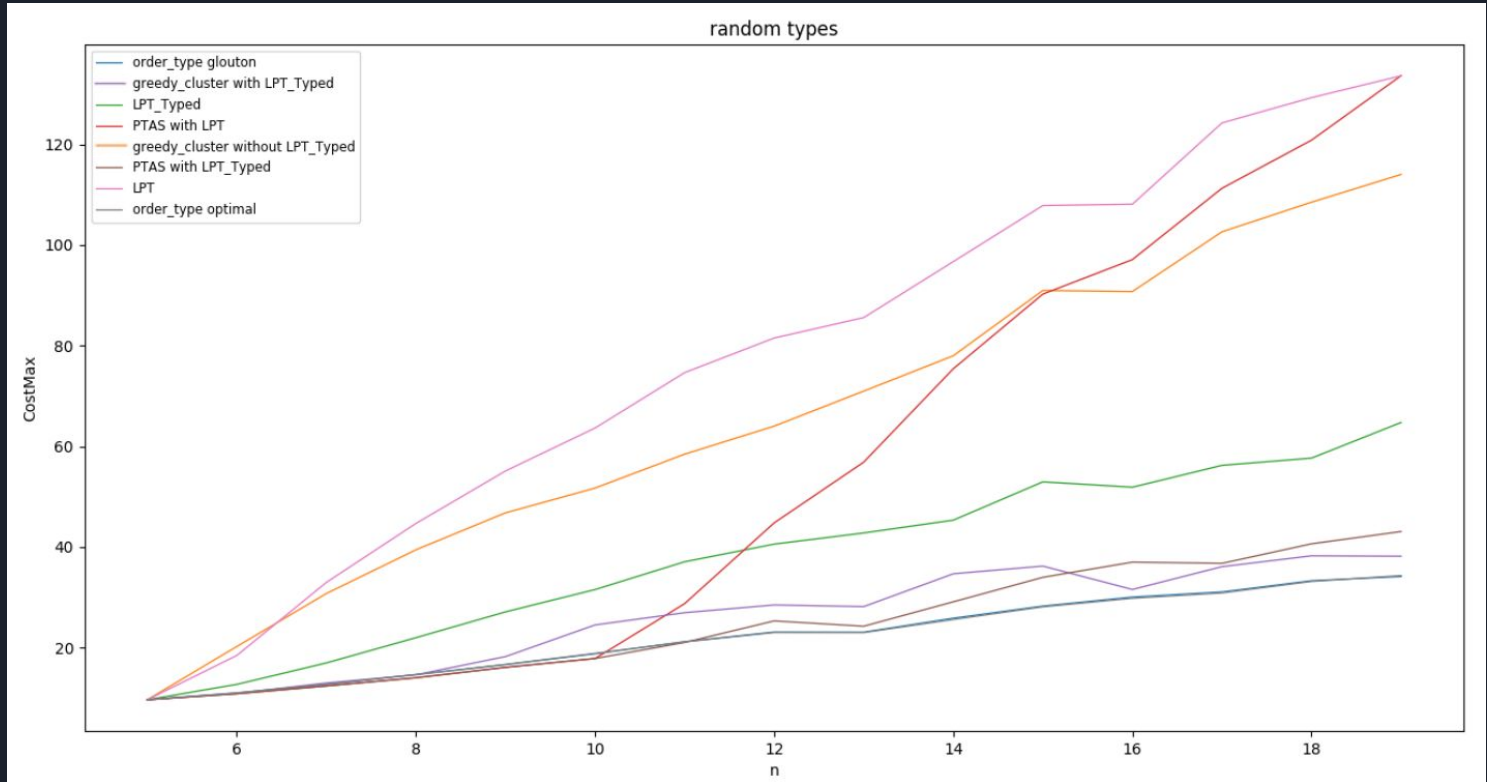
Max cost
VS
nbTasks
(3 machines)



Comparaisons

Comparaison des algorithmes pour plus de 2 types

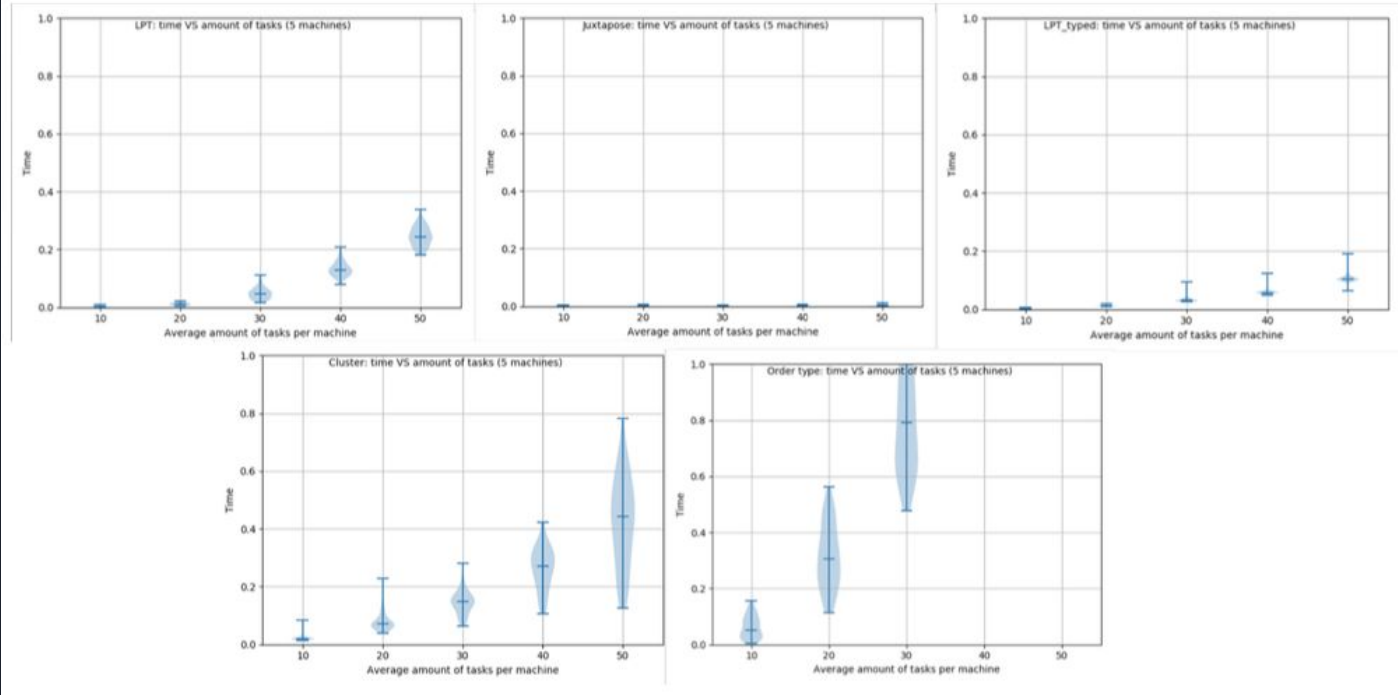
Max cost
VS
nbTasks
Random types



Comparaisons

Comparaison des algorithmes pour plus de 2 types

Figure 14 – Temps moyen d'exécution des algorithmes en fonction du nombre de tâches



Time(s)
VS
nbTypes



Conclusion

Heuristiques

Allocation optimal , coût supérieur

Critère utilitarien