UMONS

Université de Mons

UMONS

Optimisation linéaire

Envoi d'un message crypté sur un canal avec du bruit creux

Projet d'optimisation linéaire

Élèves :
Nicolas Gatta
Emilien Vancauwenberghe

Enseignant :
Nicolas Gillis

6 novembre 2023

Table des matières

1 Introduction

L'objectif de ce projet est l'étude du problème de minimisation $\min_{x' \in Rp} \|Ax' - y'\|_1$ tel que $x \in \{0,1\}^p$ qui permettra de débruité le message d'Alice ainsi que de créer nos propres messages codés grâce à l'utilisation du language octave et du kit GLPK

2 Question 1 : Modélisation du problème

Le problème de base comme annoté dans l'introduction est considérée comme difficile à résoudre. En pratique, il est courant d'utiliser la relaxation linéaire pour permettre d'obtenir un problème plus facile à résoudre et d'arrondir par la suite la solution obtenue si le bruit n'est pas trop important. La solution devient alors :

$$\min_{x' \in Rp} \|Ax' - y'\|_1 \text{ tel que } 0 \leq x' \leq 1 \tag{1}$$

Par définition de la norme 1, le problème (1) est équivalent à :

$$\min_{x' \in Rp} \sum_{i=1}^{m} |(Ax' - y')_i| \text{ tel que } 0 \leq x' \leq 1 \tag{2}$$

Cependant, nous cherchons à avoir le problème sous la forme d'un problème d'optimisation linéaire. Nous savons qu'il est possible de linéariser un problème comportant des expressions avec des valeurs absolue en posant $l_i = |(Ax' - y')_i|$ et en ajoutant deux nouvelles contraintes $l_i \geq (Ax' - y')_i$ et $l_i \geq -(Ax' - y')_i$. On obtient le problème optimisation suivant :

$$\begin{aligned} \min_{t \in R^m + x' \in Rp} \quad & \sum_{i=1}^{m} l_i \\ \text{t.q.} \quad & \forall i \in \{1,...,m\}, \quad l_i \geq (Ax' - y')_i \\ & \forall i \in \{1,...,m\}, \quad l_i \geq -(Ax' - y')_i \\ & -x' \geq -1 \\ & x' \geq 0 \end{aligned} \tag{3}$$

On met le problème (3) sous forme géométrique, ce qui nous donnes :

$$\begin{aligned} \min_{t \in R^m + x' \in Rp} \quad & \sum_{i=1}^{m} l_i \\ \text{t.q.} \quad & \forall i \in \{1,...,m\}, \quad l_i - Ax'_i \geq -y'_i \\ & \forall i \in \{1,...,m\}, \quad l_i + Ax'_i \geq +y'_i \\ & -x' \geq -1 \\ & x' \geq 0 \end{aligned} \tag{4}$$

3 Question 2 : Forme standard

Pour passer de la forme géométrique (4) à la forme standard, il faut que le problème soit mis sous la forme suivantes :

$$\begin{gathered}\text{Min } c^T x \text{ où } x \in R^n \\ \text{Tel que } Ax = b \\ x \geq 0.\end{gathered}$$

La problème (4) devient alors :

$$\begin{aligned}\min_{t \in R^m + x' \in Rp} \quad & \sum_{i=1}^{m} l_i^+ - l_i^- & & \\ \text{t.q.} \quad & \forall i \in \{1, ..., m\}, & l_i^+ - l_i^- - Ax_i' - s_{1i} &= -y_i' \\ & \forall i \in \{1, ..., m\}, & l_i^+ - l_i^- + Ax_i' - s_{2i} &= -y_i' \\ & & -x' - s_3 &= -1 \\ & & x', s_{1i}, s_{2i}, s_3, l_i^+, l_i^- &\geq 0\end{aligned} \tag{5}$$

4 Question 3 : Déchiffrer le message

Pour décrypter le message envoyé par Alice, nous avons formulé le problème (5) dans Octave, comme indiqué dans le code en annexe. En utilisant la fonction "glpk", nous avons obtenu la phrase "Alice vous flicite !". Nous pouvons facilement déduire que les caractères accentués peuvent causer des problèmes, et donc que le message qu'Alice a envoyé à Bob est en réalité "Alice vous félicite !"

5 Question 4 : Le polyèdre

Glpk utilise l'algorithme bien connu du simplexe pour résoudre des problèmes linéaires. Cela nous assure que la solution optimale que nous obtenons est l'un des coins du polyèdre associé au problème.

Comme décrit dans notre approche de modélisation, nous arrondissons la solution optimale du problème relaxé (1). Cependant, cela signifie que la solution finale que nous obtenons n'est pas nécessairement la solution du problème d'origine.

En effet, si le coin le plus proche avec des valeurs entières se trouve à l'extérieur du polyèdre défini par les contraintes, certaines de ces contraintes pourraient ne plus être respectées, ce qui signifie que la solution obtenue pourrait ne pas être un coin du polyèdre.

Cependant, si la solution arrondie est identique à la solution du problème relaxé, cela implique que la solution relaxée est elle-même une solution entière, alors la solution résultante est bien l'un des coins du polyèdre.

6 Question 5 : Génération message

Nous observons que lorsque nous altérons 41.50% des composantes de notre message, son déchiffrement devient de plus en plus aléatoire jusqu'à devenir illisible après 42.50%. Malheureusement, n'ayant pas de compétences suffisantes dans le domaine, ils nous est très difficile de déterminer si cela est surprenant ou non. Cependant, on peut effectivement remarquer que la norme d'erreur est assez importante quand le déchiffrement du message commence à devenir hasardeux.

7 Question 6 : Variable binaire

Malheureusement, après plusieurs tentatives, nous n'avons pas réussi à utiliser les variables binaires pour résoudre le problème. Voici les pistes que nous avons explorées :

— Mettre le vartype en "I" au lieu de "C"
— Essayer de modifier la upper et lower bond

8 Annexe

8.1 GenerezEtDechiffrezVosMessages

```
% Projet d'optimisation non lineaire
%
% Envoi d'un message crypte atravers un canal avec du bruit qaussien...
%
% Encodez vous-meme et dechiffrez un message:

clear all; clc;

% Message a envoyer
my_mess = "Ca c'est ma fusee";
fprintf('The encoded message is: %s \n', my_mess)

% Message sous forme binaire
[x,d]  = encoding_bin(my_mess);

% Longeur du message
n = length(x);

% Longeur du message qui va etre envoye
m = 4*n;

% Matrice d'encodage: on prend une matrice aleatoirement generee
A = randn(m,n);

% Message que l'on desire envoyer
y = A*x;

% Bruit ajoute par le canal de transmition
% = normale N(0,sigma) pour un % des entrees de y
% Define initial percenterror
initial_percenterror = 0.005;
percenterror = 0.38;
max_iterations = 10; % Number of iterations

for iteration = 1:max_iterations
    % Update percenterror for the current iteration
    percenterror += initial_percenterror;

    % Generate noisy signal
    yprime = noisychannel(y, percenterror);

    % Solve the optimization problem
    xprime = votrealgorithme(A, yprime);
```

```
    % Calculate the error
    error = norm(x - xprime);

    % Display the results
    fprintf('Iteration %d:\n', iteration);
    fprintf('Percent Error: %.2f%%\n', percenterror * 100);
    fprintf('Error Norm: %.4f\n', error);
    fprintf('The recovered message is: %s\n', decoding_bin(xprime, d));
    fprintf('\n');
end
```

8.2 votrealgorithme

```
% Votre algorithme pour resoudre
%
%   min_{0 <= xprime <= 1} ||A*xprime - yprime||_1

function xprime = votrealgorithme(A,yprime)

% !!! Ecrivez votre code ici !!!

yprime = yprime'; %On prend la transpose de yprime

%On cree le vecteur objectif
cprime = zeros(size(A,2),1); %On remplit de 0 pour chaque coefficient x

c2prime = ones(size(A,1),1); %On remplit de 1 pour chaque coefficient t

c = cat(1,c2prime, cprime);

% Cr ation du vecteur de contraintes
Aprime = zeros(2*size(A, 1), 2*size(A, 2));

% Remplissage des contraintes en utilisant A
lineCount = 1;
for x = 1:size(A, 1)
    for y = 1:size(A, 2)
        Aprime(lineCount, size(A, 1) + y) = A(x, y);
        Aprime(lineCount + 1, size(A, 1) + y) = -A(x, y);
    end
    lineCount = lineCount + 2;
end

% Remplissage des contraintes pour les variables d' cart
count = 1;
for i = 2:2:(2*size(A, 1))
    Aprime(i-1:i, count) = -1;
    count = count + 1;
end


% Cr ation du vecteur B
B = zeros(1, 2*size(yprime, 2));

rowCount = 1;
for i = 1:size(yprime, 2)
    B(1, rowCount) = yprime(1, i);
    B(1, rowCount+1) = -yprime(1, i);
```

```
    rowCount = rowCount + 2;
end

% Creation du vecteur de bornes inf rieurs
lb = zeros(size(A,2)+size(A,1),1);

% Cr ation du vecteur de bornes sup rieures
ubPrime = Inf(1, size(A, 1));
ub2Prime = ones(1, size(A, 2));

ub = [ubPrime, ub2Prime];

% Cr ation du tableau pour sp cifier le sens des in galit s
ctype = repmat('U', 1, 2 * size(A, 1));

%On  c r e  un tableau pour  s p cifier  le type des variables
%Ici on n'impose pas de variables binaires
vartype = [];
vartype = repmat('C', 1, size(A, 1) + size(A, 2));

x = glpk(c, Aprime, B, lb, ub, ctype, vartype, 1);

% Copy elements from x to xprime within a specified range
xprime(size(A,1) : size(A,1) + size(A,2)) = x(size(A,1) : size(A,1) + size(

xprime = round(xprime);
```

8.3 DecodeLeMessagedAlice

```
% Projet d'optimisation non lineaire
%
% Decodez le message d'Alice...

clear all; clc;

% load la matrice d'encodage et le message d'Alice transmis par le canal:
load messagedAlice

% Utilisez votre algorithme pour  r soudre :
%
%   min_{0 <= xprime <= 1} ||A*xprime - yprime||_2^2
%
% Rem. La solution xprime devrait  tre  un vecteur de taille 98x1
xprime = votrealgorithme(A,yprime);

% Affichez le resultat:
fprintf('The recovered message is: %s \n', decoding_bin(xprime,d));
```

8.4 decoding-bin

```
% Decode a binary vector into a sentence

function [mess,y] = decoding_bin(x,d)

x = max(x,0);
x = min(x,1);
x = round(x);
k = 1;
for i = 1 :  length(x)/d
  for j = 1 : d
    y(i,j) = dec2bin(x(k)); k = k + 1;
  end
end
mess = char(bin2dec(y))';
```

8.5 encoding-bin

```
% Encode a sentence into a binary vector

function [x,d] = encoding_bin(mess)

xi = dec2bin(mess+0);
[m,d] = size(xi);
x = [];
for i = 1 : m
    x = [x; double(xi(i,:)')-48];
end
```

8.6 noisychannel

```
% Perturbe percenterror% des entr es de y al atoirement

function y_n = noisychannel(y,percenterror)

m = length(y);
K = floor(m*percenterror);
I = randperm(m,K);

y_n = y;
y_n(I) = rand(K,1)*mean(y);
```