INTRODUCTION À LA RECHERCHE OPÉRATIONNELLE: PROBLÈMES MATHÉMATIQUES

BRICAUD PIERRE
LUCIEN MACE
GROSSMANN—LE MAUGUEN ARTHUR

CSI 3
2023-2024

SOMMAIRE

- PARTIE 1: Calcul d'utilité d'un sac à dos

→ 1A méthode algorithmique

→ 1B méthode heuristique

→ Comparaison des 2 algorithmes

- PARTIE 2 : Calcul du nombre de wagons nécessaire
- Résultats obtenus et temps d'exécution moyens
- Méthodes d'optimisation utilisées
- Sources

PARTIE 1A - MÉTHODE ALGORITHMIQUE DE CALCUL D'UTILITÉ D'UN SAC À DOS

Algorithme:

Algorithme Sac à dos (liste des objets, capacité max):

1. Initialiser un sac à dos vide.
2. Générer toutes les combinaisons possibles d'objets à partir de la liste d'objets.
3. Pour chaque combinaison de la liste :
    a) Vérifier si la somme des masses des objets de la combinaison est inférieure ou égale à la capacité maximale :
        i. Si oui, calculer l'utilité totale de la combinaison.
    b) Mettre à jour la combinaison optimale en fonction de l'utilité totale maximale trouvée jusqu'à présent.
4. Retourner la combinaison optimale de sac à dos avec les objets sélectionnés.

PARTIE 1B - MÉTHODE HEURISTIQUE DE CALCUL D'UTILITÉ D'UN SAC À DOS

Algorithme:

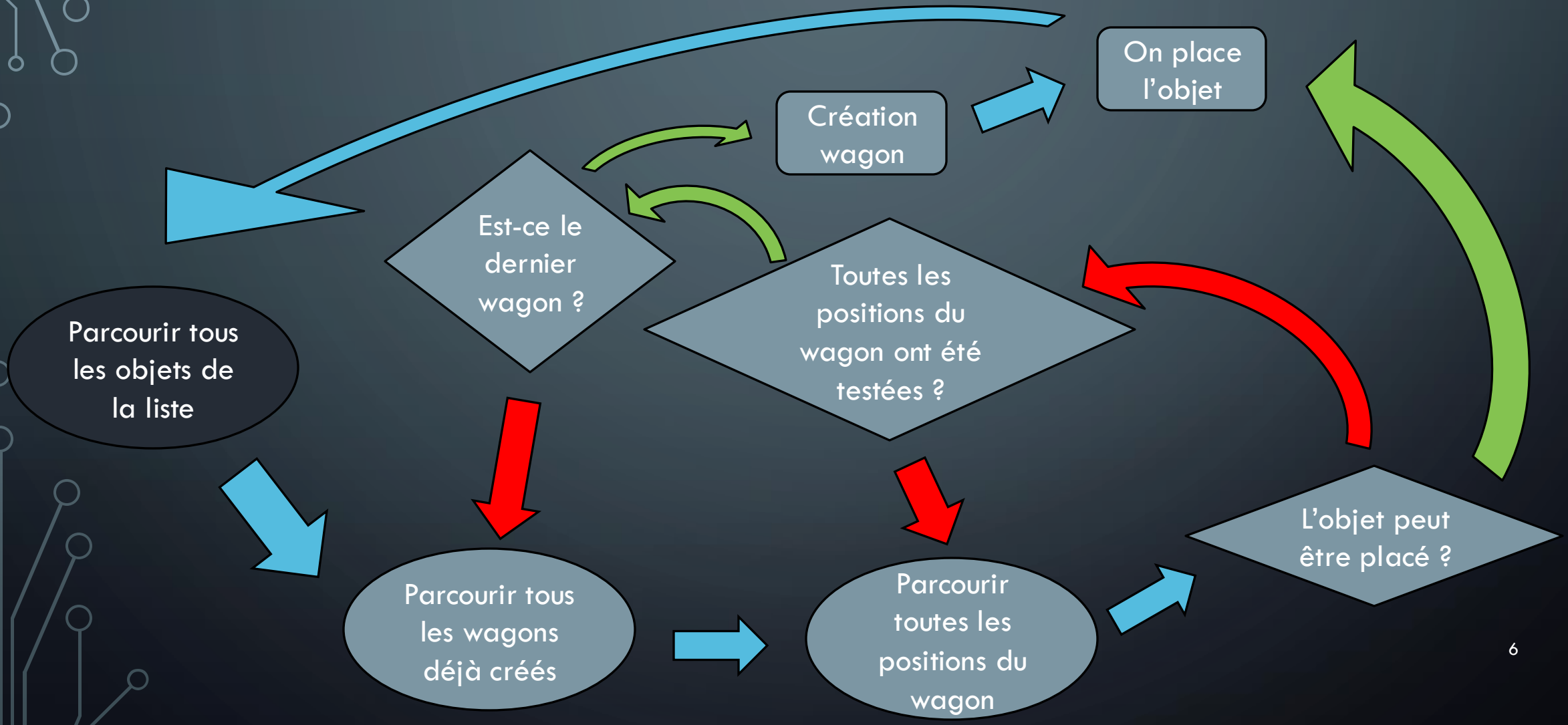
Algorithme Sac à dos (liste des objets, capacité max, ratio charge, profondeur, tri des extrêmes):

1. Initialiser un sac à dos vide.
2. On trie les objets par score = utilité/masse
3. On met les éléments ayant le plus de chiffres après la virgule et qui ne se combinent pas à la fin de la liste de tous les éléments
4. On met une partie des meilleurs dans le sac et on teste quelques-uns après afin de trouver la meilleure combinaison
5. On garde l'ensemble avec la meilleure utilité en dessous du poids maximum autorisé

COMPARAISON DES 2 ALGORITHMES

Algorithme	1A Algorithmique	1B Heuristique
Complexité	$O(n * 2^n)$	$O(n * n^{profondeur}) = O(n^{profondeur+1})$
Avantage(s)	Renvoie toujours le meilleur résultat	Renvoie un résultat plus rapidement et avec autant de précisons si les bons paramètres sont rentrés
Inconvénient(s)	Temps d'exécution long	Résultat peut-être moins précis si les mauvais paramètres sont rentrés

PARTIE 2 - MÉTHODE HEURISTIQUE DE CALCUL DU NOMBRE DE WAGONS NÉCESSAIRE



PARTIE 2 - MÉTHODE HEURISTIQUE DE CALCUL DU NOMBRE DE WAGONS NÉCESSAIRE

Complexité: $o(n^2)$

Avantages: Algorithme renvoie des résultats précis

Inconvénient: Algorithme long à exécuter

RÉSULTATS OBTENUS ET TEMPS D'EXÉCUTION MOYENS*

* (temps d'exécution mesurés 3 fois sur MSI Prestige 15 A10SC intel core I7 10th gen + 16gb RAM branché sur secteur)

Code	Partie 1 A	Partie 1 B (PARAM: sort extremes; charge ratio; max_charge; depth)	Partie 2 online D1	Partie 2 offline D1	Partie 2 online D2	Partie 2 offline D2	Partie 2 online D3	Partie 2 offline D3
Résultats/ score + paramètres si nécessaires	7,6 pour C=0,6 15,05 pour C=2,0 17,85 pour C=3,0 19,95 pour C=4,0 22,00 Pour C=5,0	7,6 pour (True; 1, 0.6; 0) 15,05 pour (True; 1; 2; 0) 17,85 pour (True; 0,8; 3; 3) 19,95 pour (True; 0,8; 4; 3) 22,00 pour (True; 0,8; 5; 4)	45	44	32	29	21	18
Temps d'exécution en secondes	21,63	0,001	0,001	0,001	5,504	8,991	480,5 (8min)	783,14 (13min)

MÉTHODES D'OPTIMISATION UTILISÉES

- Ne pas lire depuis le fichier Excel (utiliser la fonction get_data de data.py et non choose_read_excel de util.py)

→ GAIN: 0,4s en moyenne

- Arrondir moins souvent les résultats des calculs afin de perdre moins de temps

→ GAIN: 23s en moyenne

- Retirer les valeurs ayant trop de décimales et posant un problème pour obtenir le résultat le plus élevé

→GAIN: 19s en moyenne

- Possibilité de fermer un wagon quand il semble très compliqué d'y ajouter les éléments

→ GAIN: ??s en moyenne

- Ne pas tester un wagon si la place restante est inférieure au volume de l'objet que l'on essaie de rentrer

→ GAIN: ??s en moyenne

SOURCES

- https://fr.wikipedia.org/wiki/Probl%C3%A8me_de_bin_packing
- Le problème de bin-packing en deux-dimensions, le cas non-orienté : résolution approchée et bornes inférieures. Joseph El Hayek https://theses.hal.science/tel-00158728
- https://fr.wikipedia.org/wiki/Th%C3%A9orie_de_la_complexit%C3%A9_(informatique_th%C3%A9orique)
- https://fr.wikipedia.org/wiki/Classe_de_complexit%C3%A9