IFT-4001 (Hiver 2016) Projet d'exploration

Alexandre Cormier (111 101 150)
Alexandre Picard-Lemieux (111 103 625)
Patrick Côté (111 103 743)
Vincent Beaudoin (111 103 778)

24 avril 2016

1 Introduction

L'optimisation combinatoire est une branche de l'informatique et des mathématiques appliquées. C'est la recherche d'une solution au coût minimal d'un problème dont l'espace des solutions est discret.

Pour le projet d'exploration, nous devons apprendre quelque chose sur les solveurs ou l'optimisation combinatoire qui n'a pas été mentionné dans le cours et en faire mention dans notre rapport. Nous ferons une étude de la viabilité de la programmation par contraintes pour attaquer la cryptographie classique par substitution.

Ce rapport sera séparé en plusieurs parties. Nous commencerons par faire une description du problème. Ensuite, nous proposerons des approches. Par la suite, le rapport sera finalisé avec le protocole d'expérimentation, les résultats ainsi qu'une discussion sur ces résultats.

2 Description du problème

Le chiffrement par substitution est une façon de cacher un message en remplaçant chaque lettre par une autre. On parle de substutition monoalphabétique lorsque qu'une lettre est toujours remplacée par la même autre lettre. Autrement, on parle de substitution polyalphabétique.

Un exemple de chiffrement par substitution monoalphabétique est le chiffre de César. Pour ce chiffre, on associe à chaque lettre sa position dans l'alphabet, commençant par 0. La clé est une lettre et chaque lettre du message est décalée d'un nombre de positions correspondant à la clé. Par exemple, le message ABC chiffré avec la clé B devient BCD. Les calculs de décalage se font en modulo 26, tel que XYZ chiffré avec la même clé devient YZA.

Le chiffre de vigenère est un exemple de chiffrement par substitution polyalphabétique. La clé est composée d'une ou plusieurs lettres et le message est placé divisé en partie de la même longueur que la clé. Si les différentes parties du message sont placées l'une en-dessous de l'autre, il suffit d'appliquer le chiffre de César à chaque colonne i avec la $i^{ème}$ lettre de la clé. Par exemple, le message ABCD chiffré avec la clé BC devient BDDF.

La substition est toujours une composante à la base de la cryptographie moderne, mais ces méthodes de chiffrement simples ne sont plus utilisées comme systèmes cryptographiques à part entière puisqu'elles sont vulnérables à différentes attaques. Il est possible, notamment, d'analyser la fréquence relative des lettres dans le message chiffré et comparer avec la fréquence relative des lettres dans la langue du message d'origine pour trouver les clés les plus probables.

Dans le cadre de ce projet, nous étudierons une méthode alternative, utilisant la programmation par contrainte, pour retrouver la clé de tels chiffrements à partir du message chiffré. Il est évidemment impossible pour un programme automatisé de trouver la clé de façon certaine, car il n'y a pas nécessairement moyen de reconnaître le message d'origine lorsque la bonne clé est trouvée. Par contre, nous pourrons ordonner les clés les plus probables selon, notamment, la fréquence relative de chaque lettre dans le message déchiffré.

L'objectif ici est d'étudier la viabilité d'une telle approche, par contrainte, pour l'attaque du chiffrement par substitution. Nous étudierons l'efficacité de l'approche selon la longueur du message et, dans le cas du chiffre de Vigenère, de la longueur de la clé.

Par exemple, en prenant le message encrypté LXFOPVEFRNHR, le solveur doit trouver que la clé la plus probable est LEMON et que le texte original est ATTACKATDAWN. L'implémentation utilisée utilisera les fréquences des lettres trouvées d'une langue donnée. [1] Cette implémentation sera donc plus efficace si la chaine est longue et si elle utilise des vrais mots de la langue choisie.

3 Approche proposée

La façon la plus préconisée pour attaquer le chiffrement de Vigenère est l'analyse de fréquences. C'est aussi l'approche que nous proposons, seulement elle sera implémentée à l'aide de la programmation par contraintes. Il s'agit de calculer la fréquence de chaque lettre dans le message déchiffré et de comparer avec la fréquence de chaque lettre dans la langue ciblée.

3.1 Modèle de base

Le modèle présenté ici est le premier modèle conçu, purement théorique, avant toute tentative de l'implémenter dans le solveur de contraintes.

3.1.1 Modèlisation du chiffrement de Vigenère

Soit :
— n la longueur de la chaîne à déchiffrer ;
— k la longueur de la clé ;
— l le nombre de lettres dans l'alphabet de la langue ciblée ;
— $\mathbb{N}_n$ l'ensemble des nombres naturels modulo n ;
— $\mathbb{Z}_n$ l'ensemble des nombres naturels modulo n ainsi que leur opposé.

On a donc, $(\forall i : 1 \leq i \leq n)$:
— $C_i \in \mathbb{N}_l$ la $i^{ème}$ lettre de la chaîne chiffrée ;
— $P_i \in \mathbb{N}_l$ la $i^{ème}$ lettre de la chaîne déchiffrée.

Et aussi, $(\forall i : 1 \leq i \leq k)$:
— $K_i \in \mathbb{N}_l$ la $i^{ème}$ lettre de la clé.

On a la contrainte suivante :

$$P_i + K_{i \bmod k} \equiv C_i \mod l \qquad \forall i : 1 \leq i \leq n \tag{1}$$

1. Wikipedia, [En ligne]. `https://en.wikipedia.org/wiki/Letter_frequency` (Page consultée le 3 avril 2016)

3.1.2 Modélisation de l'analyse de fréquences

Soit, $(\forall i : 1 \leq i \leq l)$:
— f_i la fréquence de la $i^{ème}$ lettre de l'alphabet dans la langue ciblée ;
— $Z_i \in \mathbb{N}_{n+1}$ le nombre de fois qu'apparaît la $i^{ème}$ lettre de l'alphabet dans le message déchiffré.

Et soit la contrainte suivante :

$$Z_i = Count(i, P) \qquad \forall i : 1 \leq i \leq l \tag{2}$$

Nous devons avoir une mesure d'évaluation des solutions trouvées. La distance euclidienne entre la fréquence des lettres dans le message déchiffré et celle dans la langue ciblée est un choix intuitif. La fonction objectif est donc la suivante :

$$\min \sqrt{\sum_{i=1}^{n} (\frac{Z_i}{n} - f_i)^2} \tag{3}$$

ou de façon équivalente :

$$\min \sum_{i=1}^{n} (\frac{Z_i}{n} - f_i)^2 \tag{4}$$

3.2 Modèle implémenté

Le modèle de base est simple, mais il présente quelques défis à l'implémentation [2], notamment en lien avec les nombres réels pour les fréquences. On doit aussi ajouter quelques variables et contraintes intermédiaires.

3.2.1 Variables et contraintes intermédiaires

Pour la modélisation du (dé)chiffrement, il faut ajouter des variables supplémentaires, $(\forall i : 1 \leq i \leq n)$. Soit donc :
— $I_i \in \mathbb{N}_{2l-1}$ la $i^{ème}$ lettre du texte intermédiaire.

La contrainte 1 est alors séparée en deux contraintes distinctes :

$$P_i + K_{i \bmod k} = I_i \qquad \forall i : 1 \leq i \leq n \tag{5}$$

$$I_i \bmod l = C_i \qquad \forall i : 1 \leq i \leq n \tag{6}$$

De plus, il faut séparer la fonction objectif en plusieurs variables et contraintes. On introduit donc les variables intermédiaires suivantes, $(\forall i : 1 \leq i \leq l)$:
— $F_i \in [0, 1]$ la fréquence de la $i^{ème}$ lettre de l'alphabet dans le message déchiffré ;
— $D_i \in [-1, 1]$ la différence entre la fréquence de la $i^{ème}$ lettre dans le message déchiffré et celle dans la langue ciblée ;
— $A_i \in [0, 1]$ cette différence élevée à la puissance 2.

Il faut aussi ajouter une variable à minimiser :

2. Nous avons fait l'implémentation avec le solveur Choco.

— $S \in [0, l]$ la somme des différences de fréquences au carré.

Pour lier ces variables ensemble, il faut de nouvelles contraintes :

$$F_i = \frac{Z_i}{n} \qquad \forall i : 1 \leq i \leq l \tag{7}$$

$$D_i = F_i - f_i \qquad \forall i : 1 \leq i \leq l \tag{8}$$

$$A_i = D_i^2 \qquad \forall i : 1 \leq i \leq l \tag{9}$$

$$S = \sum_{i=1}^{n} A_i \tag{10}$$

3.2.2 Gestion des nombres réels

Pour simplifier l'implémentation avec le solveur, nous avons adopté une stratégie pour les transformer en entiers. Nous avons choisi le niveau de précision désiré, selon la précision des fréquences dont nous disposons [3] pour les langues cibles.

Soit donc :

— p la précision choisie en nombre de décimales.

On applique donc la modification suivante aux fréquences relatives des lettres de la langue ciblée, $(\forall i : 1 \leq i \leq l)$:

$$f_i = f_i * 10^p \qquad \forall i : 1 \leq i \leq l \tag{11}$$

On doit aussi multiplier le compte des lettres, alors on introduit des variables intermédiaires, $(\forall i : 1 \leq i \leq l)$:

— $M_i \in \mathbb{N}_{10^p(n+1)}$ le nombre de fois qu'apparaît la $i^{ème}$ lettre de l'alphabet dans le message déchiffré, multiplié par 10^p.

On ajoute donc la contrainte suivante :

$$M_i = Z_i * 10^p \qquad \forall i : 1 \leq i \leq l \tag{12}$$

Et la contrainte (7) devient plutôt :

$$F_i = \frac{M_i}{n} \qquad \forall i : 1 \leq i \leq l \tag{13}$$

Puisqu'on se retrouve ainsi avec de grands nombres, il faut se méfier du dépassement d'entiers (*integer overflow*). On peut donc simplifier la fonction objectif en remplaçant le carré par la valeur absolue :

$$\min \sum_{i=1}^{n} |\frac{Z_i}{n} - f_i| \tag{14}$$

Cette nouvelle fonction objectif n'est pas équivalente à la première, mais elle permet d'éviter des problèmes et nos résultats sont même meilleurs avec celle-ci que nos résultats préliminaires avec la distance euclidienne.

On doit aussi modifier le domaine des variables reliées à la fréquence. Soit, $(\forall i : 1 \leq i \leq l)$:

3. Wikipedia, [En ligne]. `https://en.wikipedia.org/wiki/Letter_frequency` (Page consultée le 24 avril 2016)

— $F_i \in \mathbb{N}_{10^p+1}$ la fréquence de la $i^{ème}$ lettre de l'alphabet dans le message déchiffré ;
— $D_i \in \mathbb{Z}_{10^p+1}$ la différence entre la fréquence de la $i^{ème}$ lettre dans le message déchiffré et celle dans la langue ciblée ;
— $A_i \in \mathbb{N}_{10^p+1}$ la valeur absolue de cette différence.

Et de même pour la variable à minimiser :
— $S \in \mathbb{N}_{l(10^p+1)}$ la somme des valeurs absolues des différences de fréquences.

Finalement, la division de la contrainte 7 doit maintenant être une division entière.

Au final, nous avons des listes de n, k et l variables. Le nombre de variables est donc borné par $\theta(n + k + l)$.

Chaque variable dans les liste de n ont l ou $2l - 1$ valeurs possibles.
Chaque variable dans les liste de k ont l valeurs possibles.
Chaque variable dans les liste de l ont $n + 1$, $10^p(n + 1)$ ou $2(10^p(n + 1))$ valeurs possibles.
Le nombre de valeurs est donc borné par $\theta(nl + kl + l(10^p n))$. p peut toutefois être considéré comme une constante (nous utilisons 5) et donc on obtient $\theta(l(n + k))$.

Nous avons n contraintes de type (5) et (6), l contraintes de type (2), (8), (9), (12) et (13) ainsi qu'une contrainte de type (10). Le nombre de contraintes est donc borné par $\theta(n + l)$.

3.3 Amélioration des heuristiques

3.4 Recherche des n meilleures solutions

4 Protocole d'expérimentation

Nous avons choisi de faire des tests sur des instances de différentes longueurs de message et de clé, des messages qui respectent bien la distribution de lettres anglaises, d'autre qui ne la respectent pas. Les métriques que nous allons observer durant les expérimentations sont le temps de réponse, le nombre de retours arrière, ainsi que le nombre de solutions trouver.

La première instance est une instance qui est considérée comme étant un message long ayant une clé de longueur 4 et qui devrait avoir une distribution correcte de la fréquence des lettres.
Voici le texte brut : FRIEN DSROM ANSCO UNTRY MENLE NDMEY OUREA RSICO METOB URYCA ESARN OTTOP RAISE HIMTH EEVIL THATM ENDOL IVESA FTERT HEMTH EGOOD ISOFT INTER REDWI THTHE IRBON ESSOL ETITB EWITH CAESA RTHEN OBLEB RUTUS HATHT OLDYO UCAES ARWAS AMBIT IOUSI FITWE RESOI TWASA GRIEV OUSFA ULTAN DGRIE VOUSL YHATH CAESA RANSW ERDIT HEREU NDERL EAVEO FBRUT USAND THERE STFOR BRUTU SISAN HONOU RABLE MANSO ARETH EYALL ALLHO NOURA BLEME NCOME ITOSP EAKIN CAESA RSFUN ERAL

Voici le texte encrypté : YKBXG WLKHF TGLVH NGMKR FXGEX GWFXR HNKXT KLBVH FXMHU NKRVT XLTKG HMMHI KTBLX ABFMA XXOBE MATMF XGWHE BOXLT YMXKM AXFMA XZHHW BLHYM BGMXK KXWPB MAMAX BKUHG XLLHE XMBMU XPBMA VTXLT KMAXG HUEXU KNMNL ATMAM HEWRH NVTXL TKPTL TFUBM BHNLB YBMPX KXLHB MPTLT ZKBXO HNLYT NEMTG WZKBX OHNLE RATMA VTXLT KTGLP XKWBM AXKXN GWXKE XTOXH YUKNM NLTGW MAXKX LMYHK UKNMN LBLTG AHGHN KTUEX FTGLH TKXMA XRTEE TEEAH GHNKT UEXFX GVHFX BMHLI XTDBG VTXLT KLYNG XKTE

La deuxième instance est une instance qui est considérée comme étant un message d'une courte longueur ayant une clé de longueur 1 et qui a une bonne distribution de la fréquence des lettres. De plus, cette instance va être réutilisée pour faire des tests avec des clés de longueur 2,3,4,5.
Voici le texte brut : GENIUS WITHOUT EDUCATION IS LIKE SILVER IN THE MINE

La troisième instance est une instance qui est considérée comme un long message ayant une très bonne distribution de la fréquence des lettres que nous avons testées sur une clé de 1 et de deux.
Voici le texte brut : HEREUPON LEGRAND AROSE WITH A GRAVE AND STATELY AIR AND BROUGHT ME THE BEETLE FROM A GLASS CASE IN WHICH IT WAS ENCLOSED IT WAS A BEAUTIFUL SCARABAEUS AND AT THAT TIME UNKNOWN TO NATURALISTS OF COURSE A GREAT PRIZE IN A SCIENTIFIC POINT OF VIEW THERE WERE TWO ROUND BLACK SPOTS NEAR ONE EXTREMITY OF THE BACK AND A LONG ONE NEAR THE OTHER THE SCALES WERE EXCEEDINGLY HARD AND GLOSSY WITH ALL THE APPEARANCE OF BURNISHED GOLD THE WEIGHT OF THE INSECT WAS VERY REMARKABLE AND TAKING ALL THINGS INTO CONSIDERATION I COULD HARDLY BLAME JUPITER FOR HIS OPINION RESPECTING IT

La quatrième instance est une instance ayant une longueur courte et qui ne respect pas très bien la distribution des lettres. Nous l'avons testé sur une clé de longueur 1.
Voici le texte brut : ZJXQKB

5 Résultats

Voici les résultats sans l'heuristique que nous avons créée.

Instances	**Résultats**				
	Longueur clé	**Retour arrière**	**Temps de réponse**	**Nb solutions**	**Présent**
1	4	219 421	93,992 secondes	158	Oui
2	1	55	0,373 secondes	21	Oui
2	2	1369	1,467 secondes	82	Oui
2	3	61 985	8,274 secondes	121	Oui
2	4				
2	5				
3	1	51	1,018 secondes	18	Oui
3	2	1323	4,099 secondes	65	Oui
4	1	63	0,198 secondes	17	Non

Voici les résultats avec l'heuristique que nous avons créée. * Les résultats sont ceux sans l'heuristique.

6 Discussion

7 Conclusion

Instances	**Résultats**				
	Longueur clé	**Retour arrière**	**Temps de réponse**	**Nb solutions**	**Présent**
1	4	219 421	93,992 secondes	158	Oui
2	1	55	0,373 secondes	21	Oui
2	2	1369	1,467 secondes	82	Oui
2	3	61 985	8,274 secondes	121	Oui
2	4				
2	5				
3	1	51	1,018 secondes	18	Oui
3	2	1323	4,099 secondes	65	Oui
4	1	63	0,198 secondes	17	Non