Conception et implantation d'un système d'aide à la décision

Affectation par mariages stables

HAI902I - Aide à la décision

FABRE Charlotte - *21814533*
HADDAD Gatien - *21903888*
POINTEAU Gabrielle - *21917975*

Accès à l'interface web

Table des matières

1 Introduction

1.1 Présentation du sujet

Le domaine de l'aide à la décision répond au besoin de certains utilisateurs d'être guidés lors de leur processus de prise de décision. Pour cela, un ensemble de systèmes d'aide à la décision permettent de prendre en compte des préférences et de proposer en retour des décisions en fonction de ces préférences.

Dans le cadre de la matière HAI902I - Aide à la décision, nous avons réalisé un projet de groupe dans lequel notre objectif était de nous intéresser à l'implémantation d'un système d'aide à la décision. Il s'agissait pour cela de proposer une implémentation de l'algorithme des mariages stables et une méthode d'évaluation de satisfaction des préférences, ainsi qu'une interface pour pouvoir effectuer des tests sur des listes de préférences générées aléatoirement.

1.2 Choix du langage

Afin de réaliser les différentes étapes de ce projet, et notamment pour faciliter la résalisation d'une interface graphique au visuel satisfaisant, nous avons choisi d'implémenter le projet dans les langages du web.

Nous avons ainsi choisi d'utiliser HTML et CSS pour la partie visuelle, et JavaScript pour rendre la navigation intéractive ainsi que pour implémenter l'algorithme des mariages stables. Nous avons également utilisé Boostrap pour améliorer le rendu visuel. Cela nous a notamment permis de proposer une application web *responsive* donc facilement accessible sur mobile.

1.3 Présentation du plan

Pour présenter les aboutissements de notre projet, notre rapport se structurera autour de ses étapes clés, de la génération des préférences à l'intégration des représentations compactes, avec un accent particulier sur l'implémentation de l'algorithme des mariages stables, et l'analyse et l'évaluation des résultats obtenus par rapport aux préférences.

2 Algorithme des Mariages Stables

Dans cette première partie, nous présentons l'algorithme des mariages stables ainsi que l'implémentation que nous en avons faite.

2.1 Présentation de l'Algorithme

L'algorithme des mariages stables est un système d'aide à la décision qui permet la résolution de problèmes d'affectation, en établissant des correspondances entre les entités de deux ensembles, tout en essayant de respecter au mieux les préférences de chaque entité. Dans notre cas, les deux ensembles concernés sont des étudiants et des établissements, et les préférences sont :

- Pour les établissements : les étudiants qu'ils veulent recruter en priorité
- Pour les étudiants : les établissements qu'ils veulent intégrer en priorité

Pour présenter cet algorithme, nous commencerons par expliquer rapidement la façon dont nous avons choisi sa déclinaison « Affectation différée », puis nous décrirons son fonctionnement.

2.1.1 Choix de la déclinaison

Il existe plusieurs déclinaisons de l'algorithme des mariages stables. Les plus couramment utilisées, et celles que nous avons vues en cours, sont l'algorithme de Boston et l'algorithme d'acceptation différée.

Pour choisir la déclinaison que nous avons implémentée, nous avons repris les trois critères d'évaluation d'un système d'aide à la décision présentés dans le cours : efficacité, équité, non-manipulabilité.

Nous savons que, malgré se popularité, l'algorithme de Boston ne respecte aucun de ces trois critères, c'est donc naturellement que nous avons choisi l'algorithme d'acceptation différée.

2.1.2 Description de l'algorithme

Une des caractéristiques de l'algorithme des mariages stables, et en particulier de l'algorithme d'acceptation différée, est sa flexibilité quant à la priorité attribuée aux préférences des étudiants par rapport à celles des établissements. Dans notre cas, nous avons opté pour accorder une priorité aux préférences des étudiants plutôt qu'à celles des établissements.

Il est important de souligner que ce choix est personnel et dépend des objectifs spécifiques du projet. En privilégiant les préférences des étudiants, notre implémentation vise à maximiser la satisfaction individuelle des étudiants dans le processus d'affectation, reflétant ainsi une préférence personnelle au sein de la flexibilité offerte par l'algorithme des mariages stables.

L'algorithme des mariages stables par affectation différée prend en entrée les préférences des étudiants et des établissements. Il initialise un ensemble d'étudiants libres (qui correspond aux étudiants non affectés à un mariage), dans lequel on place l'ensemble des

étudiants. L'algorithme entame alors un processus itératif jusqu'à ce que tous les étudiants soient appariés.

Durant chaque itération, l'algorithme parcourt les étudiants libres et explore pour chacun d'eux leurs préférences d'établissements.
En premier lieu, il tente de les marier avec le premier établissement disponible dans leurs préférences. On considère alors ces deux cas :

- L'établissement en premier choix de l'étudiant est libre
- L'établissement est déjà affecté

Si l'établissement n'a pas encore de mariage, un mariage est établi et l'étudiant est retiré de la liste des étudiants libres.

Dans le cas où l'établissement a déjà un mariage, l'algorithme va alors comparer les préférences de l'établissement sur l'étudiant actuel et l'étudiant déjà marié à cet établissement. Si l'étudiant actuel est préféré, le mariage est mis à jour avec le nouvel étudiant, et l'étudiant précédemment apparié redevient libre. Sinon, l'étudiant actuel reste libre.

Ce processus se répète jusqu'à ce que tous les étudiants soient appariés, garantissant ainsi des mariages stables en respectant au mieux les préférences définies.

Pour éviter au programme de boucler, l'algorithme vérifie en premier lieu si les nombres d'étudiants et d'établissements sont égaux.

S'il y a plus d'établissements que d'étudiants, cela ne pose pas de problème étant donné que l'on va s'arrêter dès que tous les étudiants sont affectés à un mariage.

Dans le cas où il y aurait plus d'étudiants que d'établissements, l'algorithme doit vérifer si l'on a intérogé au moins une fois tous les étudiants. Si c'est le cas et que l'ensemble des étudiants libres reste inchangé par rapport à l'itération précédente, l'algorithme se termine, garantissant alors une stabilité dans les affectations.

2.2 Implémentation de l'algorithme

Nous avons donc implémenté l'algorithme d'acceptation différée en JavaScript en suivant la logique de l'algorithme décrit précédemment.

Dans un premier temps, notre programme récupère le fichier de préférences fourni par l'utilisateur. Le fichier va alors être parsé pour récupérer les préférences individuelles de chaque étudiant et établissement.

Initialement, tous les étudiants sont considérés comme libres, et le processus itératif se poursuit jusqu'à ce que tous les étudiants soient affectés ou qu'aucun nouvel appariement ne soit possible.

Une fois l'affectation terminée, la fonction renvoie les affectations à notre programme principal, qui se charge d'afficher les différentes informations concernant les mariages, à savoir :

- Les affectations pour chaque étudiant et pour chaque établissement affecté
- Les étudiants ou établissements qui n'ont pas été affectés (dans le cas d'un déséquilibre entre le nombre d'étudiants et d'établissements)

3 Implémentation du contexte et de l'interface

Afin d'optimiser la structuration de notre interface graphique, nous l'avons divisée en cinq parties distinctes, chacune remplissant un rôle spécifique :

1. Génération de préférences aléatoires
2. Calcul des affectations
3. Visualisation des préférences
4. Visualisation des affectations
5. Mesure de la satisfaction

Un guide de l'utilisation de l'interface et de ces différentes parties pourra être retrouver en annexe du rapport, en section 7.1.

3.1 Génération des préférences aléatoies

Pour générer les préférences des étudiants et des établissements, nous avons mis en place une fonction *generer_preferences()* qui, pour chaque étudiant (resp. établissement), génère un tableau d'entiers uniques, représentant les indices des établissements (resp. étudiants) dans un ordre aléatoire.

Par exemple dans le cas où il y aurait 2 étudiants et 5 établissements, une des représentations possibles des préférences pour les étudiants serait la suivante :

```
{
    "1": [ 1, 4, 3, 5, 2 ],
    "2": [ 4, 1, 5, 3, 2 ]
}
```

Ici, l'étudiant n°1 place l'établissement n°1 en premier choix, puis l'établissement n°4, etc.

Pour initier la génération de préférences aléatoires, l'utilisateur doit remplir deux champs de saisie, spécifiant respectivement le nombre d'étudiants et d'établissements (figure 1 page 5).

L'utilisation du bouton *Télécharger les préférences* indique alors à notre programme de générer un fichier de préférences (figure 2 page 5). Ce fichier contient toutes les informations essentielles pour les appariements, à savoir le nombre d'étudiants et d'établissements, ainsi que les préférences respectives de chaque entité.

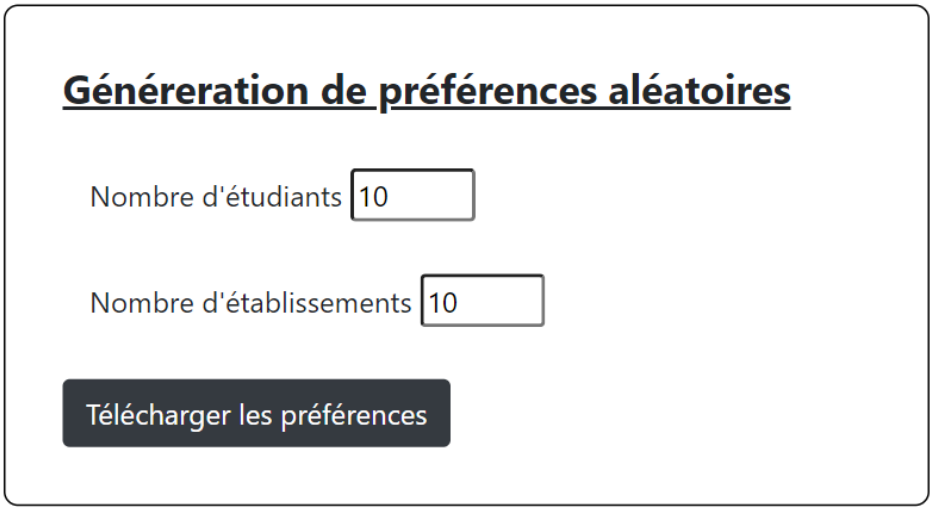


FIGURE 1 – Champs de génération des préférences

```
1 {
    "nbEtudiants": 3,
    "nbEtablissements": 3,
    "preferencesEtudiants": {
        "1": [2, 1, 3],
        "2": [1, 2, 3],
        "3": [1, 2, 3]
    },
    "preferesEtablissement": {
        "1": [1, 3, 2],
        "2": [2, 1, 3],
        "3": [2, 1, 3]
    }
}
```

FIGURE 2 – Exemple de fichier de préférences

Plutot que de générer des préférences et de les utiliser directement dans notre programme, nous avons choisi de générer un fichier téléchargable, et ce, pour deux raisons.

La première est que cette méthode permet de générer un grand nombre de fichier de préférences aléatoires à la suite, ce qui sera utile plus tard pour effectuer de nombreux tests.

Dans un second temps, cette méthode nous permettait de créer et de modifier nos propres fichiers de préférences, pour nous permettre d'effectuer des tests précis sur l'algorithme des mariages stables ou de génération de préférences.

3.2 Calcul des affectations

En fournissant au programme un fichier de préférences, ce dernier est alors en mesure de calculer un ensemble d'affectations à l'aide de l'algorithme des mariages stables (voir partie 2.1.2).

Appliqué à notre exemple de préférences précédent (figure 2), l'algorithme renverrait une affectation dans ce format :

```
{
    "1": 1,
    "2": 2,
    "3": 3
}
```

FIGURE 3 – Exemple de fichier d'affectation

3.3 Visualisation des préférences

Une fois le calcul des affectations réalisé, le programme met en place l'affichage des préférences pour chaque étudiant et chaque établissement (figure 4 page 6). Cet affichage permet de visualiser les préférences directement dans notre interface plutot que de devoir ouvrir le fichier téléchargé.

Nous observons ainsi deux tableaux distincts, l'un correspondant aux étudiants et l'autre aux établissements, où chaque numéro représente l'indice correspondant à un établissement ou un étudiant. Chaque ligne est alors formatée avec "Étu N°i : $1 \succ 2 \succ 3$" indiquant que, pour l'étudiant i, l'établissement n°1 est préféré à l'établissement n°2, qui est lui-même préféré à l'établissement n°3 (et réciproquement pour les établissements).

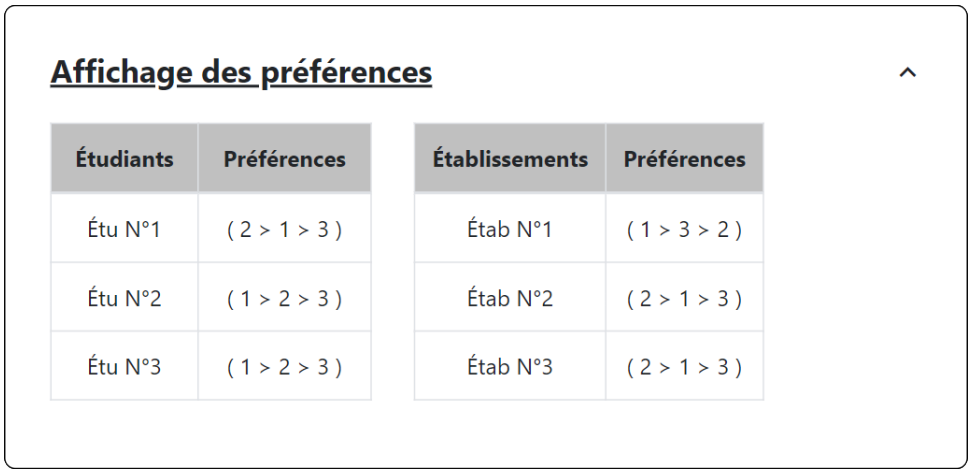


FIGURE 4 – Affichage des préférences

3.4 Visualisation des affectations

Ensuite, une étape importante du programme intervient : la visualisation des affectations. Ces appariements sont représentés dans un tableau où chaque ligne correspond

à un mariage. Les deux colonnes indiquent avec qui chaque étudiant (et respectivement chaque établissement) a été apparié.

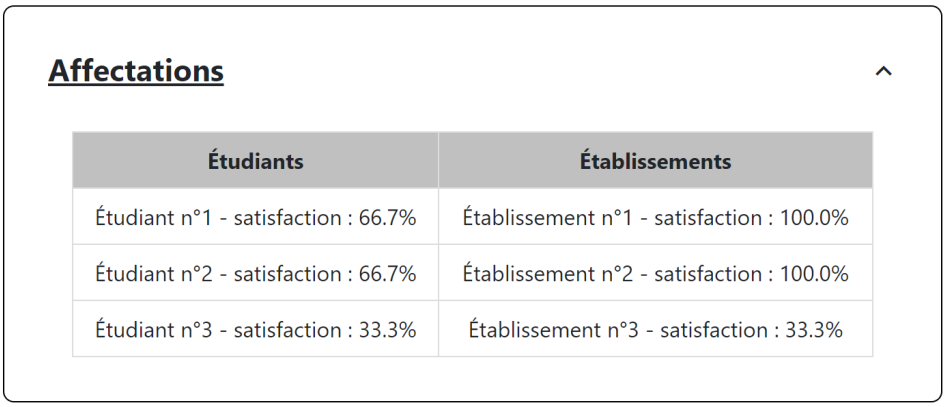


FIGURE 5 – Affichage des affectations

Dans la figure 5 qui reprend notre exemple précédent, nous constatons avec la première ligne que l'étudiant n°1 à été apparié avec l'établissement n°1

Dans chaque case du tableau, est également indiquée la satisfaction de l'individu concerné, dont nous détaillons le calcul dans la partie suivante.

3.5 Mesure de la satisfaction

Pour finir, cette dernière partie concerne le calcul de la satisfaction. Cette étape est cruciale car elle sert à évaluer la qualité des affectations générées. Pour cela, trois métriques sont ici prises en comptes : la satisfaction moyenne des étudiants, la statisfaction moyenne pour les établissements et enfin une moyenne globale pondérée.

Le calcul de la satisfaction de chaque étudiant se fait en regardant la position de l'établissement dans ses préférences (et réciproquement). Un pourcentage de satisfaction est attribué en fonction de cette position. Plus un pourcentage est elevé, plus l'individu a obtenu un choix qu'il préférait. Ainsi, ce calcul reflète la pertinence des appariements par rapport aux préférences individuelles.

Si l'on reprend notre exemple précédent, les préférences de l'étudiant n°1 étaient "$2 \succ 1 \succ 3$". Or, il a été affecté à l'établissement n°1, sa satisfaction est donc de $\frac{2}{3}$, soit 66.7%.

Avec ce calcul, on constate que la satisfaction minimale qu'un individu peut obtenir en étant apparié avec son dernier choix est de $1/n$ (n étant le nombre de préférences de l'individu) et non 0. En effet, la satisfaction atteint 0 lorsque l'individu n'est simplement pas apparié, ce qui arrive lors des cas de déséquilibre entre le nombre d'étudiants et d'établissements.

La satisfaction globale, quant à elle, n'est pas simplement la moyenne des deux moyennes des satisfactions précédentes, mais est calculée en fonction des nombres respec-

tifs d'étudiants et d'établissements, donnant ainsi une mesure équilibrée de la satisfaction générale.

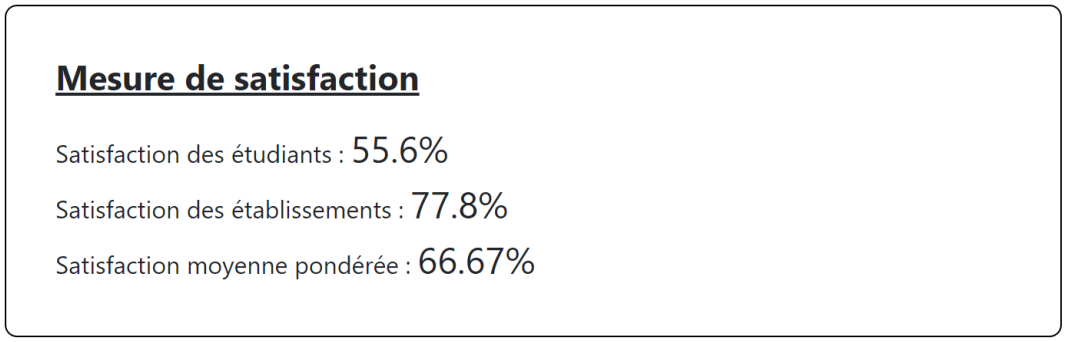


FIGURE 6 – Affichage de la satisfaction

4 Analyse des résultats

Après avoir présenté l'algorithme des mariages stables ainsi que l'implémentation du projet, nous nous attardons désormais sur l'analyse des résultats obtenus. Cette phase est importante pour évaluer l'efficacité de notre implémentation du système d'aide à la décision basé sur l'algorithme des mariages stables.

Pour effectuer cette analyse, nous reprenons le critère d'évaluation des affectations que nous venons de présenter, et nous l'appliquons à un certains nombre d'instances-tests

4.1 Choix des paramètres de test

Pour effectuer des tests pertinents afin d'évaluer les affectations proposées par notre implémentation de l'algorithle des mariages stables, les paramètres ont été choisis de façon à représenter une large gamme de scénarios différents.

4.1.1 Premier cas : cardinalités égales

Pour les premiers paramètres de tests, nous avons décidé de nous intéresser au cas le plus simple : celui dans lequel il y a autant d'étudiants que d'établissements. En effet, dans ce cas, chaque étudiant et établissement possède une affectation à la fin de l'algorithme.

Dans ce cas des cardinalités égales, nous avons proposé un certain nombre d'instances de tailles différentes, de façon à voir évoluer la satisfaction en fonction de l'échelle de grandeur. Ainsi, pour n le nombre d'étudiants et d'établissements, nous avons testé

- $n = 10$
- $n \in [50, 500[$ avec un pas de 50
- $n \in [500, 1000[$ avec un pas de 100
- $n \in [1000, 5000[$ avec un pas de 500

Nous nous sommes arrêtes à $n = 5000$ pour garder un temps de calcul quasi-immédiat, puisqu'à partir de 5000 il y avait plus de 5 secondes de calcul.

4.1.2 Second cas : cardinalités différentes

Dans un second temps, nous nous sommes intéressés aux instances plus complexes dans lesquelles les cardinalités des ensembles d'étudiants et d'établissements diffèrent.

Pour cela, nous avons choisi de prendre 1000 étudiants et établissments confondus au total, tel que le nombre d'étudiant soit n et le nombre d'établissements $1000 - n$. Nous avons alors fait varier n dans l'intervalle $[100, 900]$, par pas de 100.

4.2 Test et évaluation des jeux de données

Une fois les paramètres choisis pour les deux cas de cardinalités égales et différentes, nous avons lancé les tests sur les différentes instances. Ainsi, pour chaque couple de paramètres {nombre d'étudiants, nombre d'établissements}, nous avons généré 5 fichiers d'instances sur lesquels nous avons calculé une moyenne des résultats de satisfaction.

À partir des résultats obtenus, nous avons produit 3 graphes représentant la satisfaction selon les différents critères : satisfaction des étudiants, satisfaction des établissements et satisfaction globale.

4.2.1 Résultats pour les cardinalités égales

Dans ce premier cas, nous visualisons la satisfaction des entités (étudiants et établissements) sur la figure présentée à la page 10. Dans ce graphe, le nombre d'étudiants et d'établissements considérés est le même, et on présente la satisfaction en fonction de ce nombre.

Comme le précise la légende, les courbes bleues et rouges représentent respectivement les satisfactions des étudiants et des établissements, tandis que la courbe orange représente la satisfaction globable pondérée.

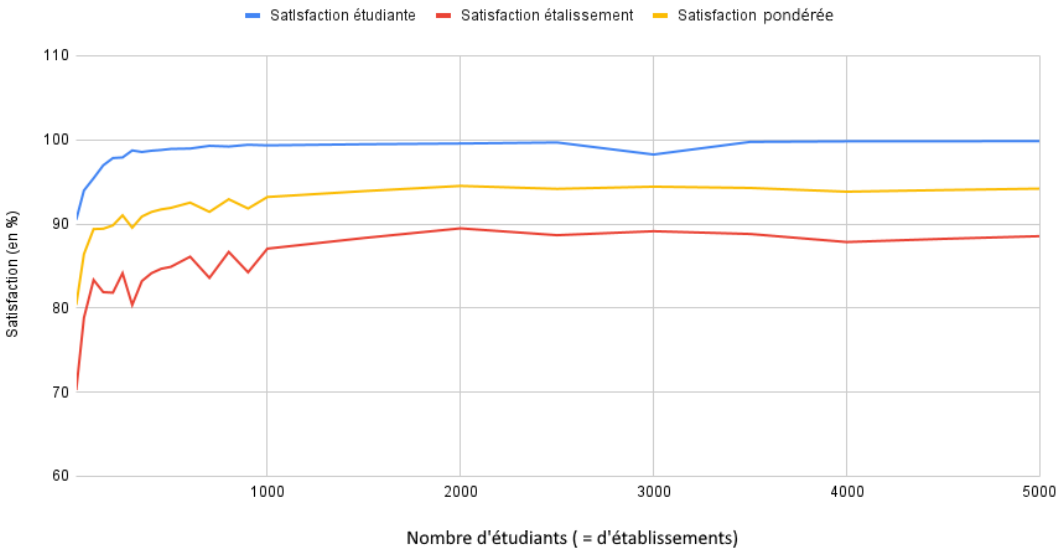


FIGURE 7 – Satisfaction en fonction du nombres d'étudiants

Sur le graphique, il apparaît que la satisfaction des trois courbes augmente, surtout entre 0 et 1000 étudiants et établissments, et que la satsifaction moyenne des étudiants semble converger vers 100%.

Dans un premier temps, les variations que l'on observe entre 0 et 1000 sont dues à notre méthode d'échantillonage. Comme nous avons effectué plus de tests à cet intervale, la courbe semble moins stable à cet endroit là.

Pour ce qui est de la convergence, ce résultat est dû au fait que notre calcul de la satsifaction fait intervenir le nombre de préférences. En effet, un étudiant qui obtient son second choix lorsqu'il y a 10 établissements va obtenir une satisfaction de 9/10, soit 90%. Lorsque ce dernier nombre est plus élevé, disons 1000 établissements, l'étudiants ayant eu son second choix se retrouve alors avec une satisfaction de 999/1000, soit 99,9%.

Plus généralement, si on note $(n-k)/n$ la satisfaction de l'étudiant ayant eu son k-ième choix, on se rend compte que, peu importe le k moyen, la satisfaction tendra vers

100% lorsque n tend vers l'infini.

Une seconde observation que l'on peu effectuer est que la satisfaction moyenne des étudiants est toujours plus grande que celle des établissements. Cela s'explique par l'implémentation de l'agorithme des mariages stables que nous avons choisie.

En effet, comme nous l'avons expliqué plutôt, ce sont les préférences des étudiants qui sont privilégiées par rapport à celles des établissmements. Ainsi, l'algorithme cherche à proposer les meilleures affectations possibles aux étudiants, même si cela peut se faire au détriment des préférences des établissements.

La satisfaction des établissements (courbe rouge) est ainsi la plus basse car ceux-ci se « contentent » des propositions qui leurs sont faites par les étudiants : ils ne choisissent pas leurs étudiants préférés parmi l'ensemble des étudiants, mais parmi ceux qui se proposent à eux.

4.2.2 Résultats pour les cardinalités différentes

Dans ce second cas, nous visualisons à nouveau le graphe des satisfactions en fonction du nombre d'étudiants et d'établissments. À nouveau, la légende est la suivante : la satisfaction des étudiants est en bleu, celle des établissments est en rouge et la satisfaction globale pondérée est en orange.

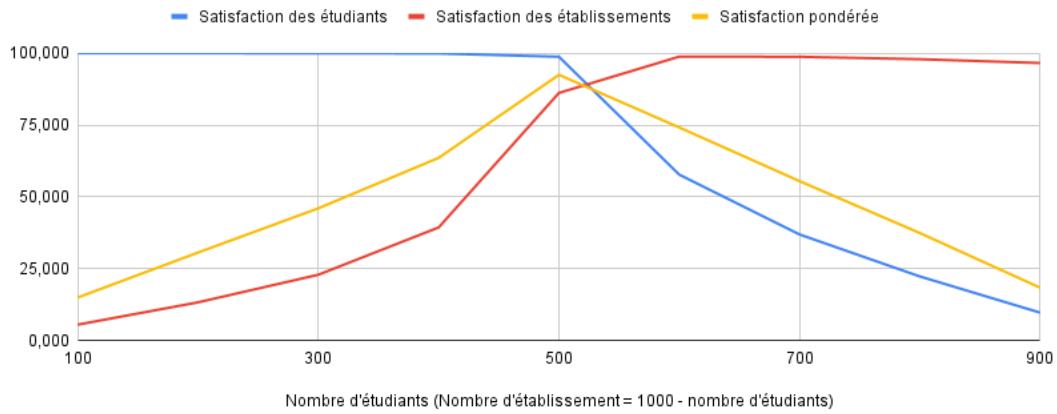


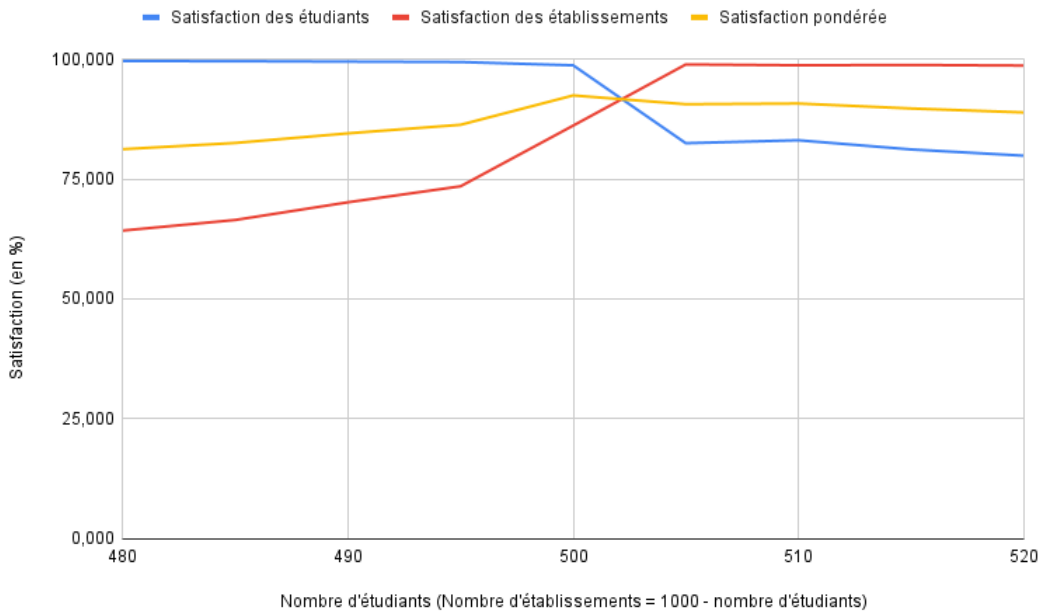
FIGURE 8 – Graphe de la satisfaction en fonction du ratio d'étudiants

Sur ce second graphe, les courbes de satisfaction des étudiants et des établissements s'apparentent à des sigmoïdes, dont l'une est normale ou montante (pour les établissments) et l'autre est inversée ou descendante (pour les étudiants). La courbe de satisfaction pondérée a alors la forme d'une courbe en cloche de Gaüs, puisqu'elle est calculée à partir d'une moyenne des deux autres courbes. Ce rendu s'explique par nos choix de paramètres : nous avons fait varier les couples de nombre d'étudiants et d'établissments de (100, 900) à (900, 100) en utilisant un pas de 100 pour chacune des deux valeurs.

Ainsi, la partie gauche du graphe correspond aux couples dans lesquels le nombre d'étudiants est supérieur à celui d'établissements, puisque le premier varie entre 100 et 500 et le second entre 500 et 900. Il est alors assez compréhensible que la satisfaction des établissments soit très réduite par rapport à celle des étudiants, puisque la quantité

d'établissements non-affectés fait chuter la satisfaction du groupe. De façon symétrique, dans la partie droite du graphe, la satisfaction des étudiants est négativement éffectée par la non-satisfaction des étudiants non-affectés, de plus en plus nombreux lorsqu'on se déplace vers la droite.

La situation optimale apparaît alors être le cas dans lequel il y a 500 étudiants et 500 établissements. En effet, la satisfaction globale connaît son maximum lorsque les deux cardinalités sont équilibrées, comme on peut le voir sur le graphe, puisqu'aucun étudiant ou établissement non-affecté ne pèse dans la mesure de la satisfaction.



Pour terminer la lecture et l'analyse de ce graphique, nous proposons d'effectuer un *zoom* sur la partie centrale, dans laquelle le nombre d'étudiants et d'établissements s'équilibre, faisant augmenter la satisfaction globale. Pour cela, nous faisons évoluer le nombre d'étudiants de 460 à 520 et le nombre d'établissements de 540 à 580, en réglant le pas à 10. Le graphique représentant ce nouveau groupe de tests est présenté à la page 12.

Sur ce nouveau graphique, on remarque que la satisfaction globale maximum se situe à 500 étudiants et 500 établissements. Cela confirme notre analyse précédente : dans notre modèle la satisfaction globale est meilleure lorsque le nombre d'étudiants et d'établissements est identique.

Cette dernière analyse nous permet de réfléchir au point suivant : dans un modèle dans lequel l'instatisfaction des établissements ou étudiants non-affectés ne serait pas pris en compte ou serait très faiblement pondéré dans le calcul de la satisfaction globale, est-ce que les résultats des tests nous indiqueraient toujours que l'équilibre des cardinalités est meilleur ? Cette question est une des perspectives de continuité de ce projet.

5 Intégration des représentations compactes

Pour conclure ce projet, nous nous sommes intéressé à l'intégration des représentations compactes dans le modèle que nous avons proposé. Pour cela, nous avons considéré deux pistes différentes.

D'une part, nous avons réfléchi à la façon dont notre modèle pourrait prendre en compte des préférences exprimées dans des représentations compactes.

D'autre part, nous nous sommes intéressés à comment ajouter des critères de choix pour pouvoir représenter les préférences graphiquement en utilisant les CP-net.

5.1 Lecture de représentations compactes

Pour cette première piste de réflexion, nous avons voulu réfléchir à l'idée de lancer l'algorithme sur des préférences exprimées sous la forme de représentations compactes plutôt que sur un fichier tel que nous l'avons proposé plus tôt. Pour cela, nous nous intéressons aux différentes représentations compactes que nous avons pu rencontrer en cours.

5.1.1 Représentation en logique pondérée

La première des représentations compactes à laquelle nous proposons de nous intéresser est celle des logiques pondérées. Dans ces représentations, les connaissances sur les préférences sont présentées dans des bases de connaissances dans lesquelles chaque critère de préférence est associé à une valeur qui peut représenter un coefficient de plausibilité (logique possibiliste) ou une pénalité (logique de pénalité).

Dans le cas où les choix sont constitués de plusieurs critères de préférence (entrée, plat, dessert par exemple), une valeur est calculée pour chaque choix en fonction du sous-choix fait pour chaque critère.

Dans notre cas, une préférence correspond à un seul critère : l'établissement pour un étudiant, et inversement. Ainsi, soit n le nombre d'étudiants et m le nombre d'établissements, les préférences seraient exprimées sous la forme suivante :

- $\Sigma^i = \{(eta_1, v_1), (eta_2, v_2), ..., (eta_m, v_m)\}\ \forall i = 1..n$
- $\Sigma^j = \{(etu_1, w_1), (etu_2, w_2), ..., (etu_n, w_n)\}\ \forall j = 1..m$

Ainsi, pour chaque étudiant i, il existe une base de connaissances qui contient l'ensemble des universités pondérées. De même, pour chaque établissement j, il existe une bases de connaissances qui contient l'ensemble des étudiants pondérés.

À partir de ces bases de connaissances qui représentent les préférences des étudiants et établissements, nous voulons déduire un fichier de la forme de nos fichiers actuels. Pour cela, nous voulons déduire un ordre de préférence à partir des pondérations.

Nous faisons alors une remarques sur la forme des bases de connaissances acceptées : nous n'autorisons pas deux élément d'une même base à avoir la même pondération. Autrement exprimé, $\forall i, j.\ i \neq j \Rightarrow v_i \neq v_j \wedge w_i \neq w_j$.

En effet, pour obtenir un ordre de préférence à partir d'une pondération, nous voulons simplement prendre les éléments dans l'ordre croissant/décroissant de leur poids (en

fonction du type de pondération). Ainsi, pour avoir un ordre strict et non un pré-ordre, il ne doit pas y avoir d'égalité entre les pondération.

L'ordre obtenu est également total, puisque pour tout étudiant i, $|\Sigma^i| = m$, et pour tout établissement j $|\Sigma^j| = n$.

À partir de telles bases de connaissances sur les préférences à un seul critère, on peut donc facilement établir un ordre strict et total sur les préférences, ce qui nous permet de construire un fichier de la bonne forme.

5.1.2 Représentation en logique conditionnelle et représentation graphique

Après avoir réussi à proposer une façon de traiter des préférences représentées en logique pondérée, nous voulons essayer de faire la même chose pour deux autres types de représentations compactes : la représentation en logique conditionnelle et la représentation graphique.

Dans ces deux représentations, les préférences sont présentées sous l'angle des dépendences entre leurs critères. Or, puisque dans notre cas chaque préférence correspond à un seul critère, ces deux représentations n'auraient que très peu d'intérêt.

- Dans le cas de la représentation en logique conditionnelle, il n'y a aucune façon d'exprimer les préférences des étudiants pour les établissements et inversement, puisque les préférences des étudiants ou des établissements de sont pas conditonnées dans notre modèle.
  Pour les conditionner en restant dans le cadre d'une préférence uni-critère, on pourrait par exemple proposer un modèle étendu dans lequel on pourrait composer des groupes d'amis voulant étudier ensemble parmi les étudiants, et dans lesquels on désignerait un ami principal.
  Soit par exemple $\{etu_1, etu_2, etu_3\}$ un sous-ensemble des étudiants et soit etu_1 l'ami principal, on pourrait par exemple écrire $eta_1 >_{etu_1} eta_2 : eta_1 >_{etu_2} eta_2$ et $eta_1 >_{etu_1} eta_2 : eta_1 >_{etu_2} eta_2$, c'est-à-dire que si le premier étudiant a une préférence entre les deux premiers établissements, les deux autres étudiants ont la même préférence pour rester avec leur ami.
- Dans le cas de la représentation graphique, on pourrait exprimer les préférences des étudiants et des établissements en proposant simplement un graphe dont aucun des sommets ne seraient adjacents, puisque un arc représente une dépendance. Un tel graphe comporterait alors autant de sommets que d'étudiants et d'établissements, et pour chaque sommet on inscrirait directement la liste des préférences de l'étudiant ou de l'établissement à côté de son sommet. Le format serait déjà le même que nos fichiers, et cela présenterait très peu d'intérêt.

Les préférences qui sont composées d'un seul critère ou paramètre n'ont donc aucun intérêt pour ces deux dernières représentations compactes. Dans la partie suivante, on s'intéresse alors à l'idée d'ajouter des critères aux préférences des étudiants et établissements.

5.2 Ajout des paramètres

L'idée consiste à utiliser une représentation graphique des préférences, en particulier les CP-nets. Nous devons introduire des paramètres supplémentaires pour les étudiants et les établissements afin de comparer les préférences entre elles.

Prenons un exemple, un étudiant peut avoir une préférence pour une ville par rapport à une autre, et ensuite, en fonction de ce choix, préférer un type d'établissement à un autre. Ainsi, voici quelques critères de préférence qu'on pourrait faire choisir aux étudiants :

- Ville : *Montpellier, Paris, Bordeaux, ...*
- Type de filière : *Classe préparatoire aux grandes écoles (CPES), Licence, BTS, DUT, ...*
- Type d'établissement : *Lycée (CPES), Université (CPES), ...*
- Filière : *Scientifique, Littéraire, Économique, Juridique, ...*

Dans le processus de préférence, chaque étudiant serait confronté à plusieurs choix, ainsi nous allons donner un exemple à partir des critères énoncés ci-dessus. En suivant une approche similaire à celle de l'algorithme de mariages stables, où le choix des étudiants prime sur celui des établissements, examinons ces préférences une par une dans un exemple.

Considérons alors les 4 variables binaires suivantes : V (Ville), F (Filière), TF (Type de Filière) et E (type d'Établissement) avec les domaines suivants :

— $\mathrm{Dom}(V) = \{V_m, V_p\}$ avec m : Montpellier, p : Paris ;

— $\mathrm{Dom}(F) = \{F_s, F_e\}$ avec s : scientifique, e : économique ;

— $\mathrm{Dom}(TF) = \{TF_c, TF_d\}$ avec c : CPES, d : DUT ;

— $\mathrm{Dom}(E) = \{E_l, E_u\}$ avec ly : lycée, u : université ;

Un étudiant exprime alors des préférences partielles :

(P_1) : il préfère la ville de *Montpellier* à celle de *Paris*,

(P_2) : il préfère une filière *scientifique* qu'une filière *économique*,

(P_3) : quand c'est la ville de *Montpellier*, si la filière est *scientifique*, il préfère la *CPES* comme type de filière ; sinon, il préfère le *DUT*

(P_4) : quand c'est la ville de *Paris*, il préfère la *CPES* comme type de filière

(P_5) : quand c'est la *CPES*, il préfère le *lycée* comme établissement ; sinon, il préfère l'*université*

Nous pouvons alors représenter ces préférences partielles à l'aide d'un graphique : le CP-net. Il représente chaque variable liée et les préférences associées. On peut alors le voir sur la figure 9 à la page 16.

Ainsi, à partir du CP-net nous pouvons montrer l'ordre des préférences partielles associé sur la figure 10 à la page 16.

Par ailleurs, l'algorithme des mariages stables privilégie les choix des étudiants en attendant que tous les étudiants soient affectés. On aurait aussi pu choisir de privilégier les établissements, et ajouter des critères comme l'âge des étudiants (> 20 ans, < 50 ans), les notes (> 10, < 10) ou encore le diplôme (Bac "scientifique", Bac "litteraire") pour réaliser des préférences partielles.

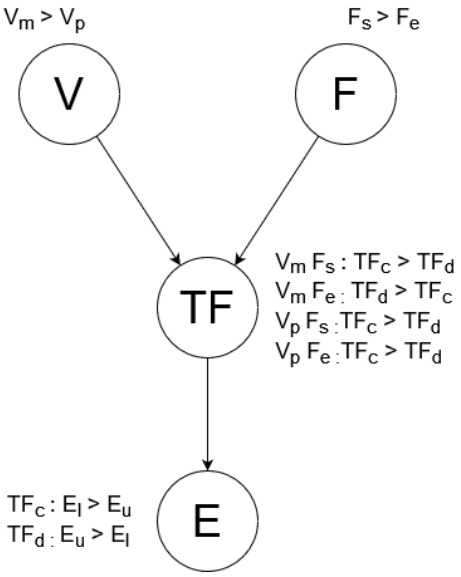


FIGURE 9 – Graphe du CP-net associé aux préférences énoncées

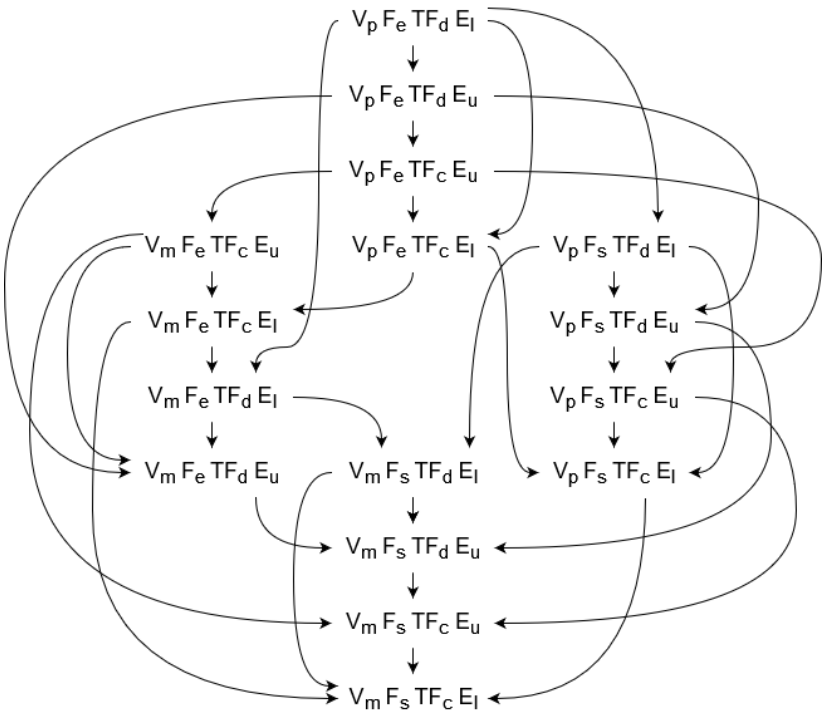


FIGURE 10 – Graphe des ordres de préférences partielles associées au CP-net

6 Conclusion

Ce projet de conception d'un système d'aide à la décision a abouti à une compréhension approfondie de l'algorithme des mariages stables par affectation différée, de son implémentation, et de son application dans le contexte de la gestion des préférences entre des étudiants et des établissements.

Le découpage du projet en différentes étapes clés nous à permis de visualiser notre avancement et de voir le projet se construire au fur et à mesure.

Dans un premier temps, l'implémentation du contexte nous a permis de prendre en main le sujet et nous a galvanisé pour la suite du développement de l'algorithme des mariages stables.

L'analyse des résultats a quant à elle permis, dans un second temps, de tester l'algorithme dans différentes conditions, notamment avec des nombres d'étudiants et d'établissements différents. Cela nous a au passage permis de corriger des erreurs algorithmiques spécifiques à certains jeux de données.

Enfin, l'intégration des représentations compactes a constitué une extension intéressante du projet. Elle nous a permis d'envisager différentes perspectives pour étoffer notre travail.

En conclusion, ce projet nous a permis de mettre en lumière l'importance de l'aide à la décision dans des domaines comme la sélectivité, en démontrant comment des algorithmes tels que celui des mariages stables peuvent contribuer à résoudre ce type de problèmes complexes. Ce rapport offre un aperçu des aspects clés du projet, tout en posant les bases pour des perspectives d'améliorations futures et d'applications plus larges de ces concepts dans le vaste domaine de l'aide à la décision.

Pour aller plus loin, nous pourrions par exemple envisager des scénarios où les établissements peuvent être affectés à plusieurs étudiants. Cela permettrait de modéliser des situations plus complexes, offrant une flexibilité suplémentaire dans la résolution de problèmes d'affectations. Cette perspective d'extension ouvre des opportunités pour des applications plus réalistes de notre travail, s'inscrivant dans une vision plus globale d'optimisation des processus d'aide à la décision.

7 Annexe

7.1 ReadMe

L'interface est disponible à cette adresse :
projet-hai902iaide-a-la-decision.leogendra.repl.co/

1. **Saisie des informations initiales**
   Les deux champs de saisie représentant le choix du nombre d'étudiant et du nombre d'établissment doivent être remplis pas l'utilisateur. Les valeurs entrées doivent être validées, ce qui lance le calcul des préférences aléatoires.
2. **Téléchargement automatique du fichier**
   Dès que les préférences aléatoires des étudiants et établissements sont calculées, un fichier au format JSON les contenant est automatiquent téléchargé.
3. **Choix du fichier**
   Après avoir appuyé sur le bouton *Charger un fichier de préférences*, l'utilisateur doit choisir le ficher qui vient d'être téléchargé pour lancer l'exécution de l'algorithmes des mariages stables. Une fois le fichier choisi, l'utilisateur accède à trois nouvelles sections de l'interface : *Affichage des préférences*, *Affectations* et *Mesure de satisfaction*.
4. **Affichage des préférences**
   L'utilisateur peut visualiser les préférences aléatoires générées plus tôt.
5. **Visualisation des Affectations**
   L'utilisateur peut visualiser les affectations renvoyées par l'algorithme des mariages stables.
6. **Analyse des satisfactions**
   L'utilisateur peut visualiser la satisfaction moyenne des établissements et des étudiants sous la forme de pourcentages, il peut également voir la satisfaction générale sous forme d'une moyenne pondérée.