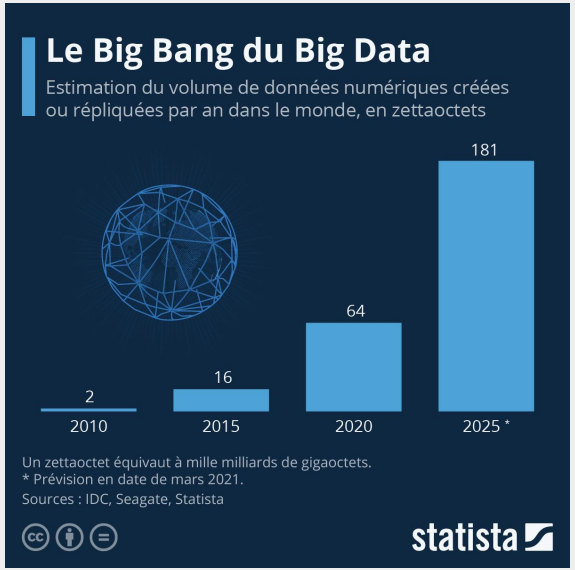
OPTIMISATION FOG COMPUTING LEVY FLIGHT MONARCH

Rachel Roux
Octave Cinquanta
ING3 IA Gr A

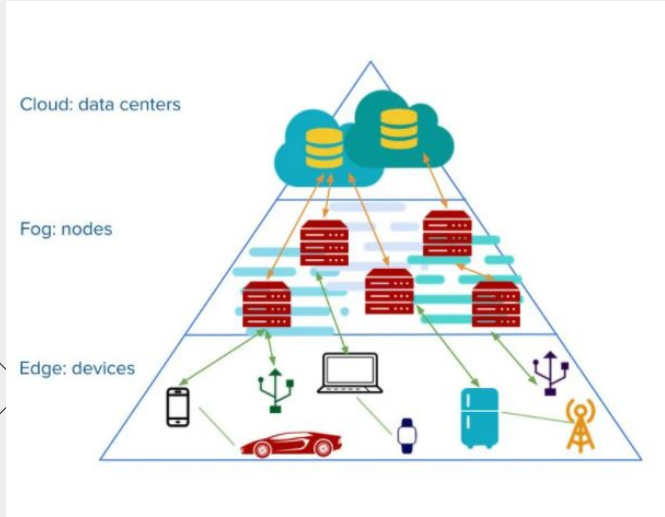
TABLE DES MATIÈRES

INTRODUCTION

BIG DATA -> UTILISATION MASSIVE DES CLOUDS -> CONSOMMATION IMPORTANTE D'ÉNERGIE ET DÉLAI TOUJOURS PLUS LONG



CRÉATION DU FOG COMPUTING POUR RÉPARTIR LE TRAITEMENT DES TÂCHES SUR TROIS NIVEAUX



PROBLÈME D'OPTIMISATION

- Basé sur le papier : "Multi-objective optimization for task offloading based on network calculus in fog environments", Qian Ren, Kui Liu, Lianming Zhang, 2022.
- Cherche à assigner au mieux les tâches au Fog Node
- Première formulation mathématiques:

$$
\begin{cases}
\min\limits_{X_i} & \sum\limits_{i=1}^{N} X_i(W_1T_{1,i} + W_2T_{2,i} + W_3T_{3,i}) \\
s.t. & \sum\limits_{i=1}^{N} Xi - M = 0 \\
& W_j \in [0,1], \forall j \in \{1,2,3\} \\
& T_{j,i} \in [0,1], \forall j \in \{1,2,3\}, \forall i \in [\![1,N]\!] \\
& X_i \in \{0,1\}, \forall i \in [\![1,N]\!]
\end{cases}
$$

$$
T_{1,i} = \frac{P_i - P^{min}}{P^{max} - P^{min}} \qquad T_{2,i} = \frac{B_i}{B_{i,S}} \qquad T_{3,i} = \frac{D_i - D^{min}}{D^{max} - D^{min}}
$$

PROBLÈME D'OPTIMISATION

- Seconde formulation mathématiques: Retirer l'agrégation et modifier la fonction objectif du buffer size en hard constraint.

$$
\begin{cases}
\min\limits_{X_i} & \sum\limits_{i=1}^{N} X_i T_{1,i} \\
\min\limits_{X_i} & \sum\limits_{i=1}^{N} X_i T_{3,i} \\
s.t. & \sum\limits_{i=1}^{N} Xi - M = 0 \\
& \sum\limits_{i=1}^{N} T_{2,i} - 1 < 0 \\
& W_j \in [0,1], \forall j \in \{1,3\} \\
& T_{j,i} \in [0,1], \forall j \in \{1,2,3\}, \forall i \in [\![1,N]\!] \\
& X_i \in \{0,1\}, \forall i \in [\![1,N]\!]
\end{cases}
$$

PROBLÈME D'OPTIMISATION

- Troisième formulation mathématiques :
- Basé sur l'idée que les tâches ne sont pas identiques entre elle

[1 0 1 1 0 0 0] → [FN1 FN3 FN2 FN8 FN3 FN 4 FN1]

Table 1. Node-task table.

FN/ST	1	2	3
A	− 0.3/1	− 0.1/2	− 0.4/2
B	− 0.2/2	− 0.1/3	− 0.3/2
C	− 0.1/1	− 0.1/3	− 0.5/2

$$
\begin{cases}
\min\limits_{X_{i,j}} & \sum\limits_{i=1}^{N} X_{i,j} P_{i,j} \\
\min\limits_{X_{i,j}} & \sum\limits_{i=1}^{N} X_{i,j} D_{i,j} \\
s.t. & \sum\limits_{i=1}^{N} X_{i,j} - M = 0 \\
& \sum\limits_{i=1}^{N} \frac{B_{i,j}}{B_{i,j,S}} - 1 < 0 \\
& \sum\limits_{i=1}^{N} X_{i,j} = \sum\limits_{j=1}^{M} X_{i,j} = 1 \\
& X_i \in \{0,1\}, \forall i \in [\![1, N]\!]
\end{cases}
$$

DATASET TASK OFFLOADING

Cost_Table
Permet de simulation le power consumption

Execution_Table
Permet de simuler le Delay

Offre un dataset de la même forme que dans l'exemple donné et évite de devoir calculer chaque facteur, sachant que le délai est basé sur les courbes de service dans le calcul de réseau qui est assez compliqué.

FN/ST	1	2	3
A	− 0.3/1	− 0.1/2	− 0.4/2
B	− 0.2/2	− 0.1/3	− 0.3/2
C	− 0.1/1	− 0.1/3	− 0.5/2

LEVY FLIGHT

Cet algorithme s'inspire du comportement erratique des animaux lorsqu'ils cherchent de la nourriture. De la même manière, l’algorithme génère des petits pas et grand pas selon une gaussienne. Plus le pas est grand, plus la solution est modifiée.

LEVY FLIGHT

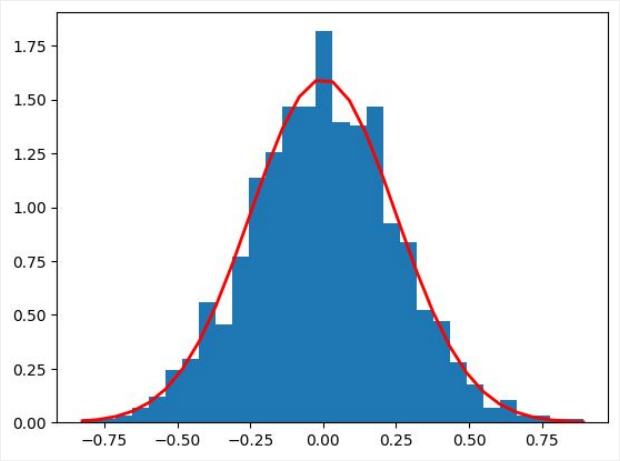
- Générer une solution aléatoire avec un tableau de N indice (pour N tâches) avec un FN attitré
- Générer un nombre aléatoire selon une gaussienne
  - si ce nombre est petit alors faire peu de modifications aléatoires
  - si ce nombre est grand alors faire beaucoups de modifications aléatoires à la fois
- Évaluer la nouvelle solution
- Si cette solution n'est pas dominé par les anciennes solutions, la rajouter au solution non dominé
- Si cette solution domine des solutions déjà enregistrée, retirer les anciennes solutions
- Critère d'arrêt : Nombre de solution non dominé suffisamment nombreuse pour créer le front pareto

https://knaidoo29.github.io/mistreedoc/levy_flight.html

LEVY FLIGHT

Paramètres :

- Loi normale de moyenne 0 et d'écart type σ 0.25
- Nombre p généré dessus:
  - Si |p| >= 3*σ (1% de chance) on change toute la solution
  - Si |p| >= 2*σ (5% de chance) on change pratiquement 75% de la solution
  - Si |p| >= σ (50% de chance) on change 50% de la solution
  - Sinon on change 25%



MONARCH

Cet algorithme s'inspire du comportement migratoire des papillons Monarques lorsqu'ils doivent s'orienter dans de nouveaux environnement en suivant certain individus qui ont de meilleurs probabilité de s'orienter. En imitant ce comportement, notre algorithme cherche à garder une mémoire à long terme sur les comportements à adopter.

MONARCH

- Créer N solutions possibles réuni dans une population et le diviser N dans deux sous populations NP1 et NP2
- Pour chaque élément de chaque solution sur NP1 :
    - Générer un nombre aléatoirement sur une loi uniforme et le comparer à une variable fixée
        - Si p est supérieur: copier un élément d'une autre solution de NP1 sur l'élément actuel
        - Si p est inférieur: copier un élément d'une autre solution de NP2 sur l'élément actuel
- Pour chaque élément de chaque solution de NP2:
    - Générer un nombre aléatoirement sur une loi uniforme et le comparer à une variable fixée
        - Si p est supérieur: copier l'élément de la meilleur solution de toute la population sur l'élément actuel
        - Si p est inférieur: copier un élément d'une autre solution de NP2 sur l'élément actuel
- Évaluer les nouvelles solutions
- Générer une nouvelle population avec toutes les solutions des rangs les moins dominés
- Critère d'arrêt : itération d'algorithme

Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization, A. Feng ,G. Wang, S. Deb, M. Lu, Xiang-Jun Zhao, 2015

Monarch

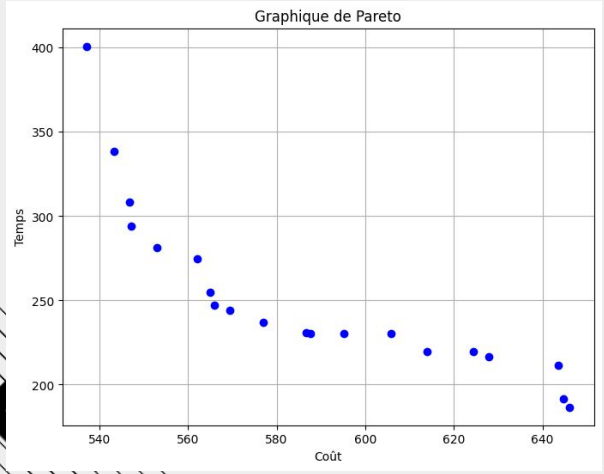
Paramètres :

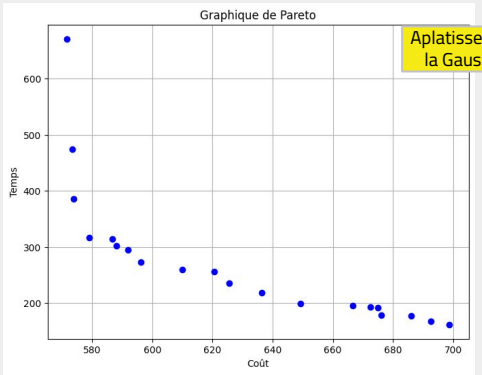
- Population totale à chaque passe: 20
- Sous population NP1 = NP2 = 10
- Paramètre p: seuil sous lequel doit être généré un nombre random de chaque génome du monarch pour copier un génome d'une des meilleures solutions
- Nombre d'itération de l'algorithme : varie entre 5 (minimum pour le pareto) et 25 (convergence trop importante)

Conclusion

Pour 40 tâches répartie sur 13 FN:

- Pour le monarque :
  - Meilleure Front pareto obtenue (15 itération, p = 0.25:
  - En 3.6 secondes
  - Les résultats restent hasardeux
- Pour le Levy Flight:
  - Bon Front pareto obtenue
  - En 2.5 secondes
  - Les résultats sont stables





Aplatissement de la Gaussienne