Année académique : 2016-2017

Université de Mons

Rapport de projet

Optimisation linéaire

Auteurs :

MAAZOUZ Mehdi
LECOCQ Alexis

Titulaire :

GILLIS Nicolas

Assistant :

VANDAELE Arnaud

Introduction

L'objectif du projet est de déchiffrer un message chiffré envoyé d'Aline à Bob via un canal contenant du bruit creux. Le message est envoyé sous forme binaire ($x \in \{0,1\}$). Aline et Bob se sont sont mis d'accord sur le choix d'une matrice d'encodage $A \in R^{m \times p}$ où $m = 4p$.

Pour ce projet 2 choix s'offrent à nous : utiliser Matlab disponible en licence académique sur le bureau à distance de l'UMons ou utiliser son alternative libre et gratuite Octave. Malheureusement, la licence académique de Matlab ne supporte pas la fonction requise "linprog". Nous avons donc opté pour Octave qui contient la fonction "glpk".

Problème d'optimisation

Aline encode le message en utilisant la matrice A et envoie le message $y = Ax \in R^m$ sur le canal. Le canal va transmettre le message bruité $y' = Ax + n$ à Bob où le vecteur de bruit n ne contient qu'un petit nombre d'entrées non nulle. En présence de ce type de bruit, une bonne approche est de minimiser la norme 1 de l'erreur. En termes mathématiques, afin de récupérer le message d'Aline, nous devons résoudre le problème d'optimisation suivant:

$$min_{x' \in R^p} \sum_{i=1}^{m} \left| Ax'_i - y'_i \right| tel\ que\ x' \in \{0, 1\}^p$$

Problème d'optimisation linéaire

Malheureusement, le problème ci-dessus est combinatoire et difficile à résoudre. En pratique, il est courant d'utiliser la relaxation continue suivante :

$$min_{x' \in R^p} \sum_{i=1}^{m} \left| Ax'_i - y'_i \right| tel\ que\ x' \in [0,\ 1]^p$$

et d'arrondir la solution obtenue. Si le bruit n'est pas trop grand, $x' \approx x$, nous pourrons récupérer le message d'Aline.

Forme standard

Objectif : $min_{x' \in R^p} \sum_{i=1}^{m} \left| Ax'_i - y'_i \right|$

Contraintes :

- $\forall i \in \{1, ..., p\},\ x'_i \geq 0$
- $\forall i \in \{1, ..., p\},\ -x'_i \geq -1$

Sous forme standard, les variables sont toutes positives. La première contrainte est donc implicitement vérifiée sous forme standard, ce qui nous permet de la supprimer.

La forme standard ne permet pas l'utilisation de la valeur absolue. Comme vu au cours, nous pouvons remplacer une valeur absolue | x | par une nouvelle variable t_i en imposant les contraintes suivantes :

$t_i \geq x$ et $t_i \geq -x$

ce qui revient à prendre le maximum de x et -x, c'est-à-dire la valeur absolue.

Ce qui nous donne le problème suivant :

Objectif : $min_{x' \in R^p} \sum_{i=1}^{m} t_i$

Contraintes :

- $\forall i \in \{1, ..., p\}, -x'_i \geq -1$
- $\forall i \in \{1, ..., m\}, t_i - Ax' \geq -y'$
- $\forall i \in \{1, ..., m\}, t_i + Ax' \geq y'$

La forme standard ne permet pas d'utiliser les inégalités. Comme vu au cours, nous allons ajouter une variable d'écart par inégalité afin de la transformer en égalité. Ce qui nous donne le problème suivant :

Objectif : $min_{x' \in R^p} \sum_{i=1}^{m} t_i$

Contraintes :

- $\forall i \in \{1, ..., p\}, -x'_i - q_i = -1$
- $\forall i \in \{1, ..., m\}, t_i - Ax' - r_i = -y'$
- $\forall i \in \{1, ..., m\}, t_i + Ax' - s_i = y'$

La forme standard ne permet pas d'utiliser des variables réelle (seulement des variables positives). Comme vu au cours, nous allons transformer les variables t_i par deux variables positives tp_i - tm_i. Ce qui nous donne le problème suivant :

Objectif : $min_{x' \in R^p} \sum_{i=1}^{m} tp_i - tm_i$

Contraintes :

- $\forall i \in \{1, ..., p\}, -x'_i - q_i = -1$
- $\forall i \in \{1, ..., m\}, tp_i - tm_i - Ax' - r_i = -y'$
- $\forall i \in \{1, ..., m\}, tp_i - tm_i + Ax' - s_i = y'$

Résolution du problème avec Octave

Nous avons entré les contraintes trouvées ci-dessus dans le logiciel Octave afin de déchiffrer le message d'Aline. Voir code en annexe 1. Le message obtenu est "Aline vous f<E9>licite!". Nous devinons le "é" de "félicite" mais Octave ne semble pas gérer correctement l'UTF-8 (l'édition des fichiers ayant également posé problème).

Sommet du polyèdre correspondant ?

Comme la fonction glpk utilise l'algorithme du simplex pour résoudre les problèmes d'optimisation linéaire, la solution optimale est évidemment un sommet du polyèdre.

Cependant, notre algorithme n'utilise pas directement la solution optimale du problème relaxé mais la solution arrondie. La solution trouvée par notre algorithme n'est donc pas forcément solution du problème initial. En effet, il est possible que certaines contraintes ne soient plus respectées une fois la solution arrondie, dans le cas où le sommet en nombres entiers le plus proche se trouve en dehors du polyèdre des contraintes. D'où, la solution donnée par notre algorithme n'est pas forcément un sommet du polyèdre correspondant.

Dans le cas où l'arrondi de la solution égale la solution du problème relaxé (solution en nombres entiers) alors la solution est également solution du problème initial et dans ce cas, la solution se trouve sur un sommet.

Bruit maximum

Nous avons généré le message "Coucou petite perruche !". Ce message est déchiffrable dans la plupart des cas entre 0 et 40% de bruit. A partir de 40%, il devient rare de pouvoir retrouver le message original. Il n'y a pas vraiment de niveau de bruit maximum étant donné qu'à 40%, il est parfois possible de déchiffrer le message grâce à notre algorithme, parfois pas. Si la solution du problème relaxé n'est pas en nombres entiers et qu'on n'a pas une idée de la forme des données de départ, il est impossible de déterminer si les données de départ sont correctes (si la solution arrondie est optimale). Il est clair que plus on approche des 40%, moins nous avons de chances de déchiffrer le message original avec notre algorithme alors que plus on approche du 0% de bruit, plus nous avons de chances de déchiffrer le message.
Dans le cas où la solution au problème relaxé est égale à la solution arrondie, alors nous sommes certains d'avoir la solution optimale du problème initial et de déchiffrer le bon message original.
Dans le cas où la solution au problème relaxé est différente de la solution arrondie, alors nous pouvons utiliser l'arrondi de la solution mais nous n'avons aucune certitude sur l'optimalité de cette solution (aucune certitude de déchiffrer le message original).

Variables binaires

Nous avons ajouté un nouvel argument “integer” à votrealgorithme afin de définir si la solution obtenue doit être optimale en nombres entiers. Voir code en annexe 1. Lorsque cette valeur est vraie, nous demandons à la fonction glpk de nous retourner une valeur en nombres entiers et il n’est donc plus utile d’arrondir la solution.

Ce problème peut être rapide lorsque l’algorithme du simplex retourne une solution en nombres entiers. Mais lorsque la solution du simplex retourne une solution en nombres réels, nous ne pouvons pas utiliser cet algorithme pour résoudre le problème de départ. Nous sommes alors obligés d’utiliser un algorithme “Branch and Bound” qui met alors beaucoup plus de temps à résoudre le problème. L’algorithme “Branch and Bound” permet en effet de déchiffrer un message avec plus de bruit mais prendra beaucoup plus de temps.

Conclusion

Ce projet nous a permis de manipuler et différencier des problèmes d’optimisation linéaire et d’optimisation en nombres entiers. Lorsque l’algorithme du simplex retourne une solution entière comme solution d’un problème en nombres entiers relaxé en problème d’optimisation linéaire, celui-ci permet d’obtenir la solution optimale très rapidement. Si l’algorithme du simplex retourne une solution en nombres réels, nous devons alors utiliser un algorithme de type “Branch and Bound” qui met beaucoup plus de temps à trouver la solution optimale.

Dans le cas d’un transfert de message chiffré sur un canal de communication rapide, on préférera surement renvoyer le message en cas de bruit important alors que sur un canal de communication lent, on préférera dans certains cas passer plus de temps à déchiffrer le message.