Problèmes d'optimisation

une approche gloutonne

{Benjamin.Wack,Jean-Marc.Vincent}@univ-grenoble-alpes.fr

Laboratoire LIG, Équipe Inria POLARIS

DIU EIL
Grenoble 2020

ALGORITHMIQUE ET MODÉLISATION

1 OPTIMISATION

2 LE PROBLÈME D'ALLOCATION

3 STRATÉGIE GLOUTON

4 EXERCICES

5 LE PROBLÈME DE L'ARBRE COUVRANT

PROBLÈME D'OPTIMISATION

Un problème d'optimisation a les caractéristiques suivantes :

- Une solution est un **sous-ensemble** d'une des données du problème
- Il existe en général plusieurs solutions **admissibles**
- À chaque solution (admissible) est associée une **valeur** (en général un coût ou un gain)

Le problème d'optimisation consiste non seulement à trouver une solution admissible, mais à trouver une solution de valeur **minimale** (pour un coût) ou **maximale** (pour un gain).

PROBLÈME D'OPTIMISATION

Un problème d'optimisation a les caractéristiques suivantes :

- Une solution est un **sous-ensemble** d'une des données du problème
- Il existe en général plusieurs solutions **admissibles**
- À chaque solution (admissible) est associée une **valeur** (en général un coût ou un gain)

Le problème d'optimisation consiste non seulement à trouver une solution admissible, mais à trouver une solution de valeur **minimale** (pour un coût) ou **maximale** (pour un gain).

Un exemple : le rendu de monnaie

- **Problème** : On dispose de pièces de valeurs $v_1, v_2, \ldots$ (en nombre illimité).
  On cherche à fournir une somme d'argent S en monnaie.
- **Solution** : une suite de (valeurs de) pièces ayant pour total S
- **Meilleure solution** : celle utilisant le moins possible de pièces

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

STRATÉGIE GLOUTONNE

Il s'agit d'**une stratégie particulière** pour résoudre un problème d'optimisation.

STRATÉGIE GLOUTONNE

Il s'agit d'**une stratégie particulière** pour résoudre un problème d'optimisation.

Algorithme glouton

Un **algorithme glouton** est un algorithme qui construit une telle solution :

- élément par élément **sans jamais revenir en arrière**
- en se basant sur des considérations **locales**.

La séquence construite est un optimum global ; ses sous-séquences sont des optimums des sous-problèmes.

STRATÉGIE GLOUTONNE

Il s'agit d'**une stratégie particulière** pour résoudre un problème d'optimisation.

Algorithme glouton

Un **algorithme glouton** est un algorithme qui construit une telle solution :

- élément par élément **sans jamais revenir en arrière**
- en se basant sur des considérations **locales**.

La séquence construite est un optimum global ; ses sous-séquences sont des optimums des sous-problèmes.

Remarque :

- Il n'existe pas toujours un algorithme glouton pour résoudre un problème d'optimisation.
- Il s'agit d'une caractéristique de l'algorithme mais surtout de la structure du problème (notamment c'est impossible s'il existe des optima locaux).

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ALGORITHME GLOUTON POUR LE RENDU DE MONNAIE

Algorithme glouton

1. Tant que $S > 0$:
2. Choisir la pièce de plus grande valeur v inférieure à S
3. Recommencer avec $S - v$.

ALGORITHME GLOUTON POUR LE RENDU DE MONNAIE

Algorithme glouton

1. Tant que $S > 0$:
2. Choisir la pièce de plus grande valeur v inférieure à S
3. Recommencer avec $S - v$.

Pièces du système européen

$S = 16$ avec pièces de 10, 5, 2, 1 :

- $16 - 10 = 6$
- $6 - 5 = 1$
- $1 - 1 = 0$

3 pièces

ALGORITHME GLOUTON POUR LE RENDU DE MONNAIE

Algorithme glouton

1. Tant que $S > 0$:
2. Choisir la pièce de plus grande valeur v inférieure à S
3. Recommencer avec $S - v$.

Pièces du système européen

$S = 16$ avec pièces de 10, 5, 2, 1 :

- $16 - 10 = 6$
- $6 - 5 = 1$
- $1 - 1 = 0$

3 pièces

Exemple curieux

$S = 16$ avec pièces de 9, 8 et 1 :

- $16 - 9 = 7$
- $7 - 1 = 6$
- ...

8 pièces alors que $8 + 8 = 16$ en 2 pièces
et pour le système américain ?

Un système de valeurs de pièces pour lesquelles la stratégie gloutonne est optimale pour toute somme S est dit **canonique**

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ALGORITHMIQUE ET MODÉLISATION

1 OPTIMISATION

2 LE PROBLÈME D'ALLOCATION

3 STRATÉGIE GLOUTON

4 EXERCICES

5 LE PROBLÈME DE L'ARBRE COUVRANT

ALLOCATION

Le problème

- ensemble d'objets : tâches, périodes, espace mémoire,...
- choisir un sous-ensemble d'objets
- contraintes : incompatibilité, séquencement,
- fonction à optimiser : temps d'exécution, utilisation de mémoire, recouvrement,...

Énoncé d'un problème

On dispose d'une salle pouvant être louée pour une durée variable.
On choisit parmi un ensemble de n demandes de location celles qui seront acceptées.

- Données : les dates de début d_i et de fin f_i de chaque demande i.
- Solution optimale = satisfaisant le plus de demandes.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ALGORITHMIQUE ET MODÉLISATION

1 OPTIMISATION

2 LE PROBLÈME D'ALLOCATION

3 STRATÉGIE GLOUTON

4 EXERCICES

5 LE PROBLÈME DE L'ARBRE COUVRANT

ALGORITHME GLOUTON GÉNÉRIQUE

Initialiser un ensemble *Utilisables*
Initialiser *Sol* := ∅ `// ensemble vide ou suite vide ou ...`

while *Sol* **est incomplète et** *Utilisables* **n'est pas vide do**
- Sélectionner *x* dans *Utilisables* `// selon critère glouton`
- **if** *x* **compatible avec** *Sol* **then** `// dans certains problèmes`
  `// c'est toujours le cas`
  - Ajouter *x* à *Sol*
- **if condition then** `// true ou x incompatible ou ...`
  - Retirer *x* de *Utilisables*

Renvoyer *Sol*

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

COMPLEXITÉ

Le choix du prochain élément est en principe efficace car il ne considère qu'une partie des données :

- $\log n$ dans une FAP
- 1 pour les pièces de monnaie (si leurs valeurs sont préalablement triées)
- ...

COMPLEXITÉ

Le choix du prochain élément est en principe efficace car il ne considère qu'une partie des données :

- $\log n$ dans une FAP
- 1 pour les pièces de monnaie (si leurs valeurs sont préalablement triées)
- ...

La complexité de l'algorithme sera en général de la forme

$$\mathcal{O}(n \times f(n))$$

où f (le coût du choix) est une fonction sub-linéaire.

PREUVE DE CORRECTION

Schéma de preuve générique

Pour la boucle principale, on maintient l'invariant :

Il existe une solution optimale dont *Sol* est un préfixe.

La preuve de cet invariant se fait en utilisant les propriétés de l'opération **Sélectionner** et de la mise à jour de *Utilisables*.

PREUVE DE CORRECTION

Schéma de preuve générique

Pour la boucle principale, on maintient l'invariant :

Il existe une solution optimale dont *Sol* est un préfixe.

La preuve de cet invariant se fait en utilisant les propriétés de l'opération **Sélectionner** et de la mise à jour de *Utilisables*.

Lorsqu'il y a plusieurs solutions optimales, il est souvent nécessaire de faire appel à une propriété « d'échange » :

Soit S' une solution qui diffère de la gloutonne *Sol* en construction par le dernier choix effectué : on peut transformer S' en une autre solution aussi bonne et qui contient *Sol*.
Supposons que $Sol \subseteq S$ une solution optimale et que le choix glouton ajoute à *Sol* un élément $x \notin S$.

Alors il existe un élément $y \in S \backslash Sol$ tel que $S \cup \{x\} \backslash \{y\}$ soit une autre solution au moins aussi bonne que S.

PROBLÈME DE CHOIX DES ACTIVITÉS

Énoncé du problème

On dispose d'une salle pouvant être louée pour une durée variable.
On choisit parmi un ensemble de n demandes de location celles qui seront acceptées.

- Données : les dates de début d_i et de fin f_i de chaque demande i.
- Solution optimale = satisfaisant le plus de demandes.

PROBLÈME DE CHOIX DES ACTIVITÉS

Énoncé du problème

On dispose d'une salle pouvant être louée pour une durée variable.
On choisit parmi un ensemble de n demandes de location celles qui seront acceptées.

- Données : les dates de début d_i et de fin f_i de chaque demande i.
- Solution optimale = satisfaisant le plus de demandes.

CHOIX_ACTIVITES (demandes)
Trier les demandes par **date de fin** croissante.
Sol := ∅
dispo := 0 `// Première date disponible`

PROBLÈME DE CHOIX DES ACTIVITÉS

Énoncé du problème

On dispose d'une salle pouvant être louée pour une durée variable.
On choisit parmi un ensemble de n demandes de location celles qui seront acceptées.

- Données : les dates de début d_i et de fin f_i de chaque demande i.
- Solution optimale = satisfaisant le plus de demandes.

CHOIX_ACTIVITES (demandes)
Trier les demandes par **date de fin** croissante.
Sol := ∅
dispo := 0 `// Première date disponible`
for *i* **:= 1 to** *n* **do**

PROBLÈME DE CHOIX DES ACTIVITÉS

Énoncé du problème

On dispose d'une salle pouvant être louée pour une durée variable.
On choisit parmi un ensemble de n demandes de location celles qui seront acceptées.

- Données : les dates de début d_i et de fin f_i de chaque demande i.
- Solution optimale = satisfaisant le plus de demandes.

CHOIX_ACTIVITES (demandes)
Trier les demandes par **date de fin** croissante.
$Sol := \emptyset$
$dispo := 0$ `// Première date disponible`
for i **:= 1 to** n **do**
  if $d_i \geqslant dispo$ **then** `// demande acceptable`
    Insérer i dans Sol

PROBLÈME DE CHOIX DES ACTIVITÉS

Énoncé du problème

On dispose d'une salle pouvant être louée pour une durée variable.
On choisit parmi un ensemble de n demandes de location celles qui seront acceptées.

- Données : les dates de début d_i et de fin f_i de chaque demande i.
- Solution optimale = satisfaisant le plus de demandes.

CHOIX_ACTIVITES (demandes)
Trier les demandes par **date de fin** croissante.
$Sol := \emptyset$
$dispo := 0$ `// Première date disponible`
for i **:= 1 to** n **do**
  if $d_i \geqslant dispo$ **then** `// demande acceptable`
    Insérer i dans Sol
    $dispo := f_i$

PREUVE DE L'ALGORITHME GLOUTON

Remarque : l'algorithme choisit toujours la demande 1.

PREUVE DE L'ALGORITHME GLOUTON

Remarque : l'algorithme choisit toujours la demande 1.

Propriété d'échange

Soit S une solution optimale, que l'on ordonne par date de fin croissante sans perte de généralité.
Si la première demande de S est une demande $k \neq 1$, alors $T = (S \backslash \{k\}) \cup \{1\}$ est également une solution optimale :

PREUVE DE L'ALGORITHME GLOUTON

Remarque : l'algorithme choisit toujours la demande 1.

Propriété d'échange

Soit S une solution optimale, que l'on ordonne par date de fin croissante sans perte de généralité.
Si la première demande de S est une demande $k \neq 1$, alors $T = (S \backslash \{k\}) \cup \{1\}$ est également une solution optimale :

- $f_1 \leqslant f_k$ donc la demande 1 est compatible avec toutes les autres demandes de S (qui ont des dates postérieures à f_k).

PREUVE DE L'ALGORITHME GLOUTON

Remarque : l'algorithme choisit toujours la demande 1.

Propriété d'échange

Soit S une solution optimale, que l'on ordonne par date de fin croissante sans perte de généralité.
Si la première demande de S est une demande $k \neq 1$, alors $T = (S \backslash \{k\}) \cup \{1\}$ est également une solution optimale :

- $f_1 \leqslant f_k$ donc la demande 1 est compatible avec toutes les autres demandes de S (qui ont des dates postérieures à f_k).
- T est de même taille que S.

PREUVE DE L'ALGORITHME GLOUTON

Remarque : l'algorithme choisit toujours la demande 1.

Propriété d'échange

Soit S une solution optimale, que l'on ordonne par date de fin croissante sans perte de généralité.
Si la première demande de S est une demande $k \neq 1$, alors $T = (S \backslash \{k\}) \cup \{1\}$ est également une solution optimale :

- $f_1 \leqslant f_k$ donc la demande 1 est compatible avec toutes les autres demandes de S (qui ont des dates postérieures à f_k).
- T est de même taille que S.

Par ailleurs, si S est une solution optimale pour le problème initial, alors $S' = S \backslash \{1\}$ est une solution optimale pour le sous-problème $\{i \mid d_i \geqslant f_1\}$.
En effet une meilleure solution que S' à ce sous-problème fournirait une meilleure solution que S au problème initial, simplement en lui ajoutant la demande 1.

PREUVE DE L'ALGORITHME GLOUTON (2)

On distingue alors trois possibilités sur la **première demande** de $S \setminus Sol$:

PREUVE DE L'ALGORITHME GLOUTON (2)

On distingue alors trois possibilités sur la **première demande** de $S \setminus Sol$:

- c'est une demande $k < i$: impossible car dans ce cas, à l'itération k, on a refusé la demande k qui était incompatible avec la dernière demande de Sol.

PREUVE DE L'ALGORITHME GLOUTON (2)

On distingue alors trois possibilités sur la **première demande** de $S \setminus Sol$:

- c'est une demande $k < i$: impossible car dans ce cas, à l'itération k, on a refusé la demande k qui était incompatible avec la dernière demande de Sol.
- c'est la demande $k = i$: alors $Sol \cup \{i\}$ est encore un préfixe de S.

PREUVE DE L'ALGORITHME GLOUTON (2)

On distingue alors trois possibilités sur la **première demande** de $S \backslash Sol$:

- c'est une demande $k < i$: impossible car dans ce cas, à l'itération k, on a refusé la demande k qui était incompatible avec la dernière demande de Sol.
- c'est la demande $k = i$: alors $Sol \cup \{i\}$ est encore un préfixe de S.
- c'est une demande $k > i$: on montre qu'on peut la remplacer par i.

Propriété d'échange

Si la première demande de $S \backslash Sol$ est une demande $k > i$, alors $T = (S \backslash \{k\}) \cup \{i\}$ est également une solution optimale :

PREUVE DE L'ALGORITHME GLOUTON (2)

On distingue alors trois possibilités sur la **première demande** de $S \backslash Sol$:

- c'est une demande $k < i$: impossible car dans ce cas, à l'itération k, on a refusé la demande k qui était incompatible avec la dernière demande de Sol.
- c'est la demande $k = i$: alors $Sol \cup \{i\}$ est encore un préfixe de S.
- c'est une demande $k > i$: on montre qu'on peut la remplacer par i.

Propriété d'échange

Si la première demande de $S \backslash Sol$ est une demande $k > i$, alors $T = (S \backslash \{k\}) \cup \{i\}$ est également une solution optimale :

- $f_k \geqslant f_i$ donc la demande i est compatible avec toutes les autres demandes de S (qui ont des dates postérieures à f_k).

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

PREUVE DE L'ALGORITHME GLOUTON (2)

On distingue alors trois possibilités sur la **première demande** de $S \backslash Sol$:

- c'est une demande $k < i$: impossible car dans ce cas, à l'itération k, on a refusé la demande k qui était incompatible avec la dernière demande de Sol.
- c'est la demande $k = i$: alors $Sol \cup \{i\}$ est encore un préfixe de S.
- c'est une demande $k > i$: on montre qu'on peut la remplacer par i.

Propriété d'échange

Si la première demande de $S \backslash Sol$ est une demande $k > i$, alors $T = (S \backslash \{k\}) \cup \{i\}$ est également une solution optimale :

- $f_k \geqslant f_i$ donc la demande i est compatible avec toutes les autres demandes de S (qui ont des dates postérieures à f_k).
- T est de même taill e que S.

UFR IM²AG UNIVERSITÉ Grenoble Alpes

ALGORITHMIQUE ET MODÉLISATION

1 OPTIMISATION

2 LE PROBLÈME D'ALLOCATION

3 STRATÉGIE GLOUTON

4 EXERCICES

5 LE PROBLÈME DE L'ARBRE COUVRANT

EXERCICES DE COLORIAGE

Coloriage

Un algorithme glouton pour colorier un graphe :

- prendre les sommets dans un ordre quelconque
- attribuer à chaque sommet la plus petite couleur non utilisée par ses voisins déjà coloriés

Combien de couleurs nécessite cet algorithme ? Est-il optimal ?

Coloriage (2)

Un algorithme glouton un peu plus malin pour colorier un graphe :

- prendre les sommets par degré décroissant
- attribuer à chaque sommet la plus petite couleur non utilisée par ses voisins déjà coloriés

Que penser de cet algorithme par rapport au précédent ? Est-il optimal ?

EXERCICES D'ORGANISATION

Couverture par des intervalles

On se donne n réels $\{x_1, x_2, \ldots, x_n\}$. On souhaite trouver le nombre minimal K tel que K intervalles $I_1, I_2, \ldots I_K$ chacun de longueur 1 recouvrent tous les points, c'est-à-dire que tout point appartient à au moins l'un de ces intervalles. On considère ici des intervalles fermés (dont les bornes sont incluses).

1. Donner un exemple (avec n petit) montrant qu'il peut y avoir plusieurs recouvrements minimaux (plusieurs familles d'intervalles pour le même K minimal). Donner également un exemple où ce recouvrement minimal est unique.
2. Écrire un algorithme glouton qui détermine la valeur minimale de K en construisant les K intervalles.
3. Écrire la preuve de votre algorithme glouton.

Ordonnancement

Dans un petit restaurant, un serveur (seul) doit servir n clients ; il sait qu'il lui faudra t_i minutes pour servir le client numéro i. Le mécontentement de chaque client se mesure au temps qu'il doit attendre avant d'être servi.
Afin de créer le moins de mécontentement possible, l'objectif est ici de minimiser le total des temps d'attente de tous les clients.
Comment faut-il procéder et pourquoi ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

ALGORITHMIQUE ET MODÉLISATION

1 OPTIMISATION

2 LE PROBLÈME D'ALLOCATION

3 STRATÉGIE GLOUTON

4 EXERCICES

5 LE PROBLÈME DE L'ARBRE COUVRANT

UFR IM²AG
UNIVERSITÉ Grenoble Alpes