Pb d'optimisation du sac à dos

- Caractéristiques des objets à cambrioler;

objet	A	B	C	D
masse	13 Kg	12 Kg	8 Kg	10 Kg
valeur marchande	700 €	400 €	300 €	300 €

- Variable de décision binaire; l'objet est mis dans le sac (1) ou non (0)
- Contrainte; capacité du sac du cambrioleur (30 kg)
- Objectif; maximiser le gain du cambriolage

Pb du sac à dos

Sans coder, déterminer plusieurs solutions possibles;

1. Une solution réalisable non optimale
2. Une solution non réalisable
3. Une solution optimale localement; si A est mis dans le sac, quel est la meilleure solution?
4. Une ou la solution optimale?

Pb du sac à dos

Plusieurs solutions possibles;

1. Une solution réalisable non optimale; A
2. Une solution non réalisable; A, B, C, D ou 2 A
3. Une solution optimale localement; si A est mis dans le sac, quel est la meilleure solution?
4. Une solution optimale; ?

Pb du sac à dos

1. Une solution réalisable non optimale (A); les contraintes ne sont pas saturées (il reste de la place dans le sac)
2. Une solution non réalisable (A, B, C, D ou 2 A). Une contrainte n’est pas satisfaite

Pb du sac à dos

3. Une solution optimale localement (A, B); si on procède par étape (sans retour arrière sur une décision) avec un algorithme « glouton », une solution optimale localement n’est pas nécessairement une solution optimale globalement

Pb du sac à dos

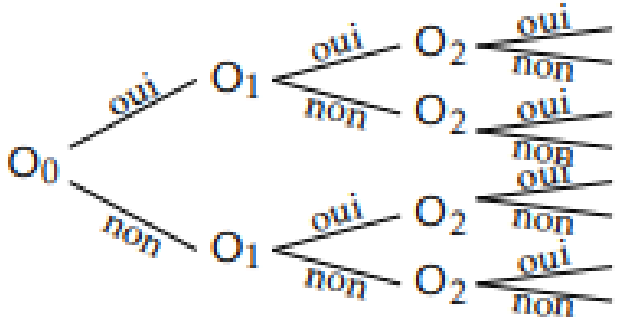
4. Une solution optimale se calcule soit par un algorithme de type « force brute » (en énumérant toutes les solutions possibles) ou par programmation linéaire (pulp) ou programmation dynamique (algo récursif)

Pb du sac à dos - complexité

Combien de combinaisons d'objets à considérer sans tenir compte de la capacité du sac?

Pb du sac à dos - complexité

Si n représente le nombre total d'objets, il y a 2^n solutions à explorer;



O_0	O_1	O_2
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Fig. 11.4 – Représentation des huit solutions de rangement dans le cas $n = 3$.

Pb du sac à dos – Complexité

Complexité exponentielle en $O(e^n)$;

- e est une constante
- n est la « taille » du pb (ici le nombre d'objets)

Le temps de calcul pour résoudre ce type problème en énumérant toutes les solutions (2^N) croît exponentiellement avec sa taille.

Méthode gloutonne - définition

Le problème est résolu par étape (sans retour arrière), à chaque étape on prend un choix « localement optimale » selon une règle de décision choisie (il en existe plusieurs)

Algorithme glouton - démo

1. Calculer la valeur massique de chaque objet

valeur massique des objets

objet	A	B	C	D
valeur massique	54 €/Kg	33 €/Kg	38 €/Kg	30 €/Kg

2. Trier les objets par valeur massique décroissante**; A, C, B, D**

Algorithme glouton - démo

3. Construire une solution réalisable par étape; tant que les contraintes sont satisfaites (capacité du sac), on ajoute un objet en le choisissant localement de façon optimale (max valeur massique);
   - 1ère étape; A (capacité 30-13=17)
   - 2ème étape; …
4. Afficher la solution

Algorithme glouton - démo

- 1ère étape; A (capacité 30-13=17)
- 2ème étape; C (capacité 17-8=9)
- 3ème étape;
  B (capacité 9-12=-3)
  C (capacité 9-10=-1)
- FIN
- Solution;
  item dans le sac; A, C
  sac de 21 KG valant 1000$

Algorithme glouton - démo

- A,C n’est pas une solution optimale globale
- Pas de retour arrière dans un algo glouton; les objets mis dans le sac à une étape donnée, reste dans le sac (A)
- À une étape donnée; l’algo glouton choisit la meilleure solution réalisable localement

Algorithme glouton - code

- Structure de données; dictionnaires de données (POO ou non)
- Trier un dictionnaire avec sorted() et une expression lambda
- Choisir une règle gloutonne