Automatisation de la cryptanalyse des cryptosystèmes classiques à l'aide d'algorithmes modernes

Helder Brito
O'nel Hounnouvi

Table des matières

1 Substitution monoalphabétique

1.1 Introduction

La substitution monoalphabétique est l'une des plus anciennes méthodes de chiffrement. Elle consiste à remplacer dans le message clair une lettre donnée de l'alphabet par une autre lettre. Voici un exemple :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

Le message *SUBSTITUTION* devient *PRYPQFQRQFLK*.

L'alphabet latin comporte 26 lettres. Cela permet donc de construire $26! = 4 \times 10^{26}$ permutations, soit de l'ordre de 2^{88}. Sachant qu'environ 2^{58} secondes se sont écoulées depuis la création de l'univers, il serait impossible d'explorer toutes les permutations. Ce chiffre donne une impression de sûreté qui est toutefois trompeuse...

1.2 Cryptanalyse

La substitution monoalphabétique possède de grosses faiblesses structurelles. Les chiffres utilisant cette méthode sont « faciles » à casser par analyse fréquentielle. Notre analyse ici sera basée sur l'analyse des fréquences d'apparition des n-grammes dans le message chiffré.

Un *n-gramme* est une séquence de n lettres consécutives dans un texte. Par exemple, dans le mot *CRYPTANALYSE*, les bigrammes ($n = 2$) incluent *CR*, *RY*, *YP*, ..., tandis que les trigrammes ($n = 3$) sont *CRY*, *RYP*, *YPT*, ... En utilisant un dictionnaire de référence contenant les fréquences relatives des n-grammes dans un large corpus de textes en français, il est possible d'estimer la probabilité qu'un texte donné soit écrit dans cette langue.

Pour évaluer la qualité des solutions potentielles et identifier celle qui correspond le mieux à du français, nous attribuons un score à chaque solution avec une *fitness function*.

La fonction de score utilisée dans ce projet est basée sur la somme des probabilités logarithmiques des n-grammes. C'est la fonction de log-vraisemblance. Elle est définie comme suit :

$$score = -\sum \log(frequence(c_1 \dots c_n))$$

L'objectif ici est de minimiser cette fonction à cause du changement de signe. La solution ayant le plus petit score est celle qui sera « le plus » français.

1.3 Métaheuristiques

Une *métaheuristique* est un algorithme d'optimisation visant à résoudre des problèmes pour lesquels on ne connaît pas de méthode classique plus efficace. Les métaheuristiques sont généralement des algorithmes stochastiques [1] itératifs, qui progressent vers un optimum global (c'est-à-dire l'extrémum global d'une fonction).

1. Un processus stochastique est un processus qui intègre des éléments d'aléatoire, c'est-à-dire dont l'évolution est déterminée par des phénomènes aléatoires.

1.3.1 Hill Climbing

L'idée générale pour trouver la clef de déchiffrement est la suivante :

1. Partir d'une clef aléatoire ;
2. Utiliser la clef pour déchiffrer le cryptogramme ;
3. Calculer le score du texte obtenu ;
4. Si ce score est meilleur que le score précédent, adopter cette nouvelle clef comme clef courante ; sinon, conserver l'ancienne ;
5. Modifier légèrement la clef courante ;
6. Revenir à l'étape 2 tant que la clef correcte n'a pas été trouvée.

Il arrive cependant fréquemment que l'algorithme se retrouve bloqué : il n'arrive plus à améliorer la clef actuelle, bien que la véritable clef n'ait pas encore été trouvée. Pour éviter qu'il ne tourne indéfiniment dans cette impasse, nous introduisons un compteur qui mesure le nombre de tentatives infructueuses de modification de la clef. Si, après 200 itérations consécutives, aucune amélioration n'a été constatée, l'algorithme s'arrête et retourne la meilleure clef obtenue jusque-là.

Pour faire une analogieLe randonneur part d'un point au hasard dans la montagne. À chaque pas, il regarde autour de lui et choisit toujours de descendre si possible. Il répète ce processus en allant toujours vers le bas. Mais, s'il atteint un endroit où aucun pas ne le fait descendre davantage, il s'arrête – même s'il est juste coincé dans un petit creux, et non dans la vraie vallée.

Modification de la clef

Il nous faut maintenant définir comment générer une nouvelle clef à partir de celle en cours (étape 5 de l'algorithme). Pour cela, on effectue une permutation aléatoire de deux lettres dans la clef actuelle.

$$\texttt{QWERTZUIOPASDFGHJKLXCVBNM} \rightarrow \texttt{QGERTZUIOPASDFWHJKLXCVBNM}$$

1.3.2 Hill Climbing optimisé

Comme son nom l'indique, cette méthode est une version améliorée du Hill Climbing classique. Le principal inconvénient de l'approche standard est sa tendance à rester bloquée dans un minimum local, empêchant ainsi de découvrir la véritable clef.

L'algorithme optimisé propose une solution à ce problème : au lieu d'abandonner après un certain nombre d'échecs, il effectue un redémarrage aléatoire. Plus précisément, si aucune amélioration n'est observée après 200 itérations consécutives, une nouvelle clef complètement aléatoire est générée, et l'algorithme recommence depuis l'étape 1.

Cette stratégie permet d'explorer plusieurs régions de l'espace des solutions, augmentant ainsi significativement les chances d'atteindre le minimum global — autrement dit, la clef correcte.

1.3.3 Recuit simulé

1.4 Différence entre Hill Climbing et Recuit simulé

Pour bien comprendre les différences entre ces deux méthodes d'optimisation, utilisons l'analogie suivante.

1.4.1 Hill Climbing

Imaginons un randonneur qui cherche le point le plus bas dans une vallée, mais qui ne voit que ce qui est autour de lui à la lumière de sa lampe frontale. Le randonneur commence à un point au hasard et, à chaque pas, il regarde autour de lui et choisit de descendre si possible. Il répète ce processus en allant toujours vers le bas. Cependant, si le randonneur atteint un endroit où aucun pas ne le fait descendre davantage, il s'arrête, même s'il est coincé dans un petit creux plutôt qu'au fond de la vallée.

Cela peut être efficace si le terrain est simple, mais le randonneur risque de se retrouver bloqué dans un minimum local, au lieu de trouver le minimum global.

1.4.2 Recuit simulé

Le recuit simulé est une stratégie plus astucieuse. Au début, le randonneur est prêt à faire des pas au hasard, même si cela le fait remonter légèrement. Cela lui permet de sortir des petits creux et de continuer à descendre vers le bas. Au fil du temps, la lampe frontale devient plus précise, et le randonneur ne fait plus que des pas qui le font descendre. Cette approche permet au randonneur de ne pas se laisser piéger dans un minimum local et d'augmenter ses chances d'atteindre le minimum global.

Le recuit simulé est donc plus efficace pour trouver "le point le plus bas global" d'un terrain complexe, même si cela prend parfois plus de temps.

1.5 Résultat

1.5.1 Hill climbing

1.5.2 Hill Climbing optimisé

1.5.3 Recuit simulé