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,

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.

qui progressent vers un optimum global (c'est-à-dire l'extrémum global d'une fonction).

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 analogie, le 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

Le recuit simulé est une méthode d'optimisation inspirée du processus de recuit en métallurgie, où un matériau est chauffé puis refroidi lentement pour atteindre un état de faible énergie. Notre algorithme est présenté de la façon suivante :

1. **Initialisation :**
   (a) Partir d'une clé aléatoire $C1$ et d'une température initiale T_0
   (b) Utiliser la clé pour dechiffrer le cryptogramme
   (c) Calculer le score du texte obtenu
2. **Boucle principale :**
   (a) Générer une solution voisine $C2$ en faisant une légère modification (voir 1.3.1)
   (b) Calculer le changement de coût, défini par

$$\Delta = score(C2) - score(C1).$$

   (c) Si $\Delta \leq 0$, accepter $C2$ (la solution s'améliore ou reste équivalente)
   (d) Sinon, accepter $C2$ avec une probabilité donnée par

$$P_{\text{accept}} = \exp\left(-\frac{\Delta}{T}\right).$$

   (e) Mettre à jour la température avec le coefficient de refroidissement α après un nombre d'itérations :

$$T \leftarrow \alpha T, \quad \text{avec } 0 < \alpha < 1.$$

3. **Critère d'arrêt :** Terminer l'algorithme après un nombre prédéfini d'itérations, puis retourner la solution finale s.

Le recuit simulé échappe aux minima locaux en introduisant une étape d'acceptation probabiliste des solutions moins bonnes. Concrètement, au lieu d'accepter uniquement les modifications qui améliorent le score, l'algorithme accepte une solution voisine avec la probabilité P_{accept}.

Au début, la température T est élevée, ce qui rend l'expression $\exp\left(-\frac{\Delta}{T}\right)$ relativement grande. Cela permet donc à l'algorithme d'accepter des solutions moins bonnes et d'explorer plus librement l'espace de recherche, en sautant potentiellement hors d'un minimum local.

Au fur et à mesure que l'algorithme progresse, la température est progressivement abaissée (refroidissement), ce qui diminue la probabilité d'accepter des solutions moins performantes. Ainsi, en phase finale, le recuit simulé affine la solution dans un voisinage qui se rapproche d'un minimum global.

2 Résultats

Pour évaluer l'efficacité des différentes méthodes métaheuristiques, nous avons réalisé trois jeux de tests pour chaque algorithme. Dans chacun de ces tests, nous avons utilisé des textes chiffrés de longueurs différentes : 110, 509 et 1150 caractères.

L'objectif était d'observer l'évolution du score (log-vraisemblance) en fonction du nombre maximum de permutations autorisées pendant l'exécution de l'algorithme. Cela permet d'analyser la rapidité de convergence, la stabilité, ainsi que la capacité de chaque méthode à s'approcher du minimum global.

Les résultats obtenus sont présentés sous forme de graphiques pour une comparaison visuelle claire entre les méthodes. Ils sont tous disponibles en annexe.

2.1 Hill Climbing

Impact de la longueur du texte :
Plus le texte est long, plus les *scores moyens diminuent*, en particulier pour les *trigrammes* ($n = 3$) et *quadrigrammes* ($n = 4$). Cela s'explique par le fait qu'un texte plus long fournit *davantage de contexte statistique*, facilitant la détection de séquences plausibles en français.

Observations principales :

- **Scores limités malgré plus de permutations :** L'algorithme atteint souvent un *plateau* de performance, même lorsque le nombre de permutations autorisées est élevé. (Voir figure 1)
- **Problème des minima locaux :** Le hill climbing classique a du mal à *s'échapper des minima locaux*, ce qui réduit son efficacité, surtout pour des critères exigeants comme les *quadrigrammes*. (Voir figure 2)
- **Dépendance à la clef initiale :** Les résultats peuvent fortement varier selon la *clef de départ*, ce qui rend l'algorithme *peu fiable* dans certains cas. (Voir figure 3)

Remarque : Le *nombre de permutations effectivement réalisées* peut être très inférieur au maximum autorisé. Par exemple, si l'algorithme ne progresse plus, il peut s'arrêter alors qu'il reste encore de nombreuses permutations possibles.

2.2 Hill Climbing optimisé

Lorsque l'on applique la version optimisée du hill climbing, on observe une amélioration notable de la qualité des résultats, notamment pour les trigrammes ($n = 3$) et quadrigrammes ($n = 4$). Grâce au mécanisme de relance (génération d'une nouvelle clef aléatoire après 200 itérations sans amélioration), l'algorithme parvient à explorer davantage l'espace des solutions.

Observations principales :

- **Amélioration des scores moyens** : Pour un même nombre de permutations, les scores finaux sont généralement supérieurs à ceux obtenus par le hill climbing standard, en particulier pour les grands n-grammes.
- **Stabilité accrue** : Les résultats sont plus réguliers, avec une variance réduite par rapport au hill climbing simple. (Voir figure 5) Cela signifie que l'algorithme est plus fiable, et ne dépend pas autant du hasard de la clef initiale.
- **Gain d'efficacité sur les grands textes** : Le bénéfice de la relance devient particulièrement visible avec les textes plus longs. (Voir figure 6) La diversité des séquences statistiques donne plus de chances à une clef correcte d'être distinguée par le score.

Remarque : Ces observations ne semblent pas tout à fait correspondre dans le cas où les textes sont courts : il n'y à presque aucun changement avec le hill climbing classique.

2.3 Recuit simulé

Annexes

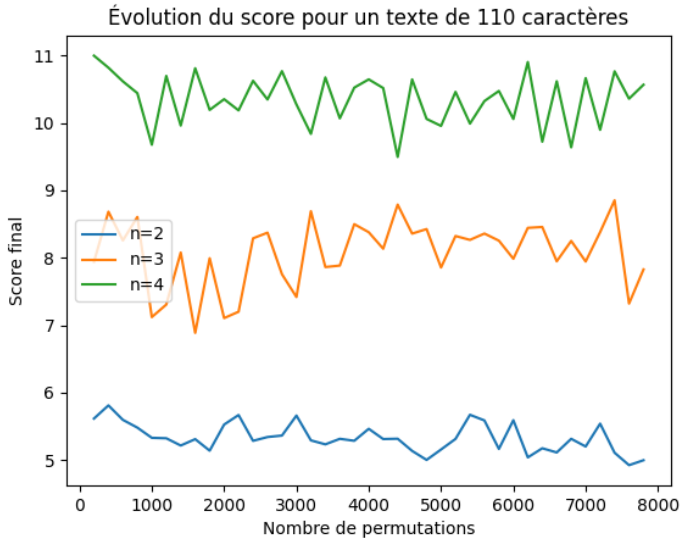

FIGURE 1 – Hill Climbing sur un texte de 110 caractères.

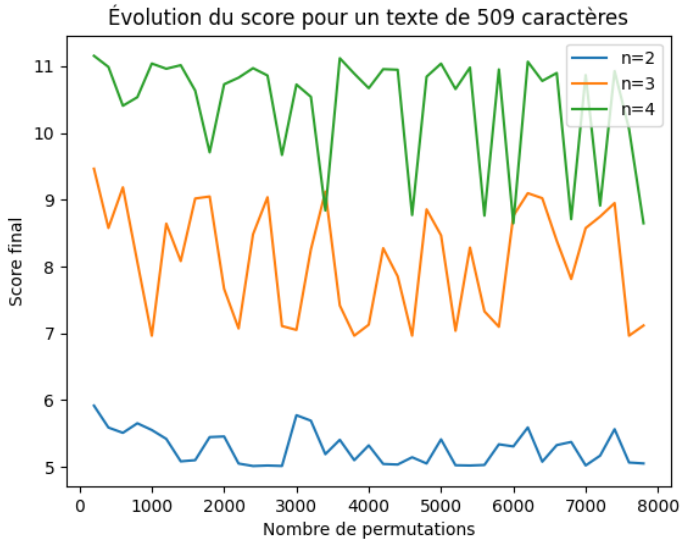

FIGURE 2 – Hill Climbing sur un texte de 509 caractères.

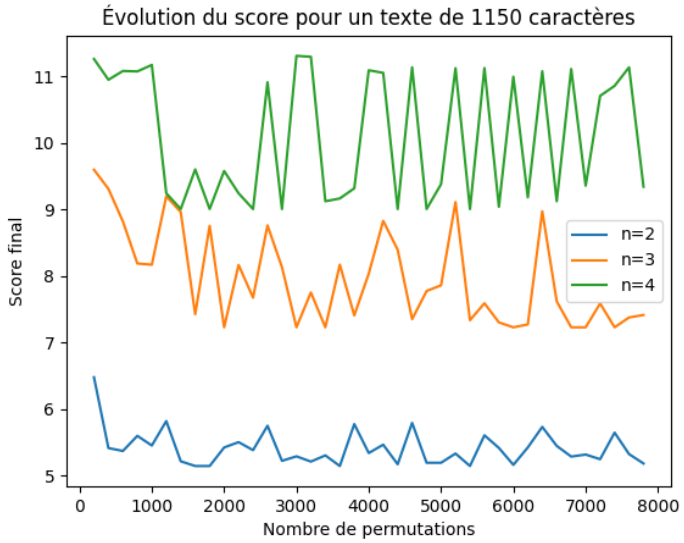

FIGURE 3 – Hill Climbing sur un texte de 1150 caractères.

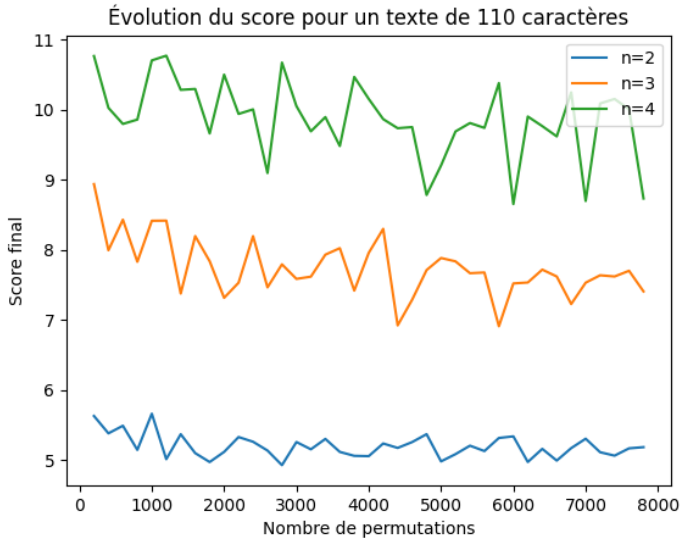

FIGURE 4 – Hill Climbing optimisé sur un texte de 110 caractères.

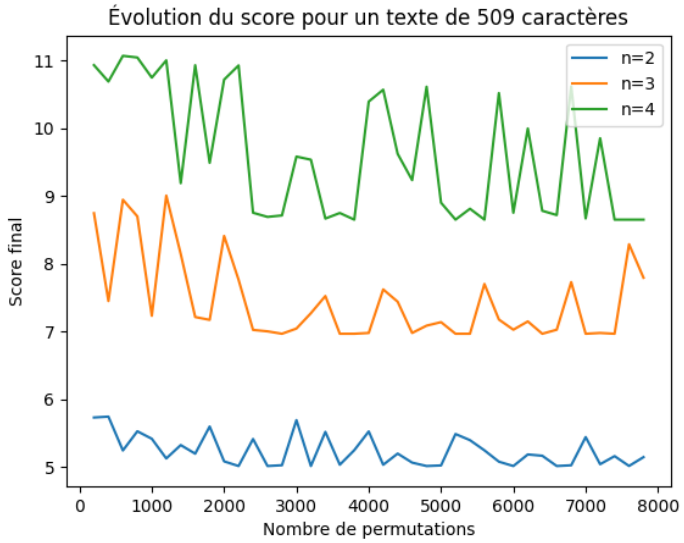

FIGURE 5 – Hill Climbing optimisé sur un texte de 509 caractères.

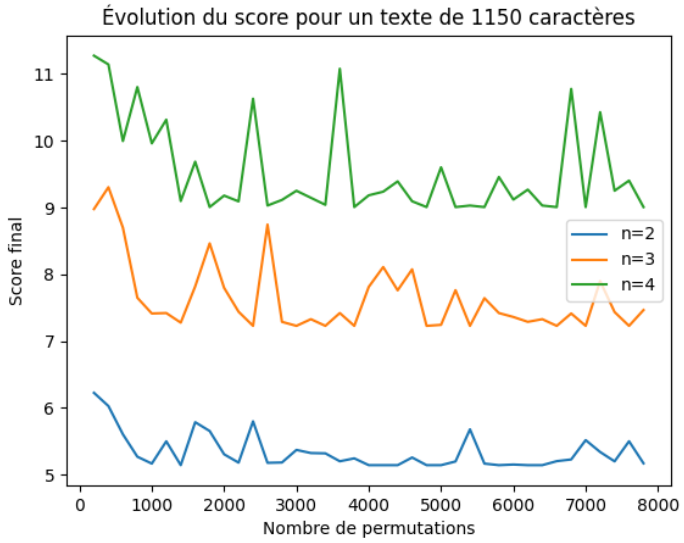

FIGURE 6 – Hill Climbing optimisé sur un texte de 1150 caractères.