



ÉCOLE CENTRALE CASABLANCA
OPTIMISATION COMBINATOIRE MINI-PROJET

Ordonnancement d'un atelier robotisé

Achraf Mansouri et Sami Youssef

Année académique 2023-2024

l'action est la clé fondamentale de tout succès

Picasso

Introduction

Dans le domaine de la recherche opérationnelle, les problèmes de planification et d'ordonnancement occupent une place prépondérante, à l'intersection de la théorie des algorithmes et des applications industrielles. Le présent rapport, issu de nos travaux dans le cadre du module "Optimisation Combinatoire" lors du travail dirigé numéro 5, s'attaque à un cas spécifique de cette vaste thématique. Notre étude se concentre sur un tel problème, qui, se classe majoritairement dans la catégorie NP-Difficile, à l'exception de quelques cas particuliers.

Notre travail se porte sur un scénario représentative des défis inhérents à ce type de problèmes. Il s'articule autour d'un atelier robotisé, simulé par une configuration précise : un frigo initial contenant les éprouvettes à traiter, deux bains de traitement successifs, et un frigo final pour le dépôt des éprouvettes post-traitement. Ce système est géré par un robot unique, capable de manipuler une seule éprouvette à la fois et contraint de suivre des itinéraires définis pour ses déplacements entre les différents postes. La tâche consiste à traiter un nombre n d'éprouvettes (initialement $n=3$), chaque éprouvette devant passer un temps déterminé dans chaque bain.

Dans ce contexte, notre livrable se propose de dégager une séquence optimale pour le traitement des éprouvettes, dans le but de minimiser le temps total de traitement. Il s'articule autour de quatre axes principaux : la proposition d'une heuristique de construction, l'élaboration d'une méta-heuristique d'amélioration, la formulation mathématique et la résolution du problème via un solveur, et enfin, l'approche basée sur la théorie des graphes pour une représentation et une résolution alternatives du problème.

Chapitre 1

Méta-heuristique de construction

1.1 Description du problème

Dans le cadre de notre étude, nous examinons un atelier automatisé, dont le schéma est présenté dans la figure ci-dessous. Cet atelier se compose d'un réfrigérateur initial, qui abrite les éprouvettes nécessitant un traitement, et de deux bains de traitement distincts, suivis d'un réfrigérateur final pour le stockage des éprouvettes après leur traitement. Un unique robot, opérant au sein de cet atelier, est chargé de la manipulation des éprouvettes. Capable de gérer seulement une éprouvette à la fois, ce robot est contraint de suivre un parcours prédéterminé, tel qu'illustré dans la figure, pour se déplacer entre les réfrigérateurs et les bains de traitement.

Le processus de traitement est conçu pour un nombre n d'éprouvettes, débutant initialement avec $n=3$. Le protocole de traitement exige que chaque éprouvette soit immergée, d'abord dans le premier bain, pour une durée strictement comprise entre 12 et 16 secondes, puis dans le second bain, pour une durée s'étalant de 12 à 13 secondes.

La configuration spécifique de l'atelier permet d'accueillir les n éprouvettes tant dans les bains que dans les réfrigérateurs.

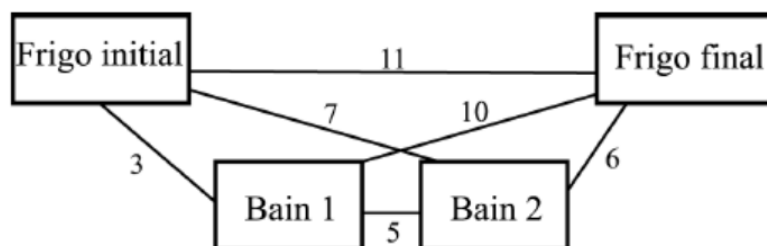


FIGURE 1.1 – Atelier robotisé. Extrait de [TD 5].

1.2 Heuristique

Dans les sous-sections suivantes, nous présentons trois solutions à ce problème. Cependant, les deux premières s'avèrent moins efficace, car elle ne prend pas en compte l'objectif principal de minimiser le temps de déplacement du robot. En revanche, la deuxième solution semble plus optimale et mieux adaptée.

1.2.1 Solution 1

Dans cette heuristique, nous commençons par immerger la première éprouvette dans le bain 1. Une fois son traitement terminé, nous la remplaçons dans le frigo initial. Ensuite, nous procédons à l'immersion de l'éprouvette 2 dans le bain 1 pour son traitement, suivie de son retour dans le frigo initial. Nous faisons la même chose avec la dernière éprouvette. Cette séquence est répétée pour toutes les éprouvettes concernant le bain 2, à la différence que cette fois-ci, elles sont transférées vers le frigo final. le temps total estimé serait de 129 unités

1.2.2 Solution 2

Dans cette heuristique, nous traitons une éprouvette à la fois. Cela signifie que nous prenons l'éprouvette 1 et la déplaçons vers le bain 1, où nous attendons jusqu'à la fin de son traitement, en respectant le temps minimal requis. Ensuite, nous la transférons vers le bain 2, en appliquant la même procédure que pour le bain 1, avant de la déplacer vers le réfrigérateur final. Nous répétons ces mêmes étapes pour les éprouvettes restantes. Cette méthode semble rudimentaire. Si nous l'appliquons à notre cas initial ($n=3$ éprouvettes), le temps total estimé serait de 136 unités.

1.2.3 Solution 3

Dans notre processus de traitement, nous débutons en prenant l'éprouvette 1 que nous plongeons dans le bain 1. Ensuite, nous revenons vers frigo initial, nous déplaçons ainsi l'éprouvette 2 vers le bain 1. nous transférons ainsi l'éprouvette 1 vers le bain 2. Nous reviendrons ensuite vers le bain 1 pour prendre l'éprouvette 2 et la déplacer vers le bain 2. Nous prenons L'éprouvette 1 vers le frigot final, on revient pour faire la m chose avec l'éprouvette 2. A ce stade, nous revenons vers le frigot initial pour déplacer l'éprouvette 3. nous la déplaçons, successivement, vers le bain 1, le bain et le frigot finale, le temps total estimé serait de 99 unités.

À chaque étape de ce processus, le robot attend que chaque éprouvette ait achevé son traitement suffisant avant de passer à l'étape suivante.

Une description détaillée de ce processus est disponible dans le fichier Excel joint à ce rapport.

```

▶ def temps_total(n):

    if n % 2 == 0:      # Si n est pair
        k = n // 2
        temps_total = k * 50 + (k - 1) * 11
    else:
        k = n // 2 # Si n est impair
        temps_total = k * 50 + k * 11 + 38

    return temps_total

print(temps_total(6)) # Si n est pair et égale à 6
print(temps_total(5)) # Si n est impair et égale à 5

```


 172
 160

FIGURE 1.2 – Code illustrant notre méta-heuristique

1.3 Méta-heuristique de construction

Dans cette section, nous abordons le cas où le nombre d'éprouvettes, noté n , est inconnue. Nous considérons par la suite que nous avons n éprouvettes. Nous avons trouvé qu'il est judicieux de traiter deux cas : soit n est pair, soit n est impair. Nous avons opté pour une approche d'étude de cas afin d'explorer les scénarios correspondant à ces deux situations. Cette démarche s'inspire de notre heuristique initiale pour développer une méta-heuristique adaptative.

Le code illustré dans la figure ci-dessus résume notre approche pour calculer le temps total en fonction de la parité de n .

1.4 Formulation mathématique

Objectif

Minimiser le temps total nécessaire pour que toutes les éprouvettes passent par les étapes de traitement.

Variable de décision

x_{ij} : est une variable qui a pour valeur 1 si l'éprouvette i est traitée dans la source j , 0 sinon.

$$Z = \sum_{i=1}^n \sum_{j=1}^n (t_{ij} + d_{ij}) x_{ij}$$

Contraintes

Posons tous les variables qui vont nous aider à formuler notre problème mathématiquement.

- $r(i)$ La ressource i , par exemple $r(0)$ représente le frigot initial et, $r(1)$ et $r(2)$ représente le bain 1 et le bain 2, successivement, et $r(3)$ représente le frigot final
- $R(S)$ la position actuel de robot,
- $w_{r(i),r(j)}$ le temps nécessaire pour se déplace de $r(i)$ vers $r(j)$,
- $x_{\alpha,i}$ (resp. $y_{\alpha,i}$) le moment où l'éprouvette alpha entre à (resp. sort de) $r(i)$,
- $b_{\alpha,i}$ (resp. $e_{\alpha,i}$) début de traitement de l'éprouvette α (resp. ends) on sample α ,
- d_i (resp. d^j) la min (resp. max) durée de traitement pour la ressource i (la ressource j) (on considère que $d_0 = 0$ and $d_3 = \infty$),
- $t_\alpha(S)$ la tache actuelle de l'éprouvette α ,
- $s(S)$ la dernière éprouvette que le robot a traité,
- $T(S)$ le moment où le robot est prêt à se déplacer pour effectuer une nouvelle tâche ,

Vous trouverez en pièce jointe un doc qui résume les contraintes.

Considérons que notre robot traite deux taches de déplacer les éprouvettes alpha et gamma :

- C1 : le temps de l'arrivée de l'éprouvette gamma à sa prochaine ressource doit égale au temps de mettre l'échantillon alpha dans sa propre ressource plus le temps de déplacement.

-C2 : Évidente dans notre cas

-C3 : le temps de début de traitement doit égale au le temps où l'éprouvette entre à la source (pour le bain 1 et le bain 2 : $r(1)$ et $r(2)$)

-C4 : le temps de fin de traitement doit etre le temps où l'éprouvette doit quitte la source (pour le bain 1 et le bain 2 : $r(1)$ et $r(2)$)

-C6/C7 : chaque éprouvette doit être traitée exactement une fois dans le bain 1 et une fois dans le bain 2

-C8 : Chaque éprouvette doit passer par le bain 1 avant le bain 2

1.5 Résolution par solveur de notre programme linéaire

```

C: > Users > SAMI YOUSSEF > Downloads > # Importer matplotlib.py > ...
1  from pulp import LpProblem, LpMinimize, LpVariable, lpSum, value
2
3  # Données
4  n = 3 # le nombre d'éprouvettes à traiter
5  Tmin_B1 = 12 # durée minimale pour le bain 1
6  Tmax_B1 = 16 # durée maximale pour le bain 1
7  Tmin_B2 = 12 # durée minimale pour le bain 2
8  Tmax_B2 = 13 # durée maximale pour le bain 2
9
10 # Création du problème
11 probleme = LpProblem("Ordonnancement", LpMinimize)
12
13 # Définition des variables de décision
14 x = LpVariable.dicts("x", [(i, j) for i in range(1, n + 1) for j in range(1, 3)], cat="Binary")
15 time_in_bath_1 = LpVariable.dicts("time_in_bath_1", [(i, j) for i in range(1, n + 1) for j in range(1, 3)], lowBound=0)
16 time_in_bath_2 = LpVariable.dicts("time_in_bath_2", [(i, j) for i in range(1, n + 1) for j in range(1, 3)], lowBound=0)
17
18 # Définition de la fonction objectif
19 probleme += lpSum(time_in_bath_1[i, j] + time_in_bath_2[i, 3 - j] for i in range(1, n + 1) for j in range(1, 3)), "z"
20
21 # Contraintes
22 probleme += lpSum(x[i, 1] for i in range(1, n + 1)) <= lpSum(x[i, 2] for i in range(1, n + 1)), "Contrainte1"
23
24 for i in range(1, n + 1):
25     |   probleme += Tmin_B1 * x[i, 1] <= time_in_bath_1[i, 1] <= Tmax_B1 * x[i, 1], f"Contrainte2_B1_{i}"
26     |   probleme += Tmin_B2 * x[i, 2] <= time_in_bath_2[i, 2] <= Tmax_B2 * x[i, 2], f"Contrainte2_B2_{i}"
27
28 for i in range(1, n + 1):
29     |   for j in range(1, 3):
30     |   |   |   probleme += x[i, j] - x[i, j % 2 + 1] == 0, f"Contrainte4_{i}_{j}"
31
32 # Résolution du problème
33 probleme.solve()
34
35 # Affichage des résultats
36 print(f"État de la résolution: {probleme.status}")
37

```

FIGURE 1.3 – Code python partie 2

```

37
38 # Affichage du temps total
39 temps_total = value(probleme.objective)
40 print(f"Le temps total pour traiter les {n} échantillons est de {temps_total} secondes")
41
42 # Affichage des variables de décision
43 for i in range(1, n + 1):
44     |   for j in range(1, 3):
45     |   |   |   print(f"x[{i},{j}] = {value(x[i, j])}")

```

FIGURE 1.4 – Code python partie 2

1.6 Algorithme de résolution

Étape 0 : Initialiser le programme partiel S en définissant

$$x_{\alpha,0} = 0; \quad t_{\alpha}(S) \leftarrow 0; \quad h_{\alpha} \leftarrow 0; \quad h_k \leftarrow d_0; \quad \text{pour } \alpha = 1, \dots, n$$
$$c \leftarrow 0; \quad (\text{compteur d'itération})$$

Étape 1 : Selon une certaine règle \mathcal{R}_1 , choisir un échantillon α avec $t_{\alpha}(S) < p$

Étape 2 : Mettre $c \leftarrow c + 1$; appeler $\text{MOVE}(\alpha)$; régler $S \leftarrow S_{\alpha}$ et actualiser h_{α} et h_{β} pour $\beta = 1, \dots, n$; si S est un programme partiel réalisable satisfaisant la contrainte (5) alors passer à l'étape 7

Étape 3 : Soit O l'ordre partiel induit par S .

Étape 4 : Utiliser l'algorithme de Bellman pour soit calculer le chemin le plus long de a à tous les autres sommets dans G , soit pour trouver un circuit de longueur positive dans G ; Si G ne contient aucun circuit de longueur positive alors régler $S \leftarrow S_G$ et passer à l'étape 7;

Étape 5 : Basé sur une certaine règle \mathcal{R}_2 , choisir un échantillon β avec $t_{\beta}(S) > 0$;

Étape 6 : Régler $C \leftarrow t_{\beta}(S)$ et $G \leftarrow G - \{\beta\}$; aller à l'étape 4

Étape 7 : Si $c = n \cdot p$ alors STOP sinon aller à l'étape 1

Voici la fonction $\text{MOVE}(\alpha)$

$$y_{\alpha,t} \leftarrow h_{\alpha}$$
$$e_{\alpha,t} \leftarrow \min\{h_{\alpha}, \min_{i < \alpha} (e_{i,t} + L_{i,t}) + d_{\alpha}\}$$
$$h_{\alpha,t} \leftarrow \min\{e_{\alpha,t} - d_{\alpha}, \min_{i < \alpha} (e_{i,t} + L_{i,t})\}$$
$$x_{\alpha,t+1} \leftarrow y_{\alpha,t} + w_{r(\alpha),r(t+1)}$$

Fin de MOVE