Première NSI - Algorithmique

Les algorithmes gloutons - 4. Cours

qkzk

2020/08/01

Principe général d'un algorithme glouton

Généralités

Les algorithmes dits *gloutons* (en anglais **greedy algorithm**) servent à résoudre certains problèmes d'**optimisation**.

Un problème d'optimisation : on cherche à construire une solution à un problème qui optimise une **fonction objectif**. Un problème d'optimisation se définit comme :

- un ensemble fini d'éléments, E,
- une solution au problème est construite à partir des éléments de E : c'est par exemple une partie de E ou un multi-ensemble d'éléments de E ou une suite (finie) d'éléments de E ou une permutation de E qui satisfait une certaine contrainte.
- à chaque **solution** S est associée une fonction objectif $v(S)$: on cherche donc une solution qui maximise (ou minimise) cette fonction objectif.

On peut utiliser un algorithme d'approximation pour résoudre le problème d'optimisation : il fournit toujours une solution mais pas forcément une solution optimale. Bien sûr, on souhaite qu'il soit efficace.

Un algorithme d'approximation peut être:

- déterministe - pour une entrée donnée, il donnera toujours la même solution (heuristiques gloutonnes, optimum local, tabou...)
- non déterministe : recuit simulé, algorithme génétique...

Le principe d'une méthode gloutonne :

- Avaler tout ce qu'on peut = Construire au fur et à mesure une solution en faisant les choix qui paraissent optimaux localement

On procède de façon séquentielle, en faisant à chaque étape le choix qui semble localement le meilleur. * On ne revient jamais en arrière. * Il s'agit d'une progression *descendante*, à chaque étape on fait un choix puis on résoud un problème plus petit issu de ce choix.

Dans certains cas, cela donnera finalement la meilleure solution : on parlera d'algorithmes gloutons exacts.

Dans d'autres, non, on parlera d'heuristiques gloutonnes.

En général, le fait que le résultat soit correct est facile, le fait qu'il soit optimal n'est pas évident.

Le schéma de la méthode gloutonne

Il est basé sur un critère local de sélection des éléments de E pour construire une solution optimale. En fait, on travaille sur l'objet "solution partielle" - "début de solution"- et on doit disposer de :

- `select` : qui choisit le meilleur élément restant selon le critère glouton.
- `complete?` qui teste si une solution partielle est une solution (complète).
- `ajoutPossible?` qui teste si un élément peut être ajouté à une solution partielle, i.e. si la solution partielle reste un début de solution possible après l'ajout de l'élément. Dans certains cas, c'est toujours vrai !
- `ajout` qui permet d'ajouter un élément à une solution si c'est possible.

Schémas d'algo glouton

```
// on initialise l'ensemble des "briques"
// élémentaires des solutions.
Ens.init() ;
// on initialise la solution :
// ensemble (ou suite) "vide" ou..
Sol.Init() ;
while (Non Sol.complete ?() et Ens.NonVide ?()) do
    //on choisit x selon critère glouton
    x ← Ens.select();
    if Sol.ajoutPossible(x) then
        Sol.ajout(x) ; fsi ;
    //dans certains problèmes, toujours le cas
    if CertainesConditions then
        Ens.retirer(x) ;
    // selon les cas, x considéré une fois ou plus
end
// la Solution partielle est a priori complète
return Sol ;
```

Autre schéma :

```
Algorithme vorace
Début
S <- EnsembleVide
C <- ensemble des candidats à la solution
Tant que S n'est pas une solution et C <> EnsembleVide Faire
    x <- choisir un élément de C le plus prometteur
    C <- C - x
    Si realisable(solution,x) Alors
        solution <- union(solution, x)
    Finsi
FinTantQue
Si S est une solution Alors
    Retourner S
Sinon
    Retourner pas de solution
FinSi
```

Pour sélectionner, on trie souvent tout simplement la liste des éléments selon le critère glouton au départ ; on balaye ensuite cette liste dans l'ordre.

Ceci est un schéma général qui a l'avantage et les inconvénients d'un schéma : dans certains cas, c'est encore plus simple ! par exemple, lorsque la solution recherchée est une permutation, en général l'algorithme se réduit au tri selon le critère glouton ! dans d'autres cas, les "solutions" sont un peu plus compliquées et on a besoin d'un schéma un peu plus sophistiqué.