Projet d’optimisation combinatoire



Table des matières

[Modélisation linéaire 3](#_Toc85294581)

[Description du modèle 3](#_Toc85294582)

[Fonctionnement de l’algorithme de conversion 4](#_Toc85294583)

[Interprétation des résultats du solveur 4](#_Toc85294584)

[Algorithme Glouton 7](#_Toc85294585)

[Algorithme Génétique 7](#_Toc85294586)

[Algorithme Métaheuristique 8](#_Toc85294587)

# Modélisation linéaire

## Description du modèle

Notre but est de maximiser le score total qu’aura le groupe de convive invité à la soirée.

Équation:

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, pour la plupart du temps, pas beaucoup d’autres potentiels invités, on peut rassembler les closes : pi + pj <= 1 ; pi + pk <=1 ; pi + pl <= 1 … en :

X\*pi + pj + pk + pl … <= X

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

Chaque convive est une contrainte car il restreint la possibilité de faire un groupe. Pour un problème avec 300 convives, dans notre cas, nous aurons donc 300 contraintes présentes dans ce 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 imposé 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. Puis il met dans une liste le nombre de convives ainsi que le nombre de relations (1ère ligne du programme).

La seconde étape du programme est d’écrire la ligne de maximisation dans le 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 de ces derniers inconnus à chaque convive. Par défaut, pour un invité possible 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 individus se connaissant. Lorsqu’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 invités possibles qu’ils ne connaissent pas.

Une fois que toutes les lignes ont été lus, le programme écrit les contraintes dans le fichier GLPK. Pour chaque individu de la liste des convives, ce dernier écrit une ligne de contrainte basé sur les invités qu’il ne connait pas (expliqué dans le modèle linéaire).

## Interprétation des résultats du solveur

Une image contenant texte

Description générée automatiquement

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

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

Une image contenant texte

Description générée automatiquement

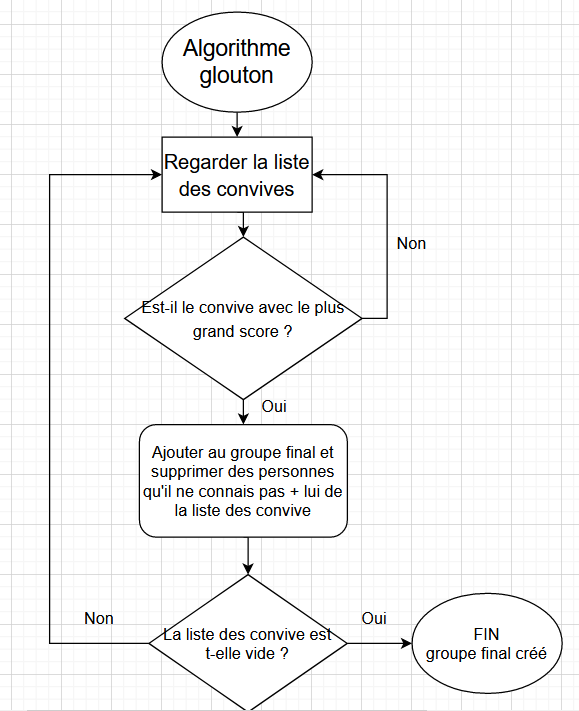
Voici, ci-dessous, le tableau des solutions trouver par le solveur GLPK avec le temps d’exécution et la mémoire utilisée :

|  |  |  |  |
| --- | --- | --- | --- |
| 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 sommes partis sur un heuristique qui est le nombre de connaissances commun maximum pouvant être présente à cette soirée. Dans l’étude de cet algorithme nous ne nous intéresserons pas à maximiser le score.

Explication de l’algorithme :



Nous regardons la liste une première 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 ce dernier ainsi que toutes les personnes qu’il ne connais pas. On réitère ces actions jusqu’à ce qu’il n’y ait plus de convive dans la liste.

Finalement on obtient un groupe où tout le monde se connait.

# Algorithme Génétique

# Algorithme Métaheuristique