Projet ANDROIDE
Mariage stable dynamique

May 8, 2022

Binome:
Xinyu Huang 3803966
Ruohui Hu 21102304
Muyang Shi 28714332

Professeur: M.Bruno Escoffier

Contents

Résumé

Nous avons emmm 0.0

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 trouvé une algorithme qui minimiser le changement de couples.

1.1 L'objective 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 consécutive), on veut aussi appliqué l'algo Gale-Shapley et programmation linéaire pour trouver le meilleux 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 probleme 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 lenéaire : 22 Février - 14 Mars
- Test, algorithme approché et exact, algorithme binary à relaxation continue 20 Mars - 5 Avril
- Tests les contraintes de stabilite dans Algo3 et Algo4,Analyse des courbes de temps,Complexite theorique : 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 . La rapport est écrit à l'aide de Overlef en Latex

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

```
def Gale_Shapley(female,male,pref_f,pref_m,opt=0):
"""
Entree : Deux ensembles finis Female et Male de cardinal n et n ;
         Deux dictionnaire de preference pref_f(cote femme) et pref_m(cote homme) ;
         opt: par defaut 0, qui decide quel cote a privilegier(opt=0,femme-optimal;
              sinon homme optimale)
Sortie : Un ensemble S de couples engages (homme ; femme) ;
"""
    # choose which side to optimize
    if opt==0:
            side_opt=female.copy()
            pref_opt=deepcopy(pref_f)
            side_des=male.copy()
            pref_des=deepcopy(pref_m)
        else:
            side_opt=male.copy()
            pref_opt=deepcopy(pref_m)
            side_des=female.copy()
            pref_des=deepcopy(pref_f)
    # The algos continue while it exists single people on the optimal side
    while side_opt:
        person=side_opt.pop()
        for person_partner in (pref_opt[person][1]):
            # when the partner is single, we form directely a couple
            if pref_des[person_partner][0]=="Single":
                pref_opt[person][0]=person_partner
                pref_des[person_partner][0]=person
                pref_opt[person][0]=person_partner
                break
            # when the partner is not single, we will compare with the competitor in
    the list of prefernce of partner
            else:
                competitor=pref_des[person_partner][0]
                pref_comp=pref_des[person_partner][1].index(competitor)
                pref_person=pref_des[person_partner][1].index(person)
                if pref_person<pref_comp:
                    pref_des[person_partner][0]=person
                    pref_opt[person][0]=person_partner
                    pref_opt[competitor][0]="Single"
                    side_opt.append(competitor)
                    break

    return [(x, y[0])  for x,y in pref_opt.items() if y[0]!='Single']
}
```

Listing 1: Gale-Shapley

3 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'algirithme 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

Fonction objective:

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

Contraintes:

- pour tout t, sum xij sur i ou j soit inf a 1 (mariage)
- $xijt >= 0$
- $xijt <= 1$
- Stabilité

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

4 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'algirithme 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 xij sur i ou j soit inf a 1 (mariage)
- $zijt >= xijt - xijt - 1$
- $zijt >= 0$

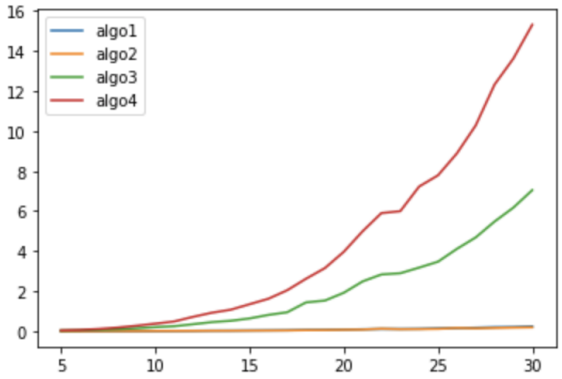
- $xijt >= 0$
- $xijt <= 1$
- Stabilité

Advantage:
1)Comparer à l'algo itérative, résoudre le problem en une seule itération 2)R2soudre le problèm de façon gloable en minimisant la difference gloable

5 Analyse des algorithmes

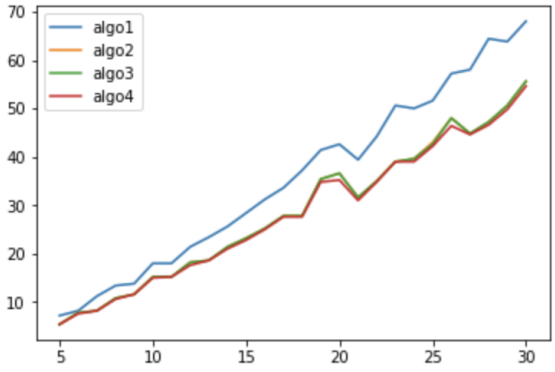
Dans notre projet, on se concentre sur un cas particulier pour le problème de mariage stable séquentiel: à chaque instant t, on fixe le nombre de personne d'un côté(par exemple femme) à valeur maximum, et on incrémente le nombre de personne de l'auntre côté au cours du temps.À chaque pas de temps, on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

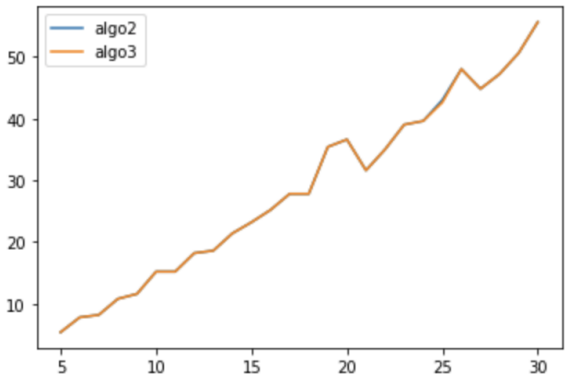
5.1 Temps de calcul



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 prends beaucoup plus de temps que algo1/2, et algo4 coût plus de temps, qui est à peu près deux fois de plus que celle de l'algo3.

5.2 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 changement de couples entre deux t différents.

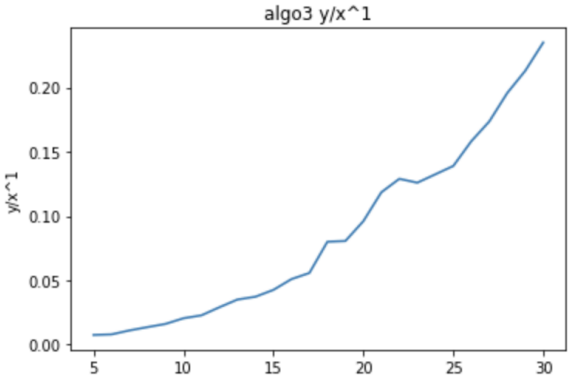
Remarque 2 : À partir de la figure de gauche, on peut aussi remarquer que la courbe de algo3 est très proche de cela de algo4, qui est logique car les deux algos prennent en compte à minimiser la différence entre t consécutive. Et algo4 est une borne inférieure de algo3 car il minimise les changements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

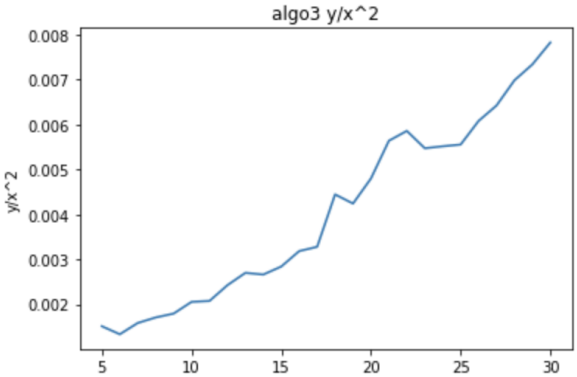
Remarque 3 : On ne peut pas trouver la courbe à partir de la courbe de gauche. En faisant la commande "print", j'ai remarqué que la valeur de algo2 est très proche de la valeur de algo3(une seule différence dans notre cas est quand n=25, algo2 est pire que algo3. C'est une remarque un peu magique mais 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 limité seulement le côté femme), c'est pourquoi ils sont proches.

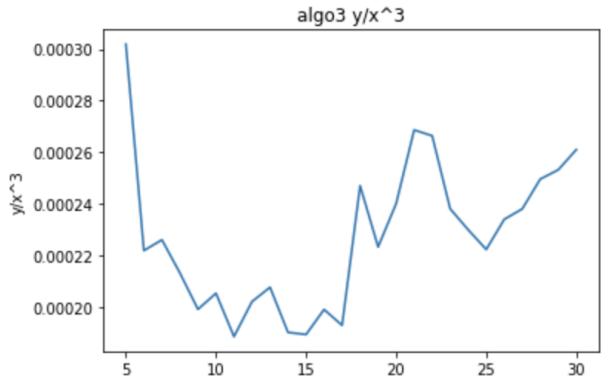
5.3 Complexité en temps de calcul

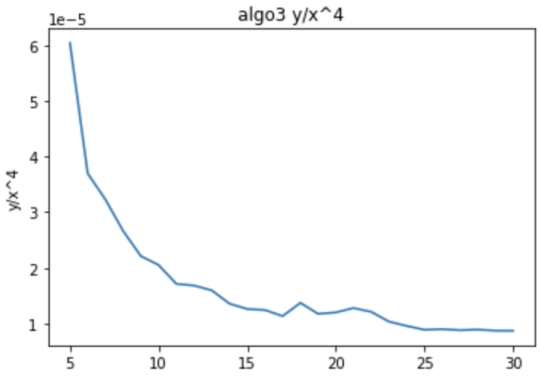
5.3.1 Interpolation:division par la fonction devinnée

Interpolation pour l'algo3:

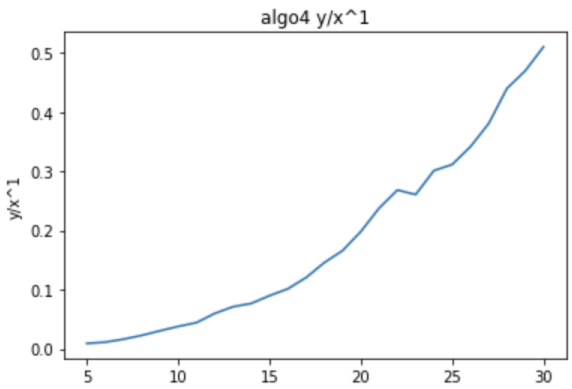


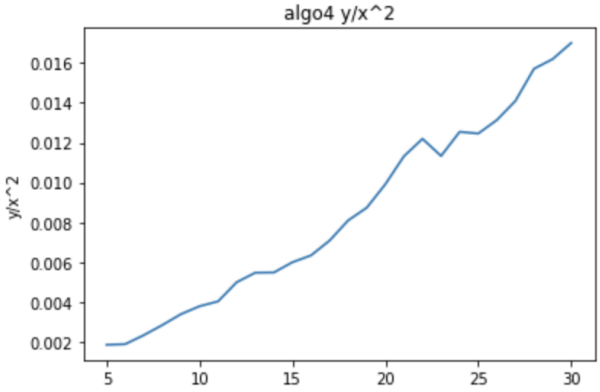


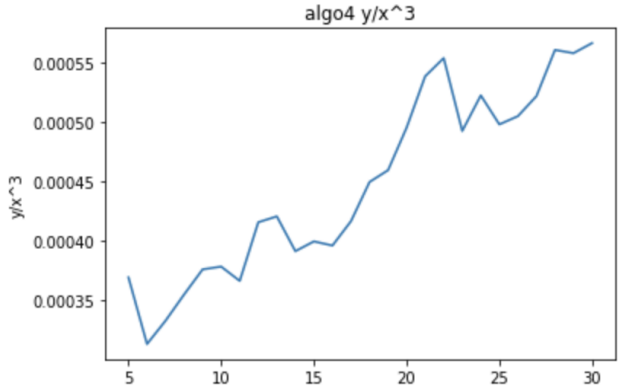


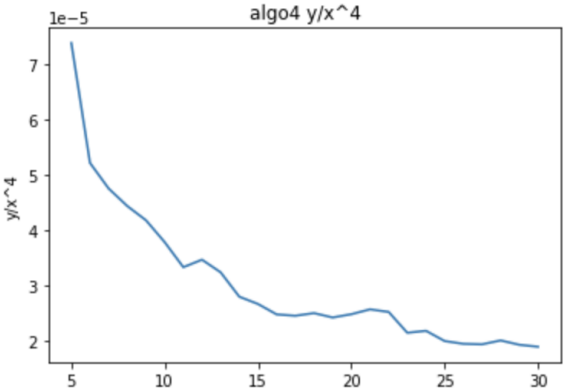


Interpolation pour l'algo4:

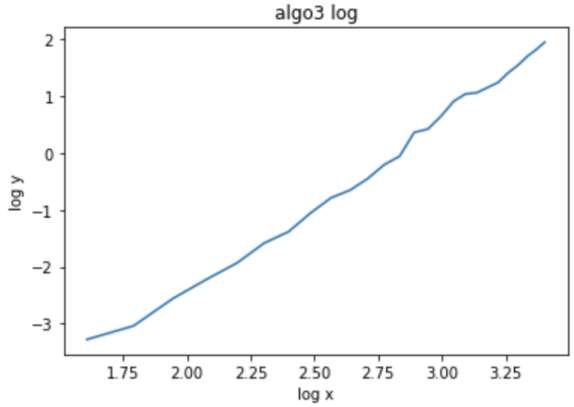


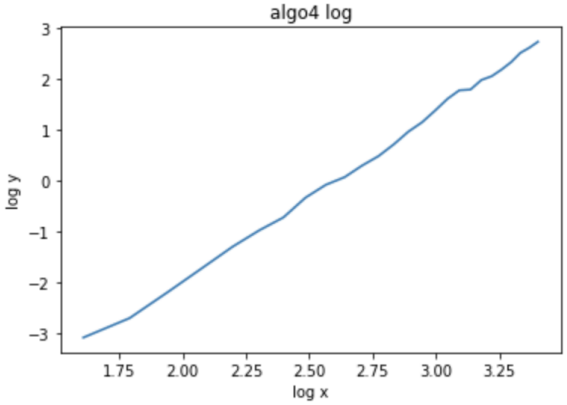






5.3.2 Interpolation:passer à log





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) $Gradient = 4 \div 1.25 = 3.2$

5.4 Conclusion

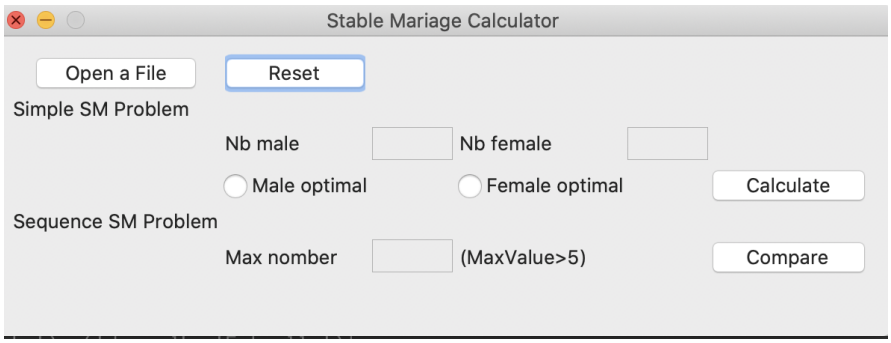
Donc on peut en conclure que la complexité des deux algos:

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

6 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 :

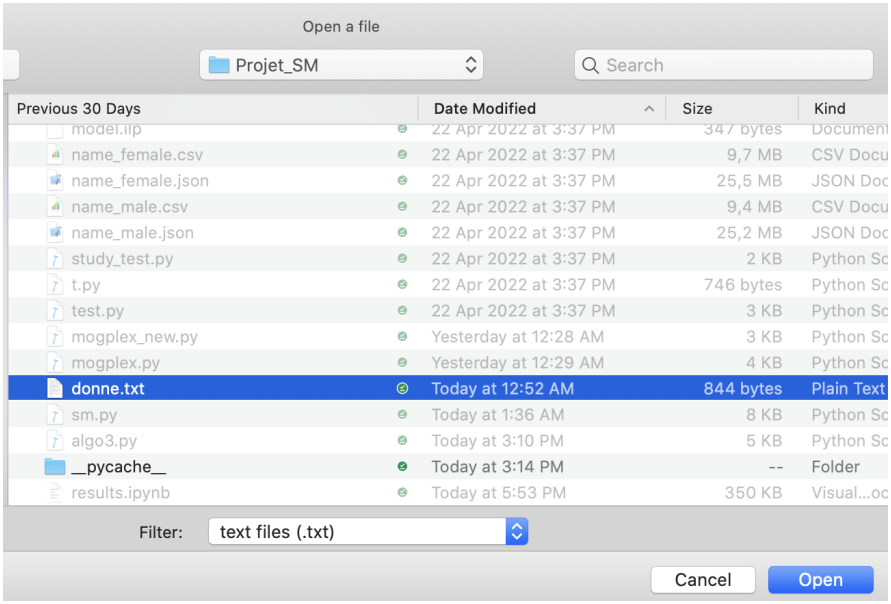


Fonctionnalité:1) problème de mariage stable 2) problème de mariage stable séquentiel

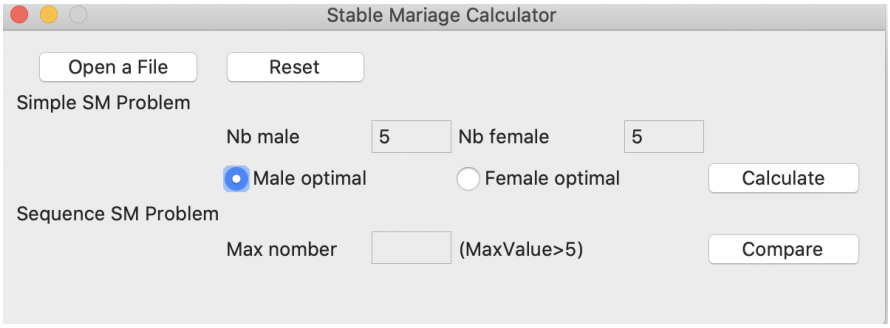
Données utilisés:1) listes des prénoms et listes de préférence fournis par défaut stable 2) données chargés depuis un ficher en cliquant sur ***Open a File***.

Exemple 1: Calculer les couples stables à partir du fichier

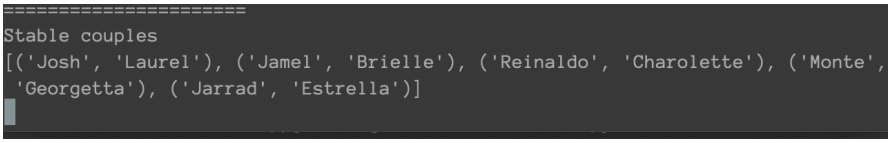
- Étape 1: ouvrir le fichier(Ici le fichier *donne.txt*)



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

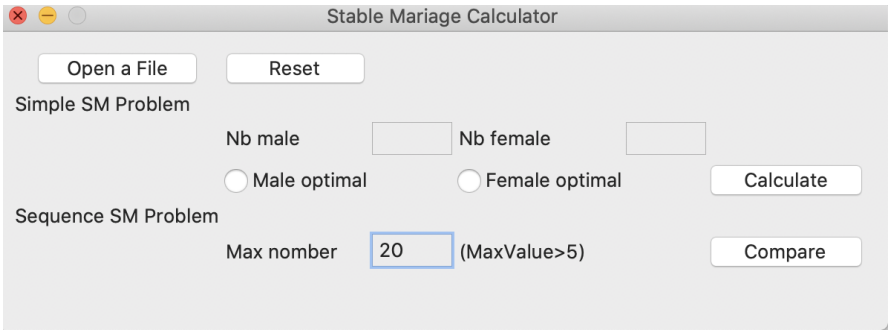


- Étape 3: Cliquer sur le bouton Calculate et l'affichage sur le terminal est bien les couples obtenus.

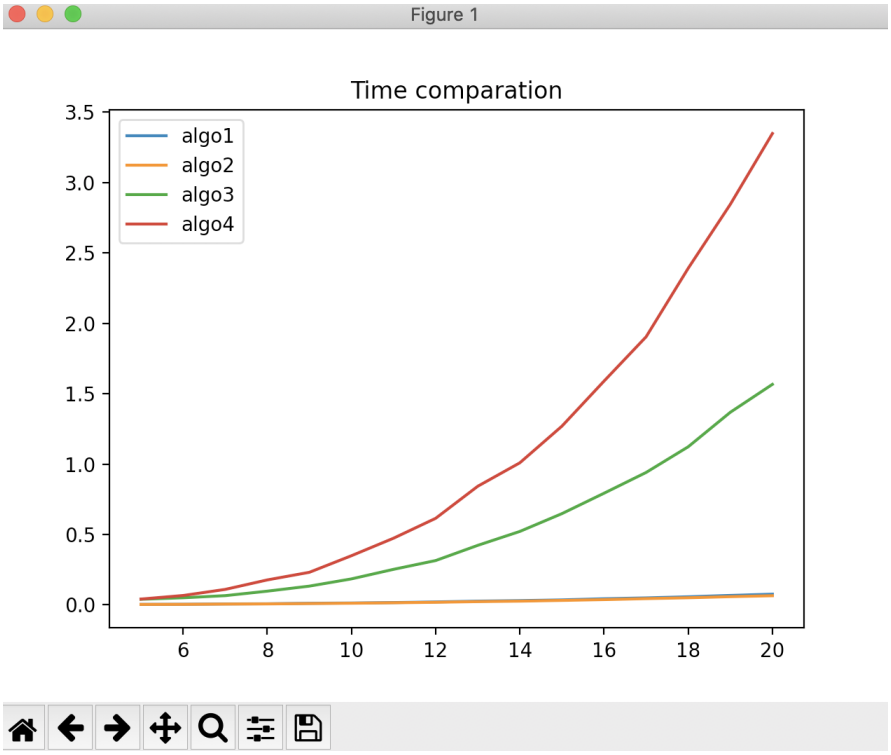


Exemple 2: Comparer les différents algos pour le problème de mariage stable séquentiel

- Étape 1: Chosir le ***valeur maximum***



- Étape 2: Cliquer sur le bouton ***Compare*** et les courbres appraissent.



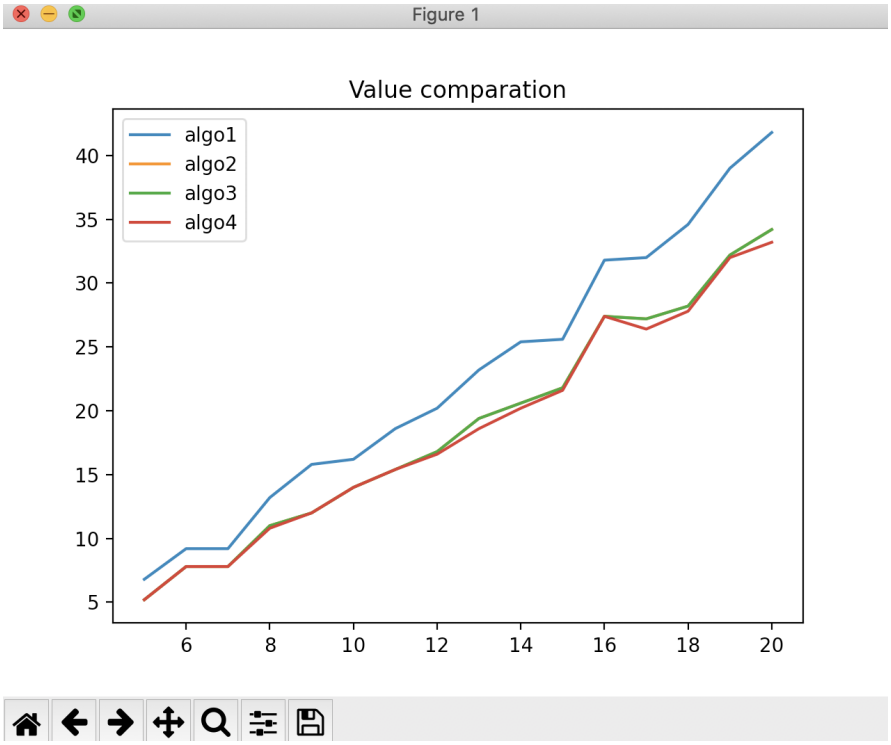


Figure 1
Value comparation
algo1
algo2
algo3
algo4
40
35
30
25
20
15
10
5
6
8
10
12
14
16
18
20

7 Annexe