Algorithmes gloutons

N. TSOPZE

Département d'Informatique - Université de Yaoundé I

Plan du chapitre

un problème $\Rightarrow$ plusieurs solutions
une solution $\Rightarrow$ une valeur

problème d'optimisation

recherche d'une solution de valeur optimale (minimum ou maximum)

algorithme de résolution:

- reconnaissance d'une solution
- évaluation d'une solution
- sélection d'une des meilleures solutions

Optimisation

Cas possibles:

1. problème de petite taille $\Rightarrow$ énumération possible
2. problème de grande taille $\Rightarrow$ énumération impossible
3. problèmes faciles $\Rightarrow$ algorithme de complexité polynomiale
4. problèmes "difficiles" $\Rightarrow$ tous les algorithmes connus sont de complexité exponentielle

Plan du chapitre

étapes

1. Transformation du problème d'optimisation en un problème dans lequel on fait un choix à la suite duquel on se retrouve avec un seul sous-problème à résoudre.
2. Démonstration qu'il y a toujours une solution optimale du problème initial qui fait le choix glouton, de sorte que le choix glouton est toujours approprié.
3. Démonstration que, après avoir fait le choix glouton, on se retrouve avec un sous-problème tel que, si l'on combine une solution optimale du sous-problème et le choix glouton que l'on a fait, on arrive à une solution optimale du problème originel.

Deux éléments clefs: propriété du choix glouton + sous-structure optimale

choix glouton

idée

arriver à une solution globalement optimale en effectuant un choix localement optimal

1. quand on considère le choix à faire, on fait le choix qui paraît le meilleur pour le problème courant, sans tenir compte des résultats des sous-problèmes;
2. faire le choix qui semble le meilleur pour l'instant, puis résoudre le sous-problème qui survient après le choix;
3. Choix ne dépend jamais de futur.

Propriété

Un problème exhibe une sous-structure optimale si une solution optimale du problème contient les solutions optimales des sous-problèmes.

résumé

remarque 1: faire toujours le choix qui semble être le meilleur pour l'instant;

remarque 2: Choix localement optimal espèrant qu'il sera la solution globalement optimal;

remarque 3: Possiblité de ne pas toujours obtenir la solution optimale;

remarque 4: Tous les problème n'admettent pas une solution gloutonne.

Plan du chapitre

Problème du choix d'activités

problème du sac à dos

contexte $S = \{a_1, a_2, ..., a_n\}$ n activités proposées qui veulent utiliser une ressource

a_i a une heure de début s_i et une heure de fin f_i, $(0 < s_i < f_i < \infty)$.

compatibilité a_i et a_j sont compatibles si les intervalles $[s_i, f_i)$ et $[s_j, f_j)$ ne se chevauchent pas

problème sélectionner un sous-ensemble, de taille maximale, d'activités mutuellement compatibles

Exemple

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

Cas 1: $\{a_3, a_9, a_{11}\}$ est un ensemble d'activités mutuellement compatibles, pas un sous-ensemble maximum,

cas 2: $\{a_1, a_4, a_8, a_{11}\}$ plus grand est un sous-ensemble maximum d'activités mutuellement compatibles

- $S_{ij} = \{a_k \in S / f_i \leq s_k < f_k \leq s_j\}$
- ajout de a_0 et a_{n+1} telles que: $f_0 = 0$ et $s_{n+1} = \infty$; alors, $S = S_{0,n+1}$ et les intervalles de i et j sont donnés par $0 \leq i, j \leq n+1$.
- trier les activités : $f_0 \leq f_1 \leq f_2 \leq ... \leq f_n < f_{n+1}$

principe

toute solution d'un sous-problème non vide S_{ij} contient une certaine activité a_k, et que toute solution optimale contient en elle des solutions optimales des instances de sous-problème S_{ik} et S_{kj}.
$A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$

1. $n \leftarrow longueur(s)$
2. $A \leftarrow \{a_1\}$
3. $i \leftarrow 1$
4. pour m allant de 2 à n faire
    1. si $s_m \leq f_i$ alors
        1. $A \leftarrow A \cup \{a_m\}$
        2. $i \leftarrow m$
5. retourner A

sac à dos

Entrées : $X = \{x_1, x_2, ..., x_n\}$ donc l'objet i a un poids p_i, $x_i \in \{0, 1\}$

d'un sac donc le poids max W

Sortie : $X_1 \subset X$ tel que $\forall x_i \in X_1 \sum_{i=1}^{|X_1|} x_i p_i \leq W$

Problème: Comment trouver X_1 ?

Deux variantes:

1. variable entière
2. variable fractionnaire

Variantes

Variable entière : considérer le chargement de valeur maximale pesant au plus W. Si l'on retire l'objet j du sac, le chargement restant doit être le meilleur que puisse contenir le sac pour un poids maximum de $W - w_j$ à partir des $n - 1$ objets initiaux, j étant exclus

Variable fractionnaire : considérer que si l'on retire un poids w d'un objet j dans le chargement optimal, le reste du chargement doit être le meilleur que le sac puisse contenir pour un poids maximum de $W - w$ à partir des $n - 1$ objets initiaux, et des $w_j - w$ kilos de l'objet j.