

Les algorithmes gloutons - cours - qkzk

1. Généralités

Optimiser un problème, c'est déterminer les conditions dans lesquelles ce problème présente une caractéristique spécifique. Par exemple, déterminer le minimum ou le maximum d'une fonction est un problème d'optimisation. On peut également citer la répartition optimale de tâches suivant des critères précis, le problème de rendu de monnaie, le problème du sac à dos, la recherche d'un plus court chemin dans un graphe, le problème du voyageur de commerce. De nombreuses techniques informatiques sont susceptibles d'apporter une solution exacte ou approchée à ces problèmes. Certaines de ces techniques, comme l'énumération exhaustive de toutes les solutions, ont un coût machine qui les rend souvent peu pertinentes au regard de contraintes extérieures imposées (temps de réponse de la solution imposé, moyens machine limités).

Les techniques de *programmation dynamique* ou d'*optimisation linéaire*, certains algorithmes numériques peuvent apporter une solution. Les *algorithmes gloutons* constituent une alternative dont le résultat n'est pas toujours optimal. Plus précisément, ces algorithmes déterminent *une* solution optimale en effectuant successivement des *choix locaux*, jamais remis en cause. Au cours de la construction de la solution, l'algorithme résout une partie du problème puis se focalise ensuite sur *le* sous-problème restant à résoudre. Une différence essentielle avec la *programmation dynamique* est que celle-ci peut remettre en cause des solutions déjà établies. Au lieu de se focaliser sur un sous-problème, elle explore les solutions de *tous* les sous-problèmes pour les combiner finalement de manière optimale.

Le principal avantage des *algorithmes gloutons* est leur facilité de mise en œuvre. En outre, dans certaines situations dites *canoniques*, il arrive qu'ils renvoient non pas *un* optimum mais *l'optimum* d'un problème. Nous présentons de telles situations dans la suite, en montrant les avantages mais aussi les limites de la technique.

2. Rendu de monnaie

Un achat dit *en espèce* se traduit par un échange de pièces et de billets. Dans la suite de cet exposé, les pièces désignent indifféremment les véritables pièces que les billets. Supposons qu'un achat induise un rendu de 49 euros. Quelles pièces peuvent être rendues ? La réponse, bien qu'évidente, n'est pas unique. Deux "pièces" de 20 euros, une pièce de 5 euros et deux pièces de 2 euros conviennent. Mais quarante-neuf pièces de 1 euro conviennent également ! Si la question est de *rendre la monnaie avec un minimum de pièces*, le problème change de nature. Mais la réponse reste simple : c'est la première solution proposée. Toutefois, comment parvient-on à un tel résultat ? Quels choix ont été faits qui optimisent le nombre de pièces rendus ? C'est le *problème du rendu de monnaie* dont la solution dépend du *système de monnaie* utilisé.

Dans le système monétaire européen, les pièces prennent les valeurs 1, 2, 5, 10, 20, 50, 100 euros. Pour simplifier nous nous intéressons seulement aux valeurs entières et oublions l'existence des billets de 200 et 500 euros. Rendre 49 euros avec un minimum de pièces est un problème d'optimisation. En pratique, sans s'en rendre compte généralement, tout individu met en œuvre un *algorithme glouton*. Il choisit d'abord la plus grande valeur de monnaie, parmi 1, 2, 5, 10, 20, 50, contenue dans 49 euros. En l'occurrence, deux fois la pièce de 20 euros. La somme de 9 euros restant à rendre, il choisit une pièce de 5 euros puis deux pièces de 2 euros. Cette stratégie gagnante pour la somme de 49 euros l'est-elle pour n'importe quelle somme à rendre ? On peut montrer que la réponse est positive pour le système monétaire français. Pour cette raison, un tel système de monnaie est qualifié de *canonique*.

D'autres systèmes ne sont pas canoniques. L'algorithme glouton ne répond alors pas de manière optimale. Par exemple, avec le système $\{1, 3, 6, 12, 24, 30\}$, l'algorithme glouton répond en proposant le rendu $49 = 30 + 12 + 6 + 1$, soit 4 pièces alors que la solution optimale est $49 = 2 \times 24 + 1$ soit trois pièces. La réponse à cette difficulté passe par la *programmation dynamique*, thème abordé en terminale.

2.1 Un algorithme glouton

Considérons un ensemble de n pièces de monnaie de valeurs :

$$v_1 < v_2 < \dots < v_n$$

avec $v_1 = 1$. On suppose que ce système est canonique. On peut noter le système de pièces :

$$S_n = \{v_1, \dots, v_n\}$$

Désignons par s une somme à rendre avec le minimum de pièces de S_n . L'algorithme glouton sélectionne la plus grande valeur v_n et la compare à s .

- Si $s < v_n$, la pièce de valeur v_n ne peut pas être utilisée. On reprend l'algorithme avec le système de pièces S_{n-1} .
- Si $s \geq v_n$, la pièce v_n peut être utilisée une première fois. Ce qui fait une première pièce à comptabiliser, de valeur v_n , la somme restant à rendre étant alors $s - v_n$. L'algorithme continue avec la même système de pièces S_n et cette nouvelle somme à rendre $s - v_n$.

L'algorithme est ainsi répété jusqu'à obtenir une somme à rendre nulle.

Remarque. Il s'agit effectivement d'un algorithme glouton, la plus grande valeur de pièce étant systématiquement choisie si sa valeur est inférieure à la somme à rendre. Ce choix ne garantit en rien l'optimalité globale de la solution. Le choix fait est considéré comme pertinent et permet d'avancer plus avant dans le calcul. Toutefois, comme nous l'écrivions plus haut, si le système monétaire est canonique, la solution est optimale. Pour savoir si le système est canonique,

on peut se contenter du principe suivant : la somme des n premières pièces doit être inférieure à la pièce $n + 1$ ($1 + 2 < 5$, $1 + 2 + 5 < 10$ etc.)