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"

Cette Partie est faite pour la génération aléatoires des préférences des étudiants et des instituts. Pour cela, on a réaliser une méthode prenant **n** étudiants et **k** instituts, qui permet de générer **n** étudiant avec **k** préférences d'instituts pour chaque étudiant, qui seront stockés dans un fichier *json*, pour avoir la possibilité de générer plusieurs jeux de données. Pour les institus, on génère **k** instituts avec **n** préférences pour chaque institut et une capacité **q**. la capacité de chaque instituts est aussi générer aléatoirement, en respectant,la propriété que la somme des capacités de tout les instituts soit égale au nombre **n** d'étudiants.

1.2 Implémentation de l'algorithme du mariage stable

1.3 Interface de visualisation

Pour pouvoir visualiser, les affectations des étudiants aux instituts, on a pensé à une structure de graphes, avec des noeuds et des arêtes, pour mieux visualiser tout cela. Chaque institut et chaque étudiants sont représenté par un noeud, étiqueté par le numéro de ces derniers (ex : E1 ou I1). Chaque affectation est représenté par une arête, donc une arête entre un étudiant et son affectation d'instituts.

Cette visualisation est une petite application web, faite avec la bibliothèque *Dash* disponible sous python, qui permet de projeter des graphes avec un serveur web.

Sachant que nos résultats des affectations sont exportés dans des fichiers sous fromat *JSON* ou *CSV*, donc les résultats restent toujours, consultables via fichiers.

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

FIGURE 1 – Visualisation via app web

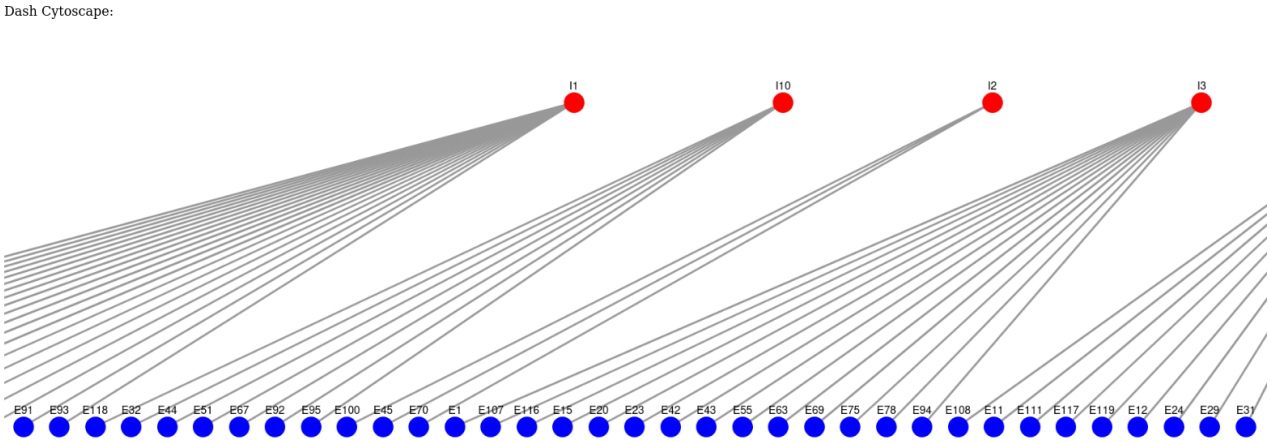


FIGURE 2 – Visualisation via fichier JSON

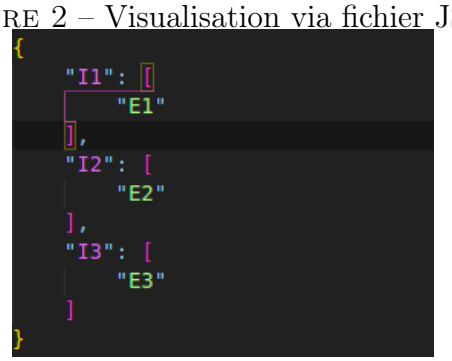
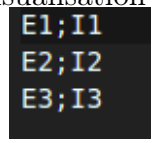


FIGURE 3 – Visualisation via fichier CSV



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