
Aide à la décision

M2 Informatique - IASD

2025–2026

Auteurs

AL AYOUBI Ibrahim

BELGUITH Rami

KAMMOUN Habib

TOUKEBRI Dhia

Encadrant

KACI Souhila



**UNIVERSITÉ DE
MONTPELLIER**

1. INTRODUCTION

Ce projet porte sur l'étude du problème du mariage stable. Ce problème consiste à appairer deux groupes, ici des étudiants et des établissements, en tenant compte de leurs préférences respectives. L'objectif est de produire un appariement stable. Un appariement est stable lorsqu'aucun étudiant et aucun établissement ne préfèrent se réappairer ensemble plutôt que conserver leur affectation actuelle. Ce projet s'inscrit dans le cadre du cours d'Aide à la Décision (IASD). Il vise à analyser l'Algorithme du mariage stable, à générer des préférences aléatoires, à mesurer la satisfaction des agents et à tester le système sur plusieurs jeux de données. Nous proposons également une extension théorique pour intégrer des représentations compactes des préférences, telles que vues dans le cours.

2. PROBLÉMATIQUE

Le problème du mariage stable consiste à appairer deux ensembles de participants, par exemple des étudiants et des établissements. Chaque étudiant exprime un classement des établissements. Chaque établissement exprime également un classement des étudiants. L'objectif est de produire un appariement stable. Autrement dit, il ne doit exister aucun couple étudiant–établissement ayant intérêt à s'associer entre eux plutôt que de conserver leurs affectations. Cette notion de stabilité est essentielle. Elle garantit qu'aucun agent ne souhaite changer unilatéralement de partenaire. Elle permet aussi d'éviter les situations de blocage ou d'injustice dans les systèmes d'affectation. Ce problème trouve des applications concrètes : affectation scolaire, recrutement hospitalier, attribution de logements universitaires ou encore marchés du travail organisés. Dans ce projet, nous analysons le modèle classique et nous proposons des outils supplémentaires pour évaluer la satisfaction des participants.

3. MÉTHODOLOGIE

Notre étude suit les quatre premières étapes définies dans le sujet.

3.1 Génération des préférences

Nous avons implanté un module capable de générer automatiquement les préférences des étudiants et des établissements. Les préférences sont aléatoires, complètes et strictes. Le générateur s'appuie sur une fonction de permutation, ce qui garantit que chaque agent dispose d'un classement cohérent.

3.2 Implémentation de l'algorithme de Gale–Shapley

Nous avons programmé la version classique de l'algorithme à propositions étudiantes. L'algorithme fonctionne par itérations simples :

- tous les étudiants commencent libres ;

- chacun propose à l'établissement le mieux classé parmi ceux qu'il n'a pas encore sollicités ;
- l'établissement accepte temporairement le meilleur candidat selon ses préférences ;
- un étudiant moins bien classé peut être rejeté et redevenir libre.

Ce processus se répète jusqu'à ce qu'aucun étudiant libre ne puisse proposer à un nouvel établissement. L'appariement obtenu est garanti stable.

4. MESURE DE LA SATISFACTION

Afin d'évaluer la qualité des appariements produits par l'algorithme de Gale–Shapley, nous avons défini une mesure de satisfaction appliquée à chaque groupe d'agents : les étudiants et les établissements.

4.1 Rang du partenaire obtenu

Pour chaque étudiant, nous calculons le rang de l'établissement auquel il est affecté dans sa liste de préférences. Un rang égal à 0 correspond au premier choix, un rang égal à 1 au deuxième choix, et ainsi de suite. Le même principe est appliqué du côté des établissements. Ces rangs constituent une mesure brute de satisfaction individuelle.

4.2 Satisfaction normalisée

Afin d'obtenir une mesure globale comparable entre scénarios de tailles différentes, nous utilisons une satisfaction *normalisée* définie sur l'intervalle $[0, 1]$.

Étant donné un ensemble de rangs r_1, r_2, \dots, r_n et une longueur maximale de liste N , la satisfaction normalisée est définie par :

$$\text{Sat} = 1 - \frac{\frac{1}{n} \sum_{i=1}^n r_i}{N - 1}.$$

Cette équation exprime une idée simple :

- si la moyenne des rangs est proche de 0 (agents satisfaits), la satisfaction est proche de 1 ;
- si la moyenne est proche de $N - 1$ (agents très insatisfaits), la satisfaction tend vers 0.

Cette mesure est cohérente, monotone et facilement interprétable.

4.3 Interprétation

La satisfaction normalisée atteint :

- 1 si tous les agents obtiennent leur premier choix ;
- 0 si tous obtiennent leur dernier choix ;
- une valeur intermédiaire si les agents obtiennent des choix moyens.

La mesure est indépendante de la taille de l'instance, ce qui permet de comparer plusieurs expériences.

4.4 Exemple

Considérons trois étudiants, avec les rangs suivants pour l'établissement auquel ils sont affectés :

$$r = [0, 2, 1].$$

Ils ont donc obtenu respectivement leur premier, troisième et deuxième choix. Nous supposons que chaque étudiant a classé $N = 3$ établissements. La moyenne des rangs vaut :

$$\bar{r} = \frac{0 + 2 + 1}{3} = 1.$$

La satisfaction normalisée s'écrit alors :

$$\text{Sat} = 1 - \frac{1}{N-1} \bar{r} = 1 - \frac{1}{2}(1) = 1 - \frac{1}{2} = 0,5.$$

Ainsi, dans cet exemple, la satisfaction du groupe étudiant est de 50%. Cette valeur reflète un compromis raisonnable : un étudiant très satisfait, un moyennement satisfait, un moins satisfait.

4.5 Métriques complémentaires

La méthode propose également d'autres indicateurs descriptifs :

- le taux de premiers choix obtenus ;
- le rang moyen et le rang médian ;
- la distribution complète des rangs ;

L'ensemble de ces mesures offre une analyse complète de la qualité d'une affectation stable.

4.6 Expérimentations

Nous avons testé le système sur plusieurs jeux de données variés. Un module dédié permet de :

- générer de multiples configurations ;
- exécuter l'algorithme ;
- collecter et archiver les résultats ;
- produire des visualisations comme des histogrammes ou des comparaisons de satisfaction.

Ces expériences permettent d'observer l'influence de la taille du problème et des préférences sur la qualité des appariements.

4.6.1 Résultats et analyse des scénarios.

Nous avons évalué quatre configurations : 50×50 , 100×100 , 200×200 et 500×500 . Pour chaque scénario, nous présentons un ensemble de quatre visualisations regroupées dans une seule figure :

1. la distribution des rangs obtenus par les établissements,
2. la distribution des rangs obtenus par les étudiants,
3. la satisfaction individuelle (par étudiant et par établissement),
4. la satisfaction globale.

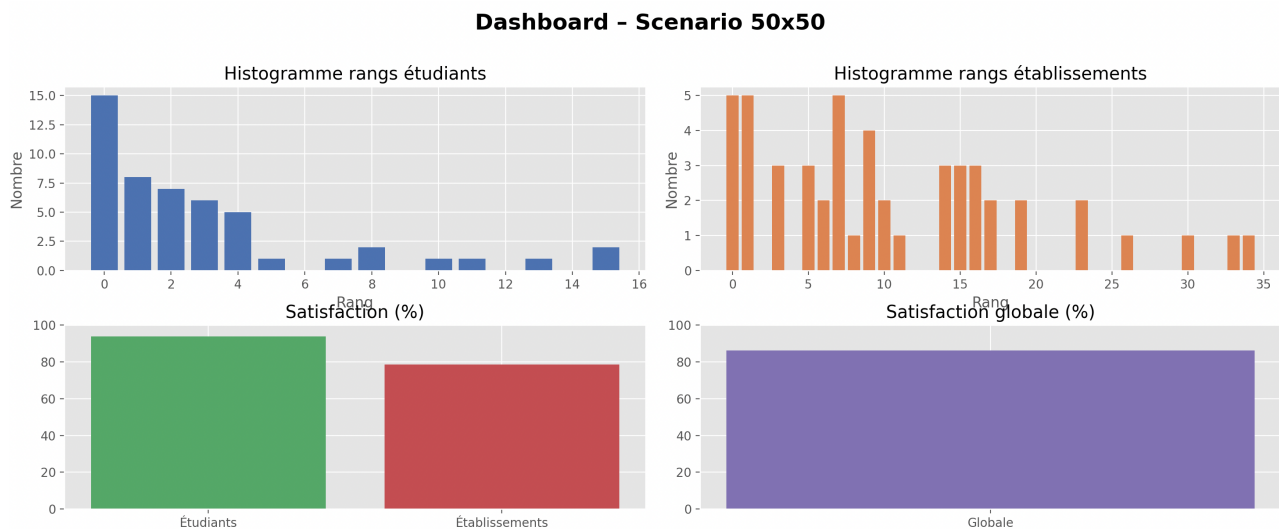


FIGURE 1 – Visualisations pour la configuration 50×50 .

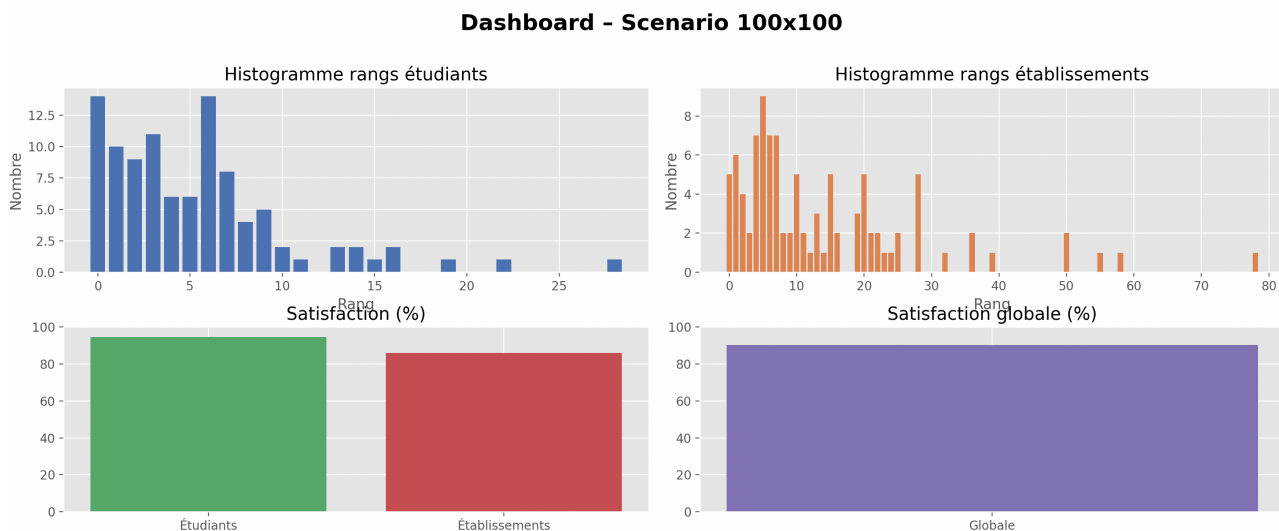


FIGURE 2 – Visualisations pour la configuration 100×100 .

Dashboard - Scenario 200x200

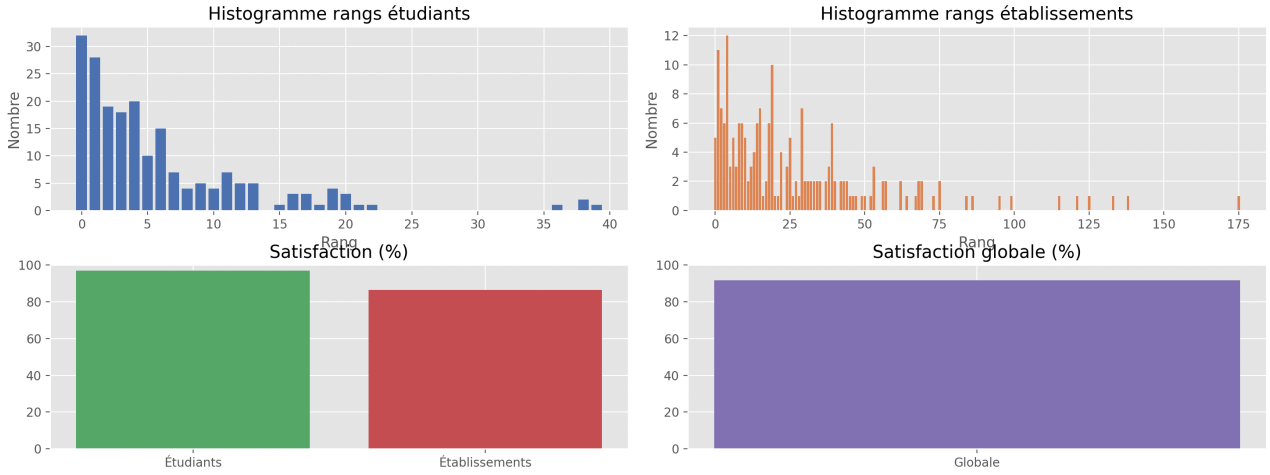


FIGURE 3 – Visualisations pour la configuration 200×200 .

Dashboard - Scenario 500x500

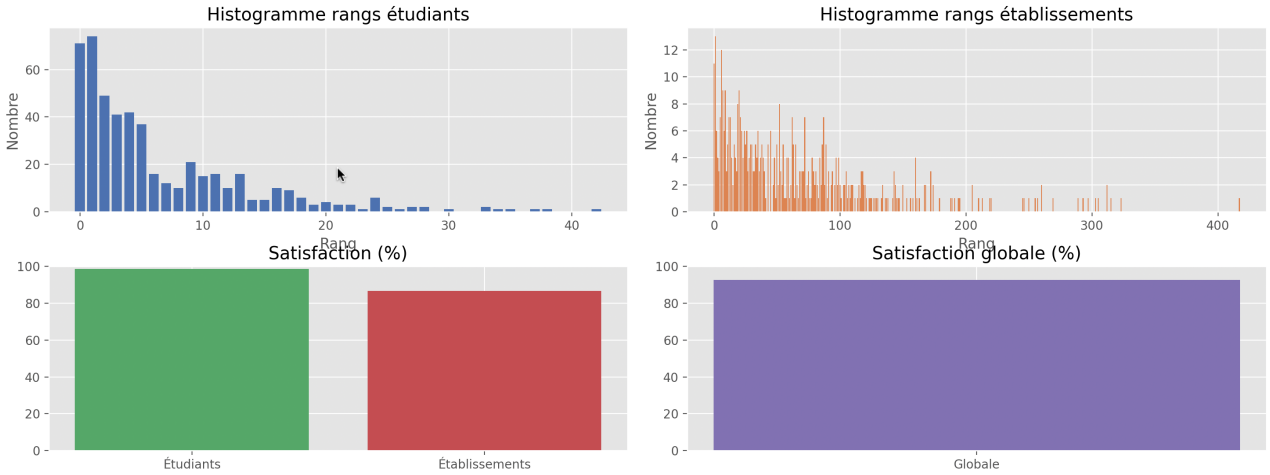


FIGURE 4 – Visualisations pour la configuration 500×500 .

Les résultats montrent clairement que les étudiants obtiennent très souvent des rangs faibles, ce qui signifie qu'ils décrochent généralement l'un de leurs premiers choix. Leur niveau de satisfaction reste donc très élevé dans les quatre scénarios (entre 92% et 97%), ce qui est normal puisque l'algorithme favorise le côté qui fait les propositions.

Pour les établissements, la situation est plus fluctuante. Leur satisfaction est correcte en 20×20 , elle monte nettement en 50×50 , puis elle redescend en 100×100 . Quand la taille de l'instance augmente, les étudiants sont plus nombreux à entrer en concurrence, et les établissements finissent souvent avec des candidats moins bien classés dans leurs préférences.

De manière générale, les expériences confirment que l'algorithme produit des appariements stables et globalement bons, mais avec un avantage marqué pour les étudiants, ce qui correspond exactement au comportement attendu de la version étudiant-proposant de Gale-Shapley.

5. EXTENSION : REPRÉSENTATIONS COMPACTES DES PRÉFÉRENCES

Les préférences complètes sont souvent coûteuses à collecter et peu réalistes dans des systèmes de grande taille. Pour rendre le modèle plus fidèle aux pratiques réelles tout en réduisant la charge d'expression imposée aux agents, nous proposons d'intégrer des représentations compactes des préférences, telles que présentées dans le cours IASD. Nous détaillons ci-dessous quatre approches adaptées à notre cadre.

5.1 Préférences par scores

Chaque agent attribue un score à chaque partenaire. Les scores permettent de dériver un classement automatiquement. Cette représentation est simple et réduit fortement la quantité d'information à fournir.

5.2 Priorités (School Choice)

Les établissements peuvent exprimer des priorités plutôt qu'un ordre complet : secteur géographique, résultats, critères sociaux, etc.

Cette représentation compacte est celle utilisée dans les systèmes réels d'affectation scolaire. L'algorithme d'acceptation différée peut fonctionner directement sur ces priorités.

5.3 CP-nets

Les CP-nets sont un formalisme puissant pour représenter des préférences structurales, conditionnelles et multicritères. Ils permettent d'exprimer des règles du type :

- « Si la formation est excellente, alors la distance importe peu. »
- « Si deux établissements ont la même réputation, je préfère le plus proche. »
- « Si l'étudiant a d'excellentes notes, la provenance de son école devient moins importante. »

Chaque CP-net définit :

- des variables de décision (ex : réputation, distance, qualité pour un étudiant ; notes, CV, établissement d'origine pour une école),
- des dépendances conditionnelles entre ces variables,
- des tables de préférences locales (CPT) qui expriment les préférences *ceteris paribus* pour toutes les configurations pertinentes.

Les CP-nets permettent ainsi aux agents d'exprimer leurs préférences de manière compacte, qualitative et intuitive, conformément au cours.

L'ordre global induit par une CP-net est un ordre partiel, mais il peut être étendu en un ordre total compatible, par exemple via les *worsening flips*. Cet ordre total peut alors servir directement de liste de préférences dans Gale-Shapley.

5.4 Logique Possibiliste

Lorsque les critères sont indépendants et peuvent être évalués séparément (réputation, distance, coût, qualité...), la logique possibiliste pondérée, présentée, constitue une méthode simple et efficace.

Chaque critère est associé à une formule pondérée (φ_i, a_i) , où a_i est un degré d'importance. Pour chaque établissement (ou étudiant), on calcule une note globale de satisfaction :

$$\pi(\omega) = 1 - \max\{a_i \mid \omega \not\models \varphi_i\}.$$

Plus $\pi(\omega)$ est élevé, plus l'option est satisfaisante.

Cette approche permet d'obtenir aisément un ordre total sur les établissements (côté étudiant) ou sur les étudiants (côté établissement), en triant selon le degré de satisfaction calculé.

5.5 Notre Choix

Parmi les différentes représentations compactes étudiées, les CP-nets apparaissent comme la solution la plus adaptée à notre système. Elles permettent de modéliser finement les dépendances entre critères, d'exprimer des préférences conditionnelles réalistes et d'obtenir automatiquement un ordre total compatible avec l'algorithme de Gale–Shapley, sans nécessiter aucune modification. Grâce à leur expressivité et à leur capacité à capturer des préférences multicritères complexes de manière compacte, les CP-nets constituent le choix naturel pour intégrer des préférences avancées dans notre modèle d'appariement.

ANNEXE A — DESCRIPTION TECHNIQUE DES MODULES ET CLASSES PYTHON

Cette annexe présente l'architecture logicielle du projet ainsi que le rôle des principales classes engagées dans l'implémentation du problème du mariage stable. L'objectif est de fournir une vue structurée et exhaustive des composants, sans reproduction du code source, conformément aux exigences du rapport.

A.1. Génération des préférences

La classe `GenerateurPreferences` constitue le point d'entrée du système. Elle est paramétrée par le nombre d'étudiants, le nombre d'établissements et une graine pseudo-aléatoire optionnelle. Sa méthode principale, `generer()`, produit deux structures de type dictionnaire contenant respectivement :

- les préférences ordinales des étudiants ;
- les préférences ordinales des établissements.

Chaque préférence est représentée sous la forme d'une permutation stricte de l'ensemble opposé. Ce mécanisme garantit la cohérence des données et leur reproductibilité en cas d'utilisation

d'une graine fixe.

A.2. Algorithme d'appariement stable

La classe `AlgorithmeGaleShapley` implémente l'algorithme de Gale et Shapley dans sa version classique à propositions du côté étudiant. Elle reçoit en entrée les deux dictionnaires de préférences.

La méthode `executer()` orchestre l'ensemble du processus d'appariement. Elle mobilise des structures internes permettant :

- de maintenir l'ensemble des étudiants non appariés ;
- de suivre, pour chaque étudiant, l'indice de la prochaine proposition ;
- d'enregistrer l'appariement courant du côté des établissements.

La comparaison entre deux candidats pour un établissement s'appuie sur une fonction interne, `_score_etablissement()`, qui transforme les listes de préférences en scores indexés, facilitant les décisions de remplacement.

Un mécanisme de journalisation, assuré par la méthode `_enregistrer_etape()`, conserve l'état du système à chaque interaction, ce qui permet une analyse a posteriori du comportement de l'algorithme.

A.3. Analyse des résultats

La classe `AnalyseurSatisfaction` permet l'évaluation approfondie des appariements produits. Elle repose sur trois types de méthodes :

- des méthodes de calcul des rangs obtenus par les étudiants et par les établissements ;
- une fonction `satisfaction_normalise()` destinée à normaliser les rangs sur une échelle continue ;
- une méthode de synthèse, `rapport_complet()`, fournissant un ensemble de métriques : rangs moyens, taux de premiers choix, distributions de rangs, satisfaction globale et coût social.

La structure retournée par `rapport_complet()` sert de base à l'analyse empirique des performances du mécanisme.

A.4. Exécution expérimentale

La classe `TesteurAlgorithme` a été conçue pour automatiser la conduite d'expériences sur différents jeux de données. Elle encapsule l'ensemble du processus, depuis la génération des préférences jusqu'à la production des indicateurs d'évaluation. Sa méthode centrale, `tester_configuration()` effectue successivement :

1. la génération des préférences ;
2. l'exécution de l'algorithme d'appariement stable ;
3. l'analyse de la satisfaction ;

4. l'archivage complet des résultats.

Chaque exécution génère un enregistrement structuré incluant les paramètres initiaux, l'appariement final, les mesures de satisfaction et l'historique détaillé du processus.

A.5. Visualisations

Le projet inclut enfin un module dédié à la visualisation, implémenté dans la classe `Visuals`. Cette classe propose une méthode principale, `dashboard()`, qui génère automatiquement un tableau de bord complet pour chaque scénario testé. Chaque dashboard regroupe plusieurs représentations graphiques :

- un histogramme des rangs obtenus par les étudiants ;
- un histogramme des rangs obtenus par les établissements ;
- une comparaison visuelle des niveaux de satisfaction des deux groupes ;
- une représentation de la satisfaction globale ;

Cette approche centralisée permet de produire, pour chaque configuration, une visualisation cohérente et complète des performances du système.

6. CONCLUSION

Ce projet nous a permis de comprendre en profondeur le fonctionnement de l'algorithme du mariage stable, tant sur le plan théorique que pratique. En générant des préférences aléatoires et en observant les résultats obtenus pour différentes tailles de scénarios, nous avons pu analyser la qualité des appariements produits et mesurer la satisfaction des deux groupes. Les expérimentations montrent que l'algorithme parvient à fournir des appariements stables et globalement satisfaisants, tout en révélant certaines variations selon la taille des instances et la structure des préférences.

L'étude des représentations compactes nous a également montré comment le modèle peut être étendu pour mieux s'adapter à des situations réelles, comme l'affectation scolaire ou d'autres systèmes d'allocation. Ce travail nous a ainsi offert une vision complète du problème, de ses applications possibles et des pistes pour aller plus loin dans la modélisation.