Jeu inspiré du film Hunger Games

Être le dernier survivant

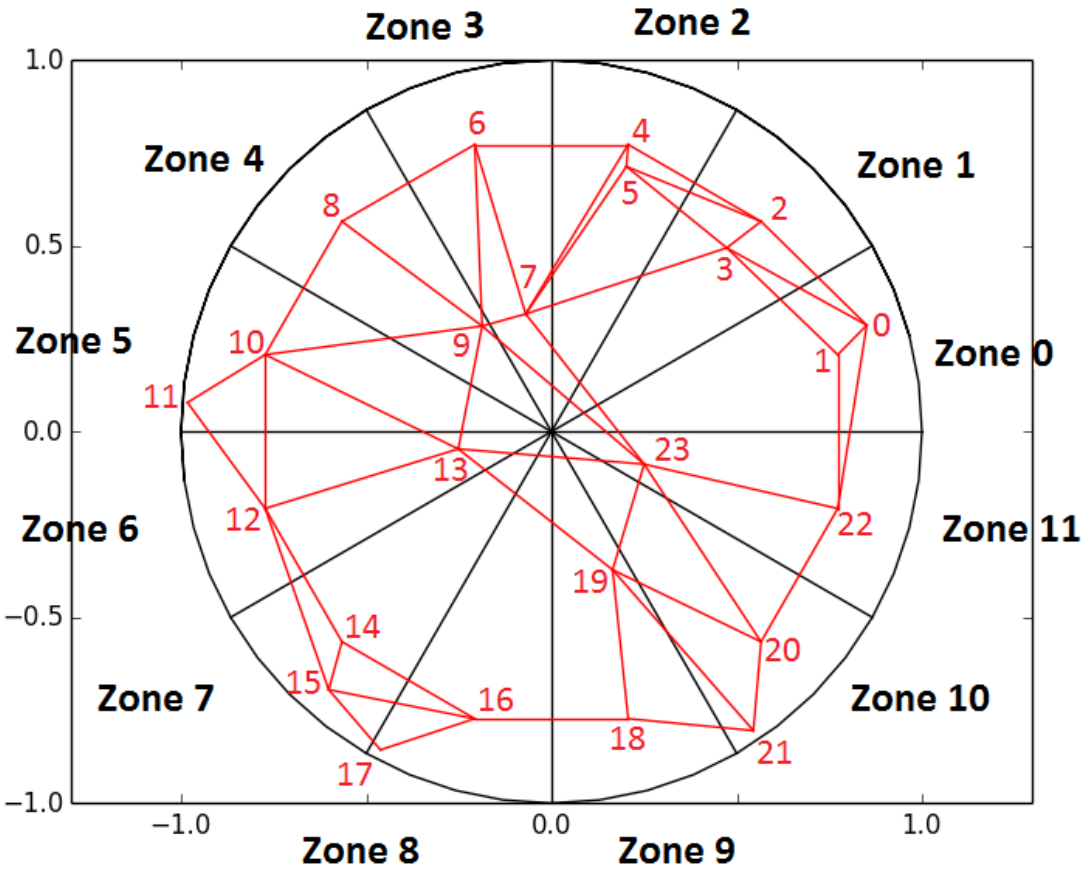
Principe du Jeu

- **Règles du jeu :** les individus sont placés dans une arène circulaire parcourue de chemins. Pour survivre, ils doivent :

  - avoir des réserves énergétiques : chasser des proies

  - combattre ou fuir les autres individus

  Le dernier survivant a gagné.



- **Simulation :** mise en place de plusieurs stratégies automatiques adaptées aux capacités de l'individu.

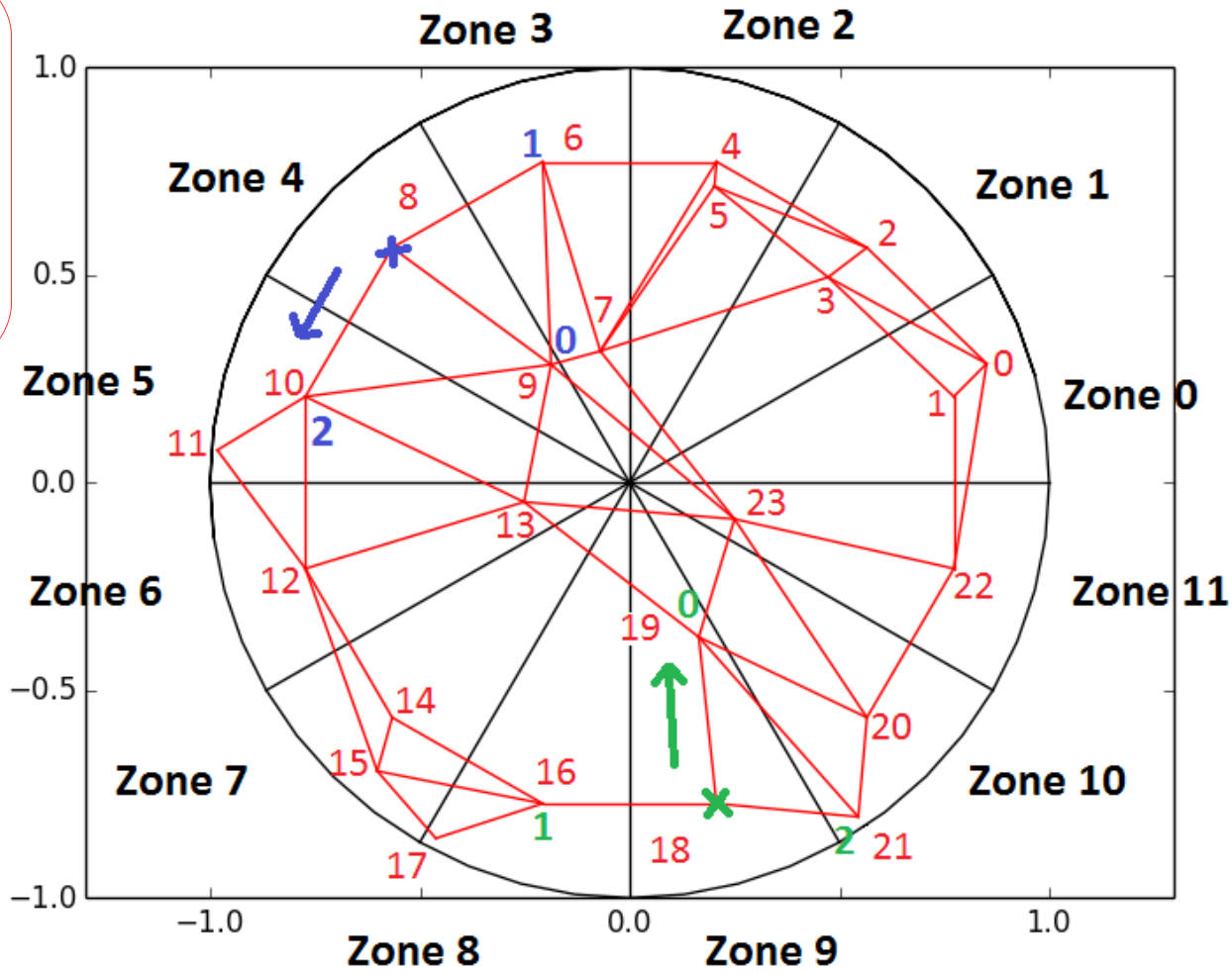
Principales variables

Arène représentée par un graphe connexe :

Nœuds : coordonnées (x,y) mémorisées dans une liste de listes L_coord_noeuds
Chemins : matrice M_connexe contenant :
- 0 si nœuds non reliés
- 1 si nœuds reliés

Individus représentés par une liste de listes :

L_individus : contient les listes individu ayant les caractéristiques des individus
individu[iNoeud] : numéro du nœud où se situe l'individu
individu[iEnergie] : réserve énergétique
individu[iForce] : capacité de force
→ stratégie de combattant
individu[iVitesse] : capacité de vitesse
→ stratégie de fuite
individu[iHabileté] : capacité de chasse
individu[iSante] : état mort ou vivant

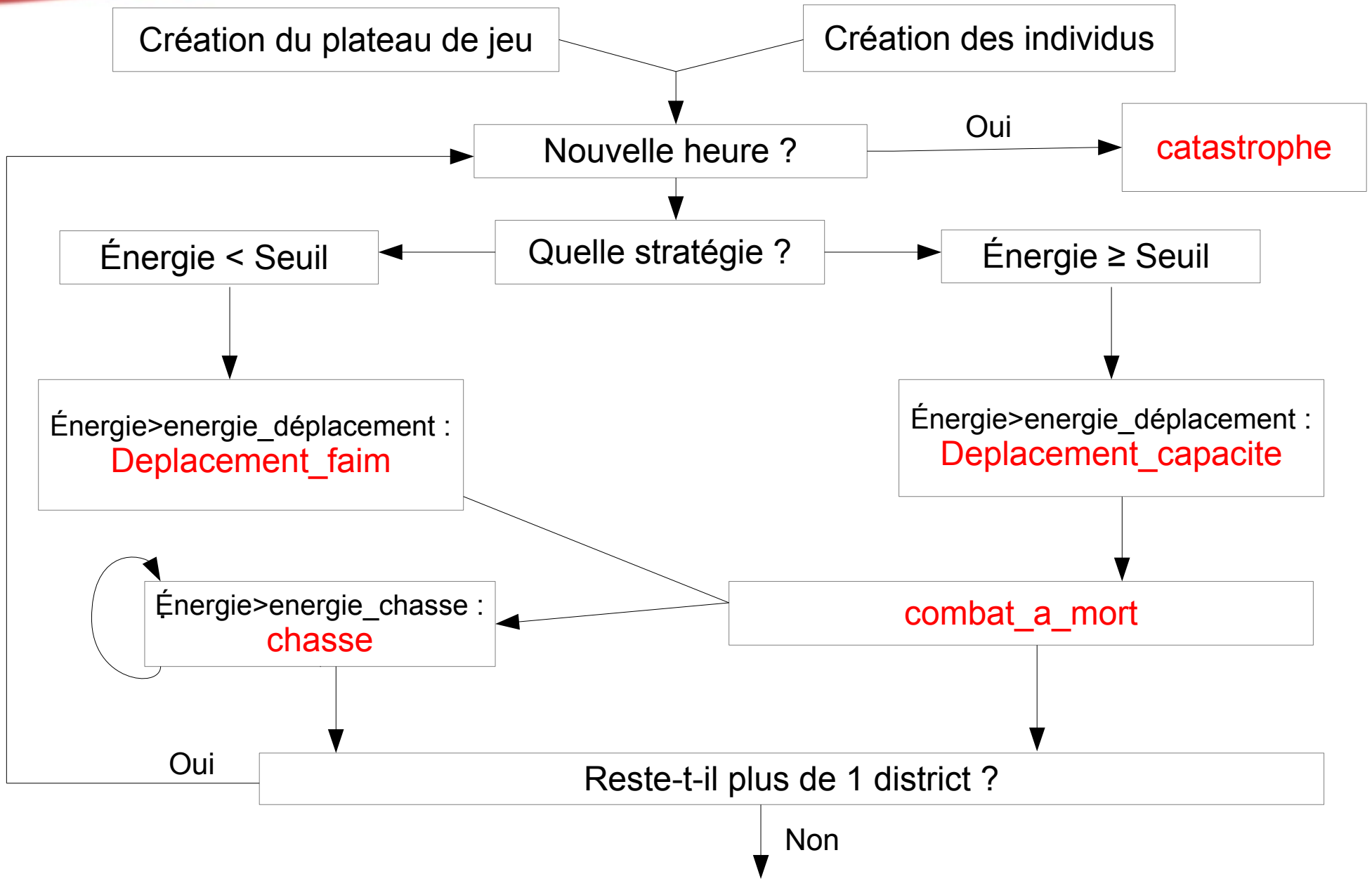


Combattant

Fuite

Exemple : liste_individus=[[8,27,**0.75**,0.25,0.8,'vivant',...],[18,4,0.2,**0.8**,0 .6,'vivant',...]]

Découpage fonctionnel



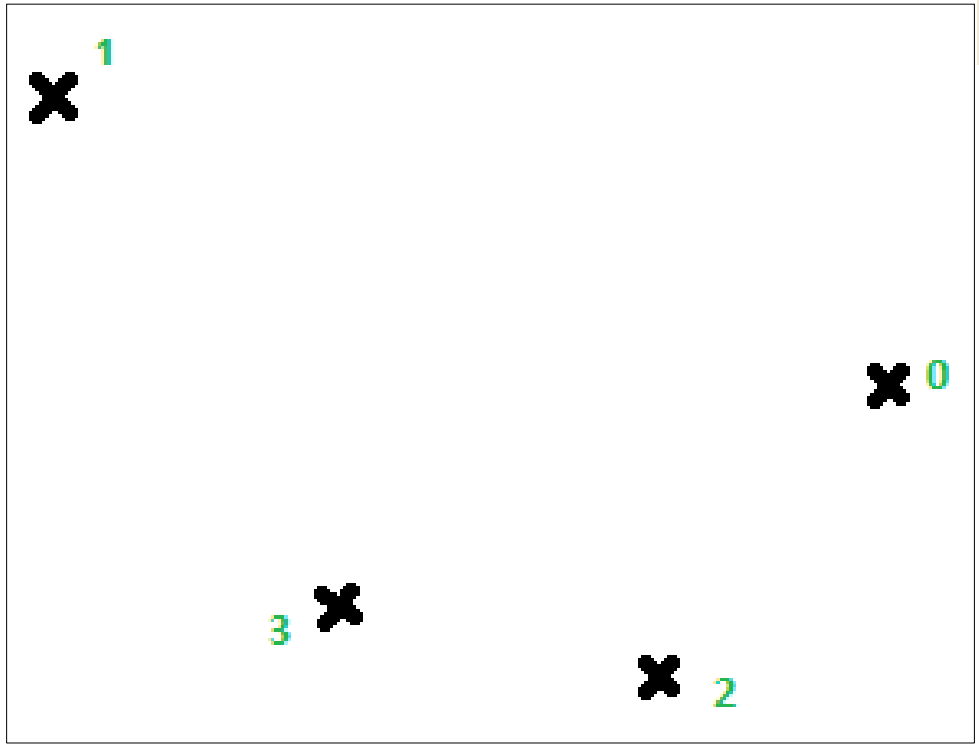
Victoire du dernier survivant ou d'un district OU défaite de tous les individus

Exemple de fonction pour la création de chemins

```
23
24  def aretes_possibles(L_coord_noeuds,d,p):
25      ''' Répertoire des arêtes que l'on peut tracer de taille inférieure ou égale à d '''
26      M_aretes=np.matrix(np.zeros((p,p)))
27      for i in range(p-1):
28          for j in range(i+1,p):
29              if distance_entre_points(L_coord_noeuds[i],L_coord_noeuds[j])<=d:
30                  M_aretes[i,j]=1
31      return M_aretes
32
```

	0	1	2	3
0	0	1	1	1
1	0	0	1	1
2	0	0	0	1
3	0	0	0	0

M_aretes



```
33  def aretes_true(L_coord_noeuds,d,p):
34      M_aretes=aretes_possibles(L_coord_noeuds,d,p)
35      M_connexe=np.matrix(np.zeros((p,p)))
36      while la.matrix_rank(M_aretes)!=0: Oui
37          i=rd.randint(0,p-2)  i = 1
38          j=rd.randint(i+1,p-1) j = 2
39          if M_aretes[i,j]==1: Oui
40              M_aretes[i,j]=0
41              L_aretes1=2*[0]
42              L_aretes1[0],L_aretes1[1]=L_coord_noeuds[i],L_coord_noeuds[j]
43              Tracable=True
44              for h in range(p-1):
45                  for k in range(h+1,p):
46                      if M_connexe[h,k]==1:  Non
47                          L_aretes2=2*[0]
48                          L_aretes2[0]=L_coord_noeuds[h]
49                          L_aretes2[1]=L_coord_noeuds[k]
50                          if aretes_se_croisent(L_aretes1,L_aretes2):
51                              Tracable=False
52              if Tracable: True
53                  M_connexe[i,j]=1
54      return M_connexe
```

M_aretes

0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

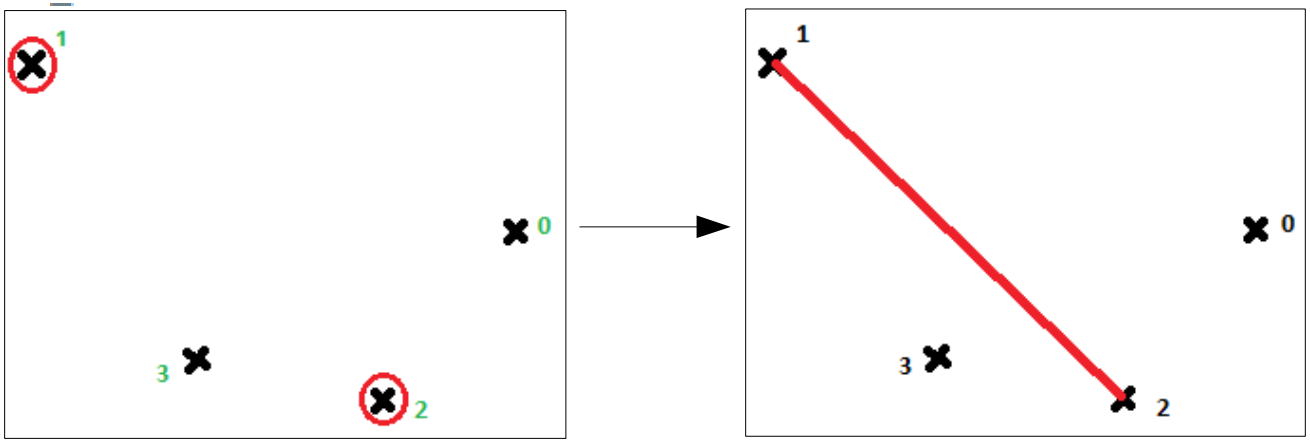
→ 0

M_connexe

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

↓

0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0



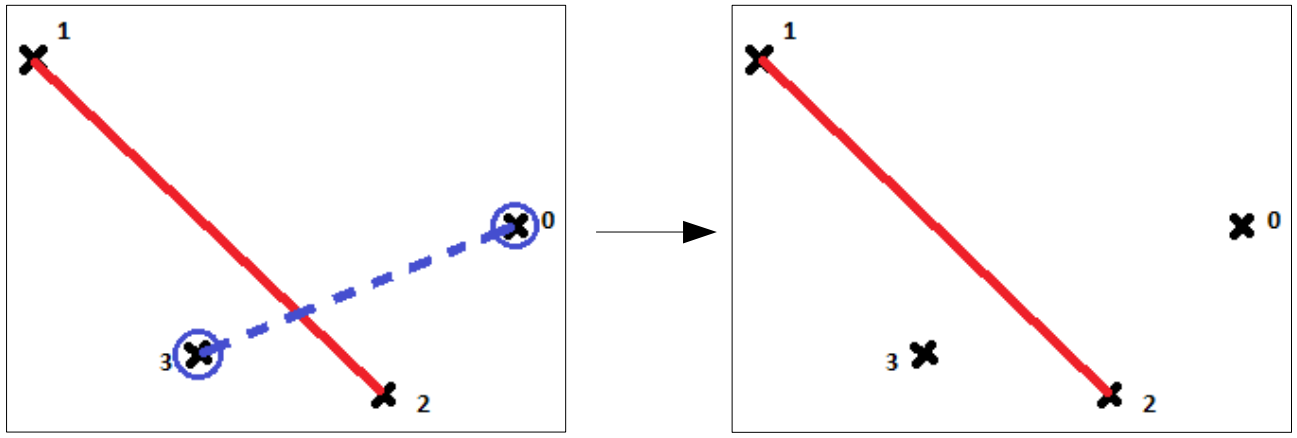
```
33  def aretes_true(L_coord_noeuds,d,p):
34      M_aretes=aretes_possibles(L_coord_noeuds,d,p)
35      M_connexe=np.matrix(np.zeros((p,p)))
36      while la.matrix_rank(M_aretes)!=0: Oui
37          i=rd.randint(0,p-2)   i = 0
38          j=rd.randint(i+1,p-1) j = 3
39          if M_aretes[i,j]==1: Oui
40              M_aretes[i,j]=0
41              L_aretes1=2*[0]
42              L_aretes1[0],L_aretes1[1]=L_coord_noeuds[i],L_coord_noeuds[j]
43              Tracable=True
44              for h in range(p-1):
45                  for k in range(h+1,p): Oui pour i = 1 et j = 2
46                      if M_connexe[h,k]==1:
47                          L_aretes2=2*[0]
48                          L_aretes2[0]=L_coord_noeuds[h]
49                          L_aretes2[1]=L_coord_noeuds[k]
50                          if aretes_se_croisent(L_aretes1,L_aretes2):Oui
51                              Tracable=False
52              if Tracable: False
53                  M_connexe[i,j]=1
54      return M_connexe
```

M aretes

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow 0$$

M_connexe

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$



Mises en perspective

- Répartition du travail :

  1) 2 personnes pour créer le plateau de jeu et 1 personne pour les fonctions concernant les individus

  2) Mise en commun pour l'écriture du script final et complexification réalisée à 3

- Critiques et amélioration :

  Graphe pas toujours connexe quand la longueur maximale pour une arête est trop petite.

  Le manque de temps nous a empêché de réaliser toutes les complexifications que nous avions envisagées

- Chiffres :

  Nombres de séances : 15

  Nombres de lignes : 97 (script) et 404 (fonctions)