
Projet ANDROIDE Mariage stable dynamique

May 21, 2022

Binome:

Xinyu Huang 3803966

Ruohui Hu 21102304

Muyang Shi 28714332

Professeur:

M.Bruno Escoffier

Contents

1	Introduction	3
1.1	L'objectif du projet est:	3
1.2	Différents algorithmes appliqués.	3
1.3	Planning estimé:	3
1.4	Description des fichiers	4
2	Algo 1/2 : Principe de l'algorithme Gale-Shapley	5
3	Algo de Base : programmation linéaire pour le problème mariage stable	6
4	Algo 3 : programmation linéaire itérative	6
5	Algo 4 : programmation linéaire globale	7
6	Aller plus loin: étudier le problème de mariage stable séquentiel	7
7	Analyse des algorithmes	8
7.1	Temps de calcul	8
7.2	Capacité de former des mariages inchangés au cours de changement des listes	9
7.3	Complexité en temps de calcul	9
7.3.1	Interpolation:division par la fonction devinée	9
7.3.2	Interpolation:passer à log	10
7.4	Conclusion	11
8	Mode d'emploi de l'interface <i>stable mariage calculator</i>	11
9	Bibliographie	15

Résumé

Dans ce projet, nous avons tout d'abord étudié le problème simple Mariage stable et le résoudre par *l'algorithme Gale-Shapley*.

Ensuite, on traite un problème plus difficile: le problème de mariage stable séquentiel en utilisant deux types de programmation linéaire : *PL séquentielle* et *PL globale*.

On résume notre travail en créant une petite interface qui facilite la résolution des deux problèmes.

1 Introduction

Le problème des mariages stables est au cœur de nombreuses procédures d'affectation, la plus connue en France étant probablement ParcourSup. Il y a dans ce problème deux types de joueurs (hommes/femmes, candidat(e)s/universités, ...), chaque joueur d'un type donnant ses préférences sur les joueurs de l'autre type (les universités classent les candidat(e)s par exemple). Le but est de trouver une affectation/un couplage vérifiant une propriété de stabilité.

L'algorithme le plus connu pour trouver une telle affectation est l'algorithme de Gale-Shapley. Nous nous intéressons dans ce projet non seulement coder les solutions en algorithme de Gale-Shapley et programmation linéaire, mais aussi étudier ce problème dans une situation dynamique : à chaque pas de temps nous réinitialise le nombre de l'homme et femme et on va trouver un algorithme qui minimise le changement de couples.

1.1 L'objectif du projet est:

- Résoudre le problème simple(**Mariage stable**) par Gale-Shapley et programmation linéaire.
- Dans le cas compliqué : **problème de Mariage stable "séquentiel"** (le nombre de femme et homme change au cours du temps et on veut minimiser le nombre de couples on change entre deux t consécutives), on veut aussi appliquer l'algorithme Gale-Shapley et programmation linéaire pour trouver le meilleur algorithme.
 - de faire des tests pour évaluer empiriquement les heuristiques implantées;
 - de réaliser une interface conviviale.

1.2 Différents algorithmes appliqués.

- **ALGO 1:** Appliquer l'algo de Gale-Shapley avec les **hommes** qui demandent
- **ALGO 2:** Appliquer l'algo de Gale-Shapley avec les **femmes** qui demandent
- **ALGO 3:** Programmation linéaire.
- **ALGO 4:** Programmation linéaire globale.

1.3 Planning estimé:

- Comprendre le problème et l'algorithme de Gale-Shapley avant la réunion : 20 Janvier - 27 Janvier

- Préparation l'algorithme Gale-Shapley, générer des préférence aléatoire : 28 Janvier - 7 Février
- Deux Programmation linéaire : 8 Février - 21 Février
- Deux algorithme Gale-Shapley, corrigé programmation linéaire : 22 Février - 14 Mars
- Test, algorithme approché et exact, algorithme binary à relaxation continue 20 Mars - 5 Avril
- Tests les contraintes de stabilité dans Algo3 et Algo4, Analyse des courbes de temps, Complexité théorique : 7 Avril - 20 Avril
- Rédaction du rapport : 23 avril - 5 Mai Cela correspond à trois mois et demi de travail. Cela est notre estimation de temps pour faire un travail de qualité. L'ensemble du projet est codé sous Python . Le rapport est écrit à l'aide de Overleaf en Latex

1.4 Description des fichiers

- name_female.json, name_female.json csv : la base de données concernant les noms des femmes/hommes
- get_csv.py : change les fichiers json au format csv
- gs.py : contenant les codes principale pour algorithmes Gale-shapley
- sm.py : le fichier principale contenant les fonctions:
 - genere_set_fm
 - genere_pref_dyn
 - genere_instance
 - calcul_difference_entre_gen
 - algo_1
 - algo_2
 - lire_entree
 - choix_algo
- moplex.py, moplex_new.py : application de linear programmation pour le problème mariage stable
- algo3.py : contenant la fonction PL itératif
- algo4.py : contenant la fonction PL globale
- interface.py : contenant les codes pour créer l'interface

2 Algo 1/2 : Principe de l'algorithme Gale-Shapley

IF opt = 0 (Femme privilege):

Initialisation: side_opt = femme
 pref_opt = pref_f
 side_des = male
 pref_des = pref_m

ELSE Initialisation par Homme privilege

WHILE: \exists free A a who has B b to propose do:

 b = first B on A's list to whom a has not yes proposed

IF pair(a',b) Then

 (a,b) become engaged

Else

IF b prefer a to a' Then

 a' \leftarrow *single*

End

End

REPEAT

```

1 def Gale_Shapley(female,male,pref_f,pref_m,opt=0):
2     """
3     Entree : Deux ensembles finis Female et Male de cardinal n et n ;
4             Deux dictionnaire de preference pref_f(cote femme) et pref_m(cote homme) ;
5             opt: par default 0, qui decide quel cote a privilegier(opt=0,femme-optimal;
6                 sinon homme optimale)
7     Sortie : Un ensemble S de couples engages (homme ; femme) ;
8     """
9     # choose which side to optimize
10    if opt==0:
11        side_opt=female.copy()
12        pref_opt=deepcopy(pref_f)
13        side_des=male.copy()
14        pref_des=deepcopy(pref_m)
15    else:
16        side_opt=male.copy()
17        pref_opt=deepcopy(pref_m)
18        side_des=female.copy()
19        pref_des=deepcopy(pref_f)
20    # The algos continue while it exists single people on the optimal side
21    while side_opt:
22        person=side_opt.pop()
23        for person_partner in (pref_opt[person][1]):
24            # when the partner is single, we form directly a couple
25            if pref_des[person_partner][0]=="Single":
26                pref_opt[person][0]=person_partner
27                pref_des[person_partner][0]=person
28                pref_opt[person][0]=person_partner
29                break
30            # when the partner is not single, we will compare with the competitor in
31            # the list of preference of partner
32            else:
33                competitor=pref_des[person_partner][0]
34                pref_comp=pref_des[person_partner][1].index(competitor)
35                pref_person=pref_des[person_partner][1].index(person)
36                if pref_person<pref_comp:
37                    pref_des[person_partner][0]=person
38                    pref_opt[person][0]=person_partner
39                    pref_opt[competitor][0]="Single"
40                    side_opt.append(competitor)
41                    break
42    return [(x, y[0]) for x,y in pref_opt.items() if y[0]!='Single']

```

3 Algo de Base : programmation linéaire pour le problème mariage stable

Pour utiliser les méthodes de PL plus avancées, on commence par PL pour le problème mariage stable:

Variable:

- x_{ij} : variable décrivant la relation entre le i-ième homme et le j-ième femme (1 si mariage entre i et j, 0 sinon)

Fonction objective:

$$z = \sum_{t=1}^n \max x_{ij} * V_{ij}$$

Soit V_{ij} une fonction d'utilité (par exemple la somme des préférence)

Contraintes:

- pour tout t, $\sum x_{ij}$ sur i ou j soit inf a 1 (mariage)
- $x_{ij} \geq 0$
- $x_{ij} \leq 1$
- Stabilité

4 Algo 3 : programmation linéaire itérative

Vu que programmation linéaire dans le cas Mariage stable est une version simplifiée, ici on s'intéresse à coder l'algorithme dans le cas "séquentiel".

Variable:

- x_{ijt} : variable décrivant la relation entre le i-ième homme et le j-ième femme (1 si mariage entre i et j, 0 sinon), à l'instant t

Vecteur de constant:

- c_{ijt-1} : le vecteur de taille $M \times N$ (nb d'homme/femme) qui est un sur une case ij si à $t-1$ l'homme i et la femme j forment un couple

Fonction objective:

$$z = \sum_{t=1}^n \max x_{ijt} * c_{ijt-1}$$

Contraintes:

- pour tout t , $\sum x_{ijt}$ sur i ou j soit inf à 1 (mariage)
- $x_{ijt} \geq 0$
- $x_{ijt} \leq 1$
- Stabilité

Avantage:

1) Comparer à l'algo Gale-shapley, la version programmation linéaire itérative est mieux parce que elle prend en compte à garder l'affectation des couples entre deux t consécutives.

5 Algo 4 : programmation linéaire globale

Vu que programmation linéaire dans le cas Mariage stable est une version simplifiée, ici on s'intéresse à coder l'algorithme dans le cas "séquentiel".

Variable:

- x_{ijt} : variable décrivant la relation entre le i -ième homme et le j -ième femme (1 si mariage entre i et j , 0 sinon), à l'instant t
- z_{ijt} : variable à minimiser

Fonction objective:

$$z = \min \sum_{t=0}^n z_{ijt}$$

Contraintes:

- pour tout t , $\sum x_{ijt}$ sur i ou j soit inf à 1 (mariage)
- $z_{ijt} \geq x_{ijt} - x_{ijt-1}$
- $z_{ijt} \geq 0$
- $x_{ijt} \geq 0$
- $x_{ijt} \leq 1$
- Stabilité

Avantage:

- 1) Comparer à l'algo itérative, résoudre le problème en une seule itération
- 2) Résoudre le problème de façon globale en minimisant la différence globale

6 Aller plus loin: étudier le problème de mariage stable séquentiel

Problème:

On veut étudier, dans une séquence de temps, le nombre de femmes et hommes change aléatoirement et on veut former les mariages pour tous les t qui minimise les changements de couples.

Ici on fixe le nombre de l'homme à valeur maximum, et le nombre de femme augment au cours du temps($min + pas * t$).

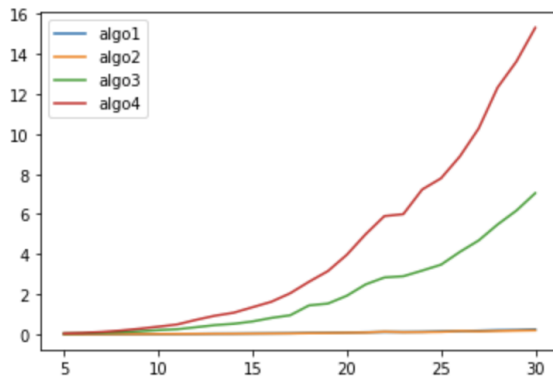
Étape:

- 1) Générer deux liste K et S, l'élément i de la liste correspond au nombre d'homme/femme à l'instant i
- 2) Choisir max hommes et femmes depuis la base de données et générer deux liste de préférences.
- 3) Selon K et S, on générer les listes de préférences spécifiques
- 4) On applique les quatres algos et comparer ses temps de calcul et valeurs.

7 Analyse des algorithmes

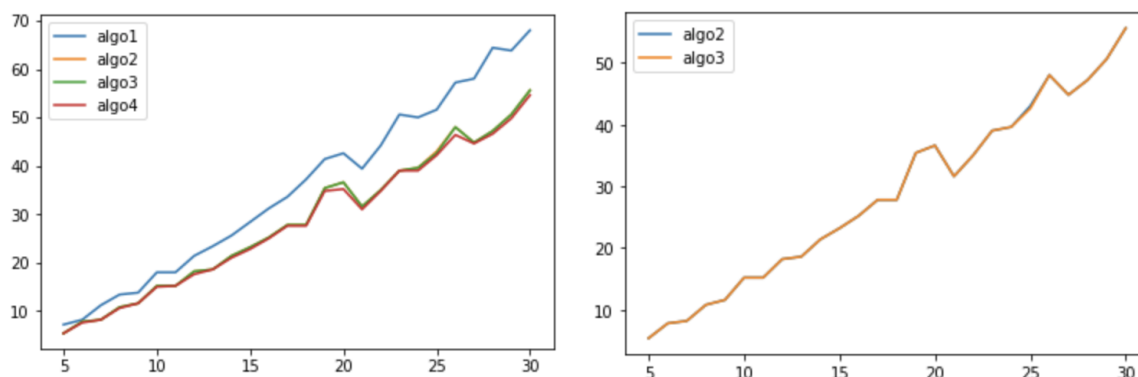
Dans notre projet, on se concentre sur un cas particulier pour le problème de mariage stable séquentiel: à chaque instant t , on fixe le nombre de personne d'un côté(par exemple femme) à valeur maximum, et on incrémente le nombre de personne de l'autre côté au cours du temps.À chaque pas de temps, on forme les mariages stables correspondants, notre objectif est de minimiser le nombre de changements de couples au cours du temps.

7.1 Temps de calcul



Interpretation : On peut bien remarquer que le temps de calcul de algo1 et algo 2 sont quasiment identiques, cela vient du fait que ces deux algos utilisent tous les deux la méthode Gale-shapley(femme optimal/homme optimal). Algo3 prend beaucoup plus de temps que algo1/2, et algo4 coûte plus de temps, qui est à peu près deux fois de plus que celle de l'algo3.

7.2 Capacité de former des mariages inchangés au cours de changement des listes



Interpretation :

Remarque 1 : À partir de la figure de gauche, on peut remarquer que l’algo 1 qui est le pire algo. Cela vient du fait que l’algo 1 n’a pas du tout pris en compte les changements de couples entre deux t différents.

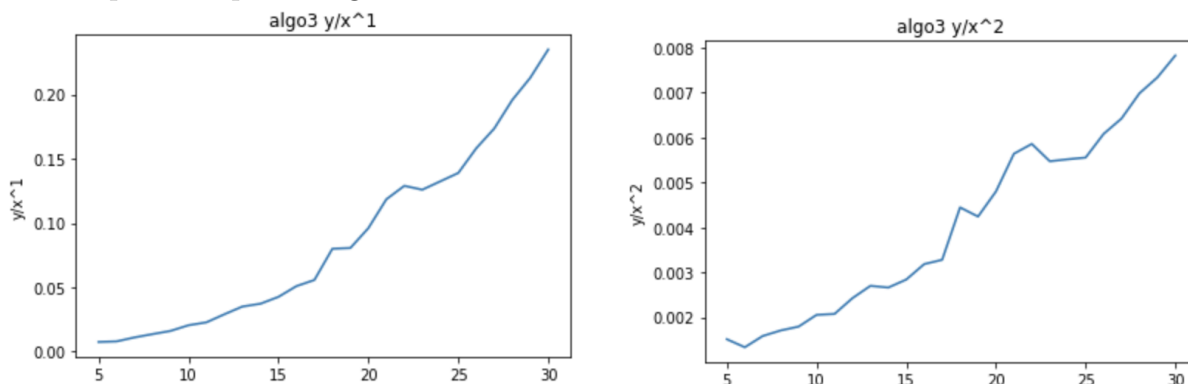
Remarque 2 : À partir de la figure de gauche, on peut aussi remarquer que la courbe de algo3 est très proche de cela de algo4, qui est logique car les deux algos prennent en compte à minimiser la différence entre t consécutive. Et algo4 est une borne inférieure de algo3 car il minimise les changements globaux et algo3 seulement considère à minimiser les changements entre les couples de temps consécutive.

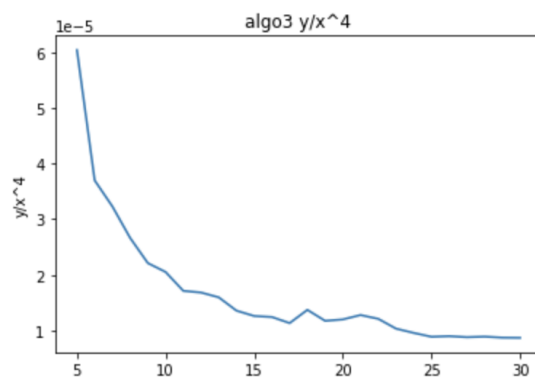
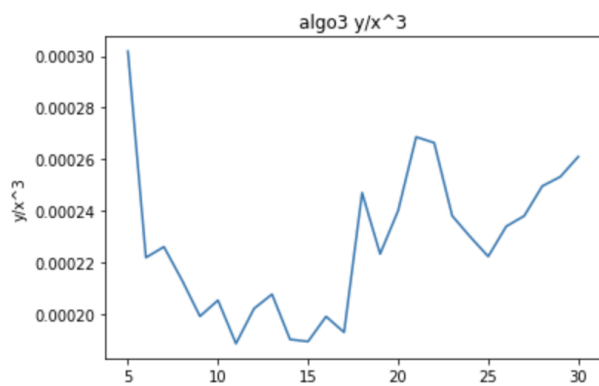
Remarque 3 : On ne peut pas trouver la courbe à partir de la courbe de gauche. En faisant la commande “print”, j’ai remarqué que la valeur de algo2 est très proche de la valeur de algo3 (une seule différence dans notre cas est quand $n=25$, algo2 est pire que algo3. C’est une remarque un peu magique mais compréhensible: dans notre algo3, on utilise une contrainte stabilité, qui est en fait une contrainte qui limite les deux côté (homme et femme), qui est plus vaste que celle de algo3 (qui limite seulement le côté femme), c’est pourquoi ils sont proches.

7.3 Complexité en temps de calcul

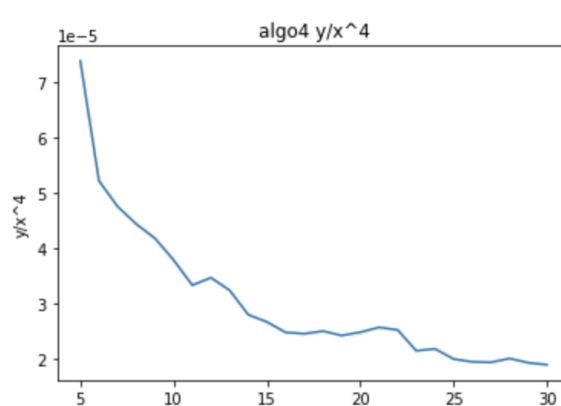
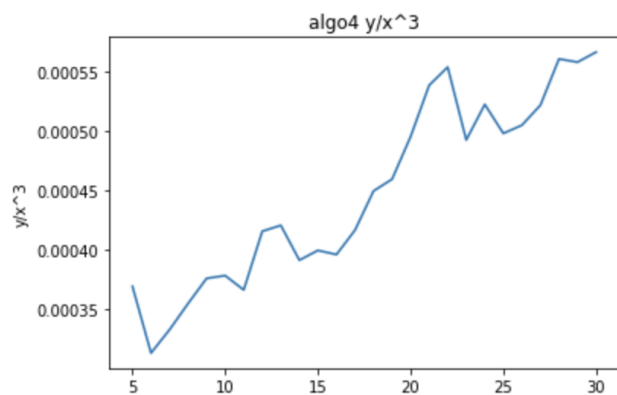
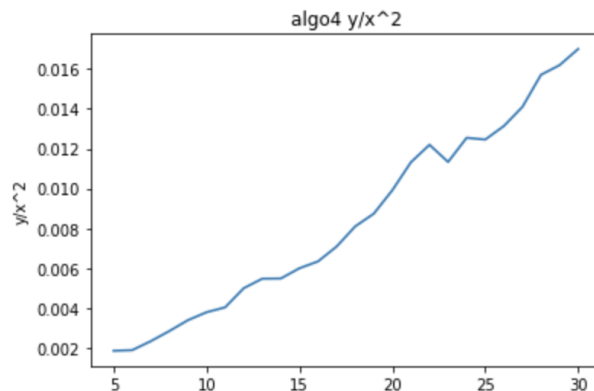
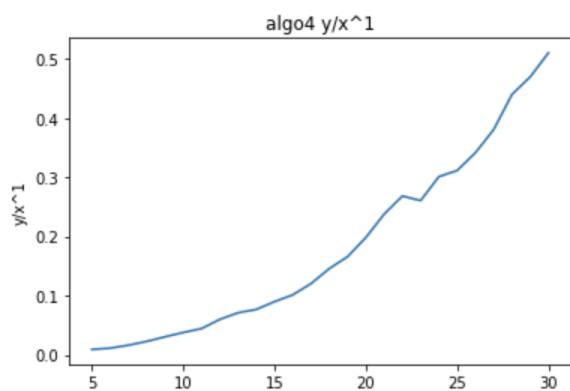
7.3.1 Interpolation: division par la fonction devinée

Interpolation pour l’algo3:

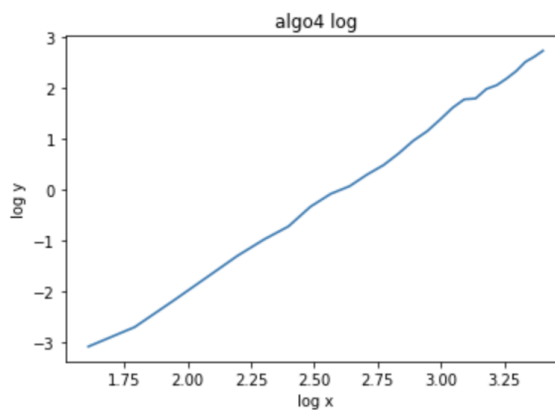
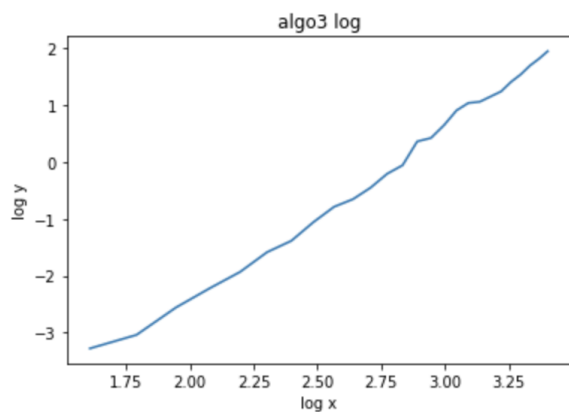




Interpolation pour l'algo4:



7.3.2 Interpolation: passer à log



Pour calculer la gradient, on prend deux points par droite :

Algo 3 : (3,0) (2.25,-2) $Gradient = 2 \div 0.75 = 2.67$

Algo 4 : (3,1) (1.75,-3) $Gradient = 4 \div 1.25 = 3.2$

7.4 Conclusion

Donc on peut en conclure que la complexité des deux algos:

- Algo3 $\theta(x^{2.67})$
- Algo4 $\theta(x^{3.2})$

8 Mode d'emploi de l'interface *stable mariage calculator*

L'interface *stable mariage calculator* est une interface pour faciliter à résoudre le problème de mariage stable.

Exécution : à partir de chemin courant, taper **python3 interface.py**

Affichage :

Fonctionnalité:1) problème de mariage stable 2) problème de mariage stable séquentiel

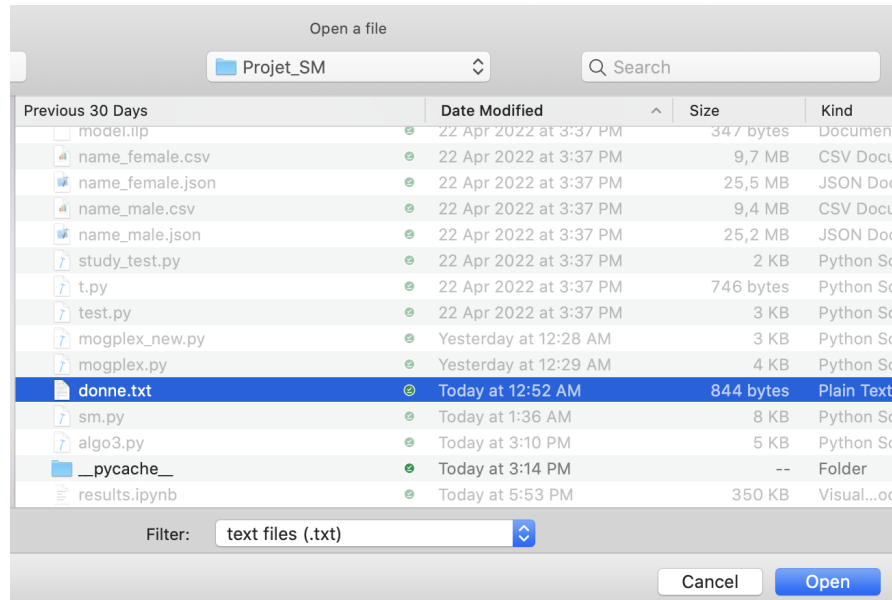
Données utilisés:1) listes des prénoms et listes de préférence fournis par défaut stable 2) données chargés depuis un fichier en cliquant sur ***Open a File***.

Format du fichier(si l'utilisateur veut utiliser ses propres données):

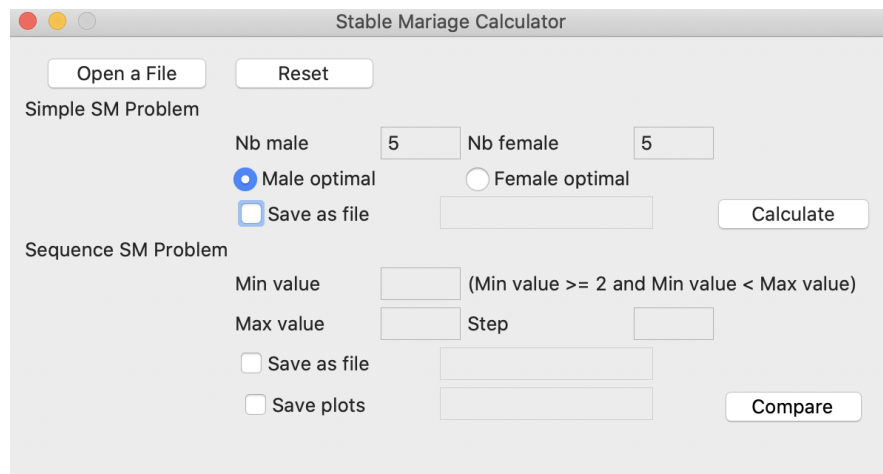
- 1) les deux premières lignes correspondant au nombre d'homme/femme
- 2) Pour les **nombre d'homme** lignes suivantes, ce sont les lignes pour caractériser la préférence des hommes: le premier élément est le nom du homme, et les éléments suivants sont les noms des femmes, classés par préférence décroissant.
- 3) Pour les **nombre de femme** lignes suivantes, de même que(2).

Exemple 1: Calculer les couples stables à partir du fichier

- Étape 1: ouvrir le fichier(Ici le fichier *donne.txt*)



- Étape 2: Remplir les cases *Nb male* et *Nb female* et choisir le côté à optimiser en crochant *male optimale* ou *female optimale*.



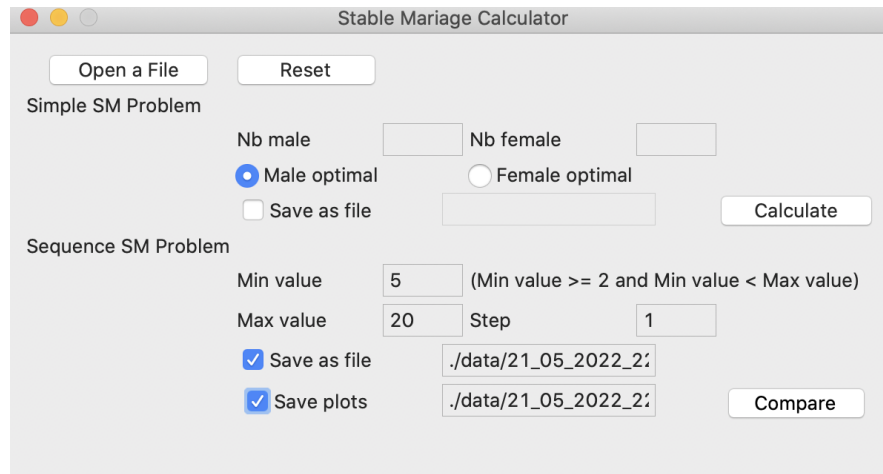
- Étape 3: Cliquer sur le bouton Calculate et l'affichage sur le terminal est bien les couples obtenus.

```
=====
Stable couples
[('Josh', 'Laurel'), ('Jamel', 'Brielle'), ('Reinaldo', 'Charolette'), ('Monte',
'Georgetta'), ('Jarrad', 'Estrella')]
```

Exemple 2: Comparer les différents algos pour le problème de mariage stable séquentiel

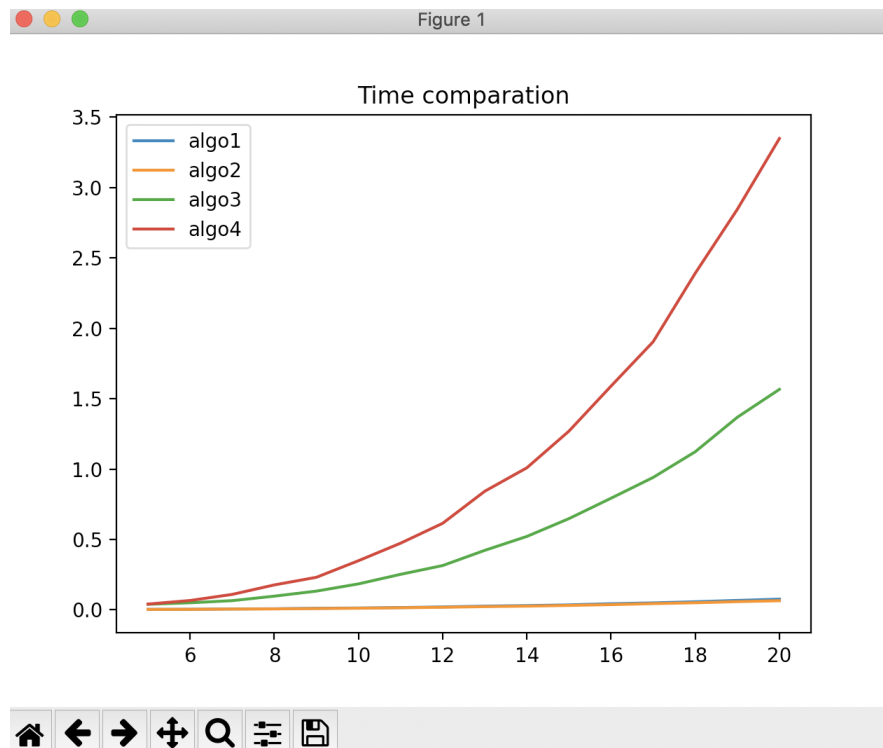
- Étape 1: Chosir le *valeur maximum*, *valeur mininum* et *pas*
Si vous voulez sauvegardez les données etou figures obtenus, crochez leles boxs

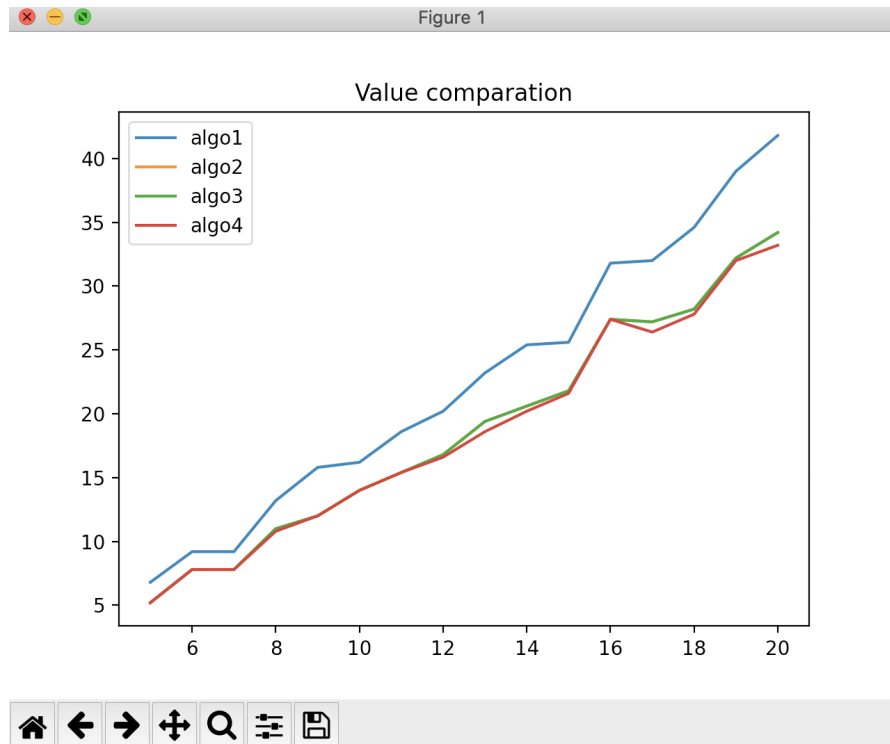
correspondantes.



The image shows a software window titled "Stable Marriage Calculator". It has two main sections: "Simple SM Problem" and "Sequence SM Problem". In the "Simple SM Problem" section, there are input fields for "Nb male" and "Nb female", radio buttons for "Male optimal" (selected) and "Female optimal", a checkbox for "Save as file", and a "Calculate" button. In the "Sequence SM Problem" section, there are input fields for "Min value" (5), "Max value" (20), and "Step" (1), with a note "(Min value >= 2 and Min value < Max value)". There are also checkboxes for "Save as file" and "Save plots" (both selected), and a "Compare" button. The file paths for saving are set to ".data/21_05_2022_21".

- Étape 2: Cliquer sur le bouton **Compare** et les courbes apparaissent.





9 Bibliographie

- [1]D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn, “Popular Matchings,” *SIAM Journal on Computing*, vol. 37, no. 4, pp. 1030–1045, 2007, doi: 10.1137/06067328X.
- [2]M. Baïou and M. Balinski, “Many-to-many matching: Stable polyandrous polygamy (or polygamous polyandry),” *Discrete Applied Mathematics*, vol. 101, pp. 1–12, Apr. 2000, doi: 10.1016/S0166-218X(99)00203-6.
- [3]D. Gale and M. Sotomayor, “Some remarks on the stable matching problem,” *Discrete Applied Mathematics*, vol. 11, no. 3, pp. 223–232, Jul. 1985, doi: 10.1016/0166-218X(85)90074-5.
- [4]M. M. Halldórsson et al., “Approximability results for stable marriage problems with ties,” *Theoretical Computer Science*, vol. 306, no. 1, pp. 431–447, Sep. 2003, doi: 10.1016/S0304-3975(03)00321-9.
- [5]R. W. Irving, “Stable marriage and indifference,” *Discrete Applied Mathematics*, vol. 48, no. 3, pp. 261–272, Feb. 1994, doi: 10.1016/0166-218X(92)00179-P.
- [6]P. Chebolu, L. A. Goldberg, and R. Martin, “The complexity of approximately counting stable matchings,” *Theoretical Computer Science*, vol. 437, pp. 35–68, Jun. 2012, doi: 10.1016/j.tcs.2012.02.029.
- [7]R. W. Irving, P. Leather, and D. Gusfield, “An Efficient Algorithm for the ‘Optimal’ Stable Marriage,” *J. ACM*, vol. 34, no. 3, pp. 532–543, Jul. 1987, doi: 10.1145/28869.28871.
- [8]D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, “Hard variants of stable marriage,” *Theoretical Computer Science*, vol. 276, no. 1, pp. 261–279, Apr. 2002, doi: 10.1016/S0304-3975(01)00206-7.
- [9]E. Ronn, “NP-complete stable matching problems,” *Journal of Algorithms*, vol. 11, no. 2, pp. 285–304, Jun. 1990, doi: 10.1016/0196-6774(90)90007-2.
- [10]K. Iwama, D. Manlove, S. Miyazaki, and Y. Morita, “Stable marriage with incomplete lists and ties,” in *In Proceedings of ICALP ’99: the 26th International Colloquium on Automata, Languages and Programming*, 1999, pp. 443–452.
- [11]C. Ng and D. S. Hirschberg, “Three-dimensional Stable Matching Problems,” *SIAM Journal on Discrete Mathematics*, vol. 4, pp. 4–245, 1991.
- [12]Á. Cseh and K. Heeger, “The stable marriage problem with ties and restricted edges,” *Discrete Optimization*, vol. 36, p. 100571, May 2020, doi: 10.1016/j.disopt.2020.100571.
- [13]A. Subramanian, “A New Approach to Stable Matching Problems,” *SIAM Journal on Computing*, vol. 23, no. 4, pp. 671–700, 1994, doi: 10.1137/S0097539789169483.
- [14]K. Iwama and S. Miyazaki, “A Survey of the Stable Marriage Problem and Its Variants,” in *International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008)*, 2008, pp. 131–136. doi: 10.1109/ICKS.2008.7.