TP

8INF852- Métaheuristiques en optimisation

HIVER 2021

Aymen Sioud

UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

Département d'Informatique et de Mathématique

Consignes générales

- **Le travail se fait par binôme (équipe de deux) obligatoirement.**
- **Les travaux devront être remis via Moodle**
- **Le détail des remises est plus bas.**

Table des matières

Mise en contexte

Vous devrez résoudre le problème d'optimisation présenté avec différentes métaheuristiques en appliquant les différentes consignes.

Le problème

Soit un ensemble de *n* tâches à ordonnancer sur une machine. Ces tâches peuvent être des threads sur un processeur, des voitures sur une chaîne d'assemblage ou encore des commandes à traiter. Afin de généraliser, nous considérons donc le traitement de *n* tâches sur une machine unique. Chaque tâche *i* des *n* tâches possède un temps de traitement (exécution) noté p_i sur la machine unique (*processing time*). Il existe de plus un temps de réglage qui est introduit lors du passage d'une tâche à une autre. Ainsi, si une tâche *j* est ordonnancée directement après une tâche *i*, il faudra alors calculer un temps de réglage noté s_{ij} (setup time). Ces temps de réglages peuvent représenter par exemple le temps de nettoyage d'un pistolet de peinture lors d'un changement de couleur, où dépendamment de la couleur à nettoyer de celle à utiliser, les temps de préparation sont asymétriques. Cela veut dire que $s_{ik} \neq s_{ki.}$

De même, nous considérons une date due pour chaque tâche *i* et noté d_i (*due date*), et finalement une date de départ $depart_i$ qui représente un temps de préparation pour la première tâche. En effet, si la tâche est ordonnancée en début de séquence, un temps de départ est introduit.

Il est supposé que les temps d'exécution, les dates dues ainsi que les temps de réglages sont des valeurs entières non négatives. Nous verrons plus loin que les dates dues peuvent être négatives dans certains cas, ce qui induit que des travaux sont déjà en retard. Dans ce travail, nous supposons que la préemption n'est pas permise.

Soit Q une séquence de tâches avec $Q = [q_0, q_1, \ldots, q_{n-1}, q_n]$ où q_j représente l'indice de la $j^{ième}$ tâche dans la séquence. La date due de la $j^{ième}$ tâche dans la séquence est notée par d_{q_j} et son temps d'exécution par p_{q_j}. Ainsi, le temps de fin d'exécution du $j^{ième}$ travail dans la séquence est égale à $C_{q_j} = \sum_{k=1}^{j}(s_{q_{k-1}q_k} + p_{q_k})$ et le retard du $j^{ième}$ travail dans la séquence est égale à $T_{q_j} = max(\, C_{q_j} - d_{q_j}, 0)$.

La fonction objectif, qui vise à minimiser le retard total (*total tardiness*) de tous les travaux de la séquence, est calculée par $\sum_{j=1}^{n} T_{q_j}$.

Afin de mieux comprendre le problème voici l'exemple suivant avec les p_i et d_i suivants :

	1	2	3	4	5	6	7	8
p_i	102	100	97	99	96	102	97	107
d_i	640	596	602	585	625	635	616	645

Ainsi que la matrice de setup s_{ij} et les temps de réglage de départ dans la première ligne.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
***depart*$_i$**	2	0	20	15	13	1	2	7
1	-	7	13	17	14	0	1	12
2	14	-	12	15	0	18	6	16
3	9	1	-	11	9	12	1	6
4	15	18	13	-	4	3	5	7
5	3	12	3	13	-	13	20	2
6	1	8	4	16	12	-	9	12
7	7	14	11	0	5	2	-	14
8	10	2	13	15	7	16	1	-

Supposons la séquence (solution) 1-2-3-4-5-6-7-8 :

Séquence	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	$\sum_{j=1}^{n} T_{q_j}$
C$_j$	104	211	320	430	530	645	751	872	
d$_j$	640	596	602	585	625	635	616	645	
T$_j$	0	0	0	0	0	10	135	227	372

Ainsi $C_1 = p_1 + départ_1 = 102 + 2 = 104$

$C_2 = C_1 + p_2 + s_{12} = 104 + 100 + 7 = 211$

$C_3 = C_2 + p_3 + s_{23} = 211 + 97 + 12 = 320$ et ainsi de suite.

Pour le calcul de T_j, $T_j = \max(0; d_j - C_j)$, Par exemple T_8 =227 = 872-645 pour la tâche 8.

À la fin, nous calculons la somme de tous les T_j pour obtenir la fonction objectif (372 pour la solution considérée).

Le but ici est de trouver la meilleure séquence avec le plus petit T_j.

Ce problème est similaire à un ATSP (Asymetric TSP).

Les instances

Afin de tester vos différents algorithmes, une batterie d'instances vous est fournie.

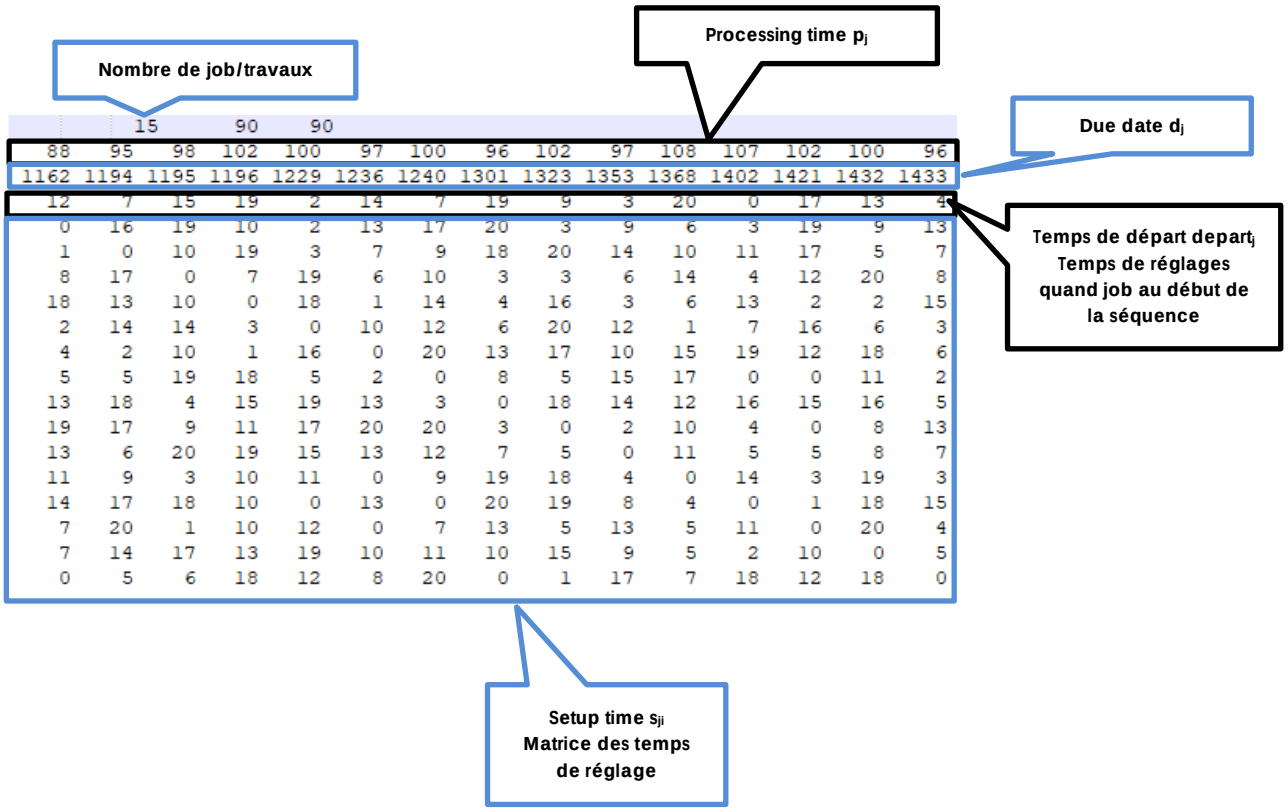
Dans la première ligne contient seulement le premier nombre est important. Il représente la taille de l'instance ou nombre de tâches. Vous pouvez ignorer les deux autres nombres.

La deuxième ligne représente les temps d'exécution (processing time) p_i.

La troisième ligne représente les dates dues (due dates) d_i.

La quatrième ligne représente les temps de réglages de départ $départ_i$.

Les lignes suivantes représentent une matrice carrée des temps de réglages (setup times). Ainsi, les n premiers nombres représentent les s_{1j}, les n deuxième les s_{2j}, etc.

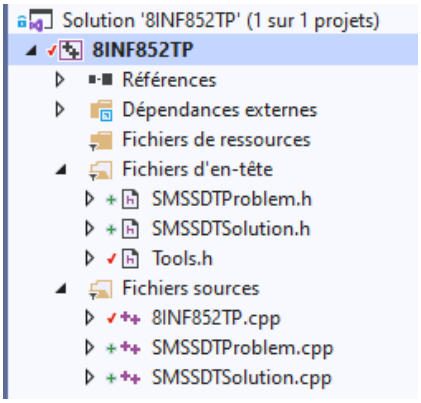



```
 15   90   90
 88   95   98  102  100   97  100   96  102   97  108  107  102  100   96
1162 1194 1195 1196 1229 1236 1240 1301 1323 1353 1368 1402 1421 1432 1433
 12    7   15   19    2   14    7   19    9    3   20    0   17   13    4
  0   16   19   10    2   13   17   20    3    9    6    3   19    9   13
  1    0   10   19    3    7    9   18   20   14   10   11   17    5    7
  8   17    0    7   19    6   10    3    3    6   14    4   12   20    8
 18   13   10    0   18    1   14    4   16    3    6   13    2    2   15
  2   14   14    3    0   10   12    6   20   12    1    7   16    6    3
  4    2   10    1   16    0   20   13   17   10   15   19   12   18    6
  5    5   19   18    5    2    0    8    5   15   17    0    0   11    2
 13   18    4   15   19   13    3    0   18   14   12   16   15   16    5
 19   17    9   11   17   20   20    3    0    2   10    4    0    8   13
 13    6   20   19   15   13   12    7    5    0   11    5    5    8    7
 11    9    3   10   11    0    9   19   18    4    0   14    3   19    3
 14   17   18   10    0   13    0   20   19    8    4    0    1   18   15
  7   20    1   10   12    0    7   13    5   13    5   11    0   20    4
  7   14   17   13   19   10   11   10   15    9    5    2   10    0    5
  0    5    6   18   12    8   20    0    1   17    7   18   12   18    0
```

Code fourni

Afin de vous faciliter la prise en main du projet, une partie de code vous est fourni. Ainsi un projet 8INF852TP en Visual C++ vous est remis avec l'énoncé.

Structure du projet



Le projet contient six fichiers différents :

- 8INF852TP.cpp
  - Contenant la fonction principale (main), il exécute un exemple simple en générant aléatoirement 1000 fois une solution pour `argv[1]` fois.
  - L'exemple calcul les temps d'exécution est les log ainsi que le résultat de la meilleure solution
- SMSSDTProblem.h/ SMSSDTProblem.cpp
  - Contient la définition du problème traité (Single Machine Sequence Setup Dependent Time)
  - Contient une fonction de lecture d'instance et d'écriture dans un stream quelconque
- SMSSDTSolution.h/ SMSSDTSolution.cpp
  - Contient la définition d'une solution
  - Contient une fonction permettant de créer une solution aléatoire et une autre permettant la sortie d'une solution avec la fonction objectif dans un stream quelconque
- Tools.h
  - Contient des fonctions généralistes tels les calculs des Ci, des Ti, de la fonction d'évaluation ou encore le log dans un fichier.

Résultat et exécution

La génération du projet permet d'obtenir un fichier exécutable 8INF852TP.exe.
Afin de l'exécuter, vous exécuterez Dans le même répertoire le fichier Execute_8INF852TP.bat qui contient la commande suivante : 8INF852TP.exe 10 PROB401.TXT (Vous pouvez aussi appeler l'invite commande de Visual Studio). Cette commande permettra d'exécuter 10 fois le programme avec le fichier en paramètre 2.

L'exécution va générer un répertoire Results ainsi qu'un fichier Report_PROB401.txt.log qui contient les temps d'exécution la meilleure séquence et sa fonction objectif.

Le fichier ExecP.bat dans le répertoire Results permet d'exécuter toutes les instances. Il faudra le mettre dans un répertoire avec toutes les instances et l'exécutable.

Il vous sera ensuite facile avec le module de données de Excel, je gérer les résultats à partir des fichiers .log générés.

Rendu #1 – Pour le 09 février 2021 – 17h

À partir du code fourni et en choisissant **un critère d'arrêt unique**, vous devrez élaborez **deux** métaheuristiques parmi les suivantes :

- VNS
- Recuit simulé
- Recherche taboue

Vous devrez aussi élaborer une méthode de descente (recherche locale).

Remise rendu#1

Vous devrez remettre :

- Un rapport unique comportant le descriptif détaillé de vos différents algorithmes avec tous les détails des différents composants de vos méthodes ainsi que leurs paramètres respectifs.
- Le rapport devra comporter les résultats avec une présentation similaire à celle de l'Annexe I.
    - Chacune de vos méthodes devra être exécutée 20 fois.
    - Pour chaque fichier et méthode, vous devrez inscrire la pire, moyenne et meilleure déviation par rapport à la meilleure solution trouvée par vos différents algorithmes et ce, en %.
    - Si le meilleur résultat de toutes vos méthodes (20 exécutions) trouvé pour l'instance 401 est de 90 par exemple par votre Méthode 1, alors vous devrez inscrire les résultats par rapport à 90.
        - Si Méthode 2 a trouvé comme meilleure solution 96, alors la déviation sera égale à :

$$\frac{96 - 90}{90} \times 100$$

- Vous devrez aussi présenter les temps d'exécution comme à l'Annexe II.
- Le zip de votre code source sous Visual Studio 2019 et son exécutable.
- Le répertoire de vos fichiers .log et les Excel de vos résultats.

Fonctionnement général

- Une rencontre par équipe sera organisée pour vous donner un retour sur votre travail après la date de remise

Rendu #2 – Pour le 09 mars 2021 – 17h

À partir du code fourni et en choisissant **un critère d'arrêt unique**, vous devrez élaborez

- Un algorithme génétique avec une population de base générée aléatoirement
- Un algorithme génétique avec la moitié de la population initiale ayant subi une descente
- Un algorithme mémétique

Remise rendu#2

Vous devrez remettre :

- Un rapport unique comportant le descriptif détaillé de vos différents algorithmes avec tous les détails des différents composants de vos méthodes ainsi que leurs paramètres respectifs.
  - Sélection
  - Croisement
  - Mutation
  - Remplacement
  - Etc.
- Le rapport devra comporter les résultats avec une présentation similaire à celle de l'Annexe I.
  - Chacune de vos méthodes devra être exécutée 20 fois.
  - Pour chaque fichier et méthode, vous devrez inscrire la pire, moyenne et meilleure déviation par rapport à la meilleure solution trouvée par vos différents algorithmes et ce, en %.
  - Si le meilleur résultat de toutes vos méthodes (20 exécutions) trouvé pour l'instance 401 est de 90 par exemple par votre Méthode 1, alors vous devrez inscrire les résultats par rapport à 90.
    - Si Méthode 2 a trouvé comme meilleure solution 96, alors la déviation sera égale à :

$$\frac{96 - 90}{90} \times 100$$

- Vous devrez aussi présenter les temps d'exécution comme à l'Annexe II.
- Le zip de votre code source sous Visual Studio 2019 et son exécutable.
- Le répertoire de vos fichiers .log et les Excel de vos résultats.

Fonctionnement général

- Une rencontre par équipe sera organisée pour vous donner un retour sur votre travail.

Rendu #3 – Pour le 13 avril 2021– 17h

À partir du code fourni et en choisissant **un critère d'arrêt unique**, vous devrez élaborez un algorithme d'optimisation par colonies de fourmi.

Remise rendu#3

Vous devrez remettre :

- Un rapport unique comportant le descriptif détaillé de vos différents algorithmes avec tous les détails des différents composants de vos méthodes ainsi que leurs paramètres respectifs.
- Résultats présentés
  - Vous devrez considérez les résultats de tous vos algorithmes du Rendu#1 et Rendu#2
- Le rapport devra comporter les résultats avec une présentation similaire à celle de l'Annexe I.
  - Chacune de vos méthodes devra être exécutée 20 fois.
  - Pour chaque fichier et méthode, vous devrez inscrire la pire, moyenne et meilleure déviation par rapport à la meilleure solution trouvée par vos différents algorithmes et ce, en %.
  - Si le meilleur résultat de toutes vos méthodes (20 exécutions) trouvé pour l'instance 401 est de 90 par exemple par votre Méthode 1, alors vous devrez inscrire les résultats par rapport à 90.
    - Si Méthode 2 a trouvé comme meilleure solution 96, alors la déviation sera égale à :

$$\frac{96 - 90}{90} \times 100$$

- Vous devrez aussi présenter les temps d'exécution comme à l'Annexe II.
- Le zip de votre code source sous Visual Studio 2019 et son exécutable.
- Le répertoire de vos fichiers .log et les Excel de vos résultats.

Fonctionnement général

- Une rencontre par équipe sera organisée pour vous donner un retour sur votre travail.

Annexe I – Présentation des résultats

Fichier	Méthode 1			Méthode 2			….
	Meilleure déviation	Moyenne déviation	Pire déviation	Meilleure déviation	Moyenne déviation	Pire déviation	
401							
402							
403							
404							
405							
406							
407							
408							
501							
502							
503							
504							
505							
506							
507							
508							
601							
602							
603							
604							
605							
606							
607							
608							
701							
702							
703							
704							
705							
706							
707							
708							

Annexe II – Présentation des temps d'exécution

Fichier	Méthode 1			Méthode 2			….
	Meilleur temps	Temps moyen	Pire temps	Meilleur temps	Temps moyen	Pire temps	
401							
402							
403							
404							
405							
406							
407							
408							
501							
502							
503							
504							
505							
506							
507							
508							
601							
602							
603							
604							
605							
606							
607							
608							
701							
702							
703							
704							
705							
706							
707							
708							