Présenter une liste de dates de manière lisible : complexité et algorithme

Guillaume Pinot

Canal TP, 20 rue Hector Malot, 75012 Paris

guillaume.pinot@canaltp.fr

Mots-clés : *complexité, algorithme,* closest string problem.

1 Introduction

Canal TP est un éditeur logiciel fournissant l'expertise et les solutions liées à l'information voyageurs y compris dans les transports en commun. Pour répondre à cette problématique, elle propose le moteur navitia disponible en logiciel libre ainsi que via le *web service* `http://navitia.io/`. Ce moteur propose entre autre un calculateur d'itinéraire et un accès aux différentes données de transport.

Dans navitia, les dates auxquelles circulent les véhicules de transport en commun sont fournies. Or, une liste de dates est peu lisible par un humain. Il est donc nécessaire de décrire sous forme condensée cette liste de dates.

2 Le problème

2.1 Description du problème

Une liste de dates constitue donc l'entrée de notre problème. La solution est une phrase, la plus courte possible, du genre *du Lundi au Vendredi du 27 avril au 31 mai 2015 sauf les 1, 8, 14 et 25 mai, avec en plus le 30 mai.*

Le résultat de ce problème doit donc décrire la liste de dates avec :

— un rythme hebdomadaire (exemple : du Lundi au Vendredi) ;
— une période de validité (exemple : du 27 avril au 31 mai) ;
— une liste de date exclues (exemple : sauf les 1, 8, 14 et 25 mai) ;
— une liste de dates incluses (exemple : avec en plus le 30 mai).

Pour que la description soit la plus simple possible, il faut minimiser le nombre d'exceptions (les dates exclues et incluses).

Il est facile de transformer la liste de dates en succession de rythmes hebdomadaires. Pour notre exemple, cela donne :

— Semaines du 27 avril et du 4 mai : Du Lundi au Jeudi ;
— Semaine du 11 mai : Du Lundi au Vendredi sauf Jeudi ;
— Semaine du 18 mai : Du Lundi au Vendredi ;
— Semaine du 25 mai : Du Mardi au Samedi.

Pour un rythme hebdomadaire donné, il est facile de généré notre résultat. Le problème est donc de trouver le rythme hebdomadaire journalier tel que le nombre d'exceptions soit minimal.

2.2 Modélisation mathématiques

Le rythme hebdomadaire peut être modélisé par une chaîne de 7 bits : un caractère par jour de la semaine, 0 pour un jour inactif, 1 pour un jour actif. La chaîne correspondant au rythme

hebdomadaire « du Lundi au Vendredi » est donc « 1111100 ».

Les données du problème peuvent alors être modélisées comme une liste de chaînes binaires de longueur 7 que l'on appellera $\mathcal{S}$.

Les données de l'exemple sont donc $\mathcal{S} = 1111000, 1111000, 1110100, 1111100, 0111110$.

Pour décrire un rythme hebdomadaire s_1 en fonction d'un autre s_2, le nombre d'exception correspond au nombre de substitutions de caractère nécessaire pour aller de s_1 à s_2. Cette distance est appelé la distance de Hamming [1]. On la notera $d(s_1, s_2)$.

Pour résoudre notre problème, nous devons donc trouver la chaîne s qui minimise la somme des distances avec $\mathcal{S}$. Ce qui nous donne la formalisation suivante :

Soit $\mathcal{S} = s_1, s_2, \ldots, s_n$, n chaînes binaires de longueur 7. Soit $d_{s_i}^{\Sigma} = \sum_{s_j \in \mathcal{S}} d(s_i, s_j)$. Le meilleur rythme hebdomadaire est représenté par la chaîne s tel que d_s^{Σ} soit minimale.

3 Comparaison avec l'existant et complexité

Le *closest string problem* est un problème très proche. Il se définit ainsi : soit $\mathcal{S} = s_1, s_2, \ldots, s_n$, n chaînes sur l'alphabet Σ de longueur m. Soit $d_{s_i}^{\max} = \max_{s_j \in \mathcal{S}} d(s_i, s_j)$. Le *closest string problem* a pour solution la chaîne s de longueur m tel que $d_s^{\max}$ soit minimale. Ce problème est NP-difficile [2] et sa complexité paramétrique est $O(nm + nd_s^{\max}(16|\Sigma|)^{d_s^{\max}})$ [3].

Notre problème est donc le *closest string problem* avec $\Sigma = \{0, 1\}, m = 7$ et une fonction objectif différente (notre problème minimise la somme d_s^{Σ} au lieu de minimiser le maximum $d_s^{\max}$). La complexité paramétrique du *closest string problem* n'étant pas exponentiel sur n, nous pouvons espérer trouver un algorithme polynomial pour résoudre notre problème (avec une constante potentiellement élevée fonction de $|\Sigma|^m = 2^7 = 128$).

4 Algorithme

Pour résoudre notre problème, nous proposons d'énumérer toutes les chaînes candidates, et d'en évaluer chaque coût. La chaîne minimisant le coût est finalement retourné. Nous avons $|\Sigma|^m = 2^7 = 128$ chaînes possibles. Évaluer une solution a pour complexité $O(mn) = O(7n) = O(n)$. Ainsi, la complexité globale de notre algorithme est $O(mn \cdot |\Sigma|^m) = O(7n \cdot 128) = O(n)$. En pratique, le temps de résolution est négligeable.

5 Conclusions et perspectives

Un petit problème pratique à première vue simple est présenté dans cet article. Ce problème se révèle finalement être très proche du *closest string problem* qui est NP-difficile. Comme la taille de certains paramètres est fixée, le problème reste malgré tout polynômial.

Ce problème pourrait être légèrement modifié pour en améliorer le résultat. Les semaines partielles sur les bords de la période de validité pourraient être prises en compte pour minimiser les exceptions. Plusieurs rythmes hebdomadaires pourraient être retournés pour condenser encore plus la phrase produite.

Références

[1] R.W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2) :147–160, 1950.

[2] J. Kevin Lanctot, Ming Lia, Bin Mab, Shaojiu Wangc, and Louxin Zhang. Distinguishing string selection problems. *Information and Computation*, 185 :41–55, 2003.

[3] Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. In *Research in Computational Molecular Biology*, volume 4955, pages 396–409. Springer, 2008.