Résolution de Problèmes
Algorithme glouton

Marie Pelleau
marie.pelleau@univ-cotedazur.fr

Master 1 - Semestre 1

Algorithme glouton

Définition

- Un **algorithme glouton** est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local, dans l'espoir d'obtenir un résultat optimum global
- Dans les cas où l'algorithme ne fournit pas systématiquement la solution optimale, il est appelé une **heuristique gloutonne**
- La facon dont on fait le choix est parfois appelée **stratégie gloutonne**

Algorithme glouton

Rendre la monnaie avec le minimum de pièces

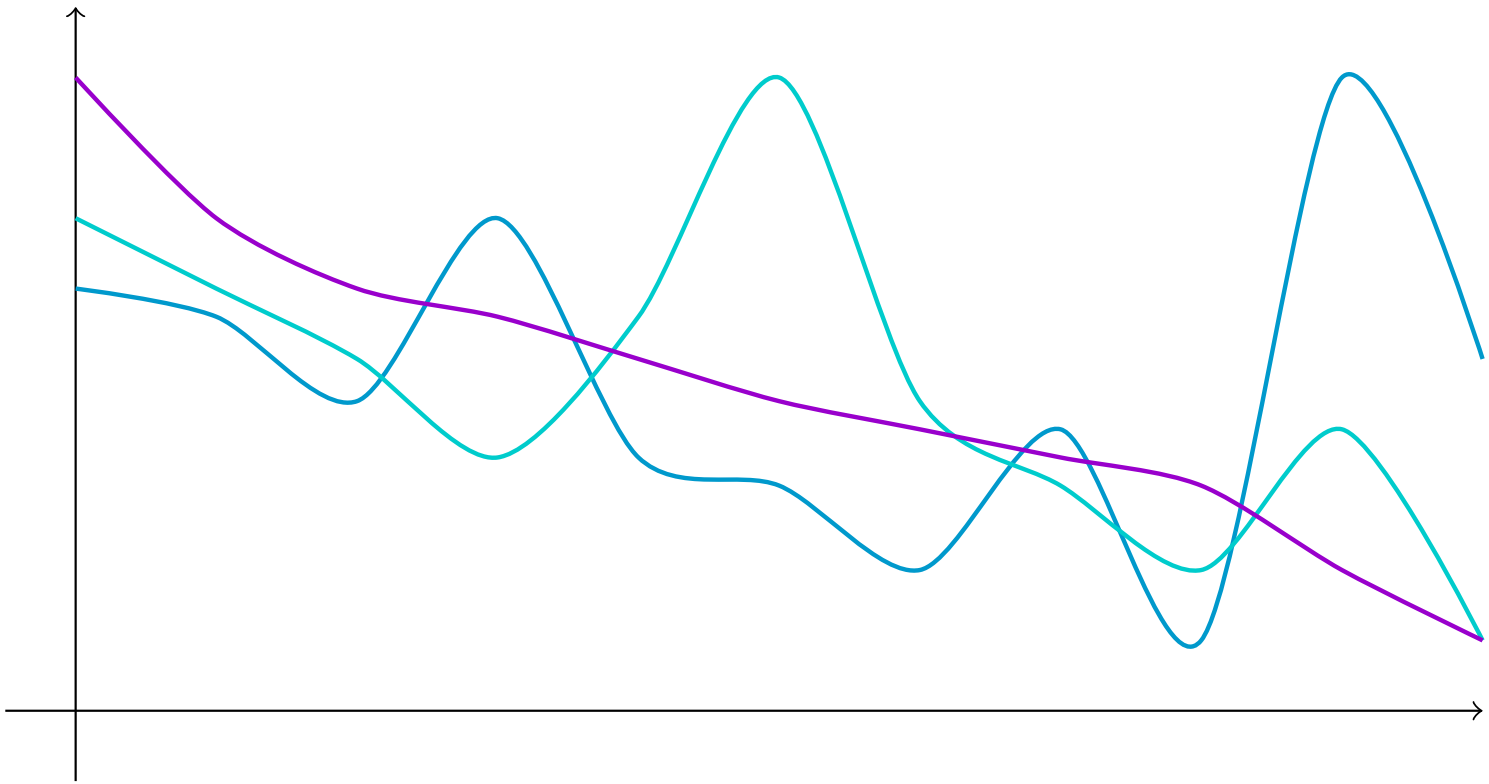
Stratégie gloutonne : à chaque étape on rend la pièce dont le montant est plus petit que ce qu'il reste à rendre et dont la valeur est la plus grande

- 37 centimes à rendre
- La plus grande pièce est 20, on rend 20 et il reste 17
- La plus grande pièce est 10, on rend 10 et il reste 7
- La plus grande pièce est 5, on rend 5 et il reste 2
- On rend 2
- On a donc rendu 20+10+5+2

- Dans le système de pièces européen l'algorithme glouton donne toujours une solution optimale
- Dans le système de pièces (1, 3, 4), l'algorithme glouton n'est pas optimal (pour 6 : 4+1+1, alors que 3+3 est optimal)

Algorithme glouton

- Principe : aller vite
- Difficulté : trouver la bonne stratégie, ou une stratégie efficace



Le sac-à-doc (knapsack)

Description

On a :

- Un Sac dans lequel on peut mettre un poids limité
- Un ensemble d'objets, chaque objet o_i a
  - Un poids : p_i
  - Une valeur : v_i

Quels sont les objets que l'on doit prendre pour maximizer la valeur transportée tout en respectant la contrainte de poids ?

- La somme des valeurs des objets pris est maximale
- La somme des poids des objets pris est $<$ poidsmax du sac

Le sac-à-doc (knapsack)

On a des sacs de métaux précieux o_i et on peut en prendre autant que l'on veut mais moins que p_i

Pour chaque sac on calcule la valeur par gramme

1. On prend le sac ayant le plus grande valeur par gramme et on rempli notre sac à dos avec le plus possible de ce sac
2. Si j'ai atteint le poids limite de mon sac à dos alors j'arrête
3. Sinon, j'ai pris entièrement le contenu d'un sac, j'élimine ce sac et je retourne en 1

Cette stratégie est optimale

Le sac-à-doc (knapsack)

Preuve d'optimalité

- On définit l'efficacité d'un objet o_i : c'est la valeur rapporté par gramme
- Supposons qu'il existe une solution optimale qui ne prend pas un gramme d'un objet o_i et qui prend un gramme d'un objet o_k d'efficacité moindre que o_i
- En échangeant 1 g de o_k par 1 g de o_i on améliore la solution

$\Rightarrow$ Contradiction

Le sac-à-doc (knapsack)

- Mettre des sacs de poudre de métaux précieux revient à accepter de couper des objets
- Si on ne peut pas couper des objets, comment fait-on ?
- Le problème devient difficile
- On fait “comme si”
- On calcule l’efficacité et on prend les objets en fonctions de leur efficacité en respectant la contrainte de poids

Le sac-à-doc (knapsack)

Exemple

On a un sac de capacité maximale 15kg et les objets suivants :

- o_1 de valeur 10 de 9kg
- o_2 de valeur 7 de 12kg
- o_3 de valeur 1 de 2kg
- o_4 de valeur 3 de 7kg
- o_5 de valeur 2 de 5kg

Le sac-à-doc (knapsack)

Modélisation

- Comment représente t'on ce problème ?
- Quelles sont les variables (les inconnues) ?
- Comment exprime t'on les contraintes ?

C'est la modélisation : représentation mathématique du problème

Le sac-à-doc (knapsack)

Les variables

- On associe à chaque objet une variable 0-1 (elle ne prend que les valeurs 0 ou 1)
- C'est une variable d'appartenance au sac à dos
- Si l'objet est pris alors la variable vaut 1 sinon elle vaut 0

Modèle

- La valeur d'un objet et son poids sont des données, donc pour l'objet o_i on a la valeur v_i et le poids p_i
- La variable d'appartenance au sac est x_i
- Le poids maximum du sac est W

Le sac-à-doc (knapsack)

Les contraintes

- $max \sum_{i=1}^{n} v_i x_i$ l'objectif
- $\sum_{i=1}^{n} p_i x_i \leq W$ somme des poids inférieure ou égal au poids maximal

Heuristique

Quand le glouton ne donne pas toujours une solution optimal c'est une **méthode heuristique**

- Vient du grec ancien *eurisko* "je trouve"
- Une heuristique est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation NP-difficile
- Une heuristique est une méthode approximative qui ne donne pas toujours la solution exacte

- On parle bien souvent de méthode heuristique
- Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propree

Évaluation d'une heuristique

Critère pratique, ou empirique

On implémente l'algorithme approximatif et on évalue la qualité de ses solutions par rapport aux solutions optimales (ou aux meilleures solutions connues) sur un banc d'essai (*benchmark*, instances d'un même problème accessible à tous)

Critère mathématique

Il faut démontrer que l'heuristique garantit des performances, il est intéressant de démontrer une garantie probabiliste, lorsque l'heuristique fournit souvent, mais pas toujours, de bonnes solutions

Remarques

- Les critères empiriques et mathématiques peuvent être contradictoires
- Souvent la solution mathématique garantie n'est pas intéressante en pratique

Choix d'activités

Description

- On considère un gymnase dans lequel se déroulent de nombreuses épreuves : on souhaite en "caser" le plus possible, sachant que deux événements ne peuvent avoir lieu en même temps (il n'y a qu'un gymnase)
- Un événement i est caractérisé par une date de début d_i et une date de fin f_i
- On dit que deux événements sont compatibles si leurs intervalles de temps le sont

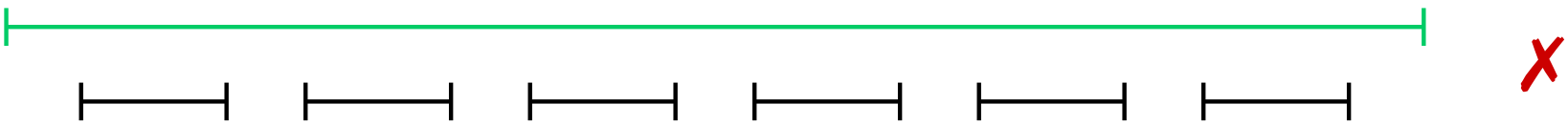
Question : Comment faire pour "caser" le maximum d'événements ?

Choix d'activités

Stratégie 1

On trie les événements par date de début croissante

Contre exemple



Choix d'activités

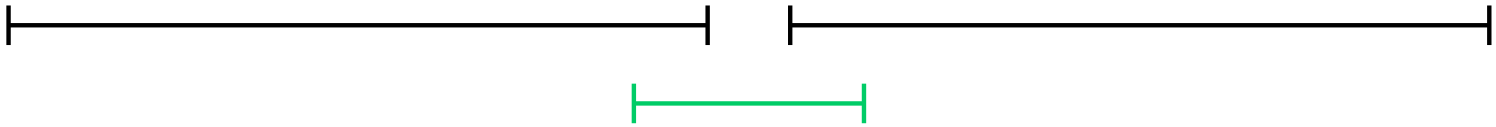
Stratégie 2

On trie les événements par durée croissante

Exemple

✓

Contre exemple



✗

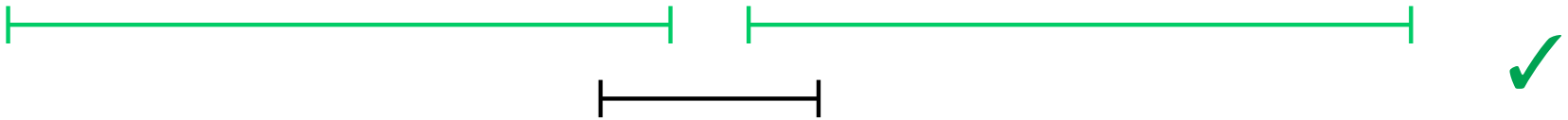
Choix d'activités

Stratégie 3

On trie les événements par date de fin croissante

Exemple

Exemple



Choix d'activités

Stratégie 3

On trie les événements par date de fin croissante

Propriété

Cet algorithme glouton est optimal

Choix d'activités

Preuve par récurrence

- Soit f_1 l'élément finissant le plus tôt, montrons qu'il existe une solution optimale contenant cet événement
- Soit une solution optimale arbitraire $O = \{f_{i1}, f_{i2}, ..., f_{ik}\}$ avec k le maximum d'événements pouvant avoir lieu
- Il y a deux possibilités
  - soit $f_{i1} = f_1$
  - soit $f_{i1} \neq f_1$, on peut remplacer f_{i1} par f_1 car il finit avant tout autre événement, et comme f_{i2} n'intersectait pas avec f_{i1}, alors f_{i2} n'intersecte pas avec f_1
    On peut donc bien trouver une solution optimale ayant comme premier événement f_1
- Ensuite, on ne considère que les événements n'intersectant pas avec f_1, et on réitère la procédure sur les événements restants, d'où la preuve par récurrence

Exercice : Choix d'activités

Description

- On considère un gymnase dans lequel se déroulent de nombreuses épreuves : on souhaite en **maximiser le temps d'utilisation** du gymnase, sachant que deux événements ne peuvent avoir lieu en même temps (il n'y a qu'un gymnase)
- Un événement i est caractérisé par une date de début d_i et une date de fin f_i
- On dit que deux événements sont compatibles si leurs intervalles de temps le sont

Question : Comment faire pour maximiser le temps d'utilisation du gymnase ?