TD 3' : Cryptanalyse

Exercice 3 (Algorithmes d'optimisation)

Le but de cet exercice est de traduire le problème de déchiffrement d'un code par permutation de l'alphabet simple (ou par codage homophonique) en un problème d'optimisation et d'en déduire un algorithme.

1. Supposons que l'on a un vecteur $\{p_\alpha\}_{\{\alpha\in\mathcal{A}\}}$ des fréquences de chaque lettre de l'alphabet $\mathcal{A}$ en anglais. Donner la formule de la vraisemblance logarithmique qu'un texte soit en anglais d'après ses fréquences.

Pour un texte donné on cherche à présent à optimiser cette quantité quand on le décrypte avec la clef `Clef`, on la notera `Score(Clef)`. Commençons pas un algorithme de *randonnée* qui consiste à partir d'une permutation aléatoire notée `TopClef` puis de calculer le score dans un ensemble de permutations *voisines* noté `Voisins(TopClef)`. Si le score d'une de ces permutations voisines est strictement supérieur on y va et on continue, si les scores de toutes les voisines sont inférieures ou égales, on a trouvé un maximum local et on s'arrête.

2. Écrire le pseudo-code de cet algorithme `Radonnée(Texte)`.
3. Pour avoir un algorithme raisonnable, on voudrait que le voisinage soit à la fois pas trop grand, disons plus petite que 10^4 et engendre toutes les permutations. En proposer un !

Le problème de cet algorithme est qu'il a des chances de tomber dans un maximum local. Pour l'améliorer, ou peut le randomiser. C'est à dire le relancer l'algorithme sur N points de départ aléatoire et éventuellement réordonner les voisins.

4. Écrire le pseudo-code d'une telle fonction `RandonnéeAvecRedémarrages(Texte, N)`.

Un autre moyen de contrevenir à ce problème est d'ajouter un aléa qui décide de suivre parfois une direction qui diminue le score avec un petite probabilité. L'algorithme se fait pour $1 \leq k \leq N$ étapes avec un paramètre de température qui décroit et tend vers 0, $T = T_0 \cdot (1 - \frac{k}{N})$. On fixe a une clef courante aléatoire, qui est la permutation sur laquelle on se trouve dans notre exploration. On teste le score des voisins de la permutation :

- Si il est supérieur à celui de la permutation actuelle, que l'on nomme `ClefCourante`, on change la clef courante pour cette permutation.
- Sinon, on note $\Delta =$ `Score(ClefCourante) - Score(ClefVoisine)` ≥ 0 et on tire un nombre X aléatoire pour la loi uniforme sur $[0, 1]$ (avec la fonction `Uniforme()`.
  - Si $\exp(-\Delta/T) > X$ alors la clef voisine devient la clef courante.
  - Sinon, on garde la clef courante.

5. Écrire le pseudo-code `RecuitSimulé(Texte, N,` T_0`)`. Penser à conserver un maximum global qui n'est pas nécessairement celui de la clef courante.
6. Quel est l'objet qui correspond aux permutation pour les codages homophoniques ?
7. Proposer un voisinage raisonnable dans ce cas.
8. Comment lister toutes les partitions de $\{1, \ldots, N\}$ par au plus m sous-ensembles ? Écrire un pseudo-code qui fait cela.
9. Comment tirer une telle partition aléatoirement ?