Optimisation combinatoire

Définition

Beaucoup de problème d'ordre pratique ou théorique nécessite de prendre, parmi un *ensemble de choix* possibles (très large), le *meilleur choix* selon un *critère* donné.

Remarque

Domaine largement étudié en informatique, en mathématiques appliquées, en sciences de gestion, en génie industriel…

Objet

Etudier un ensemble de *modèles* et *méthodes* de résolution (méthodes dites gloutonnes, programmation dynamique, métaheuristiques, etc.)

Plan

1. Introduction
2. Modèles
3. Méthodes de résolution
    - Glouton
    - Programmation dynamique
    - Méthodes heuristiques / métaheuristiques
4. Applications

1. Introduction

Exemples d'applications — Affectation de ressources

Recouvre un grand nombre de problèmes réels tels que l'affectation de chauffeurs, l'affectation des fréquences sur les antennes de télécommunications pour minimiser les interférences dans les réseaux de mobiles, ou le positionnement d'antennes. Les critères sont : le nombre de sites et la rentabilité les sites choisis. D'autres problèmes : la planification de prise de vue de Spot 5 (satellite avec 3 caméras), les tournées de véhicule ou la bioinformatique.

2. Modèles

2.1 PL (Programmation Linéaire)

Problème facile

2.2 PLNE (Programmation Linéaire en Nombre Entier)

cf. PLNE 0/1, complexité exponentielle NP-complet, méthode de résolution Branch & Bound

2.3 PNL (Programmation Non Linéaire)

2.4 Programmation mixte

2.5 Graphes

- Coloration
- Clique
- Ensemble indépendant
- PVC

2.6 CSP, Max-CSP, CSOP

• CSP <V,D,C> — Constraint Satisfaction Problem : Trouver une affectation de valeurs aux variables de manière à satisfaire toutes les contraintes.

V : ensemble de variables
D : collection de domaines de valeurs
C : contraintes

• Max-CSP : Trouver une affectation de valeurs aux variables de manière à maximiser le nombre de contraintes satisfaites.

• CSOP — Constraint Satisfaction and Optimization Problem

3. Méthodes de résolution

1. Spécifiques : tris, Dijkstra, Prim…
2. Génériques : glouton, B&B, programmation dynamique

A. Exactes : garantie la solution optimale pour un problème d'optimisation combinatoire
B. Approchées : pas de garantie sur l'optimalité des solutions trouvées, approximation.

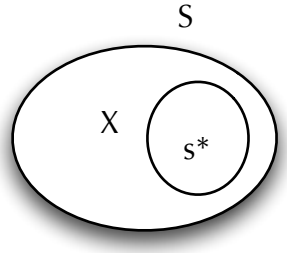
3.1 Problèmes vs. instances

Définition

Une instance d'un problème d'optimisation combinatoire est définie par un couple <S, f> où
S : ensemble de *configurations* (espace de recherche)
f : fonction S -> R, fonction objective (de coût)

Déterminer $s^* \in S$ tel que $\forall s \in X \subseteq S, f(s^*) \leq f(s)$ (*minimisation*)

s^* est la solution optimale de <S, f>



Remarques

1. X = ensemble de solutions réalisables (faisables) vérifiant les contraintes du problème
2. Configurations (*points*) vs. *solutions*
3. Maximisation
4. s^* n'est pas forcément unique
5. En pratique, traiter un problème (pratique) nécessite d'abord l'identification de S (et f) : Modélisation

3.2 Glouton (Greedy Method)

1. Prim (Krustal) pour le problème d'arbre couvrant de poids maximum d'un graphe : construction d'un arbre en fonction d'un critère de choix

2. Max-flot dans un réseau

Propriétés à vérifier pour appliquer le principe glouton

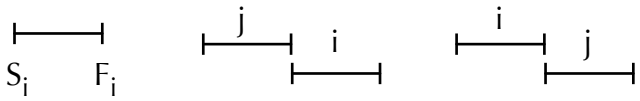
- sous-structure optimale
- "choix glouton" : une solution optimale (globale) peut-être obtenue par une série de choix localement optimaux

Principe des algorithmes gloutons

Faire une série de choix selon un critère de manière irrévocable, pour chaque choix, prendre le choix le plus favorable.

Exemple : Problème de sélection d'activités — < E, f > et E = 2^S

Soit S = {1, 2,…, n}, n activités en compétition pour occuper une ressource (e.g. salle de cours) et pour chaque activité i :



- début S_i et fin F_i ($S_i \leq F_i$).
- i et j sont compatibles si $S_i \geq F_j$ ou $S_j \geq F_i$.

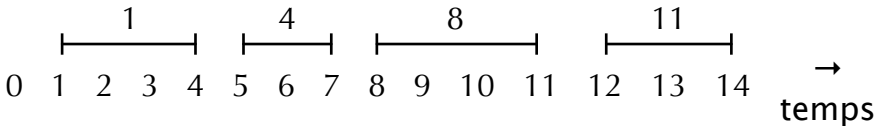
Déterminer un sous-ensemble A ⊆ S de cardinalité maximum tel que les activités de A sont compatibles deux à deux. A = max { |S'| \ S' ⊆ S et ∀i,j ∈ S', i et j sont compatibles }

Solution gloutonne — Idée

A chaque itération, l'activité à sélectionner est, parmi les activités restantes et compatibles avec activités déjà plantifiées, *l'activité i telle que fi est la plus faible* (qui finit le plus tôt).

Exemple

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



Algorithme

$F_1 \leq F_2 \leq \ldots \leq F_n$
A ← { 1 } /* selon le critère de choix, on commence avec l'activité 1 */
j ← 1 /* dernière activité selectionnée */
pour i ← 2 ***à*** n ***faire***
  si $S_i \geq F_j$ ***alors*** /* i est compatible avec les j de A */
    A ← A ∪ {i}
    j ← i
  fin si
fin pour

Optimalité de la solution

Soit S = {1, 2, …, n} l'ensemble des activités telles que $F_1 \leq F_2 \leq \ldots \leq F_n$, on veut démontrer qu'il existe une solution optimale qui applique le choix glouton défini précédemment, i.e. qui commence par l'activité 1.

Démonstration

Soit $A \subseteq S$ une solution optimale, on trie les activités de A selon l'ordre croissant de F_i ($i \in A$).

Supposons que la première activité de A est k.

(1) k = 1, OK car A suit le critère de choix

(2) $k \neq 1$, on va montrer qu'il existe une autre solution optimale B qui commence forcément par l'activité 1. Prenons $B = A - \{k\} \cup \{1\}$

- Puisque $F_1 \leq F_k$, les activités de B sont disjointes.
- Puisque $|B| = |A|$, B est donc une solution optimale et B contient aussi l'activité 1.

$\Rightarrow \exists$ toujours une solution optimale commençant par l'activité 1.

(3) Si A est optimale pour S, alors $A' = A - \{1\}$ est optimale pour $S' = \{ i \in S \mid S_i \geq F_1 \}$ (sous-problème). En effet, s'il existe une autre solution optimale B' pour S' telle que $|B'| > |A'|$ alors $B' \cup \{1\}$ est une solution optimale pour S et $|B' \cup \{1\}| > |A|$, contradiction avec le fait que A est optimale ($|A|$ maximum)

Remarques

1. Le critère de choix dépend du problème à résoudre et doit être déterminé "intelligemment".
2. En général, les algorithmes gloutons ne garantissent pas l'optimalité de la solution, c'est particulièrement vrai quand le problème à résoudre est difficile.
3. Les algorithmes gloutons sont rapides.
4. Les algorithmes gloutons se combinent très bien avec les méthodes de résolution telles que la recherche locale et les algorithmes évolutionnaires

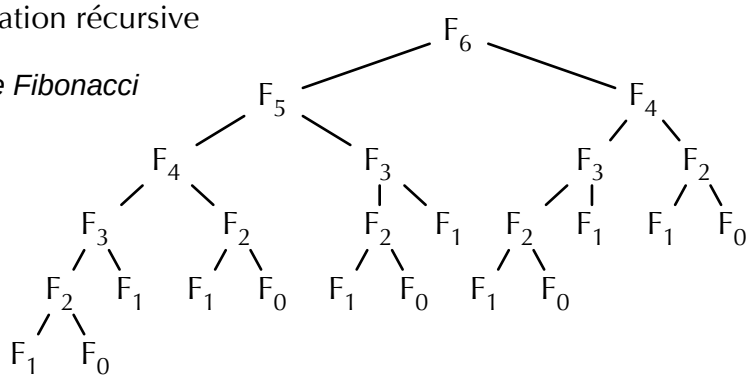
3.3 Programmation dynamique

Propriétés à vérifier

• Sous-structure optimale : relation récursive

Exemples : calcul du nombre de Fibonacci

$F_n = F_{n-1} + F_{n-2}$
$F_0 = 0$
$F_1 = 1$



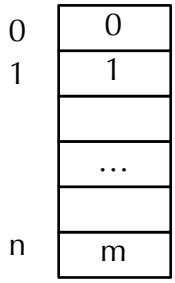
Solution récursive

Complexité $O(1.6^n)$

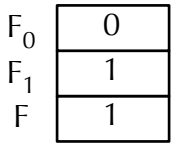
Solution itérative

Complexité O(n)

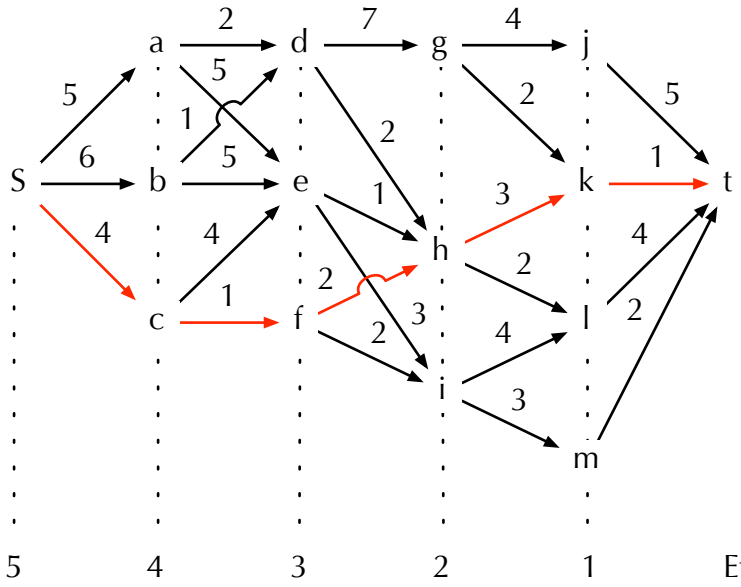
F[0] = 0
F[1] = 1
for i = 2 ***to*** n ***do***
F[i] = F[i-1] + F[i-2]



$F_0 = 0$
$F_1 = 1$
for i = 2 ***to*** n ***do***
$F = F_0 + F_1$
$F_0 = F_1$
$F_1 = F$



Problème pour le plus court chemin



Table[i] indique la décision optimale pour chaque sommet de l'étape i d'aller vers la destination t.

Table[1] =	sommet	j	k	l	m
	suivant	t	t	t	t
	coût	5	1	4	2

Table[2] =	sommet	g	h	i
	suivant	k	k	m
	coût	3	4	5

Table[3] =	sommet	d	e	f
	suivant	h	h	h
	coût	6	5	6

Table[4] =	sommet	a	b	c
	suivant	d	d	f
	coût	8	7	7

Table[5] =	sommet	S
	suivant	c
	coût	11