Projet RP – Printemps 2021

A faire en binôme. La date limite pour le rapport est le 25 mai 2021 à 17h. La date de soutenance vous sera communiquée unltérieurement. Pensez à déclarer votre binôme.

A) On va considérer le problème d'ordonnancement suivant : on se donne n tâches disponibles à l'instant 0 que l'on doit ordonnancer sur une machine. La durée d'exécution x_j de la tâche j n'est pas connue à l'algorithme et devient disponible seulement après la fin de l'exécution de la tâche. On va considérer que la préemption, c'est-à-dire la possibilité d'interrompre une tâche et la continuer plus tard est autorisée. Notre objectif est de déterminer un algorithme minimisant la somme totale des dates de fin d'exécution (dates de complétudes) de toutes les tâches.

Soit $x_1, x_2, \ldots, x_n$ les durées d'exécution des n tâches (qui ne sont pas connues dans notre cas). Si ces durées étaient connues à l'algorithme, alors l'algorithme optimal consisterait à simplement ordonnancer les tâches dans un ordre non-décroissant de leurs durées, c'est-à-dire en donnant la priorité à la tâche dont la durée est la plus courte. Dans le cas où les durées de tâches ne sont pas connues, on peut utiliser une politique ROUND-ROBIN qui garantit qu'à chaque instant t, toutes les tâches qui n'ont pas encore terminé leur exécution partagent à égalité la machine. Autrement dit, à tout instant, s'il existe k tâches toujours actives, alors chaque tâche est exécutée à une vitesse $1/k$. Ainsi, l'exécution des k tâches ayant chacune une durée réelle égale à 1 nécessiterait k unités de temps.

Dans le cadre de ce projet, nous allons considérer qu'au lieu d'ignorer totalement l'instance, l'algorithme a l'aide d'un oracle qui prédit la durée de tâches. On n'a aucune garantie concernant la qualité des prédictions fournies par l'oracle. Soit $y_1, y_2, \ldots, y_n$ les prédictions pour les durées de n tâches. Soit $\eta_i = |x_i - y_i|$ l'erreur de la prédiction pour la durée de la tâche i et $\eta = \sum_{i=1}^{n} \eta_i$ l'erreur totale. On supposera que les durées de tâches sont normalisées et que la durée de la plus petite tâche est d'au moins un.

Exemple : On considére 4 tâches avec des durées rèelles $(x_1, x_2, x_3, x_4) = (1, 1, 1, 2)$ et des durées de prédiction $(y_1, y_2, y_3, y_4) = (1, 1, 1, 1)$. Ici $\eta = 1$.

Définition : Le rapport de *competitivité* d'un algorithm A est défini comme le rapport dans le pire cas de la valeur de la fonction objectif obtenue par l'algorithme A et la valeur de l'optimum (offline).

On considére les algorithmes suivants :

- PREDICTION qui ordonne les tâches de la plus petite à la plus longue en tenant compte de la prédiction de leur durées.
- SPT qui ordonne les tâches de la plus petite à la plus longue en tenant compte de leurs vraies durées.
- ROUND-ROBIN déjà décrit ci-haut.

- PRED-ROUND-ROBIN, qui combine les algorithmes PREDICTION et ROUND-ROBIN de la manière suivante : On choisit $\lambda \in (0, 1)$ et on exécute les deux algorithmes en parallèle. ROUND-ROBIN utilise une vitesse de λ et PREDICTION utilise une vitesse de $(1 - \lambda)$. Si on compare un algorithme A qui exécute les tâches à une vitesse 1 (p.ex. ROUND-ROBIN ou PREDICTION)par rapport au même algorithme A^λ qui exécute les tâches à une vitesse λ, alors les temps de complétude de toutes les tâches dans l'ordonnancement fourni par A^λ seront augmentés d'un facteur de $1/\lambda$.

Exemple (suite) : Pour l'instance donnée dans l'exemple, PREDICTION pourrait donner un ordonnancement où les tâches seraient exécuter dans l'ordre $4, 1, 2, 3$ avec une somme de temps de complétude égale à 2+3+4+5=14. De l'autre coté, SPT donnera un ordonnancement avec une valeur de la fonction objectif égale à 1+2+3+5=11. Cette valeur est l'optimum pour cette instance. ROUND-ROBIN donnera un ordonnancement dont la somme de temps de complétude est égale à 4+4+4+5=17.

Il est demandé de :

1. Implanter l'algorithme PREDICTION.
2. Implanter l'algorithme SPT.
3. Implanter l'algorithme ROUND-ROBIN.
4. Implanter l'algorithme PRED-ROUND-ROBIN.

B) Dans cette deuxième partie, on suppose que chaque tâche a une date d'arrivée r_i. On sait que l'algorithme optimal pour la minimisation de la somme des temps de complétudes, dans le cas où les durées de tâches sont connues à l'avance, est obtenu en donnant à chaque instant la priorité à la tâche dont la durée restante est la plus faible.

Comment peut-on adapter les algorithmes vus à la première partie dans le cas avec prédictions ? Comme précédemment, les prédictions concernent uniquement les durées de tâches. Implanter les différents algorithmes et les comparer en utilisant la même méthodologie.

Simulations : Pour comparer les différents algorithmes, on va utiliser des simulations pour évaluer les rapport de competitivité de chaque algorithme en fonction de l'erreur de la prédiction. Pour ceci, il vous est demandé de générer des données synthétiques avec 50 tâches où la durée de chaque tâche est tirée indépendament selon une distribution à préciser. Pour comparer les différents algorithmes, on posera la prédiction de la durée $y_i = x_i + \epsilon_i$, où ϵ_i est tiré d'une distribution à préciser également.Pour chaque algorithme calculer le rapport de compétitivité en fonction de l'erreur de prédiction.

Dans la littéraure pour générer les durées de tâches, une distribution de Pareto avec un exposant $\alpha = 1.1$. Pour les ϵ_i, une distribution normale avec une moyenne 0 et un écart type σa été utilisée et pour un paramètre σ donné, le rapport de compétitivité sur 1000 essais indépendants où l'erreur de prediction avait l'écart type specifié. *Cependant vous*

êtes libres de concevoir l'évaluation des algorithmes selon votre méthode, notre objectif étant l'évaluation des différents algorithmes par rapport à l'erreur de la prédiction.

Mini-rapport : Un mini-rapport contenant : 1) un exemple de 10 tâches et l'exécution pour chaque algorithme, 2) Les choix pour les données, et les résultats de simulations commentés.