# 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 tâche sont disponible 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. Plus formellement, le problème d'optimisation auquel on s'intèresse consiste à trouver un ordonnancement $O$ qui minimise la fonction objectif $f$ donnée par:

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

## 2 Travail demandé

L'objectif du TP est de concevoir et d'expérimenter de façon incrémentale différents types d'heuristiques pour le problème d'ordonnacement ennoncé précédemment.

**Travail préliminaire** : Pour réaliser vos expérimentations, on vous donne plusieurs ensembles d'instances de tests avec différentes valeur de $n \in \{40, 50, 100\}$. Dans une première étape, il s'agit d'effectuer un travail préliminaire permettant de lire une instance et d'évaluer la qualité d'une solution 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).

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

**Heursitques constructives et recherche locale** : 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. 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).

**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).

**Compagne d'expérimentation et analyse** : Dans une troisième étape, il s'agit de mesurer et d'analyser la qualité de votre heuristiques. Pour cela, vous avez à disposition les solutions optimales pour chaque instance. Vous fixerez un temps d'exécution maximal pour chaque exécution (par exemple $3 \cdot n$ secondes, afin de garder un temps totale raisonable pour la réalisation de votre compagne d'expérimentation).

Pour chaque instance, on s'interesse à l'analyse du temps CPU et le nombre d'évaluations utilisées par l'algorithme pour atteindre une solution avec un facteur d'approximation $(1 + \epsilon)$ de l'optimal avec $\epsilon \in \{0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$.

**Q1.6** En effectuant une compagne d'expérimentation utilisant les instances données, quelles conclusions pouvez vous tirer quand à l'efficacité relative de votre algorithme (ou de vos algorithmes, dans le cas où différentes variantes sont débveloppées) ?
Conseil: N'hésitez pas à dessiner des courbes et/ou des tableaux rendant compte de vos observations expérimentales, d'un point de vue général, ou pour quelques instances en particulier.

**Challenge** : Dans cette dernière partie, on s'interesse à des instances de grande taille $n = 1000$.

**Q1.7** 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).