Les métaheuristiques en cryptanalyse

Didier Müller, Lycée cantonal de Porrentruy, didier.muller@divlyc.ch

Une méthode développée depuis une dizaine d'années pour décrypter des textes chiffrés consiste à rechercher la clef de déchiffrement en utilisant ce qu'on appelle des *métaheuristiques* ([1], [4], [6]). Ce sont des méthodes d'optimisation qui sont utilisées quand on ne connaît pas de méthodes plus efficaces. Parmi les plus connues, citons le *hill climbing*, le *recuit simulé*, et la *recherche avec tabous*.

Décryptement d'une substitution monoalphabétique

Une substitution monoalphabétique consiste simplement à remplacer une lettre par une autre.

La *clef* sera une suite de 26 lettres différentes, qui représentera la *table de déchiffrement*. Une clef pourrait être : **qwertzuiopasdfghjklxcvbnm** correspondant à cette table de déchiffrement :

Chiffré	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
Clair	q	w	e	r	t	z	u	i	o	p	a	s	d	f	g	h	j	k	l	y	x	c	v	b	n	m

Il faut maintenant définir un *score* qui nous donnera une estimation de la qualité du décryptement. Autrement dit, est-ce que le texte déchifré est du français ? La tâche est facile pour un humain, mais plus subtile pour un ordinateur ! Plusieurs possibilités s'offrent à nous. Toutes sont basées sur les caractéristiques statistiques de la langue : fréquences des lettres, des bigrammes, des trigrammes, des tétragrammes, etc.

La méthode qui fonctionne le mieux, selon les spécialistes, est basée sur les tétragrammes (suites de quatre lettres, aussi appelés *quadrigrammes*). Une description complète et très pédagogique est donnée dans [3]. On y trouve notamment le fichier des apparitions des tétragrammes en français et dans d'autres langues européennes.

J'ai établi mes propres statistiques sur un corpus de plus de 10 millions de lettres : 26 textes variés écrits entre 1831 et 2017 (voir détails sur [5]). La liste triée, contenant 68'183 tétragrammes différents, commence ainsi :

tétragrammes	nombres d'apparitions
MENT	30207
ELLE	28622
QUEL	23983
EMEN	22824
TION	20768
DANS	20276
IENT	19220
ESDE	18171
DELA	17248
OMME	16947

Il faudra d'abord modifier ce dictionnaire en remplaçant le nombre d'apparitions en fréquences. Cela se fait très simplement en divisant chaque nombre par le nombre total d'apparitions de tétragrammes.

Ce nouveau dictionnaire permet d'évaluer le score d'un texte en faisant le produit des fréquences de ses tétragrammes. Si un tétragramme n'est pas dans le dictionnaire, on lui donne la fréquence 10^{-10} (cette valeur semble donner de meilleurs résultats que 10^{-100} proposé dans [3]).

Mais il y a un souci informatique dû à la précision des ordinateurs : quand un nombre est vraiment très proche de 0, l'ordinateur mettra effectivement la valeur 0. Comme toutes les fréquences seront comprises entre 0 et 1, le produit donnera presque toujours 0 si le texte est long. Pour pallier ce problème, on va remplacer notre produit par la somme des logarithmes des fréquences des tétragrammes, et comme les logarithmes sont négatifs, on prendra comme score l'opposé de cette somme :

$$\text{score} = -\sum_{i=0}^{n-4} \log\left(\mathit{fréquence}\left(c_i c_{i+1} c_{i+2} c_{i+3}\right)\right)$$

Cette idée de somme de logarithmes a été proposée par **Abraham Sinkov** en 1966 déjà (voir [2, pp. 76-77]).

Pour des textes de même longueur, du fait du changement de signe, c'est celui qui aura **le plus petit score** qui devrait être « le plus » français. Vérifions.

score("KFQTDEGRNWKNIQ") = 94.79783559030287
score("CESTDUFRBNCBIS") = 77.57382632660952
score("CESTDURFANCAIS") = 50.39139100671383
score("CESTDUFRANCAIS") = 45.97301718417951

Algorithme de décryptement

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, prendre cette nouvelle clef ; sinon garder la clef précédente ;
5. modifier légèrement la clef (voir ci-dessous) ;
6. retourner en 2 tant qu'on n'a pas trouvé la bonne clef.

Il peut arriver, et c'est même assez fréquent, que l'algorithme « se coince » : impossible de trouver une meilleure clef, alors que pourtant la vraie clef n'a pas encore été découverte. Dans ce cas, il faut parfois accepter de prendre une clef moins bonne en terme de score, afin de repartir de plus belle.

Prenons une image. Vous voulez monter au sommet de l'Everest. Vous commencez votre ascension au Népal. Arrivé au sommet d'une montagne, vous constatez qu'il y en a d'autres plus élevées autour de vous ! Vous serez donc obligé de redescendre avant d'escalader une autre montagne, en espérant que ce soit bien l'Everest...

Modification de la clef

Il faut maintenant définir comment nous allons modifier notre clef pour en obtenir une nouvelle (point 5 de l'algorithme).

La première modification consistera à **échanger deux lettres** au hasard :

QWERTZUIOPASDF**G**HJKLXCVBNM → Q**G**ERTZUIOPASDF**W**HJKLXCVBNM

La seconde sera de **déplacer une lettre** dans la clef :

Q**W**ERTZUIOPASDFGHJKLXCVBNM → QERTZUIO**W**PASDFGHJKLXCVBNM

Une clef C_2 obtenue à partir de C_1 par une de ces deux modifications sera appelée *voisine* de C_1.

Recuit simulé

Essayons un algorithme de *recuit simulé*. On progresse vers la solution en passant d'une clef à l'une de ses « voisines », cette dernière étant tirée au sort. De deux choses l'une : soit la nouvelle clef est meilleure, et on la garde comme clef courante, soit elle est moins bonne, et on ne la gardera qu'avec une certaine probabilité qui diminuera avec le temps. Ceci dans le but d'éviter de se coincer dans un minimum local.

1. Choisir une « température » de départ T.
2. Générer une clef aléatoire. Appelons-la C_1.
3. Parmi toutes les clefs voisines de C_1, en prendre une au hasard et la nommer C_2.
4. Calculer $\Delta = \text{score}(C_2) - \text{score}(C_1)$
5. Si $\Delta \leq 0$, $C_1 \leftarrow C_2$; le cas échéant, mettre à jour le meilleur score et la meilleure clef. Aller à 8.
6. Générer un nombre réel aléatoire r dans l'intervalle [0 ; 1].
7. Si $r < \exp(-\Delta/T)$, $C_1 \leftarrow C_2$.
8. Diminuer T toutes les n itérations.
9. Retourner à 3, tant qu'on n'a pas décidé de s'arrêter.

Le terme *température* du point 1 fait référence au procédé sidérurgique du recuit (d'où le nom de la méthode) : on chauffe fortement un métal, puis on fait descendre la température par paliers.

Dans le recuit simulé, la décroissance de la température est un paramètre important avec lequel il faut jouer. Si la température décroît rapidement, on arrivera vite à un bon résultat, mais on risque de se coincer plus souvent dans un minimum local. En effet, quand T diminue, $\exp(-\Delta/T)$ diminue aussi, et on retiendra moins volontiers une plus mauvaise clef. On fait généralement décroître la température par paliers : on laisse la température stable un certain nombre d'itérations (encore un paramètre délicat à régler), puis on multiplie T par un coefficient $0 < c < 1$ (dernier paramètre à choisir). Remarquons encore que plus Δ est grand, plus la probabilité d'acceptation d'une mauvaise clef est faible.

Recherche avec tabous

Un autre moyen de sortir des minima locaux est d'utiliser une *liste de tabous* : on garde en mémoire les dernières clefs générées, afin de ne pas les réutiliser, ce qui provoquerait des allers-retours incessants sans progresser vers la clef cherchée. Si la liste est suffisamment longue, cela évite les blocages. La longueur de la liste des tabous est d'ailleurs le seul paramètre avec lequel on peut jouer.

1. Générer une clef aléatoire.
2. Parmi toutes les clefs voisines, prendre celle qui donne le meilleur score et qui n'est pas dans la liste des tabous.
3. Si la liste des tabous est pleine, retirer de cette file la clef la plus ancienne.
4. Mettre la nouvelle clef en queue de la liste des tabous.
5. Retourner à 2, tant qu'on n'a pas décidé de s'arrêter.

Il est à noter qu'avec cet algorithme, le hasard n'intervient qu'au tout début, quand on choisit une clef aléatoire. De plus, la solution ne va pas forcément à chaque itération, mais, à la longue, on se rapprochera d'un score minimum.

Quelques essais

Pour tester ces deux métaheuristiques, nous avons chiffré six textes en français :

- Un extrait du texte sur Leonhard Euler tiré de Wikipédia (681 lettres)
- Le poème « L'albatros » de Baudelaire (565 lettres)
- Le début de « 1984 », nouvelle traduction, éditions de la rue Dorion (4331 lettres)
- Un extrait de « La disparition » de Georges Perec (834 lettres). Il n'y a pas de E dans ce texte...
- Le début de « Salammbo » de Gustave Flaubert (255 lettres). Ce sera le texte le plus difficile à décrypter, car il est court et contient des noms propres comme Mégara, Carthage et Hamilcar.
- La fable « Le cheval et l'âne » de Jean de la Fontaine (496 lettres)

Le tableau ci-après montre les résultats obtenus. La clef correspond à la deuxième ligne de la table déchiffrante présentée au début de cet article. Les erreurs sont soulignées. On donne aussi le score du texte original et les scores des textes déchiffrés.

Nous avons pris à chaque fois le résultat du premier essai. Vu qu'il y a une part de hasard, d'autres tentatives donneront des résultats légèrement différents, parfois meilleurs, parfois moins bons. Il est à noter que, même avec des clefs légèrement inexactes, le décryptement partiel est suffisamment bon pour lire le texte clair.

À noter encore que la recherche avec tabous trouve souvent sa meilleure clef bien avant le nombre maximum d'itérations (une vingtaine d'itérations suffisent en général). Clairement, cette métaheuristique est la plus rapide.

Nous voyons quelque chose de surprenant : parfois, le score du texte décrypté est plus petit (et donc meilleur) que le texte original ! Et pourtant le texte décrypté comporte des erreurs... Cela nous indique que notre score est une bonne mesure, puisque les résultats sont tout à fait acceptables, mais pas parfaite.

Est-ce que l'usage de polygrammes plus longs (5, 6 caractères ou plus) donnerait de meilleurs résultats ? **Jean-Louis Morel**, l'auteur de [3], a essayé d'utiliser des hexagrammes. Il a obtenu de meilleurs résultats pour les cryptogrammes courts, mais pas toujours. Et pour les cryptogrammes longs, on ne gagne rien. Le problème est que le corpus doit être gigantesque. En effet, en analysant 10 millions de lettres, on ne rencontre que 0,3 % de tous les hexagrammes théoriquement possibles !

	Meilleure clef trouvée avec le recuit (mêmes paramètres que dans [3])	**Avec la recherche avec tabous** (100 itérations, longueur de la liste des tabous « infinie »)
Euler score du clair = 2733.1865257467216	PBEVDHKMNQTA**Z**WG**J**CSFIYLORUX score = 2729.173193215509 1.5 minute La bonne clef était : PBEVDH?MNQTA??GZCSFIYLORUX ? = J, K ou W	PBEVDHKMNQTA**Z**WG**J**CSFIYLORUX score = 2729.173193215509 0.7 minute J, K et W n'apparaissent pas dans le texte clair
L'albatros score du clair = 2271.81202702689	GSAWOBEDYHKMNPRTVJQXUZCFIL score = 2271.81202702689 1.3 minute La bonne clef était : GSA?OBEDYH?MNPRTV?QXUZCFIL ? = J, K ou W	GSAWOBEDYHKMNPRTV**Z**QXU**J**CFIL score = 2270.1056606881075 0.7 minute J, K et W n'apparaissent pas dans le texte clair
1984 score du clair = 17219.781643225346	UBODGMZFCHJEAILNS**K**PQ**W**YRTVX score = 17217.90039927846 11.2 minutes La bonne clef était : UBODGMZFCHJEAILNSWPQKYRTVX	UBODGMZFCHJEAILNS**K**PQ**W**YRTVX score = 17217.90039927846 5.2 minutes
La disparition score du clair = 3644.1768594249875	QBMSCDAFUHJOLYKWN**E**I**Z**PRTVX**G** score = 3625.6575229438167 2 minutes La bonne clef était : QBMSCDAFUHJOLY??NGI?PRTVXZ ? = E, K ou W	QBMSCDAFUH**G**OLYK**J**N**E**IWPRTVXZ score = 3619.837063624107 0.9 minute E, K et W n'apparaissent pas dans le texte clair
Salammbo score du clair = 1012.8072340784356	MZBEQSAHKNPURTVXWYGJDOCFIL score = 1012.8072340784356 0.6 minute La bonne clef était : M?BEQSAH?NPURTVX?YGJDOCFIL ? = K, W ou Z	MWBEQSAHKNPURTVXZYGJDOCFIL score = 1012.8072340784356 0.25 minute K, W et Z n'apparaissent pas dans le clair
Le cheval et l'âne score du clair = 1991.3634939904973	RBIFEJMLPSVAZXCGYNQUTOWKDH score = 1991.3634939904973 1.15 minute La bonne clef était : RBIFEJMLPSVA?XCG?NQUTO??DH ? = K, W, Y ou Z	RBIFEJMLPSVAKXCGZNQUTOYWDH score = 1991.3634939904973 0.5 minute K, W, Y et Z n'apparaissent pas dans le clair

À noter que j'ai fait les tests avec mes propres statistiques, puis avec celles de [3]. Les résultats sont quasiment identiques.

Une troisième métaheuristique : la voisine de la voisine

1. Générer une clef aléatoire. Appelons-la C_1.
2. Parmi toutes les clefs voisines de C_1, choisir celle qui a le plus petit score et la nommer C_2.
3. Calculer Δ = score(C_2) – score(C_1)
4. Si $\Delta < 0$, $C_1 \leftarrow C_2$; le cas échéant mettre à jour le meilleur score et la meilleure clef.
5. Si $\Delta \geq 0$, $C_1 \leftarrow$ voisine(voisine(C_2)) ;
6. Si le meilleur score n'a pas changé $k \leftarrow k$+1, sinon $k = 0$
7. Si $k < n\_max$, aller à 2.

L'idée de cette métaheuristique m'est venue de **George Lasry**. Dans sa thèse [1, pp. 40-41], il propose une variante de l'algorithme du recuit, qui est un peu un mixte entre le recuit simulé standard et la recherche avec tabous. Il explore en effet tout le voisinage de la clef (comme la recherche avec tabous), mais on sort d'un minimum local avec la technique du recuit.

On remarque que dans ma variante, il n'y a plus de température. Le point 5 de l'algorithme sert à sortir d'un éventuel minimum local : on prend comme clef courante « une voisine d'une voisine », générée aléatoirement, afin de voir s'il n'y a pas une meilleure clef dans les environs. Ici, par voisine, on entend la clef obtenue en déplaçant une lettre dans la clef courante. Le point 7 est le critère d'arrêt : si on a trouvé *n_max* fois la meilleure solution (ou du moins on l'espère), on s'arrête. On a ici fixé empiriquement *n_max* = 30.

Résultats

	Meilleure clef trouvée avec la troisième métaheuristique
Euler	PBEVDHWMNQTA**Z**KG**J**CSFIYLORUX score = 2729.173193215509, 0.35 minute
L'albatros	GSAKOBEDYHWMNPRTV**Z**QXU**J**CFIL score = 2270.1056606881075, 0.4 minute
1984	UBODGMZFCHJEAILNS**K**PQ**W**YRTVX Score = 17217.90039927846, 2.8 minutes
La disparition	QBMSCDAFUH**G**OLYK**J**N**E**IWPRTVXZ score = 3619.837063624107, 0.5 minute
Salammbo	MZBEQSAHKNPURTVXWYGJDOCFIL score = 1012.8072340784356, 0.15 minute
Le cheval et l'âne	RBIFEJMLPSVAZXCGWNQUTOYKDH score = 1991.3634939904973, 0.3 minute

Comme on le voit, les résultats sont comparables à ceux obtenus avec la recherche avec tabous. Cette méthode semble plus rapide, mais en fait non : dans tous les essais que l'on a fait, on aurait pu se contenter de 50 itérations pour la recherche avec tabous et donc diviser son temps de recherche par 2.

Les métaheuristiques sont clairement très performantes pour décrypter des substitutions monoalphabétiques. Pour d'autres types de chiffrement, on peut se référer à [1].

Références

[1] Lasry George, **A Methodology for the Cryptanalysis of Classical Ciphers with Search Metaheuristics**, Kassel University Press, 2018. Livre aussi disponible au format PDF : <http://www.uni-kassel.de/upress/online/OpenAccess/978-3-7376-0458-1.OpenAccess.pdf>

[2] Sinkov Abraham, **Elementary Cryptanalysis : A mathematical approach**, Mathematical Association of America, 1966. Livre aussi disponible au format PDF : <http://index-of.es/Varios-2/Elementary%20Cryptanalysis%20A%20Mathemathical%20Approach.pdf>

[3] Morel Jean-Louis (alias Rossignol), **Décrypter une substitution monoalphabétique**, 2015, <http://bribes.org/crypto/substitution_mono.html>

[4] Schmeh Klaus, **Breaking Historical Ciphers with Modern Algorithms** - Klaus Schmeh at 44CON 2017, <https://www.youtube.com/watch?v=r16zWEzfCTQ>

[5] Müller Didier, **Analyse des fréquences en français**, janvier 2020, <http://www.apprendre-en-ligne.net/crypto/stat/francais.html>

[6] Chopard Bastien, Tomassini Marco, **Métaheuristiques**, Presses universitaires de Perpignan, 2017