Analyse heuristique des coûts chinois

Jean-François Burnol, 7 avril 2017

Soient $n_1, \ldots, n_k$ des entiers strictement positifs premiers entre eux deux à deux. On considère la résolution générale du Problème Chinois :

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ &\cdots \quad\quad \cdots\cdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

qui est donnée, comme on sait par

$$x \equiv a_1 x_1 + \cdots + a_n x_n \pmod{n_1 \cdots n_k}$$

avec des x_j uniques modulo $P = n_1 \cdots n_k$, qui sont les solutions aux congruences

$$\begin{aligned} x_j &\equiv 1 \pmod{n_j} \\ x_j &\equiv 0 \pmod{\prod_{i \neq j} n_i} \end{aligned}$$

Dans une fiche précédente :

http://jf.burnol.free.fr/agreg170406Chinois.pdf

j'ai fait la promotion d'une méthode qui consiste à obtenir les x_j par l'algorithme suivant :

1. Calculer $P = \prod_i n_i$, et les $P_j = \prod_{i \neq j} n_i$ puis leur somme $S = \sum_j \prod_{i \neq j} n_i$.
2. Établir une identité de Bézout $US + VP = 1$.
3. Les x_j sont donnés par la formule $U_j P_j$, où l'on a noté U_j le reste dans la division euclidienne de U par n_j.

L'approche est séduisante car on n'y fait la recherche que d'une seule identité de Bézout. De plus elle est aussi théoriquement intéressante car elle prend sa source dans une remarquable formule

$$f = \frac{\sum_j \frac{a_j}{n_j}}{\sum_j \frac{1}{n_j}}$$

qui définit une fraction dont le dénominateur (après avoir multiplié en haut et en bas par P) est premier à chaque n_j et cette fraction résout le problème Chinois, à condition

d'utiliser des inverses modulaires pour interpréter ce dénominateur dans les diverses congruences.

Essayons de comparer le coût numérique entre la méthode obtenant chaque x_j individuellement ou celle passant par le U. Je vais noter X le nombre maximum des chiffres des n_i en binaire et les supposer ayant tous à peu près en effet ce nombre X de chiffres. Et k est le nombre des n_i.

Tout d'abord dans les deux cas on peut convenir qu'il faut calculer P et les $P_j = P/n_j$ et que c'est commun aux deux méthodes. Je n'essaie pas d'évaluer ce coût commun…

Quel est le coût heuristique de calcul de x_j ? Je fixe ici le déroulement de l'algorithme d'Euclide comme étant celui enseigné à l'École, avec pour les coefficients de Bézout, l'emploi d'une méthode qui met à jour des variables à chaque étape, comme usuellement en informatique.

À vrai dire, il y a là déjà une subtilité, car la toute première division euclidienne, celle de P_j par n_j, va donner un gros quotient et un reste r_j. Si l'on obtient un Bézout $un_j + vr_j = 1$, et que $r_j = P_j - Q_j n_j$, de sorte que $un_j + v(P_j - Q_j n_j) = 1$, $vP_j + (u - Q_j v)n_j = 1$, le x_j cherché sera vP_j (ou $vP_j + P$ s'il est négatif), et l'on a nul besoin du $u - Q_j v$. Donc je ne compte que le coût du Bézout avec n_j et r_j, pas avec n_j et P_j, et dans la pratique le programme informatique pour Euclide étendu qui met à jour continuellement les variables donnant les u et v finaux ne tiendra pas compte du gros quotient inital Q_j et ne sera réellement activé qu'à partir du couple (n_j, r_j).

Je supposerai aussi que les opérations de multiplication et de division sont faites naïvement [1] et je néglige le coût des additions et soustractions. Or, si les entiers tiennent sur 64bits, je crois que tout cela est très discutable car la multiplication faite au niveau du processeur ne coûtera probablement pas plus cher, mais bon je me contente d'une approche très naïve ici, et puis les vrais pros auront déjà décelé mon incompétence généralisée dans cette analyse de coût, et ça va aller en s'agravant.

Dans la première étape on divise un nombre d'environ $(k-1)X$ chiffres par un nombre de X chiffres pour obtenir le quotient et le reste. Je considère que cela coûte $\mathcal{O}(kX^2)$ (mais je me trompe peut-être). De même je compte $\mathcal{O}(kX^2)$ pour le produit vP_j (avec les notations plus haut), une fois $v < n_j$ trouvé (qui aura aussi de l'ordre de X chiffres).

1. Plus bas j'évoque un coût quadratique pour Euclide. Un peu de recherche dans la littérature m'a indiqué que c'est seulement depuis 1970 qu'on dispose d'algorithmes sous-quadratiques. De plus ce domaine est un domaine de recherches actuelles en algorithmique et en mathématiques.

Ensuite on a deux entiers d'environ X chiffres. On peut montrer que le nombre moyen d'étapes (en un certain sens) de l'algorithme d'Euclide est linéaire en le nombre initial de chiffres (pour des entiers « aléatoires »). On peut considérer heuristiquement que les quotients restent bornés et que le nombre de chiffres décroît linéairement. J'ai envie de dire [2] que la division euclidienne avec reste faite à chaque étape a donc un coût *linéaire* (et pas *quadratique*) en le nombre de chiffres encore présents. Cela donne une progression arithmétique et donc au total j'obtiens un coût $\mathcal{O}(X^2)$. Il ne faut pas oublier la mise à jour de coefficients de Bézout, je considère, par le même genre d'évaluations, que ça coûte aussi $\mathcal{O}(X^2)$ (on part de petits et on arrive à des gros avec, grosso modo, une progression arithmétique des nombres des chiffres). Il semble qu'on arrive au total à un coût de $\mathcal{O}(kX^2)$ et que c'est la première division euclidienne qui en est responsable, sinon ça serait $\mathcal{O}(X^2)$. Et ça c'est pour un x_j, donc au total semble-t-il $\mathcal{O}(k^2X^2)$.

Que est le coût heuristique pour le calcul de U ? Bon, ben P et S ont de l'ordre de kX chiffres chacun, je dois suivre Euclide-Bézout, et si j'en crois mon raisonnement précédent ça coûte $\mathcal{O}((kX)^2)$. Après il faut faire les réductions modulo les n_j, et cela semble devoir coûter aussi k fois $\mathcal{O}(kX^2)$, donc à nouveau $\mathcal{O}((kX)^2)$.

Finalement il semble que si l'on suit les méthodes de calcul de l'École pour les divisions, multiplications, et Euclide, on peut conclure que les deux approches ont un coût comparable. Ma méthode « en une fois » ne fait qu'un seul Bézout, mais les k Bézout de la méthode plus standard coûtent chacun k fois moins.

Tout cela demanderait à être confirmé sur le terrain…, si une méthode va deux fois plus vite que l'autre ça ne se voit pas dans nos grossiers $\mathcal{O}(k^2X^2)$. Et puis la vraie vie, c'est encore autre chose car pour les très grands nombres on a comme je l'ai dit dans une note de bas de page des algorithmes sous-quadratiques pour Euclide.

On peut à signaler un lien avec les polynômes interpolateurs de Lagrange, en effet la fraction rationnelle :

$$F = \frac{\sum_j \frac{u_j}{X-t_j}}{\sum_j \frac{1}{X-t_j}}$$

est en fait définie en chaque $X = t_j$ (on a le quotient de deux pôles simples en t_j donc, plus de singularité du tout en cet endroit) et y prend la valeur $\lim_{t\to t_j} F(t) = u_j$. Elle

2. À cause de résultats de ce type : *lors du déroulement de l'algorithme d'Euclide chaque quotient a plus de deux chances sur trois d'être* 1, 2, *ou* 3.

résout donc le problème de Lagrange, mais n'est pas un polynôme. Cependant, si on l'écrit sous la forme

$$F = \frac{\sum_j u_j \prod_{i \neq j}(X - t_i)}{P'}$$

avec $P = \prod_i (X - t_i)$, elle peut donner l'idée de remplacer la fraction $\frac{1}{P'}$ par l'inverse modulaire de P' modulo l'idéal $(X - t_i)$, c'est-à-dire par $P'(t_j)^{-1}$ et donc d'aboutir au polynôme

$$L = \sum_j P'(t_j)^{-1} u_j \prod_{i \neq j} (X - t_i)$$

ou plus prosaïquement de considérer la fonction $F(t)$ associée à la fraction rationnelle F, de redire qu'on peut la prolonger par continuité avec $F(t_j) = u_j$, et de constater que le vrai truc $\frac{\prod_{i \neq j}(t - t_i)}{P'(t)}$ et son remplacement $\frac{\prod_{i \neq j}(t - t_i)}{P'(t_j)}$ ont les mêmes valeurs en tous les t_i. Ce qui explique comment on passe d'une solution rationnelle à une solution polynomiale. De plus cette méthode nous indique des voies à suivre en cas de multiplicités.