Introduction à l'intelligence artificielle

Partie I : Optimisation bio-inspirée

TP 1 : Algorithmes gloutons

Wahabou Abdou
wahabou.abdou@u-bourgogne.fr
2024 - 2025

Exercice 1 : Réservation de salle d'examen

Un Directeur des études souhaite planifier l'utilisation d'une salle des épreuves prévues lors d'une journée d'examens en sachant que deux épreuves ne peuvent pas avoir lieu dans la salle au même moment. Son objectif est de placer le plus d'épreuves dans la journée et non d'utiliser la salle le plus longtemps possible.

Chaque épreuve est décrite par son intitulé, son horaire de début et son horaire de fin. La figure 1 illustre un exemple de liste de d'épreuves pour lesquelles une réservation est souhaitée. Nous pouvons par exemple noter que les épreuves de Maths et d'Informatique ne pas être toutes les deux placées (il faudra faire un choix).

Le but de cet exercice est d'implémenter un algorithme glouton qui permet de planifier un maximum d'épreuves dans cette salle. Une approche possible consiste trier les épreuves par heures de fin croissantes puis choisir l'épreuve se terminant au plus tôt. Ensuite, choisir l'épreuve se terminant au plus tôt parmi celles qui sont compatibles avec celle(s) qui est (sont) déjà choisie(s). Répéter la sélection des épreuves tant qu'il est possible d'en choisir.

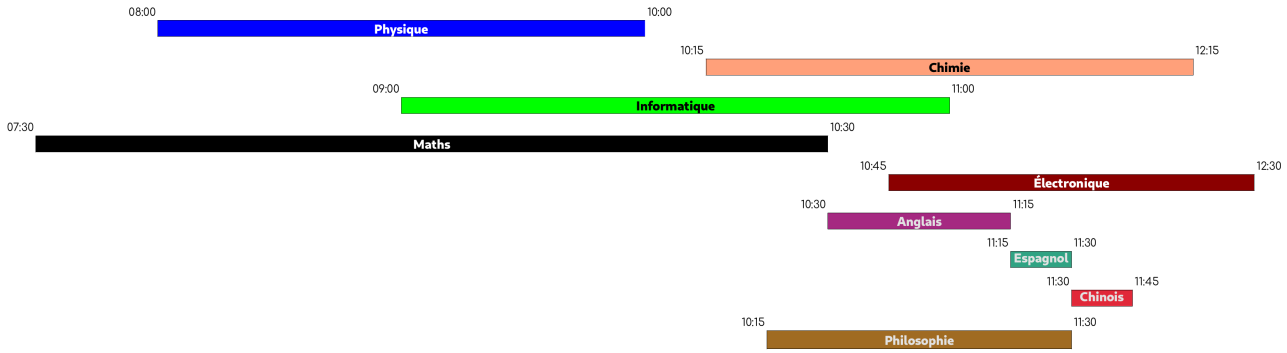

Figure 1 – Exemple d'épreuves à planifier

1. Comment identifier toutes les épreuves d'une liste qui sont en conflit avec une épreuve donnée ?
2. Complétez la méthode `eliminerConflits(Epreuve e)` de la classe `ListeEpreuves` qui permet de supprimer toutes les épreuves en conflit avec celle qui est fournie en paramètre.
3. Testez votre méthode en demandant la suppression de toutes les épreuves qui sont en conflit avec l'épreuve de Physique. Affichez la liste des épreuves restantes.
4. Testez également votre méthode en demandant la suppression de toutes les épreuves qui sont en conflit avec l'épreuve d'Espagnol. Affichez la liste des épreuves restantes.
5. Dans votre classe `Main`, commentez les lignes qui ont servi aux tests des questions 3 et 4.
6. Dans la classe `Main`, implémentez un algorithme glouton qui maximise le placement d'épreuves dans la salle.
7. Modifiez votre algorithme en utilisant un autre critère (autre que la priorisation en fonction des horaires de fin les plus proches) pour placer les épreuves. Vous serez probablement emmené à modifier la méthode `compareTo` de la classe `Epreuve`. Testez votre nouvelle version.

Exercice 2 : Bipartition

On souhaite répartir les n élément d'un ensemble E dans deux sous-ensembles E_1 et E_2 de telle sorte que la somme des éléments de E_1 soit égale à la somme des éléments de E_2 (on essayera de se rapprocher de cette égalité).

1. Implémentez les deux algorithmes proposés durant les TD (ou d'autres versions que vous ferez valider par l'enseignant).
2. Testez votre algorithme en utilisant tour à tour :
   a) $E = \{2, 10, 3, 8, 5, 7, 9, 5, 3, 2\}$
   b) $E = \{771, 121, 281, 854, 885, 734, 486, 1003, 83, 62\}$