Mini-projet Master 1 M2SI

2021-2022

Objectifs :

- Prendre conscience de la difficulté à résoudre des problèmes NP-difficile (fortement combinatoire) à l'aide de méthodes exactes.
- Prendre conscience de la capacité des métaheuristiques à trouver des solutions de très bonne qualité en un temps raisonnable.

Travail demandé :

1) Résolution des problèmes d'ordonnancement **PROBLEME 1** et **PROBLEME 2** avec la programmation dynamique (DP) et une métaheuristique (GVNS).
   La préemption n'est pas permise dans les deux problèmes.
2) Comparaison des solutions obtenues par ces méthodes et du temps d'exécution correspondants.

PROBLEME 1 : Dans ce cas, on considère ***n*** tâches à exécuter sur une seule ressource afin de minimiser ***l'objectif 1*** : $1||\sum U_i$.
Les tâches sont toutes disponibles à l'instant ***0*** avec chacune une durée opératoire p_i et une date de livraison d_i. Noter qu'après le démarrage de la première tâche aucun temps mort n'est autorisé sur la ressource (i.e. chaque solution réalisable doit commencer à t=0 et se terminer à $t=\sum p_i$).
Noter ici qu'il possible d'utiliser l'algorithme de Moore-Hodgson (AMH) pour résoudre ce problème.
En conséquence, et à titre purement illustratif, il faut inclure la comparaison de l'algorithme AMH avec DP et GVNS pour ce problème.
PROBLEME 2 : Dans ce cas, on considère ***n*** tâches à exécuter sur une seule ressource afin de minimiser ***l'objectif 2 :*** $1||\sum \alpha_i E_i + \beta_i T_i$
Les tâches sont toutes disponibles à l'instant ***0*** avec chacune une durée opératoire p_i et une date de livraison d_i. Noter qu'après le démarrage de la première tâche aucun temps mort n'est autorisé sur la ressource.

Instructions :

- Utiliser au maximum 3 structures de voisinage pour chaque phase de GVNS.
- Choisir les structures discutées en classe ou autres de votre choix.
- L'utilisation de la structure 2-opt est recommandée.
- Langage de programmation à utiliser : C ou python.
- Critère d'arrêt de GVNS : temps d'exécution de l'algorithme t_{max}. (vous pouvez augmenter t_{max} en fonction du nombre de tâches ***n***).

Evaluation :

La note finale du projet sera basée sur les éléments suivants :

- Le rapport (minimum 25 pages – maximum 100 pages).
- La qualité de la solution et la rapidité d'exécution des algorithmes.
- La clarté et l'optimisation du code.
  N.B. : solution ici veut dire à la fois la séquence et la valeur de la fonction objectif !

Instances :

Pour chaque problème, vous devez résoudre les instances suivantes :

PROBLEME 1 :

- ✓ P1_n10.txt
- ✓ P1_n50.txt
- ✓ P1_n200.txt
- ✓ P1_n500.txt

Dans chaque fichier ci-dessous :

- La première ligne contient le nombre de tâche ***n***.
- La deuxième ligne contient les durées des tâches p_i.
- La troisième ligne contient les dates d'échéance des tâches d_i.

PROBLEME 2 :

- ✓ P2_n10.txt
- ✓ P2_n100.txt
- ✓ P2_n200.txt
- ✓ P2_n500.txt

Dans chaque fichier ci-dessous :

- La première ligne contient le nombre de tâche ***n***.
- La deuxième ligne contient les durées des tâches p_i.
- La troisième ligne contient les durées d'échéance d_i.
- La quatrième ligne contient les poids α_i.
- La dernière ligne contient les poids β_i.

N.B. : pour bien paramétrer votre métaheuristique, il vous faudra probablement résoudre plusieurs instances (que vous pouvez générer aléatoirement) avant de résoudre les instances requises.

Bonus :

1) Réalisation d'une interface qui laisse le choix à l'utilisateur d'utiliser « la programmation dynamique » ou « GVNS » pour résoudre une instance donnée à partir d'un fichier *.txt. et de choisir entre ***l'objectif 1*** ou ***l'objectif 2***. Le programme devrait afficher la solution graphique (Gantt) en spécifiant les dates de début de chaque tâche dans la séquence.
2) Fournir toutes les solutions optimales pour les instances P1_n10.txt et P2_n10.txt.

Date limite :

Le travail est à rendre avant le 10/01/2021