Projet d’optimisation combinatoire



Table des matières

[Modélisation linéaire 3](#_Toc85207694)

[Description du modèle 3](#_Toc85207695)

[Interprétation des résultats du solveur 3](#_Toc85207696)

[Algorithme Glouton 3](#_Toc85207697)

[Algorithme Génétique 3](#_Toc85207698)

[Algorithme Métaheuristique 3](#_Toc85207699)

# Modélisation linéaire

## Description du modèle

Notre but est ne maximiser le score total qu’aura le groupe.

Equation :

z : score du groupe à maximiser.

i : représente un convive, i ∈ [0,1,2,...,n]

pi : représente si un convive est présent ou non, pi ∈ [0,1]

ci : représente l’intérêt d’un convive, ci ∈ N

Les contraintes : Pour chaque convive i et j, si i et j ne se connaissent pas, on ne veut pas que i et j soient tous les deux présents. On peut représenter mathématiquement cette contrainte par : pi + pj <= 1

Comme un convive i ne connaitra généralement pas plusieurs autres convives, on peut rassembler les close : pi + pj <= 1 ; pi + pk <=1 ; pi + pl <= 1 … en :

X\*pi + pj + pk + pl … <= X

X representant le nombre de convives que i ne connait pas.

Chaque convive est une contrainte car elle restreint la possibilité de faire un groupe. Pour un problème avec 300 convives par exemple, nous aurons donc 300 contraintes dans son modèle linéaire.

## Fonctionnement de l’algorithme de conversion

Le but de l’algorithme de conversion est de convertir un fichier .txt donnés selon le format imposés par le sujet en un fichier représentant le modèle linéaire définit précédemment pour le solveur GLPK.

Tout d’abord le programme, ouvre le fichier texte qui contient la liste des convives et leurs relations, il met dans une liste le nombre de convives ainsi que le nombre de relations (1ère ligne du programme).

La seconde étape du programme écrit la ligne de maximisation du fichier glpk, il en profite aussi pour construire une liste contenant la liste des valeurs binaires associées aux convives ainsi que la liste des convives qui sont des objets.

Ensuite, pour chaque convive, le programme va initialiser la liste des convives inconnus à chaque convive. Par défaut, pour un convive i, cette liste est initialisée avec tous les convives exceptées i. Nous allons dans la suite du programme retirer les connaissances des convives dans cette liste.

Le programme lit ensuite les lignes représentant deux convives se connaissant. Quand il lit une ligne, il va chercher les deux convives correspondant dans la liste des convives et supprime sa connaissance respective de leurs listes des convives qu’ils ne connaissent pas.

Une fois que toutes les lignes sont lus, le programme écrit les contraintes du fichier glpk. Pour chaque convive de la liste des convives, ce dernier écrit une ligne de contrainte basé sur les convives qu’il ne connait pas (expliquer dans le modèle linéaire).

## Interprétation des résultats du solveur

Une image contenant texte

Description générée automatiquement

Grace au solver GLPK nous avons pu avoir la résolution de notre problème linéaire. Nous avons aussi une idée du temps qu’il faudra mètre pour résoudre les algorithmes.

Cela nous a donc donnée un fichier qui contient la réponse :

Une image contenant texte

Description générée automatiquement

Voici donc le tableau des solutions trouver par le solver GLPK avec le temps d’exécution et la mémoire utilisé.

|  |  |  |  |
| --- | --- | --- | --- |
| Instance | Solution optimale | Temps (s) | Mémoire (Mb) |
| instance1.txt | 73 | 275.7 | 31.7 |
| instance2.txt | 91 | 1247.6 | 51.9 |
| instance3.txt | 84 | 787.6 | 65.1 |
| instance4.txt | 83 | 569.3 | 51.8 |
| instance5.txt | 81 | 499.9 | 48.8 |
| instance6.txt | 80 | 1068.9 | 70.6 |
| instance7.txt | 81 | 475.2 | 46.8 |
| instance8.txt | 85 | 767.4 | 60.2 |
| instance9.txt | 77 | 613.6 | 56.7 |
| instance10.txt | 91 | 516.2 | 45.6 |

# Algorithme Glouton

# Pour l’algorithme glouton nous somme partie sur un heuristique qui est le nombre de connaissance (au dernière nouvelle après ca a peut ètres changer).

Explication de l’algorithme :

Nous regardons la liste une fois, et nous prenons la personne avec le plus de connaissance. Ensuite on supprime cette personne de la liste, et toutes les personnes qu’il ne connais pas. On regarde la liste et on prend la personne qui a le plus de connaissance, on supprime de la liste lui et toutes les personnes qu’il ne connais pas. On recommence jusqu’à il n’y est plus de convive dans la liste.

Donc finalement on obtient un groupe où tout le monde se connais.

# 

# Algorithme Génétique

# Algorithme Métaheuristique