

Heuristiques — problème d’ordonnancement

M1 Informatique – Algorithmes et Complexité

Université de Lille

L’objectif de ce TP est la conception, l’implémentation et l’étude d’heuristiques pour un problème d’ordonnancement 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 $j \in \{1, 2, \dots, n\}$ possède :

- Un temps d’exécution p_j ,
- Un poids 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’intéresse 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’ordonnancement énoncé 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.

- Q1.0** En utilisant votre langage préféré, écrire un programme qui permet de lire les données d'une instance à partir du fichier correspondant. Un fichier décrivant le format des fichiers d'instances vous est donné sur moodle.
- Q1.1** Écrire un programme qui prend en argument un ordonnancement O et qui retourne la valeur de la fonction $f(O)$, pour une instance donnée du problème.
- Q1.2** Écrire un programme qui permet de générer une solution aléatoire et d'évaluer sa qualité, pour une instance donnée du problème.

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

- Q1.3** Proposer et implémenter (au moins) une **heuristique constructive**.
indication: Pensez à prendre en compte la valeur du retard d'une tâche dans l'ordonnancement.
- 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** et/ou **VND**. Veillez à spécifier les différents choix de conception (initialisation, type de mouvement, etc).

Recherche locale itérée

- Q1.5** Proposer et implémenter (au moins) une recherche locale itérée, de **type ILS**.
 Veillez à justifier/motiver les différents choix de conception (initialisation, recherche locale de base, perturbation, critère d'acceptation, critère d'arrêt).
- Q1.6** Parmi les ILS développées, tester la variante qui vous semble être la plus adaptée et reporter vos résultats (meilleure valeur objectif, temps de calcul, etc) pour quelques instances (selon le temps de calcul dont vous disposez). Veillez à justifier/motiver vos choix de conception.

Étude expérimentale

- Q1.7** Étudier la performance relative des différents algorithmes développés. Pour cela, **vous disposez sur moodle d'un fichier contenant les valeurs objectives des solutions optimales pour chaque instance (calculées de façon exacte)**. Cela vous permet de calculer l'écart relatif de la solution trouvée par l'exécution d'un algorithme, normalisé en pourcentage, relativement à la solution optimale. Ensuite, vous pouvez calculer et étudier quelques statistiques simples tels que : l'écart relatif moyen d'un algorithme en fonction du temps d'exécution, la probabilité empirique qu'un algorithme trouve une solution avec un écart donnée en fonction du temps alloué, etc.