Algorithmes en ligne

François Schwarzentruber

1 Location de ski

1€ = prix de la location pour un jour

β € = le prix de l'achat des skis, avec $\beta > 1$

1.1 Algorithme offline

— entrée : météo : une suite $x_1, \ldots, x_n$ avec $x_i \in \{beau, moche\}$
— sortie : une suite de décisions $d_i \in \{louer, acheter, skip\}$

Objectif : minimiser le coût, sachant que l'on veut skier tous les premiers beaux jours, et dès qu'un jour est moche, on ne skie pas et on part définitivement.

fonction algoOffline(x)
    $g :=$ nb de beaux jours consécutifs dans x
    si $g \leq \beta$ **alors** louer chaque jour **sinon** acheter le 1er jour

Proposition 1 L'optimum $\min(g, \beta)$ et l'algorithme offline est optimal.

1.2 Algorithme online déterministe

Il faut décider d_i avec un algorithme déterministe qui ne connaît que $x_{1..i}$.

fonction algoOnline()
    louer pendant les $\beta - 1$ premiers jours
    acheter ensuite

Définition 2 (algorithme c-compétitif) Considérons un problème de minimisation. Un algorithme online ALG est c-compétitif s'il existe une constante b telle que pour toute instance x, on ait :

$$ALG(x) \leq c \times OPT(x) + b$$

où $OPT(x)$ le coût optimal pour x, et $ALG(x)$ le coût donné par l'algorithme ALG.

Définition 3 (ratio compétitif) Soit ALG un algorithme online de minimisation. Le ratio compétitif c_{ALG} est l'infimum sur les c où ALG est c-compétitif.

Proposition 4 Le ratio compétitif de l'algorithme online donné plus haut est de $2 - \frac{1}{\beta}$.

Théorème 5 On ne peut pas mieux faire avec un algorithme online déterministe.

1.3 Algorithme online randomisé

Il faut décider $d[i]$ avec un algorithme probabiliste qui ne connaît que $x[1..i]$.

fonction algoOnlineRandom()
    choisir aléatoirement $i \in \{0, \beta - 1\}$ selon une distribution aléatoire donnée $(p_0, \ldots, p_{\beta-1})$
    louer pendant les i premiers jours
    acheter le $i+1$-ème jour

Soit $p_i := \mathbb{P}(\text{choisir } i)$.

Définition 6 (algorithme c-compétitif) Considérons un problème de minimisation. Un algorithme online randomisé $ALGR$ est c-compétitif s'il existe une constante β telle que pour toute instance x, on ait :

$$\mathbb{E}(ALGR(x)) \leq c \times OPT(x) + \beta$$

où $OPT(x)$ est la valeur optimale pour x, et $ALGR(x)$ est la valeur retournée par l'algorithme $ALGR$.

Définition 7 (ratio compétitif) Pareil. Soit $ALGR$ un algorithme online de minimisation. Le ratio compétitif c_{ALGR} est l'infimum sur les c où $ALGR$ est c-compétitif.

Proposition 8 Le meilleur ratio compétitif de l'algorithme randomisé est $c := \frac{1}{1-(1-1/\beta)^\beta}$ pour $p_i = \frac{c}{\beta}(1-(1/\beta))^{\beta-1-i}$.

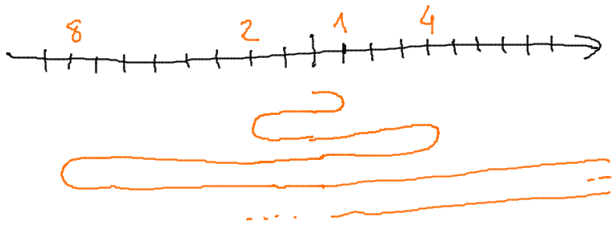
2 Robot aveugle

Un robot initialement en 0 se déplace sur la ligne $\mathbb{Z}$, qui contient une banane en d ou $-d$. But : trouver la banane en faisant le moins de mouvement possible. $OPT(x) = d$. L'algorithme offline : aller vers la banane.

Remarque 9 Seul l'information de la direction (gauche, ou droite) suffit pour avoir la performance de l'algorithme offline.

Cadre online : il ne voit rien sauf la case courante.

fonction algoOnline()
    phase 0 : se déplacer d'une case vers la droite et revenir en 0
    phase 1 : se déplacer de 2 cases vers la gauche et revenir en 0
    phase 2 : se déplacer de 4 cases vers la droite et revenir en 0
    ⋮



Proposition 10 Le ratio compétitif est de 9.

3 Arbre de Steiner

Définition 11 (arbre de Steiner) Soit un graphe complet $G = (V, E, c)$ complet pondéré positivement et vérifiant l'inégalité triangulaire ; des sommets $t_1, \ldots, t_k \in V$. Un *arbre de Steiner* est un arbre inclus dans G qui relie tous les sommets $t_1, \ldots, t_k$ entre eux et de poids minimal.

Définition 12 (terminaux) Les sommets $t_1, \ldots, t_k$ s'appellent des *terminaux*.

Définition 13 (Problème de l'arbre de Steiner online)
- entrée : un graphe complet $G = (V, E, c)$ complet pondéré positivement et vérifiant l'inégalité triangulaire ; des sommets $t_1, \ldots, t_k \in V$ donné un à un ;
- sortie : quand t_i arrive, il faut ajouter une arête à la solution courante pour que $\{t_1, \ldots, t_i\}$ soit de poids le plus petit possible.

fonction algoOnlineSteiner(G)
    $T := \emptyset$
    pour $i := 2$ à k **faire**
        quand t_i arrive, ajouter à T une arête (t_j, t_i) avec $j < i$ qui est la plus légère

Théorème 14 L'algorithme est $2 \ln k$-compétitif, où k est le nombre de terminaux.

4 Mise en cache

On considère un cache C de taille k. Le cache contient initialement des adresses. Ces adresses peuvent être des adresses mémoires pour un processeur, ou des URL pour un navigateur internet, etc.

Si on ne trouve pas une donnée dans le cache, il y a *échec*. On peut décider de choisir une donnée du cache à supprimer et à remplacer par la nouvelle donnée. But : minimiser le nombre d'échec
- entrée : $X = x_1 x_2 \ldots$ où x_i est la i-ème adresse mémoire demandée
- sortie : décider quelles données remplacer

On suppose initialement que la cache contient n'importe quoi.

4.1 Algorithme offline optimal

En cas d'échec, supprimer/remplacer une page du cache qui ne sera plus jamais visité, ou si elles sont toutes visités, celle qui sera demandé le plus tard

On note $OPT(x)$ le nombre d'échecs de cet algorithme.

4.2 Algorithme online déterministe

fonction algoOnline()
  pour $j = 1..n$ **faire**
    si la page x_j n'est pas dans le cache C **alors**
      choisir la cellule $i \in \{0, \dots, k-1\}$ selon une certaine stratégie
      $C[i] := x_j$

Quelques stratégies pour le choix de i

LRU	Least Recently Used	remplacer la page qui n'a pas été utilisée pendant le plus longtemps
FIFO	First In first Out	remplacer la page qui a été le plus longtemps dans le cache
LIFO	Last In First Out	remplacer la page la plus récemment ajoutée au cache
LFU	Least Frequently Used	remplacer la page qui a été le moins utilisée

Exemple 15 On considère un cache de taille $k = 3$ qui est 123 (3 le plus vieux et 1 le plus récent).

x_i	cache FIFO	cache algo offline opt
4	412	423
3	341	423
2	234	423
1	123	413
4	412	413
3	341	413
2	234	214
1	123	214
⋮	⋮	⋮

Proposition 16 L'algorithme online avec FIFO ou avec LRU est k-compétitif.

Remarque 17 LRU marche mieux en FIFO. Et si k augmente, le ratio valant k augmente, alors que le cache fonctionne mieux. Pourquoi ? Voir [Angelopoulous 2007] pour une analyse en moyenne + analyse concave (je ne sais pas ce que c'est).

4.3 Borne inférieure pour algorithme online déterministe

Proposition 18 S'il y a $k + 1$ pages, il n'y a pas de meilleur algorithme online : on ne peut pas faire mieux qu'un ratio compétitif de k.

4.4 Algorithme online randomisé : l'algorithme RMA

RMA = randomized marking algorithm

fonction RMA()
  $C[1..k]$:= le tableau contenant les données du cache
  $M[1..k] := [false, ..., false]$ (marques)
  pour $j := 1$ à n **faire**
    si x_j est présente dans C **alors**
      soit i avec $C[i] = x_j$
      $M[i] := true$
    sinon
      si pour tout $i = 1..k$, $M[i] = true$ **alors**
        $M[1..k] := [false, ..., false]$
      $S := \{i = 1..k \mid M[i] = false\}$
      choisir uniformément i dans S
      $C[i] := x_j$
      $M[i] := true$

Définition 19 $H_k := \sum_{i=1}^{k} \frac{1}{i}$.

Proposition 20 RMA admet un ratio compétitif de $2H_k$.

4.5 Borne inférieure pour algorithme probabiliste

Théorème 21 Supposons qu'il y ait plus de $k+1$ pages, où k est la taille du cache. Tout algorithme probabiliste pour le problème de pagination a un facteur compétitif d'au moins H_k.

5 Principe minmax de Yao

Théorème 22 (principe minmax de Yao, admis)

$$\min_{\text{ALGR}} \max_{x} \mathbb{E}(\text{coût de } ALGR(x)) = \max_{X} \min_{ALG} \mathbb{E}(\text{coût de } ALG(X))$$

où $ALGR$ dénote n'importe quel algorithme probabiliste, x une entrée, X une variable aléatoire sur les entrées, ALG n'importe quel algorithme déterministe.

Références bibliographiques

Les livres [FW98] et [BE98] (une nouvelle version est en train d'être écrite) sont les références sur le sujet. La section sur l'algorithme online pour les arbres de Steiner provient d'un cours de Tim Roughgarden.

Références

[BE98] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.

[FW98] Amos Fiat and Gerhard J. Woeginger, editors. *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*. Springer, 1998.