

|  |
| --- |
| Numerical in Python : Optimisation combinatoire  2018 |
|  |
| 17 octobre  ESIEA  Créé par : LAURENCE Camille, MAITROT Maxime, MARI Lucas |

Table des matières

[1. Modélisation linéaire 3](#_Toc85400935)

[2. GLPK 3](#_Toc85400936)

[3. Algorithme glouton 6](#_Toc85400937)

# Modélisation linéaire

Nous avons :

-N nombre d’invités potentiels

-M Nombre de lien d’amitiés

Nous cherchons à inviter le plus d’invités à la condition que chacun se connaissent.

Nous avons donc affaire à une maximisation :

Soit X une variable binaire

On maximise Z =

Sous condition que :

Si

# GLPK

Tout d’abord, il fallait réalisé un algorithme permettant de convertir les fichiers d’instances donné en fichier .lp, traitable par GLPK.

Pour cela, il fallait tout d’abord comprendre la structure des instances données, décrite le sujet.

*Rappel :*

-La première ligne est composée de N nombre total d’invités et de M nombre total de liens d’amitiés

-Les N lignes suivantes sont composées de i et ci

-Les M lignes suivants, sont composées de i et j, indiquant que l’invité i connais l’invité j, et vice versa.

Une fois la structure connue, il nous a fallut isoler chaque élément et extraire les conditions pour respecter le format d’un fichier .lp à savoir :

-L’opération à effectuer (minimize ou maximize)

-L’équation

-La « section » Subject to, énumérant les conditions, et enfin les variables Xi qui seront utilisées.

Nous avons donc procédé au développement de l’algorithme.

Une image contenant texte

Description générée automatiquementEn première étape, il fallait réussir à récupérer les informations selon le format des instances données.

Ensuite, nous avons configuré les variables N et M

Une image contenant texte

Description générée automatiquement

Puis, pour chaque n, nous créons un tableau contenant : l’indice n, une liste contenant toutes les connaissances

Une image contenant texte

Description générée automatiquement

Nous procédons ensuite à l’écriture de z (grâce au N+1 lignes de l’instance)

Une image contenant texte

Description générée automatiquement

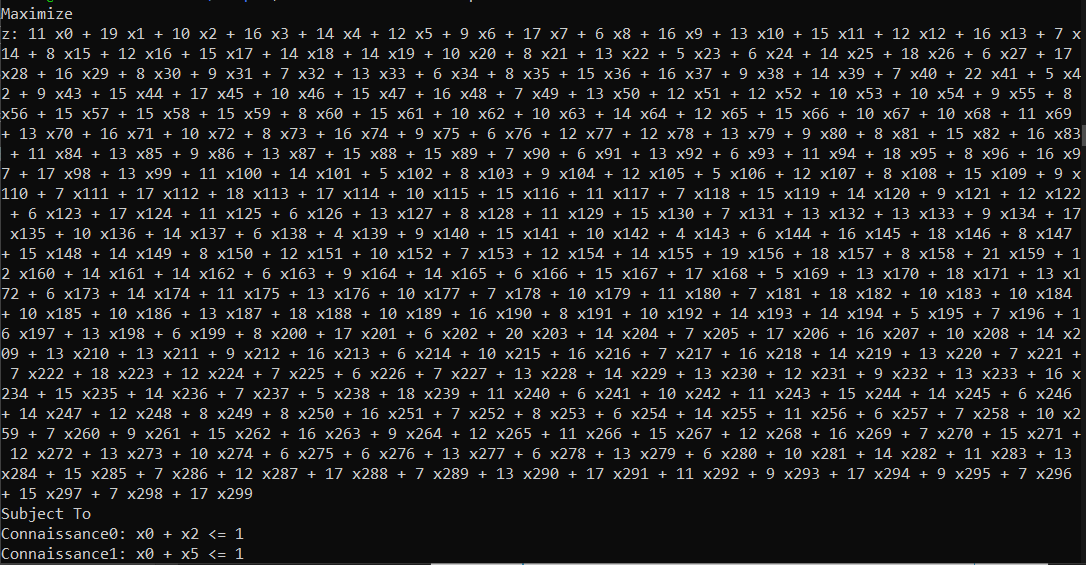
Evidemment, nous avons au préalablement créé le fichier « File2 », étant le fichier .lp

Une fois écrit, sous énumérons la condition de notre modèle linéaire pour chaque n Durant cette condition, nous prenons soit ne ne pas répéter des informations inutiles (exemple : si Vi connait Vj, on ne va pas réécrire que Vj connait Vi – ligne 49)

Une image contenant texte

Description générée automatiquement

Ainsi, voici à quoi ressemble notre fichier .lp :



Une image contenant texte

Description générée automatiquement

|  |  |  |  |
| --- | --- | --- | --- |
|  | Solution optimale | Temps | Mémoire |
| Instance1.txt | 75 | 5879.9s | 556Mb |
| Instance2.txt | 91 | 10007.0s | 717Mb |
| Instance3.txt | 84 | 10804.7s | 882.6Mb |

Pour des raisons de temps, nous n’avons pas pu faire tourner le programme GLPK sur toutes les instances. En effet, le temps de traitement étant très long, nous avons choisi ne faire que les 3 premières instances. Nous n’avons aucune piste de réflexion quant à ces temps, qui sont 5 fois plus élevé que dans le sujet.

# Algorithme glouton

Pour pallier au problème du temps de traitement de GLPK, nous avons programmé un algorithme glouton. Le but d’un algorithme glouton est de gagner significativement en temps de traitement, au dépend de la précision de la réponse.

Les algorithmes gloutons dépendent eux-même d’heuristiques, c’est-à-dire, de condition qui incrémenteront ou non le score si elles sont respectées.