Heuristiques — problème d'ordonnancement

M1 Informatique – Algorithmes et Complexité

Université de Lille

L'objecif de ce TP est la conception, l'implémentation et l'étude d'heuristiques pour un problème d'ordonnacement à l'ennoncé simple mais appartenant à la classe des problèmes NP-dur.

1 Description du problème

On considère le problème qui consiste à ordonnancer n tâches sur une machine séquentielle. La machine ne peut exécuter qu'une seule tâche à la fois. Toutes les tâches sont disponibles à l'instant 0. Chaque tâche $j \in \{1, 2, \ldots, n\}$ possède:

- Un temps d'exécution p_j,
- Un poid w_j qui renseigne sur l'importance de la tâche
- Un temps d_j qui correspond à la limite à laquelle l'exécution de la tâche doit être terminée. Au delà de cette limite, la tâche est considérée en retard.

Pour un ordonnancement O fixé (c.à.d une séquence d'exécution des tâches sur les machines), on note

- C_j le temps de complétion de la tâche j, et
- $T_j := \max\{C_j - d_j, 0\}$ le retard de la tâche j.

L'objectif est alors de minimiser la somme totale des retards pondérés. Autrement dit, on s'intèresse au problème d'optimisation qui consiste à trouver un ordonnancement O qui minimise la fonction f définie par:

$$f(O) := \sum_{j=1}^{n} w_j \cdot T_j$$

2 Travail demandé

L'objectif du TP est de concevoir et de tester de façon incrémentale différents types d'heuristiques pour le problème d'ordonnacement ennoncé précédemment. Pour réaliser vos expérimentations, on vous donne 20 instances de tests de taille moyenne avec $n = 100$, et 2 instances de grande taille $n = 1000$.

Travail préliminaire : Dans une première étape, il s'agit de lire une instance à partir d'un fichier donnée et d'évaluer la qualité d'une solution générée de façon aléatoire.

Q1.1 En utilisant votre langage préféré, écrire un programme qui permet d'**évaluer** la qualité d'une solution donnée en argument pour une instance chargée au préalable. Autrement dit, écrire un programme qui prend en argument un ordonnacement O et qui calcule la fonction $f(O)$ pour une instance donnée du problème.

Q1.2 Ecrire un programme qui permet de générer une solution aléatoire et d'évaluer sa qualité.

Heuristiques constructives et recherche locale simple : Dans une deuxième étape, il s'agit de mettre en place différents types d'heuristiques pour la résolution du problème.

Q1.3 Proposer et implémenter une (ou plusieurs) **heuristiques constructives**.
indication: Pensez à prendre en compte la valeur du retard d'une tâche dans l'ordonnacement.

Q1.4 Proposer et implémenter (au moins) un voisinage pour la conception d'heuristiques par recherche locale simple. En déduire une (ou plusieurs) recherche locale simple de type **HillClimbing** ou **VND**. Veillez à spécifier les différents choix de conception (initialisation, type de mouvement, etc).

Recherche locale itérée

Q1.5 Proposer et implémenter une recherche locale itérée, de **type ILS**. Veillez à spécifier les différents choix de conception (initialisation, recherche locale de base, perturbation, critère d'acceptation, critère d'arrêt).

Q1.6 Parmi les heuristiques développées, tester la variante qui vous semble être la plus adaptée et reporter vos résultats (meilleur valeur objectif, temps de calcul, etc) pour quelques instances (selon le temps de calcul dont vous disposez).

Q1.7 Soumettez la meilleur solution trouvée pour chaque instance sur la plateforme de test.

Bonus (non obligatoire)

Q1.8 Proposer et tester n'importe quel autre type d'algorithme qui vous semble plus efficace. Reportez vos meilleures solutions pour les différentes instances de la plateforme de test.