Université de Montpellier

Rapport du projet
Conception et implantation d'un système d'aide à la décision

El Houiti Chakib
Kezzoul Massili

30 novembre 2021

Introduction

Objectif du projet

L'objectif principal du projet est de réaliser une analyse critique de l'algorithme du mariage stable. Dans un premier temps l'objectif est d'implémenter l'algorithme de Gale et Shapley (mariage stable), pour l'affectation des étudiants aux institus, ensuite, de proposer une méthode de satisfaction pour les deux côtés et de tester cet algorithme sur plusieurs jeux de données. Finalement, c'est de proposer une représentation compacte des préférences.

Environnement de développement

Le projet a été développé sur notre propre environnement de travail. On a utilisé le langage Python, pour l'implémentation des différentes fonctionnalités, en utilisant plusieurs bibliothèques propres à Python.

Structure du projet

Pour une meilleure compréhension de l'environnement du projet, voici ci-dessous différentes informations sur les différents fichiers et répertoires du projet :

src/ Répertoire contenant les fichiers sources du projet.

- **generate.py** Fichier pour la génération automatique de jeux de données.
- **algorithme.py** Fichier implémentant les différents algorithmes, notamment celui de Gale et Shapley.
- **graphviz.py** Fichier de visualisation.
- **main.py** Fichier contenant le programme principale du projet.

data/ Répertoire contenant les différents jeux de données de différentes tailles.

output/ Répertoire contenant toutes les sorties du programme.

README.md Fichier expliquant la manière d'utiliser le programme (initialisation, compilation et exécution). Referez-vous à la section *Utilisation* de ce dernier pour plus d'informations.

Makefile Fichier qui spécifient les commandes de compilation, initialisation et autres.

1 Modèlisation et implémentation

1.1 Programme de génération de préférences aléatoires

La première partie du travail est la génération de préférences aléatoires pour les étudiants ainsi que les institutions. L'objectif ici est de générer un fichier contenant ces préférences qui pourra ensuite être interpréter par un programme. Nous avons choisis de reprèsenter les préférences par un fichier *JSON*[1]. En effet, ce format est très expressif et facile à manipuler.

1. JavaScript Object Notation est un format de données textuelles dérivé de la notation des objets du langage JavaScript. il permet de représenter de l'information structurée comme le permet XML par exemple.

```
{
  "students": {
      "E1": ["I3", "I1", "I2"],
      "E2": ["I1", "I2", "I3"],
      "E3": ["I3", "I2", "I1"]
  },
  "institutions": {
      "I1": {
          "capacities": 1,
          "preferences": ["E1", "E2", "E3"]
      },
      "I2": {
          "capacities": 1,
          "preferences": ["E1", "E3", "E2"]
      },
      "I3": {
          "capacities": 1,
          "preferences": ["E3", "E2", "E1"]
      }
  }
}
```

Listing 1 – "Exemple d'un fichier de préférences"

Pour cela, nous avons définis une fonction qui, en lui donnant en paramètre : N le nombre d'étudiants et K le nombre d'institutions, génére le fichier *JSON* ci-dessus. Ce fichier représente les préférences des étudiants et celles de institutions. On attribut à chaque étudiant une liste de K institutions générer aléatoirement et classées par ordre de préférence. La même chose est faite pour chaque institution. Mais cette fois, pour chaque institutions, on lui attribut aléatoirement en plus une capacité d'acceuil. La capacité d'acceuil est génére de sorte que la somme de toutes les capacités soit égal à N le nombre d'étudiants.

Ceci à été implémenter dans le fichier *generate.py* indépendament du reste des programmes. Les jeux de données ainsi générer sont ensuite mis dans le répertoire *data/*.

1.2 Implémentation de l'algorithme du mariage stable

La seconde partie du projet est l'implèmentation d'un algorithme de marage stable. Pour cela nous avons adapté deux implémentations de l'algorithme de *Gale & Shapley* à nos jeux de données. L'une donnant la priorité aux étudiants et la seconde aux institutions [2]. Dans ces implèmentation, nous supposant qu'il y assez de place dans les institutions pour tout les étudiants. Nous supposant aussi que la taille de la liste des préférences des étudiants (Resp. les institutions) est égale à K le nombre d'institutions (Resp. N le nombre d'étudiants). Cette condition est nécessaire afin d'assurer qu'un mariage stable existe (Voir le Théorème de Hall).

Ensuite, nous obtenons un programme principal (*main.py*), utilisable en ligne de commande, qui affiche (voir Figure 1) le temps d'exécution des deux algorithmes ainsi que la satisfaction des étudiants et des institutions, point qu'on vas aborder un peu plus loin dans ce rapport.

2. Les deux fonctions ont été définis dans le fichier *algorithme.py*

```
massy@pop-os:~/Bureau/master2/decision-aid/mtq-assignment-algorithms$ python src/main.py data/medium_preferences.json output/
Temps d'execution Gale Shapley - priorité aux étudiants: 0.002 secondes.
Temps d'execution Gale Shapley - priorité aux instituts: 0.002 secondes.
Affectation écrite dans 'output/medium_preferences_student.json'
Affectation écrite dans 'output/medium_preferences_institut.json'

Priorité aux étudiants :
        Satisfaction 'linear' des étudiants : 0.92
        Satisfaction 'poly' des étudiants : 0.86
        Satisfaction 'inverse' des étudiants : 0.74
        Satisfaction des instituts : 0.66

Priorité aux institutions :
        Satisfaction 'linear' des étudiants : 0.74
        Satisfaction 'poly' des étudiants : 0.63
        Satisfaction 'inverse' des étudiants : 0.49
        Satisfaction des instituts : 0.9
```

FIGURE 1 – Affichage produit par le programme principal

De plus, ce programme exporte dans deux fichiers sous le format *JSON*[3] les résultats des affectations (priorité aux étudiants et priorité aux institutions).

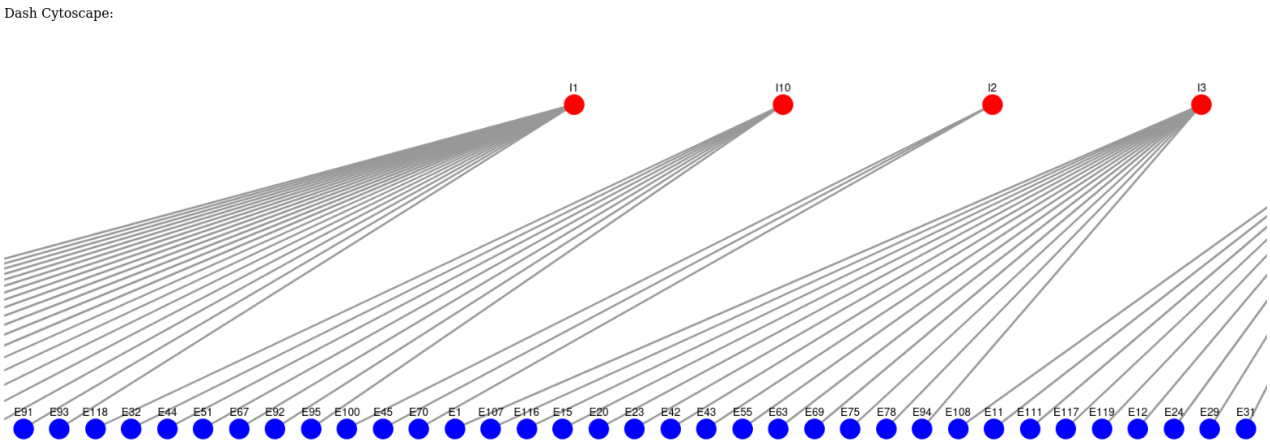
```
0  {
1    "I1": ["E1", "E2", "E11"],
2    "I2": ["E5", "E12", "E17", "E18", "E19"],
3    "I3": ["E6", "E9", "E16", "E10"],
4    "I4": ["E4", "E7", "E8", "E13", "E20"],
5    "I5": ["E3", "E14", "E15"]
6  }
```

Listing 2 – "Fichier des affectations priorité au étudiants"

1.3 Interface de visualisation

Pour pouvoir mieux visualiser les affectations des étudiants aux institution, on a pensé à une structure de graphes. Chaque institut et chaque étudiant sont représentés par un noeud étiqueté par leurs noms. Chaque affectation est représenté par une arête qui lie un étudiant à son institution.

FIGURE 2 – Visualisation en graphe



On obtient cette visualisation en utilisant le programme *graphviz.py* à qui en passe en argument un fichier *JSON* produits precedement. Ce programme utilise la bibliothèque *Dash*, qui permet de projeter des graphes interactifs via un petit serveur web en local.

3. Le format *CSV* est aussi possible mais pour la visualisation en graphe c'est le format *JSON* qui est utilisé

1.4 Méthodes de satisfaction

La satisfaction de chaque côté est nécéssaire, pour évaluer nos algorithmes et les classés. Les problèmes de satisfaction apparaîssent toujours dans les systèmes d'aid à la décison ou plus précisement dans les systèmes d'affectations. Dans notre projet, on s'est concentré sur la satisfaction des étudiants, qui est un problème très fréquent dans la vie réelle.

Satisfaction des étudiants

Il existe plusieurs manières pour calculer la satisfaction des étudiants, on a choisi des méthodes qui sont significatives. Ces méthodes ont de même des points forts et des points faibles. Toutes les méthodes sont faites, d'une façon à donner une note à chaque étudiant, selon son affectation par rapport à sa liste de préférences. Une moyenne de tout les étudiant permet de mesurer la satisfaction globale de tout les étudiants.

Linéaire Cette méthode consiste à donner une note de sat

Satisfaction des instituts

2 Extension du système

2.1 Random assignement

2.2 Li mkawda 3lihoum

3 Conclusion

3.1 Utilisation du programme

3.2 Perspectives

3.2.1 Pré-matching-Problem and Theroem de HALL