Chapitre 9 : Algorithmes d'approximation

INF4705 - Analyse et conception d'algorithmes

Gilles Pesant Simon Brockbank

École Polytechnique Montréal
gilles.pesant@polymtl.ca, simon.brockbank@polymtl.ca

Hiver 2017

Plan

Rappel

Catégories d'algorithme d'optimisation

- exact
- approximatif
- heuristique

Classification des algorithmes d’approximation

Soient c et ϵ des constantes positives et supposons que la valeur des solutions à notre problème est également positive.

Algorithme d’approximation c-absolue

Un algorithme d’approximation c-absolue calcule pour un problème d’optimisation une solution de valeur V dont l’erreur absolue par rapport à la valeur $V^\star$ d’une solution optimale est au plus c :

$$V^\star - c \leq V \leq V^\star \text{ si on maximise}$$

$$V^\star \leq V \leq V^\star + c \text{ si on minimise}$$

Classification des algorithmes d'approximation

Algorithme d'approximation ϵ-relative

Un algorithme d'approximation ϵ-relative calcule pour un problème d'optimisation une solution de valeur V dont l'erreur relative par rapport à la valeur $V^\star$ d'une solution optimale est au plus ϵ :

$$(1-\epsilon)V^\star \leq V \leq V^\star \text{ si on maximise}$$

$$V^\star \leq V \leq (1+\epsilon)V^\star \text{ si on minimise}$$

Classification des algorithmes d'approximation

Algorithme d'approximation mixte

Un algorithme d'approximation mixte calcule pour un problème d'optimisation une solution de valeur V dont l'erreur par rapport à la valeur $V^\star$ d'une solution optimale s'exprime à la fois de manière absolue et relative :

$$(1-\epsilon)V^\star - c \leq V \leq V^\star \text{ si on maximise}$$

$$V^\star \leq V \leq (1+\epsilon)V^\star + c \text{ si on minimise}$$

Des valeurs de zéro pour c et ϵ correspondent à un algorithme exact. Plus c et ϵ se rapprochent de zéro, meilleures sont les approximations.

Classification des algorithmes d'approximation

Schéma d'approximation

Un schéma d'approximation est un algorithme auquel on donne, en plus de l'exemplaire, l'ϵ désiré pour l'algorithme d'approximation ϵ-relative à résoudre.

On dira que le schéma d'approximation est *pleinement polynomial* s'il prend un temps dans $\mathcal{O}(p(n, 1/\epsilon))$ en pire cas, où n est la taille de l'exemplaire et p est un polynôme fixe en deux variables.

Plan

Mise en boîte, 1ère variante

Étant donné n objets, chacun de poids w_i, et k boîtes de capacité W , quel est le plus grand nombre d'objets qu'on puisse y mettre ?

Algorithme glouton

Candidats : objets
Critère de choix glouton : objet le plus léger, qu'on place dans la première boîte disponible

Supposons les objets en ordre non-décroissant de poids.

Ex. $k = 1$ boîte

objet	a	b	c	d
w_i	2	3	4	5

$W = 7$

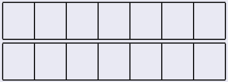
Cet algorithme est optimal pour $k = 1$.

Mise en boîte, 1ère variante

Ex. $k = 2$ boîtes

objet	a	b	c	d
w_i	2	3	4	5

$W = 7$



Mise en boîte, 1ère variante

Ex. $k = 2$ boîtes

objet	a	b	c	d
w_i	2	3	4	5

$W = 7$

Mais en pire cas cet algorithme mettra dans le sac un objet de moins que l'optimum :

- considérons une grande boîte de capacité $2W$
- soit V' la valeur optimale pour cette boîte
- soit $j = \text{argmin}_k \sum_{i=1}^{k} w_i > W$

Mise en boîte, 1ère variante

Généralisation : considérons une grande boîte de capacité kW...

En général, cet algorithme est d'approximation $(k-1)$-absolue :

$$V^\star - (k-1) \leq V \leq V^\star$$

Mise en boîte, 2e variante

Étant donné n objets, chacun de poids w_i, et des boîtes de capacité W, quel est le plus petit nombre de boítes nécessaires pour y mettre tous les objets ?

Algorithme glouton "First-Fit-Decreasing"

Candidats : objets
Critère de choix glouton : objet le plus lourd, qu'on place dans la première boîte disponible

Cet algorithme est d'approximation mixte :

$$V^\star \leq V \leq \frac{11}{9} V^\star + \frac{6}{9}$$

Plan

Sac à dos

Étant donné n objets de poids et valeur positifs w_i et v_i, $1 \leq i \leq n$ et un sac à dos pouvant supporter un poids total W, quels objets devrais-je mettre dans mon sac à dos afin de maximiser la valeur totale de son contenu ?

Algorithme glouton "densité_valeur"

Candidats : objets
Critère de choix glouton : objet avec la densité de valeur v_i/w_i la plus élevée

Sa solution peut être arbitrairement mauvaise (lorsqu'on ne peut pas fractionner les objets).

Sac à dos

Voici une façon simple de l'améliorer qui retourne une solution valant au moins la moitié de la valeur optimale :

Algorithme glouton augmenté

retourner max(densité_valeur(w, v, W), $\max\{v_i : 1 \leq i \leq n\}$)

Sac à dos

Supposons les objets en ordre décroissant de densité de valeur.
Soit ℓ le premier objet à devoir être exclus du sac ($\sum_{i=1}^{\ell} w_i > W$).

$$\begin{array}{rcl}
V & = & \max(\text{densité_valeur}(w, v, W),\ \max\{v_i : 1 \leq i \leq n\}) \\
 & \geq & (\text{densité_valeur}(w, v, W) + \max\{v_i : 1 \leq i \leq n\})/2 \\
 & \geq & (\displaystyle\sum_{i=1}^{\ell-1} v_i + v_\ell)/2 \\
 & = & (\displaystyle\sum_{i=1}^{\ell} v_i)/2 \\
 & \geq & V^\star/2
\end{array}$$

Nous avons donc un algorithme d'approximation 1/2-relative :

$$(1 - 1/2)V^\star \leq V \leq V^\star$$

Sac à dos

Ex. glouton augmenté

objet	a	b	c
w_i	10	1	80
v_i	11	1	72
v_i/w_i	1.1	1	0.9

$W = 90$

Sac à dos

Ex. glouton augmenté

objet	a	b	c
w_i	10	1	80
v_i	11	1	72
v_i/w_i	1.1	1	0.9

$W = 90$

solution optimale : $a + c$, de valeur totale 83

Sac à dos

Ex. glouton augmenté

objet	a	b	c
w_i	10	1	80
v_i	11	1	72
v_i/w_i	1.1	1	0.9

$W = 90$

solution optimale : $a + c$, de valeur totale 83

solution densité_valeur(w, v, W) : $a + b$, de valeur totale 12

Sac à dos

Ex. glouton augmenté

objet	a	b	c
w_i	10	1	80
v_i	11	1	72
v_i/w_i	1.1	1	0.9

$W = 90$

solution optimale : $a + c$, de valeur totale 83

solution densité_valeur(w, v, W) : $a + b$, de valeur totale 12

mais. . . $\max\{v_i : 1 \leq i \leq n\} = 72$

Sac à dos

Ex. glouton augmenté

objet	a	b	c
w_i	40	1	50
v_i	44	1	45
v_i/w_i	1.1	1	0.9

$W = 90$

solution optimale : $a + c$, de valeur totale 89

solution densité_valeur(w, v, W) : $a + b$, de valeur totale 45

$\max\{v_i : 1 \leq i \leq n\} = 45$

Sac à dos

Idée pour le schéma d'approximation

- Revisiter l'utilisation du patron de programmation dynamique pour ce problème
- Utiliser l'algorithme glouton_augmenté pour garantir la performance de l'approche

Sac à dos

Programmation dynamique avant (rappel)

$$V[i,j] = \max(v_i + V[i-1, j-w_i],\ V[i-1,j])$$

où $V[i,j]$ est la valeur maximum d'un contenu du sac de poids total n'excédant pas j et n'utilisant que les objets $1 \dots i$.

problème : la taille du tableau dépend de $W \Rightarrow$ pseudo-polynomial

Programmation dynamique maintenant : son dual

$$P[i,j] = \min(w_i + P[i-1, j-v_i],\ P[i-1,j])$$

où $P[i,j]$ est le poids minimum d'un contenu du sac de valeur totale j et n'utilisant que les objets $1 \dots i$.

problème : il faut savoir borner supérieurement la 2e dimension (valeur totale) du tableau en fonction de n (et de $1/\epsilon$)

Une borne supérieure pour la valeur totale du sac

Calculer $V_g = \text{glouton\_augmenté}(w, v, W)$

Donc $V^\star \leq 2V_g$

1er cas : $V_g < \frac{2n}{\epsilon}$

Puisque $V^\star < \frac{4n}{\epsilon}$, le nouvel algorithme de programmation dynamique donnera la réponse optimale (exacte) dans un temps polynomial en n et $1/\epsilon$.

2e cas : $V_g \geq \frac{2n}{\epsilon}$

On applique une mise à l'échelle des v_i afin de réduire la valeur totale du sac, au prix d'une perte de précision et ainsi de la garantie d'optimalité.

Borne supérieure, 2e cas

Posons $k = \lfloor \frac{\epsilon V_g}{n} \rfloor$

Mise à l'échelle : $v'_i = \lfloor \frac{v_i}{k} \rfloor$

Soient $X^\star$ solution optimale pour l'exemplaire originel (v_i) et X' solution optimale pour l'exemplaire modifié (v'_i)

Qualité de l'approximation donnée par X' :

$$\begin{array}{rcl} \Sigma_{i \in X'} v_i & \geq & \Sigma_{i \in X'} k v'_i \\ & \geq & \Sigma_{i \in X^\star} k v'_i \\ & > & \Sigma_{i \in X^\star} (v_i - k) \\ & = & \Sigma_{i \in X^\star} v_i - \Sigma_{i \in X^\star} k \\ & \geq & V^\star - kn \\ & \geq & V^\star - \epsilon V_g \\ & \geq & V^\star - \epsilon V^\star = (1 - \epsilon) V^\star \end{array}$$

Sac à dos
Algorithme

1. $V_g \leftarrow$ glouton_augmenté(w, v, W) ;
2. **si** $\frac{\epsilon V_g}{n} < 2$ **alors**
   1. $S \leftarrow$ prog_dyn_dual$(w, v, W, 2V_g)$;
   2. **retourner** S ; {qui sera une solution exacte}
3. $k \leftarrow \lfloor \frac{\epsilon V_g}{n} \rfloor$;
4. **pour** $i \leftarrow 1$ **à** n **faire**
   1. $v'_i \leftarrow \lfloor \frac{v_i}{k} \rfloor$; {construire l'exemplaire modifié}
5. $S \leftarrow$ prog_dyn_dual$(w, v', W, \lfloor \frac{2V_g}{k} \rfloor)$;
6. **retourner** S ; {qui sera une solution approximative ϵ-relative}

Sac à dos
Analyse

1. $V_g \leftarrow$ glouton_augmenté(w, v, W) ;
2. **si** $\frac{\epsilon V_g}{n} < 2$ **alors**
   1. $S \leftarrow$ prog_dyn_dual$(w, v, W, 2V_g)$;
   2. **retourner** S ; {qui sera une solution exacte}
3. $k \leftarrow \lfloor \frac{\epsilon V_g}{n} \rfloor$;
4. **pour** $i \leftarrow 1$ **à** n **faire**
   1. $v'_i \leftarrow \lfloor \frac{v_i}{k} \rfloor$; {construire l'exemplaire modifié}
5. $S \leftarrow$ prog_dyn_dual$(w, v', W, \lfloor \frac{2V_g}{k} \rfloor)$;
6. **retourner** S ; {qui sera une solution approximative ϵ-relative}