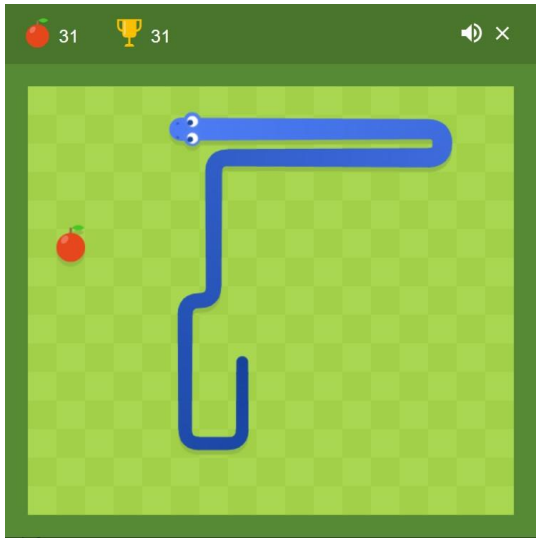
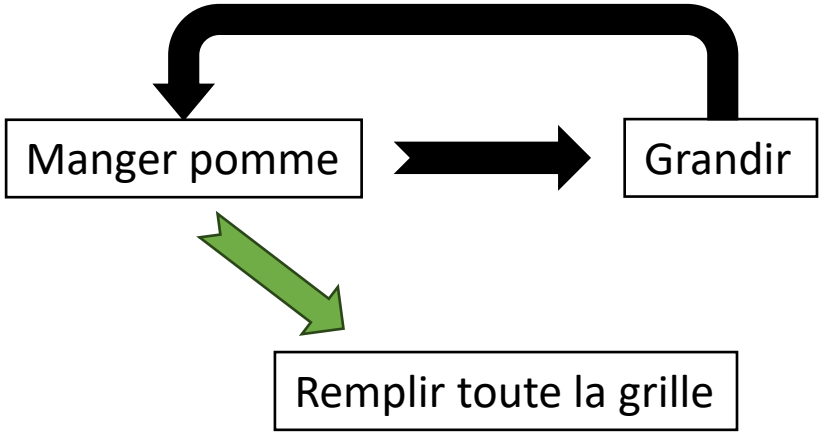
Comment gagner à coup sûr et de manière optimale au jeu Snake?

YLLOUZ Olivier – n°33867

Présentation du jeu





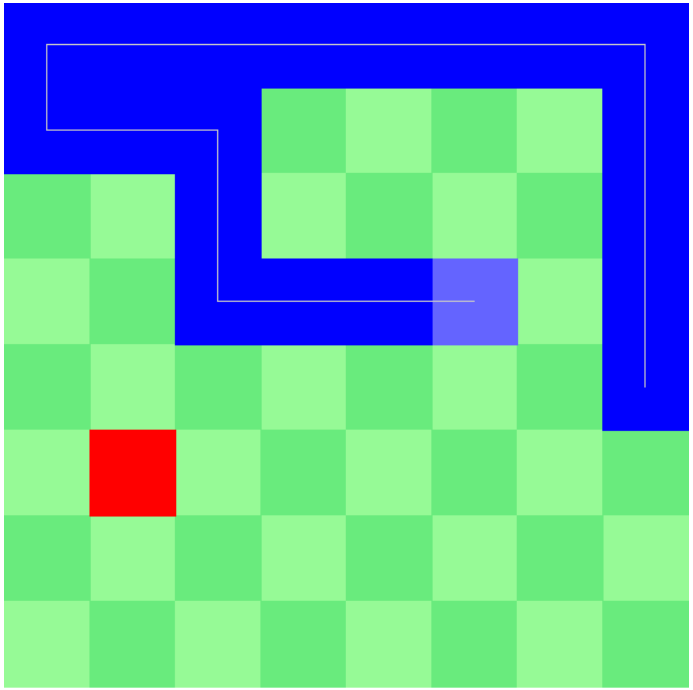
Images du jeu Snake de Google

- Grille discrète
- Temps discret

Objectifs

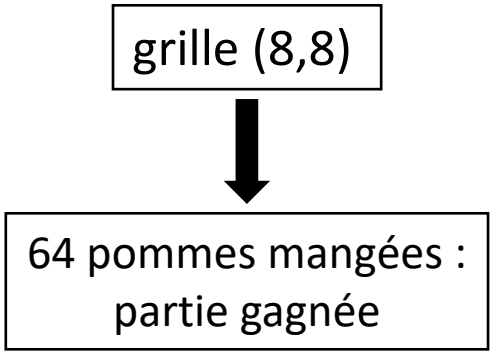
- Se rapprocher de la méthode de résolution la plus fiable et efficace possible
- Pas de programme en temps exponentiel
- Pas d'IA : maîtrise des méthodes

Programmation et terminologie



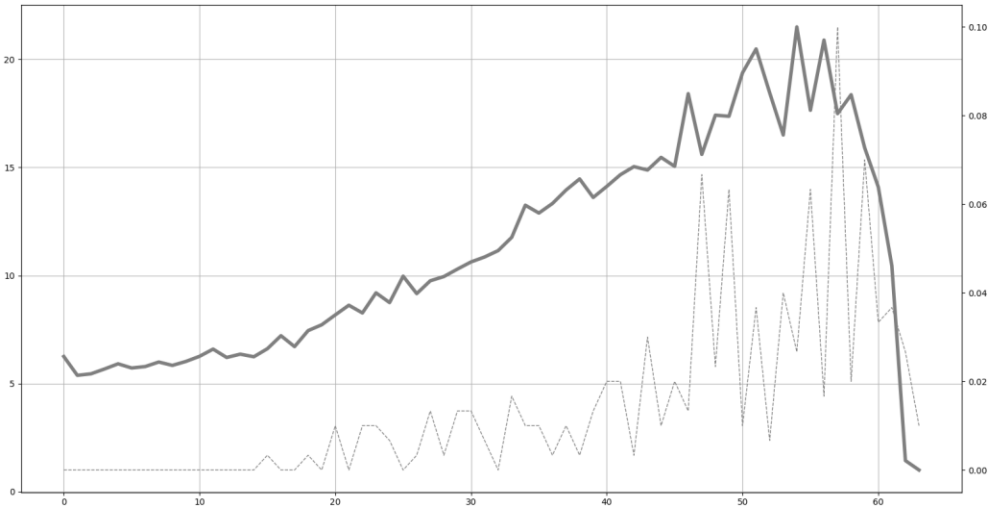
Méthodes présentées

- Plus court chemin
- Suivi de cycle Hamiltonien
- Pomme-queue
- Raccourcis sur cycle Hamiltonien
- Raccourcis sur cycle Hamiltonien 1D
- Plus court chemin sur cycle dynamique

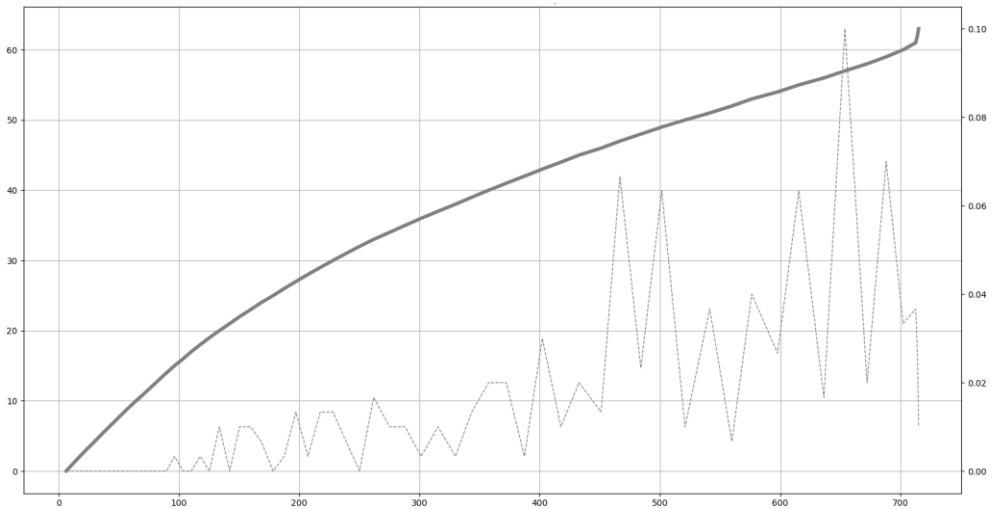


Comment juger une méthode ?

Nombre de pas en fonction du score

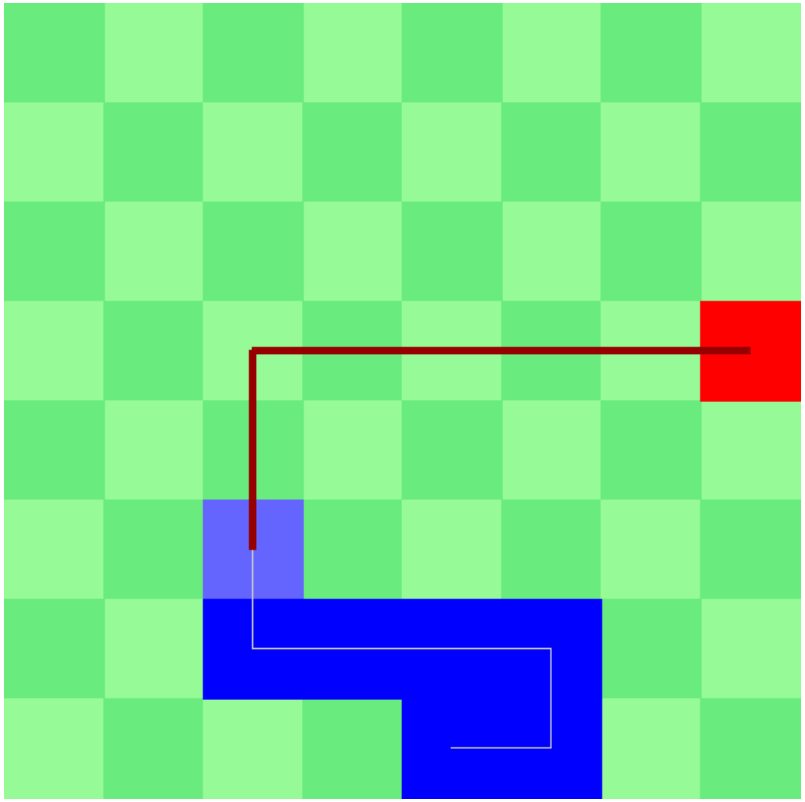


Score en fonction du nombre de pas total



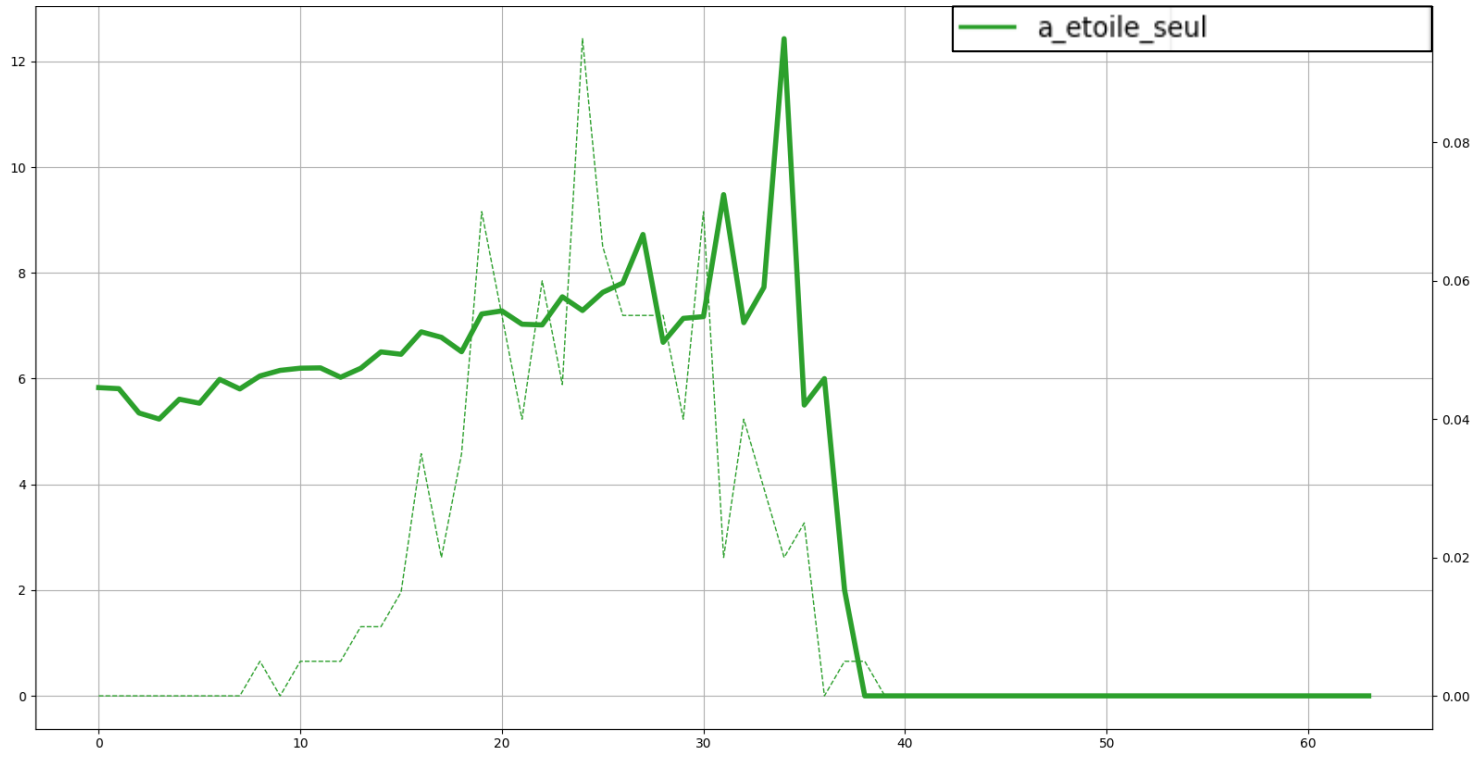
Plus court chemin Présentation

-**Plus court chemin**
-Suivi de cycle Hamiltonien
-Pomme-queue
-Raccourcis sur cycle Hamiltonien
-Raccourcis sur cycle Hamiltonien 1D
-Plus court chemin sur cycle dynamique



Plus court chemin
Résultats

200 parties

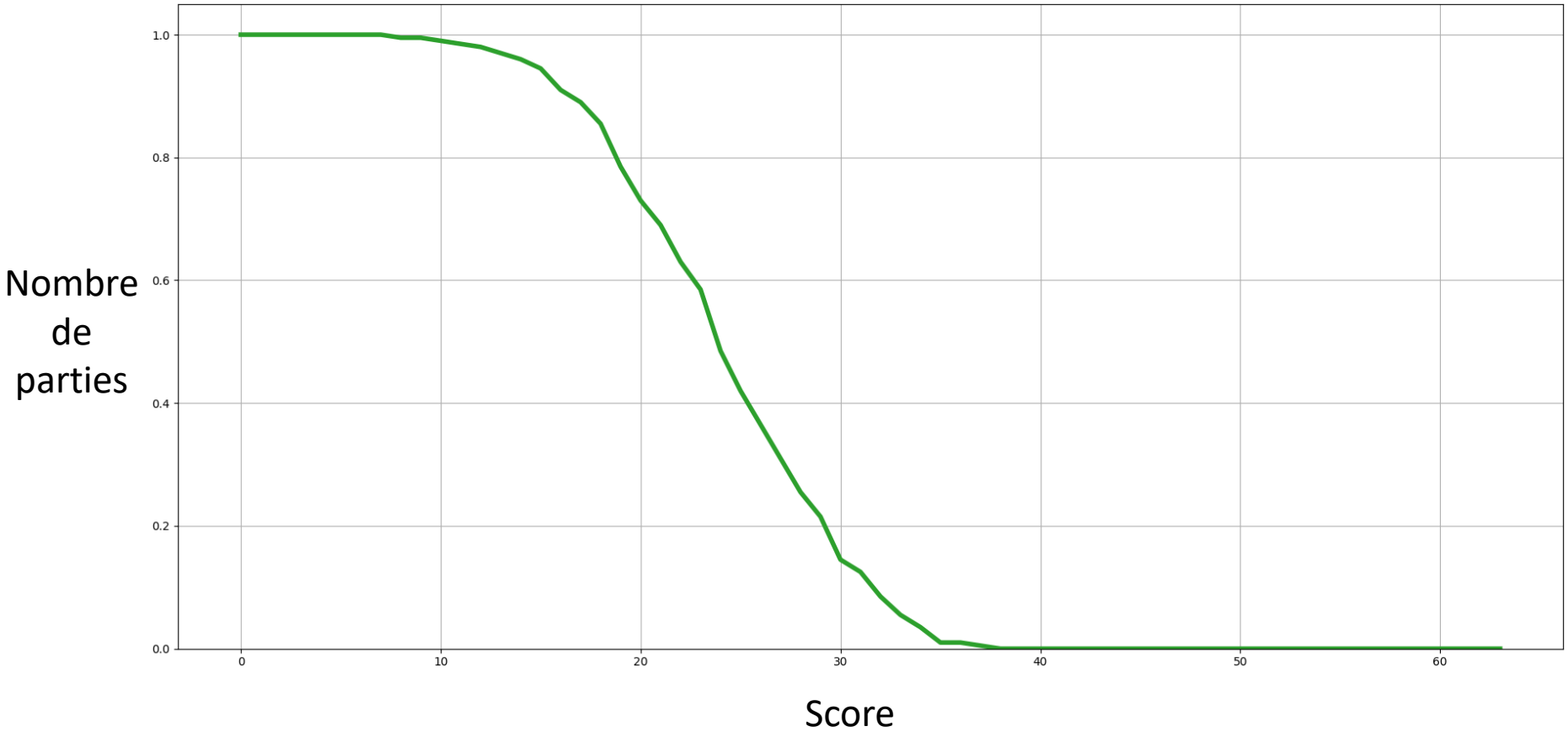


Nombre de pas

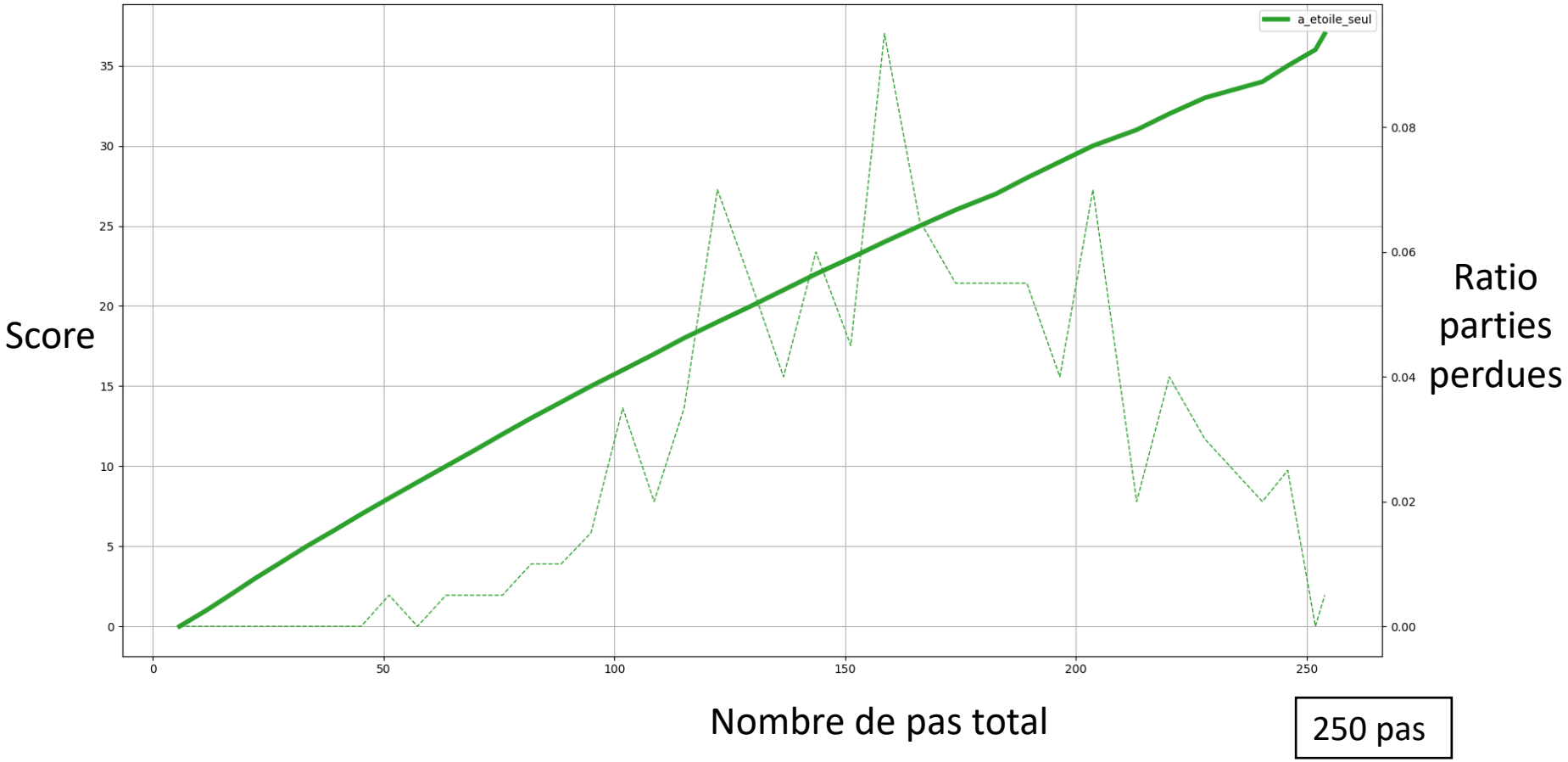
Ratio parties perdues

Score

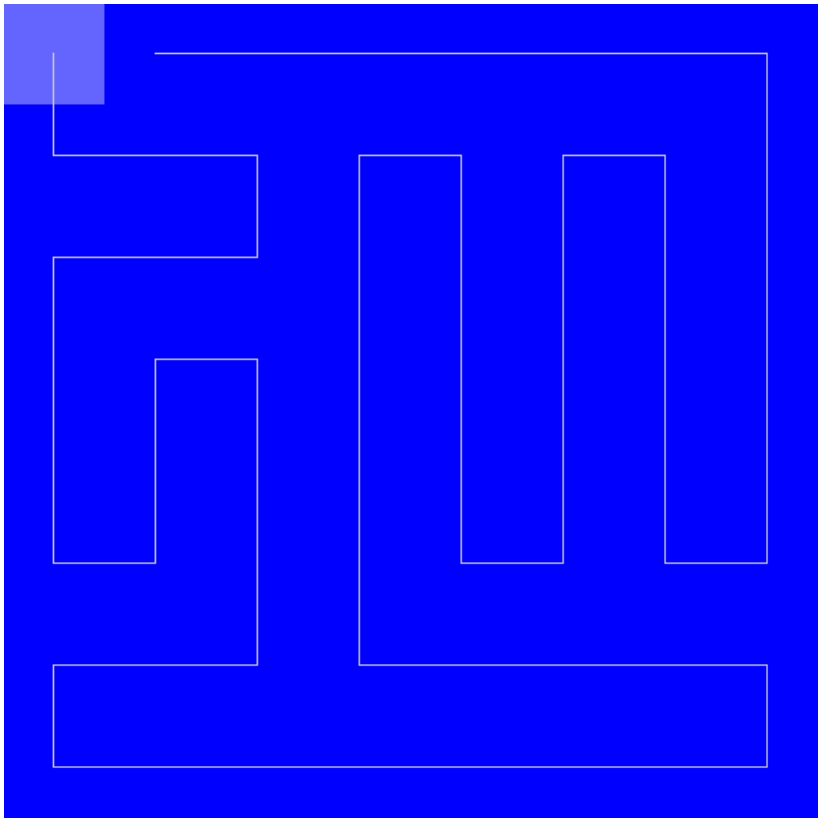
Plus court chemin
Résultats



Plus court chemin
Résultats



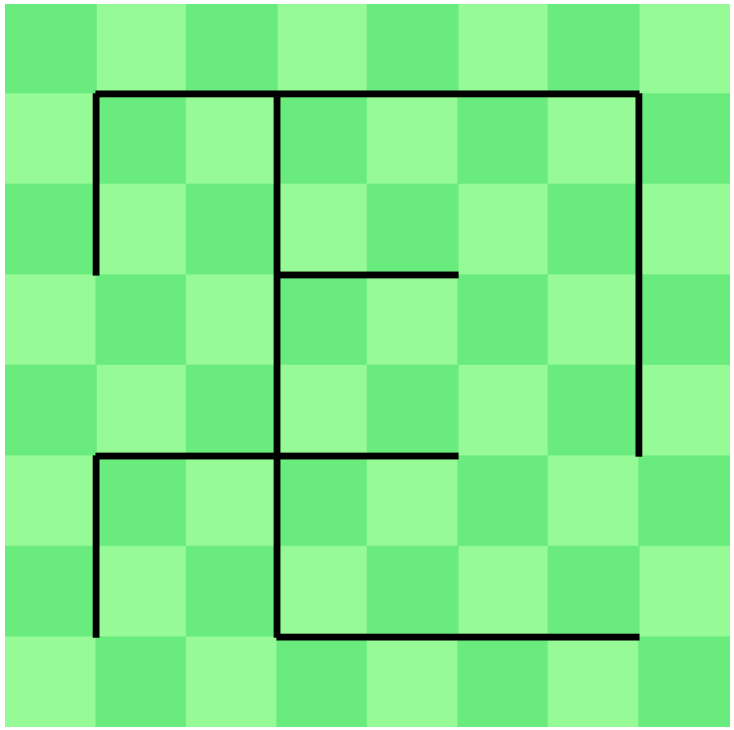
Comment être certain de gagner ?

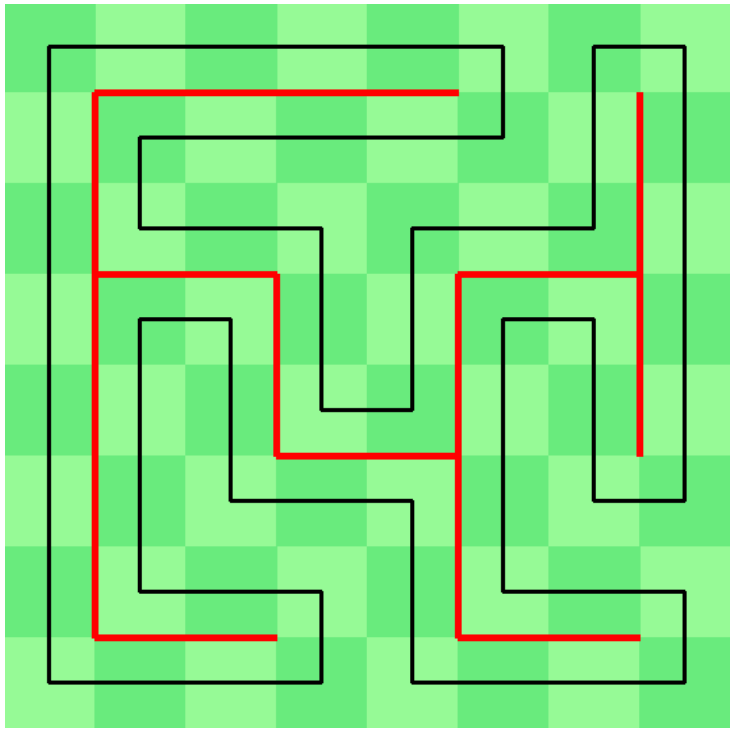


Cycle Hamiltonien

→ Chemin Hamiltonien trop peu probable

Génération de cycles Hamiltoniens





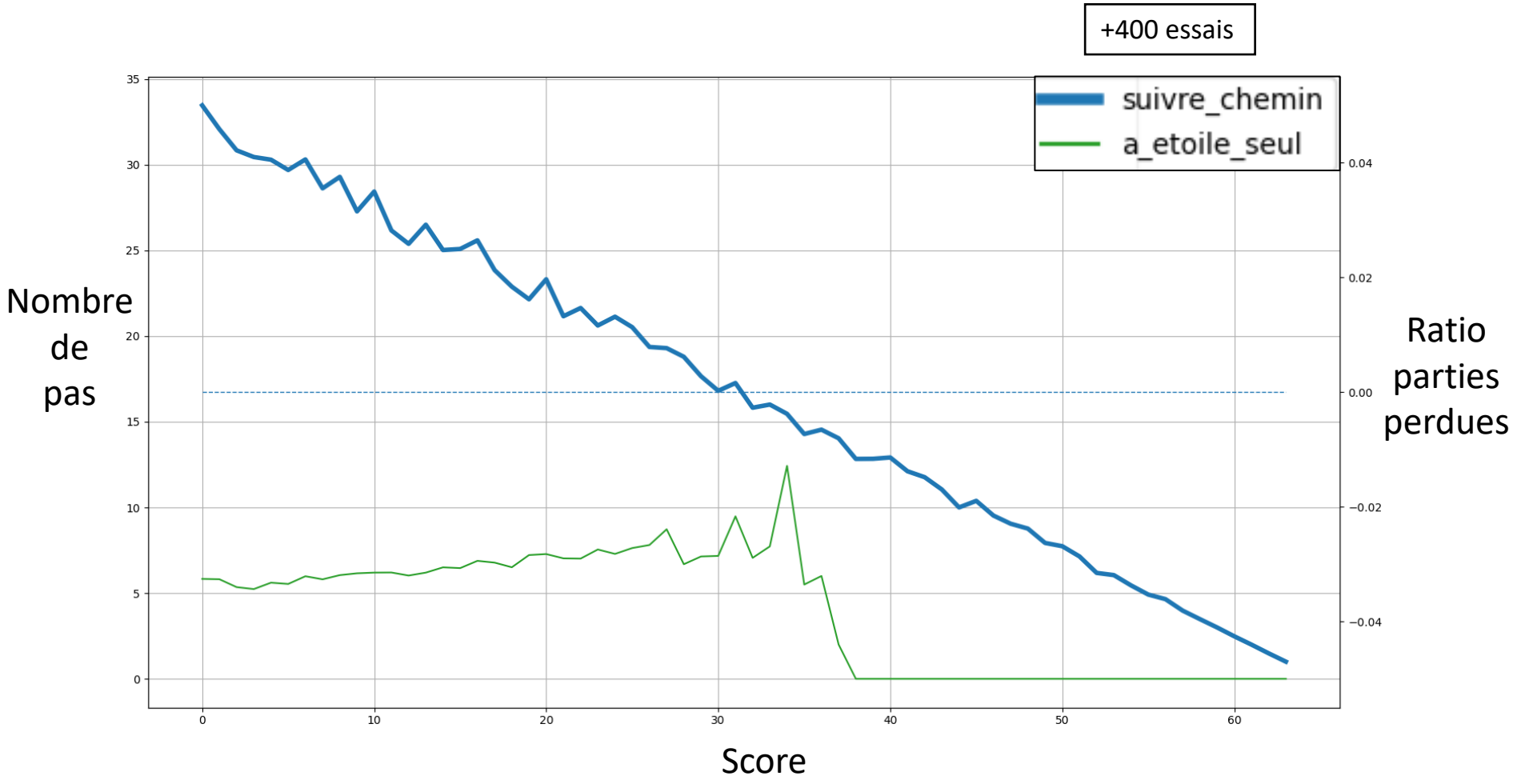
Cycles « murables »

Suivi de cycle Hamiltonien Présentation

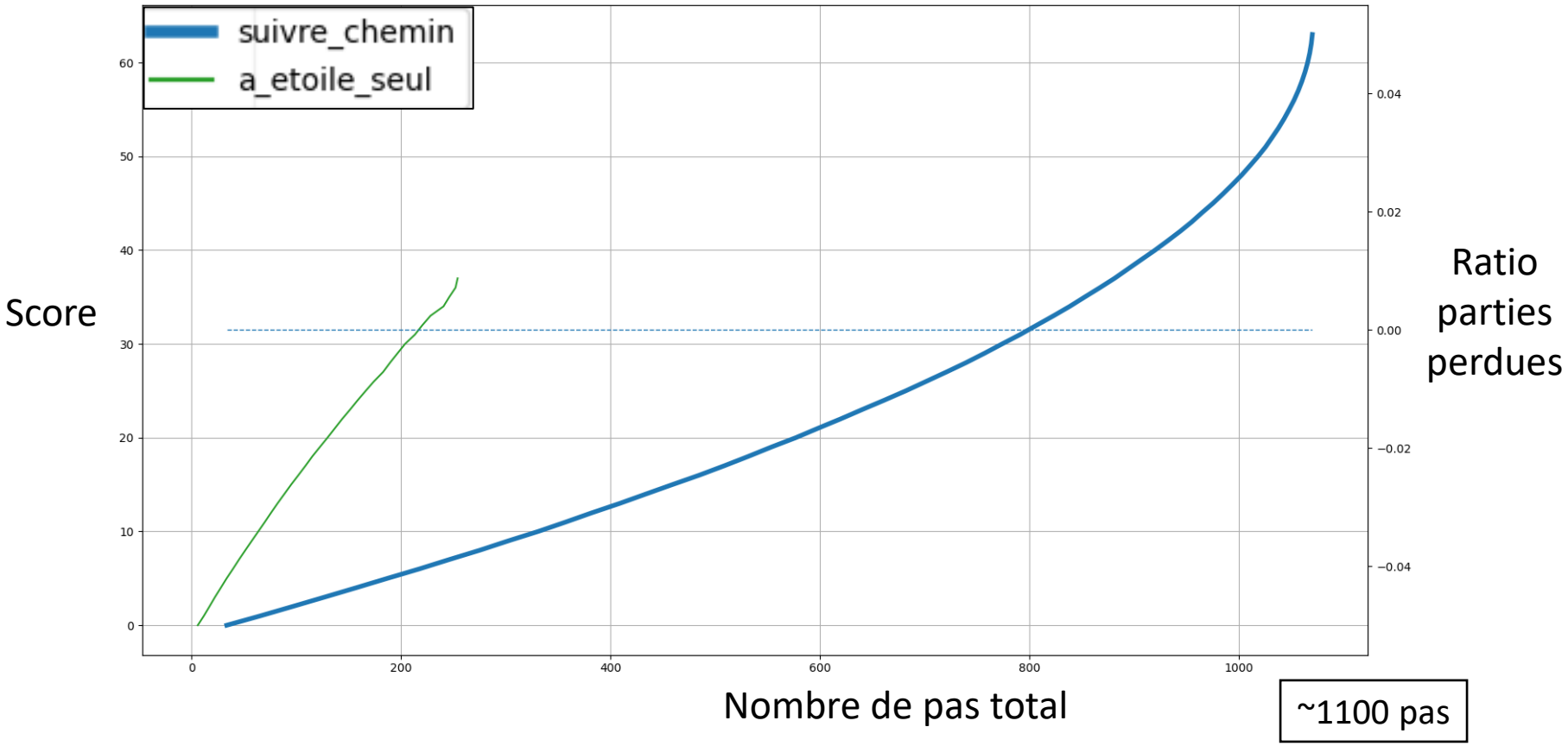
- Plus court chemin
- **Suivi de cycle Hamiltonien**
- Pomme-queue
- Raccourcis sur cycle Hamiltonien
- Raccourcis sur cycle Hamiltonien 1D
- Plus court chemin sur cycle dynamique

Suivi de cycle Hamiltonien
Résultats

+400 essais

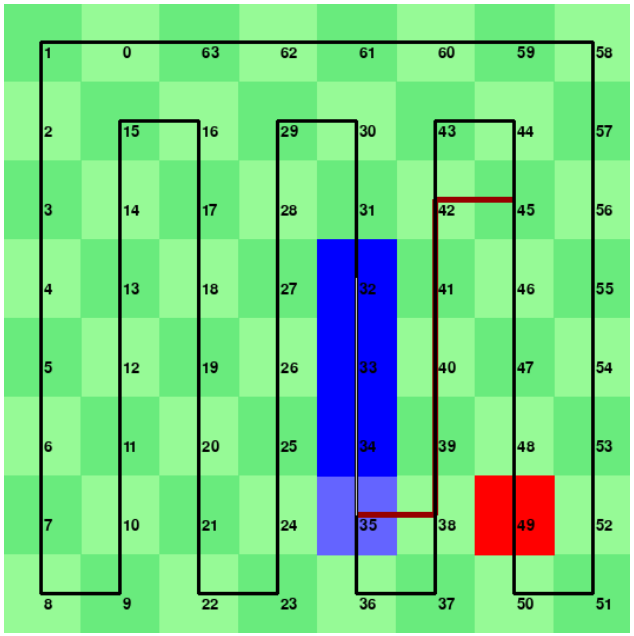


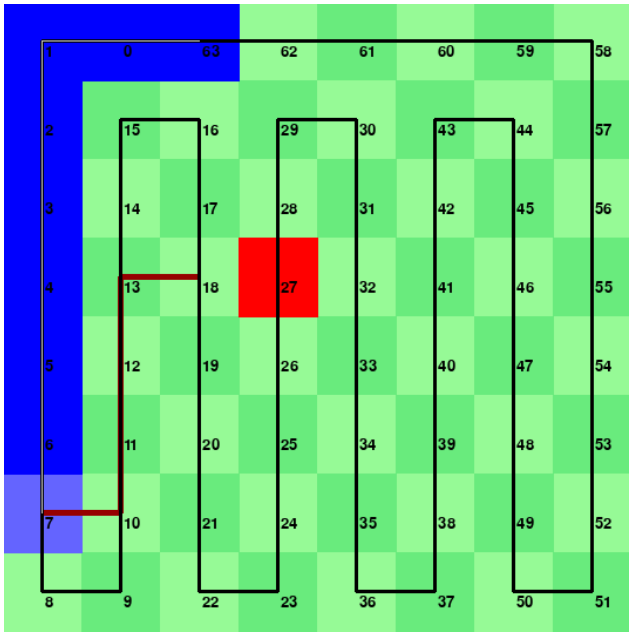
Suivi de cycle Hamiltonien
Résultats



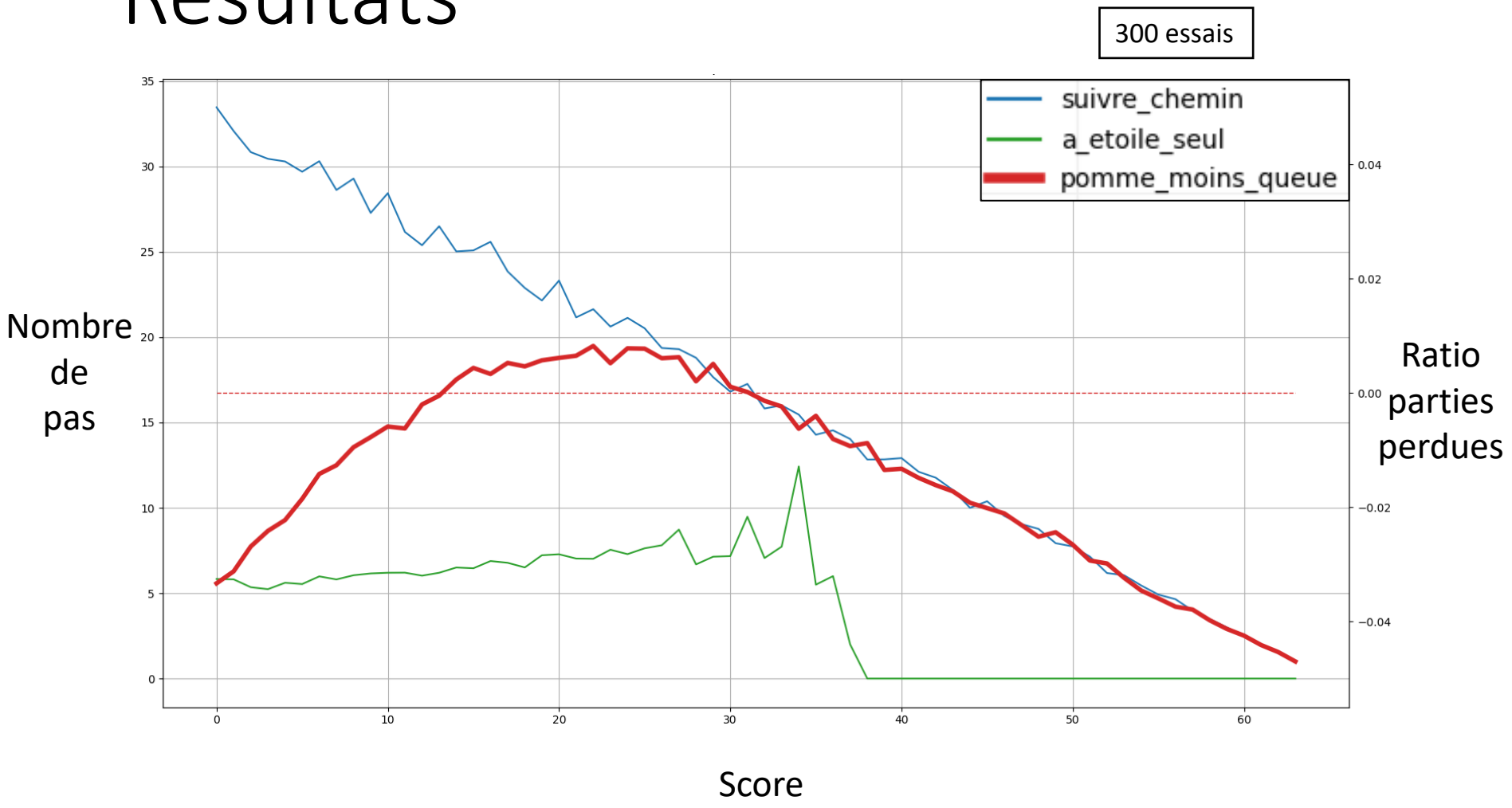
Pomme-queue
Présentation

-Plus court chemin
-Suivi de cycle Hamiltonien
-**Pomme-queue**
-Raccourcis sur cycle Hamiltonien
-Raccourcis sur cycle Hamiltonien 1D
-Plus court chemin sur cycle dynamique

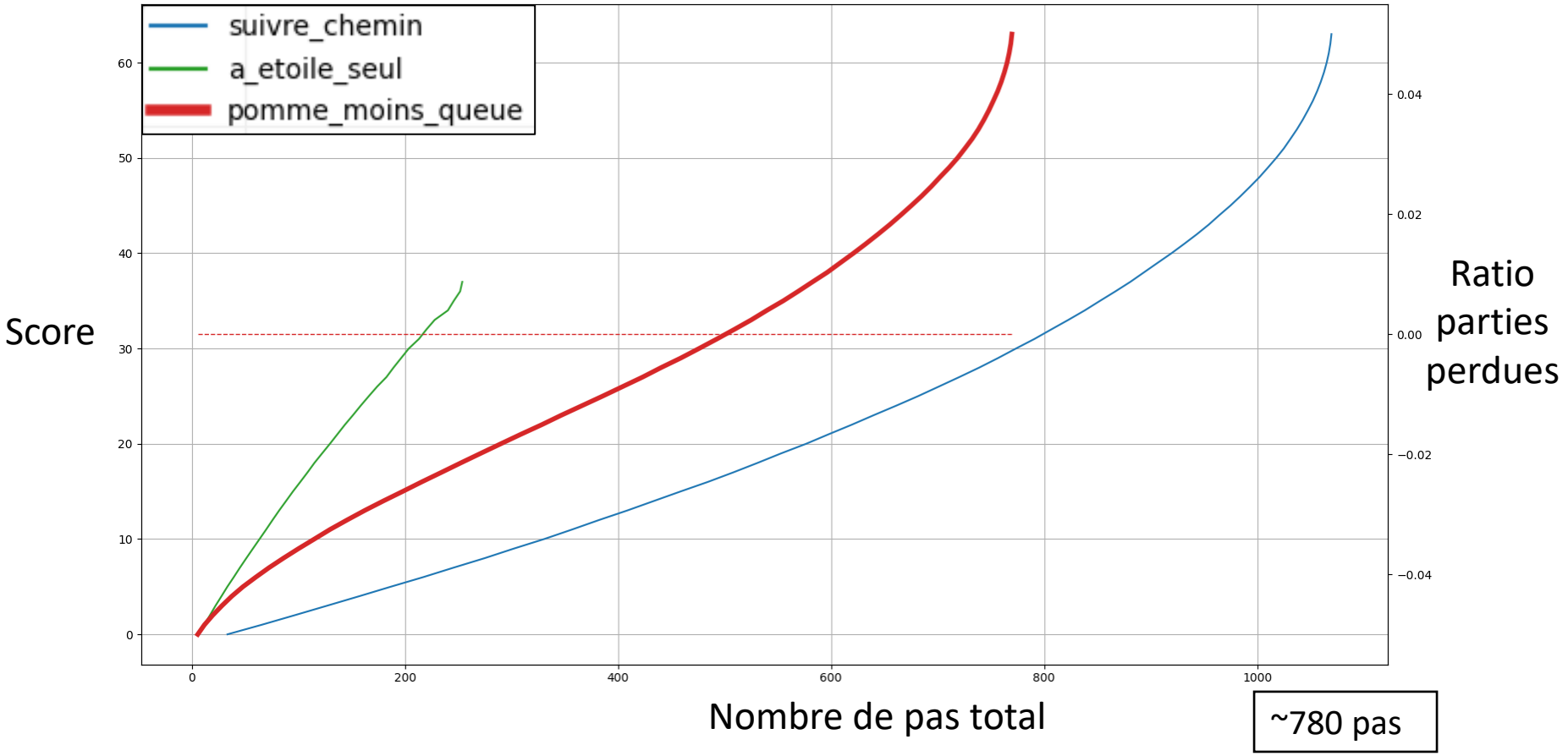




Pomme-queue
Résultats

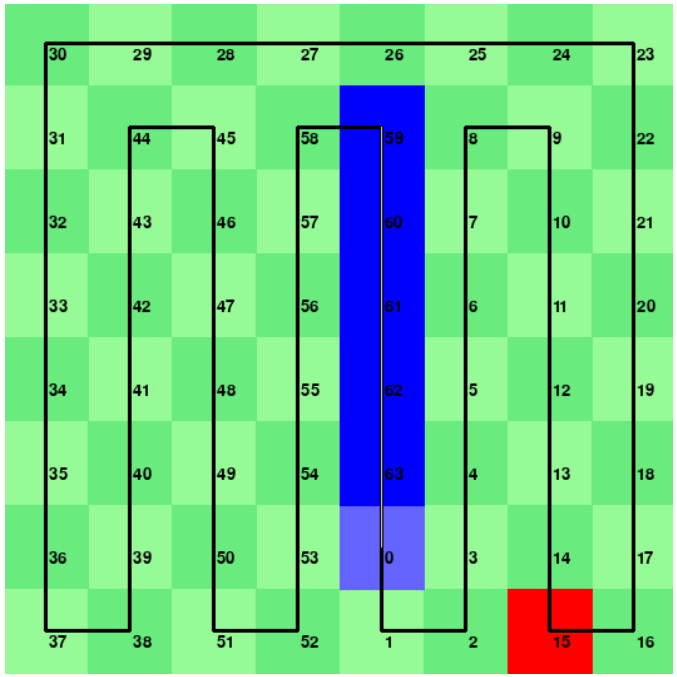


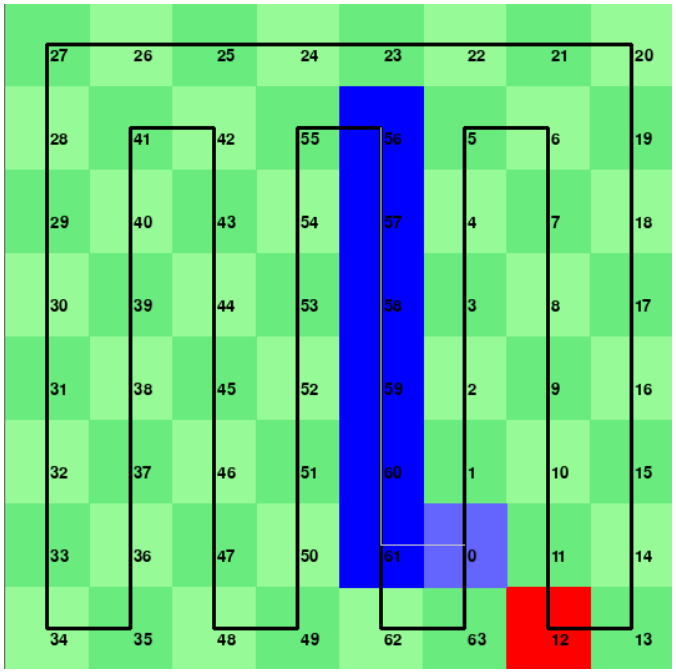
Pomme-queue
Résultats



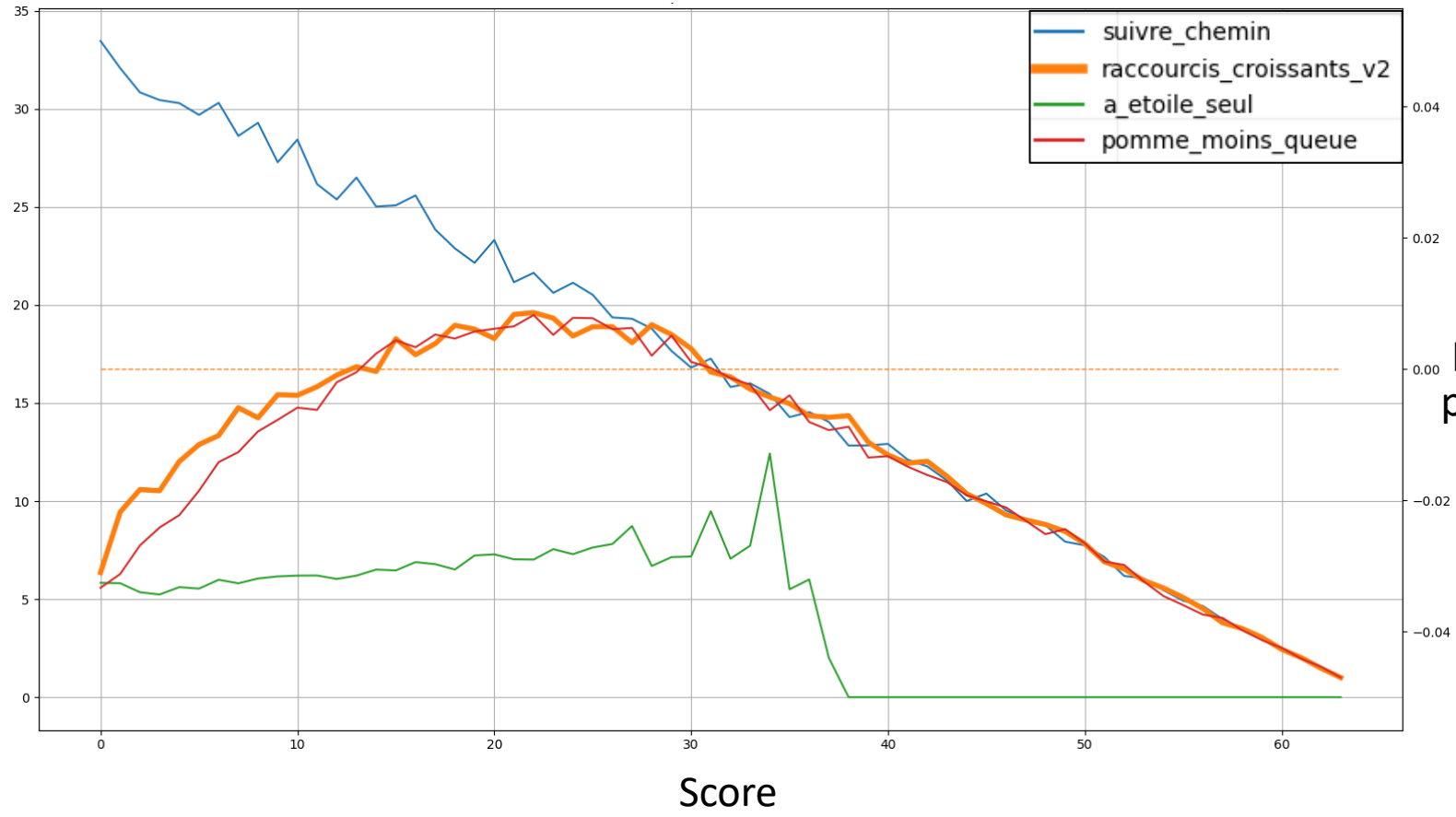
Raccourcis sur cycle Hamiltonien Présentation

-Plus court chemin
-Suivi de cycle Hamiltonien
-**Pomme-queue**
-Raccourcis sur cycle Hamiltonien
-Raccourcis sur cycle Hamiltonien 1D
-Plus court chemin sur cycle dynamique



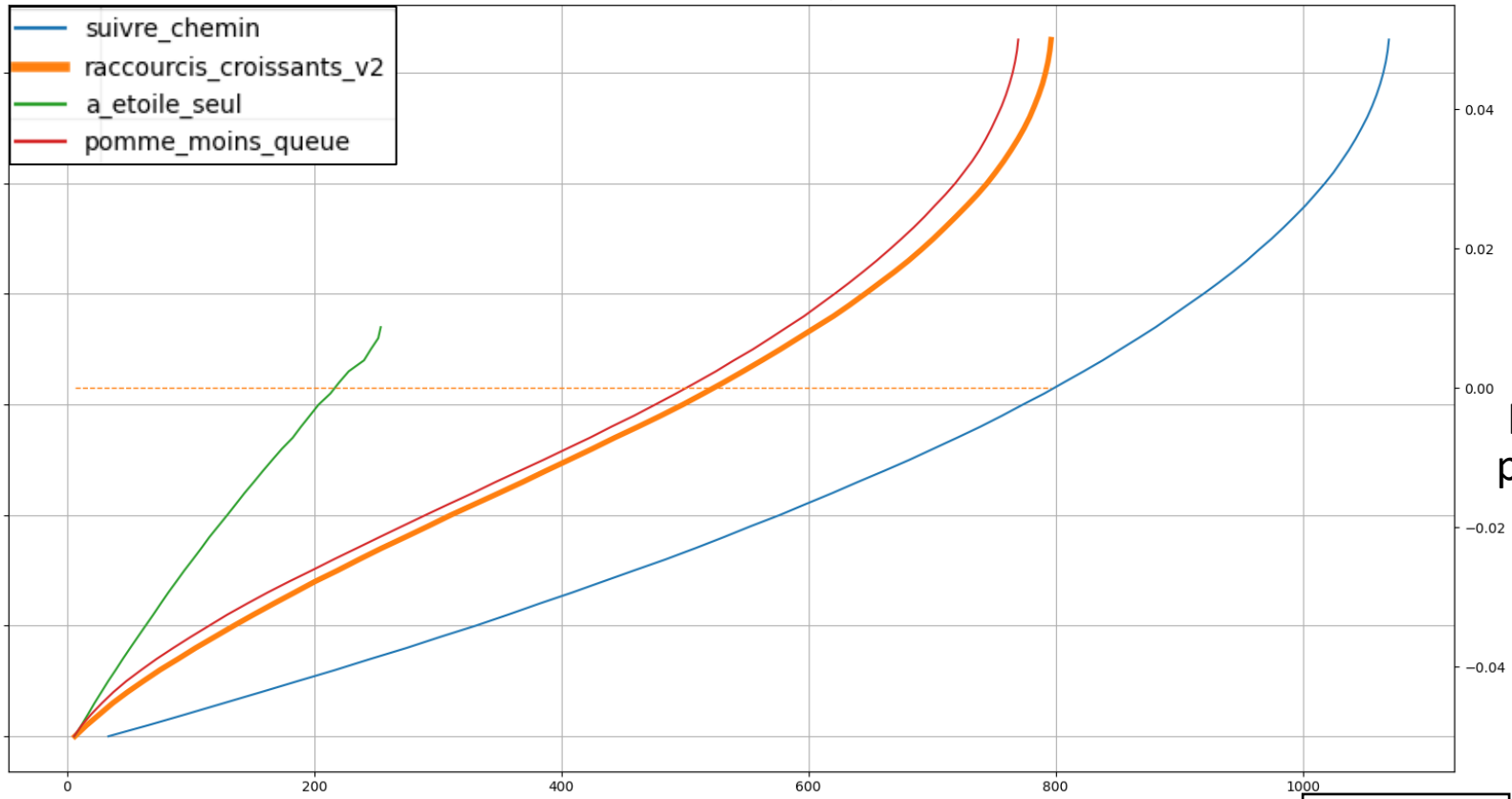


Raccourcis sur cycle Hamiltonien
Résultats



Raccourcis sur cycle Hamiltonien
Résultats

+350 essais



Score

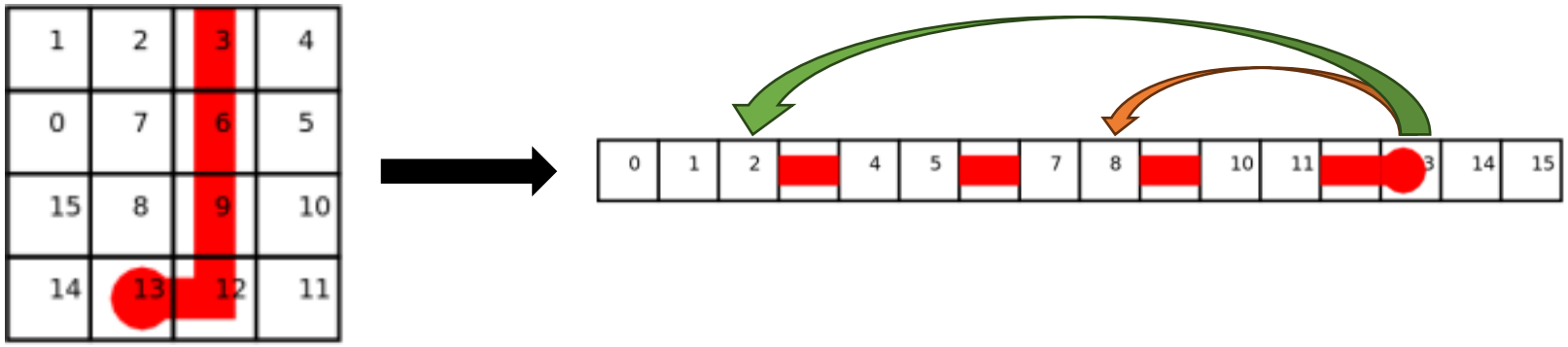
Ratio parties perdues

Nombre de pas total

~800 pas

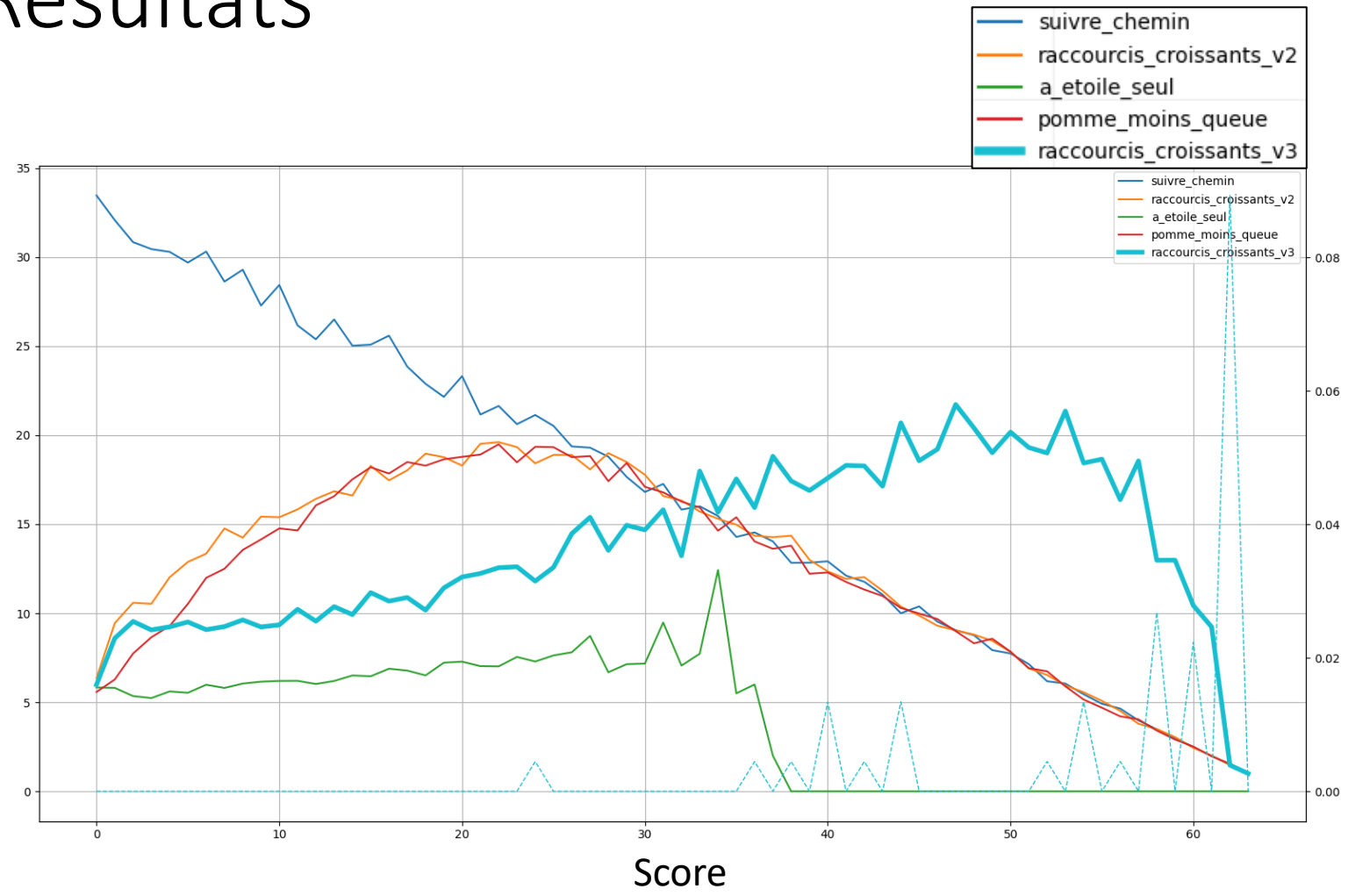
Raccourcis sur cycle Hamiltonien 1D Présentation

-Plus court chemin
-Suivi de cycle Hamiltonien
-Pomme-queue
-Raccourcis sur cycle Hamiltonien
-**Raccourcis sur cycle Hamiltonien 1D**
-Plus court chemin sur cycle dynamique

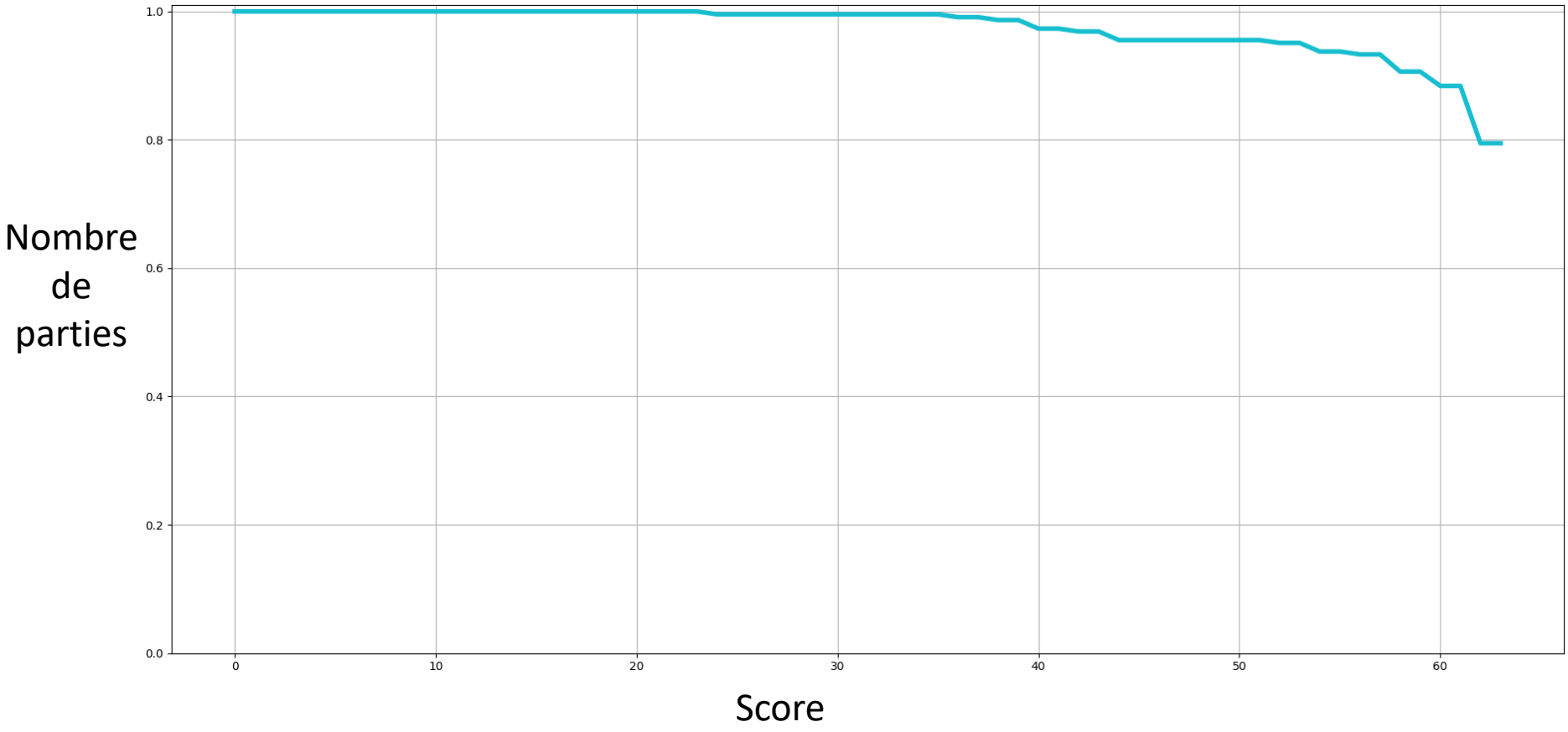


Images par John Tapsell
(cf [1])

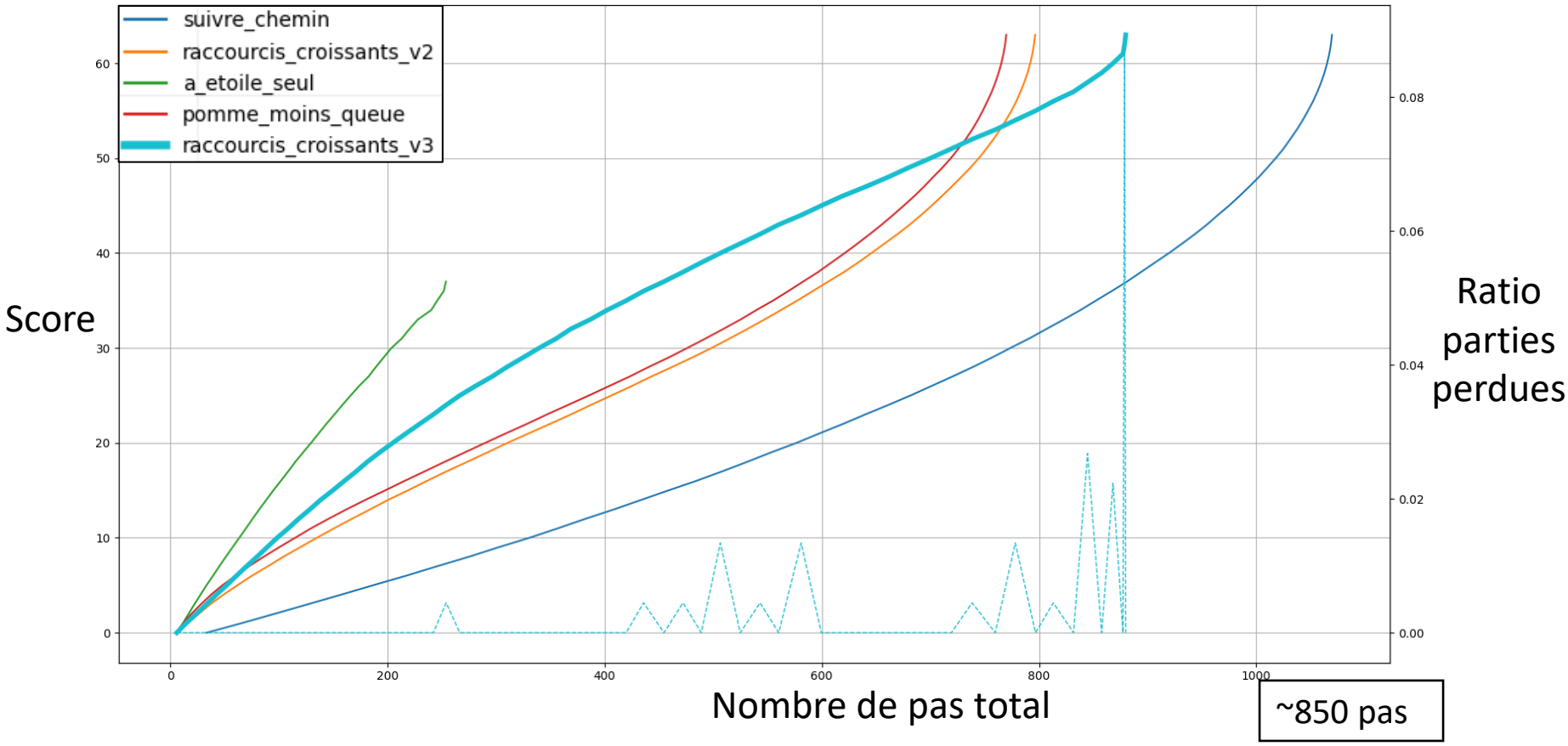
Raccourcis sur cycle Hamiltonien 1D Résultats



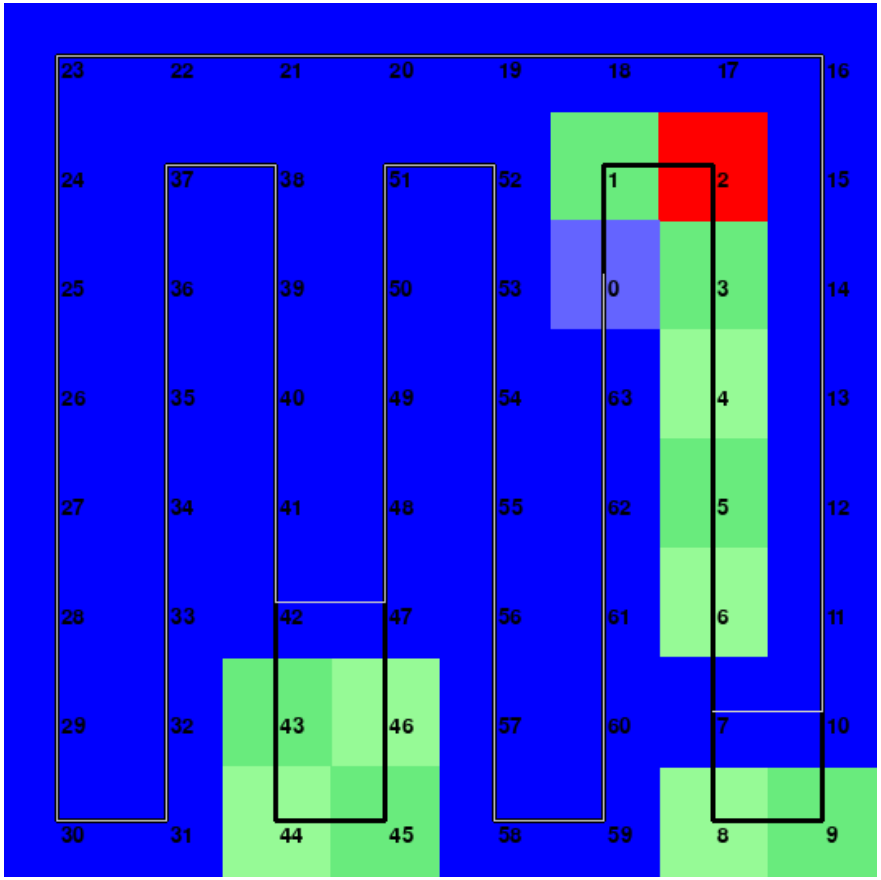
Raccourcis sur cycle Hamiltonien 1D Résultats



Raccourcis sur cycle Hamiltonien 1D Résultats

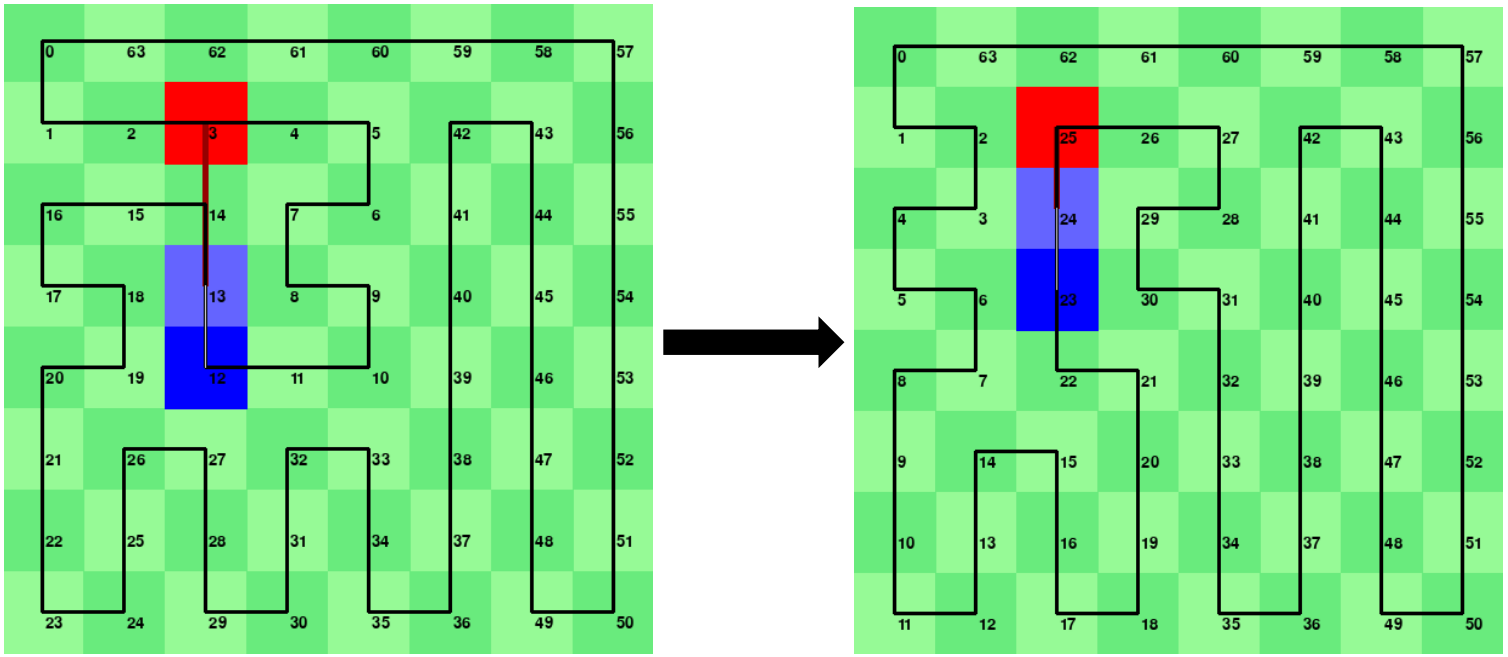


Raccourcis sur cycle Hamiltonien 1D Résultats



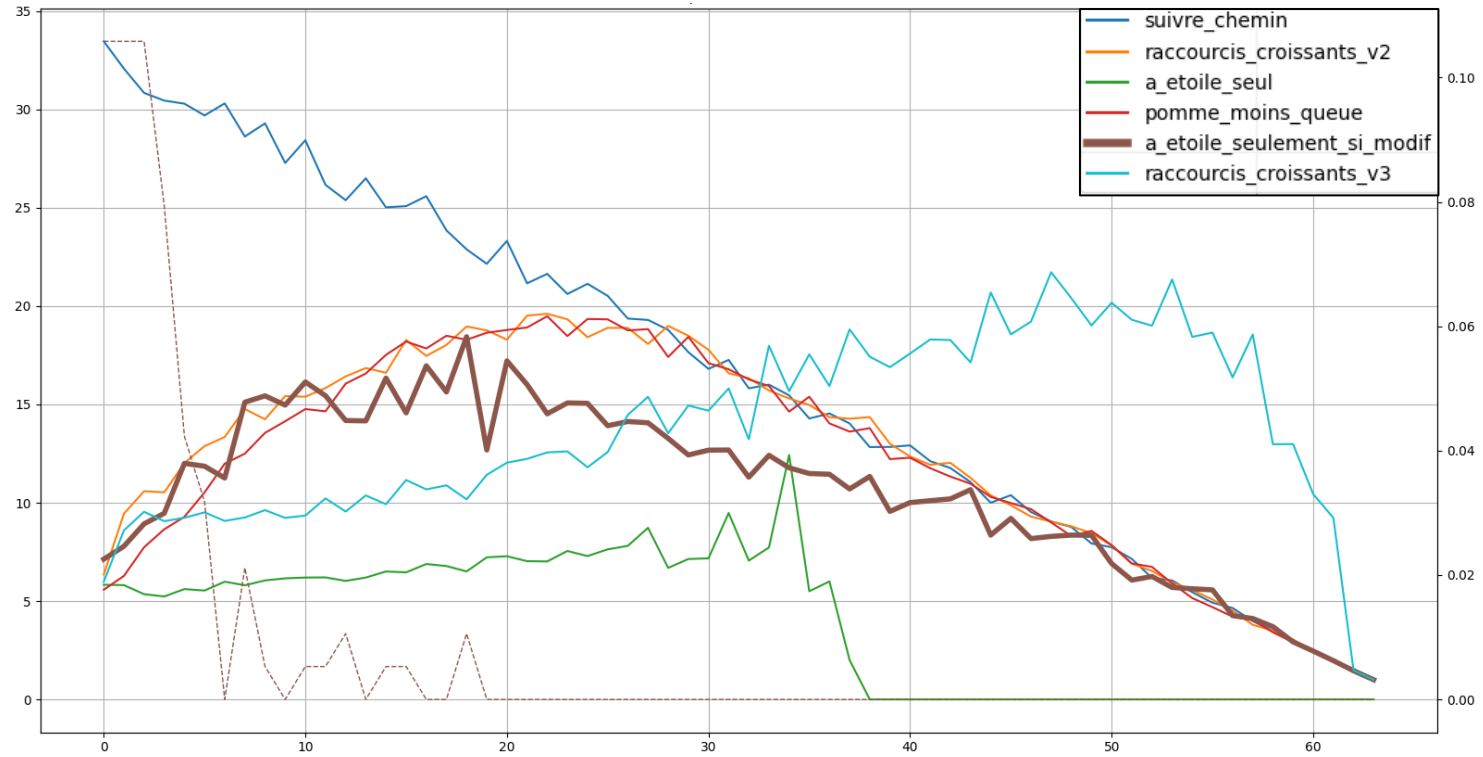
Plus court chemin sur cycle dynamique
Présentation

-Plus court chemin
-Suivi de cycle Hamiltonien
-Pomme-queue
-Raccourcis sur cycle Hamiltonien
-Raccourcis sur cycle Hamiltonien 1D
-**Plus court chemin sur cycle dynamique**



Plus court chemin sur cycle dynamique
Résultats

*Être dans une boucle virtuelle compte comme une défaite**

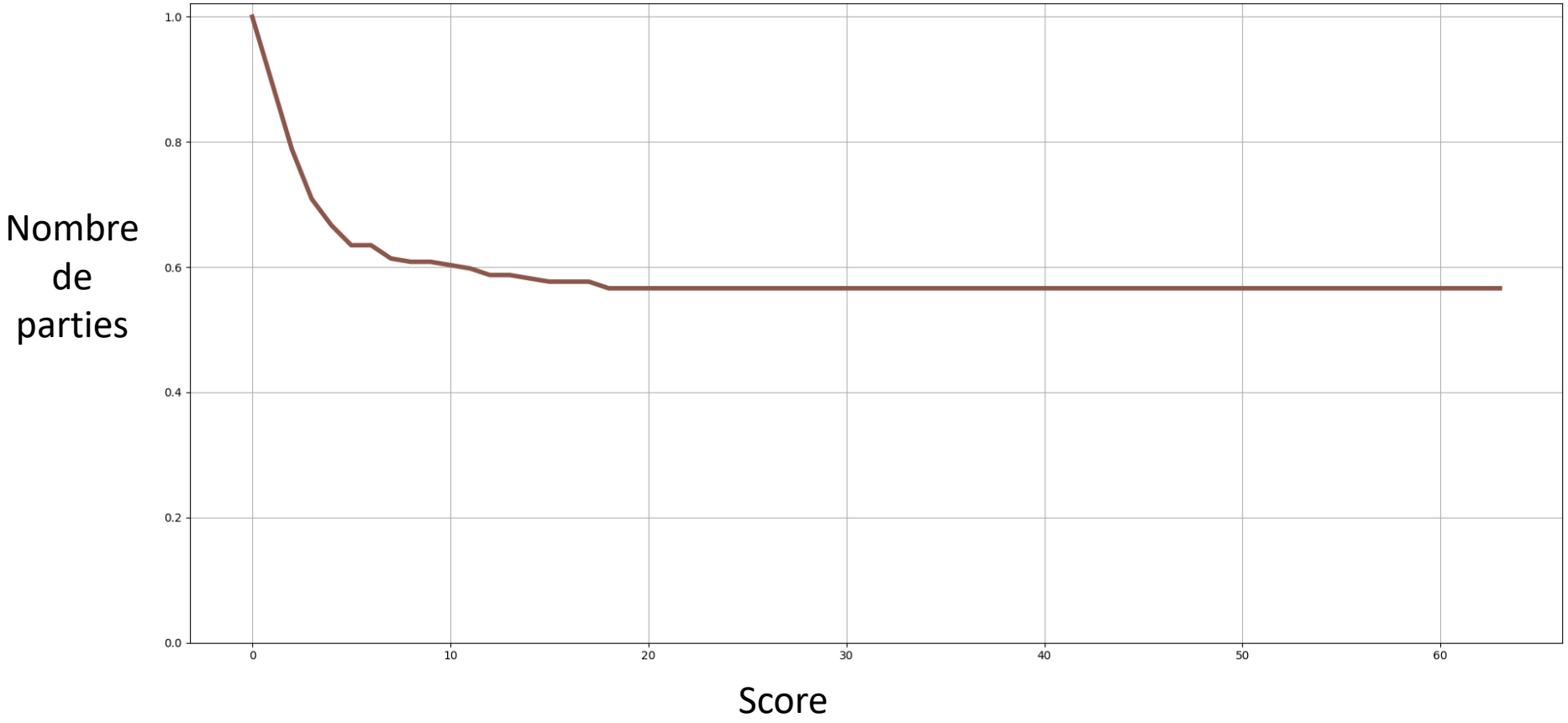


Nombre de pas

Ratio parties perdues

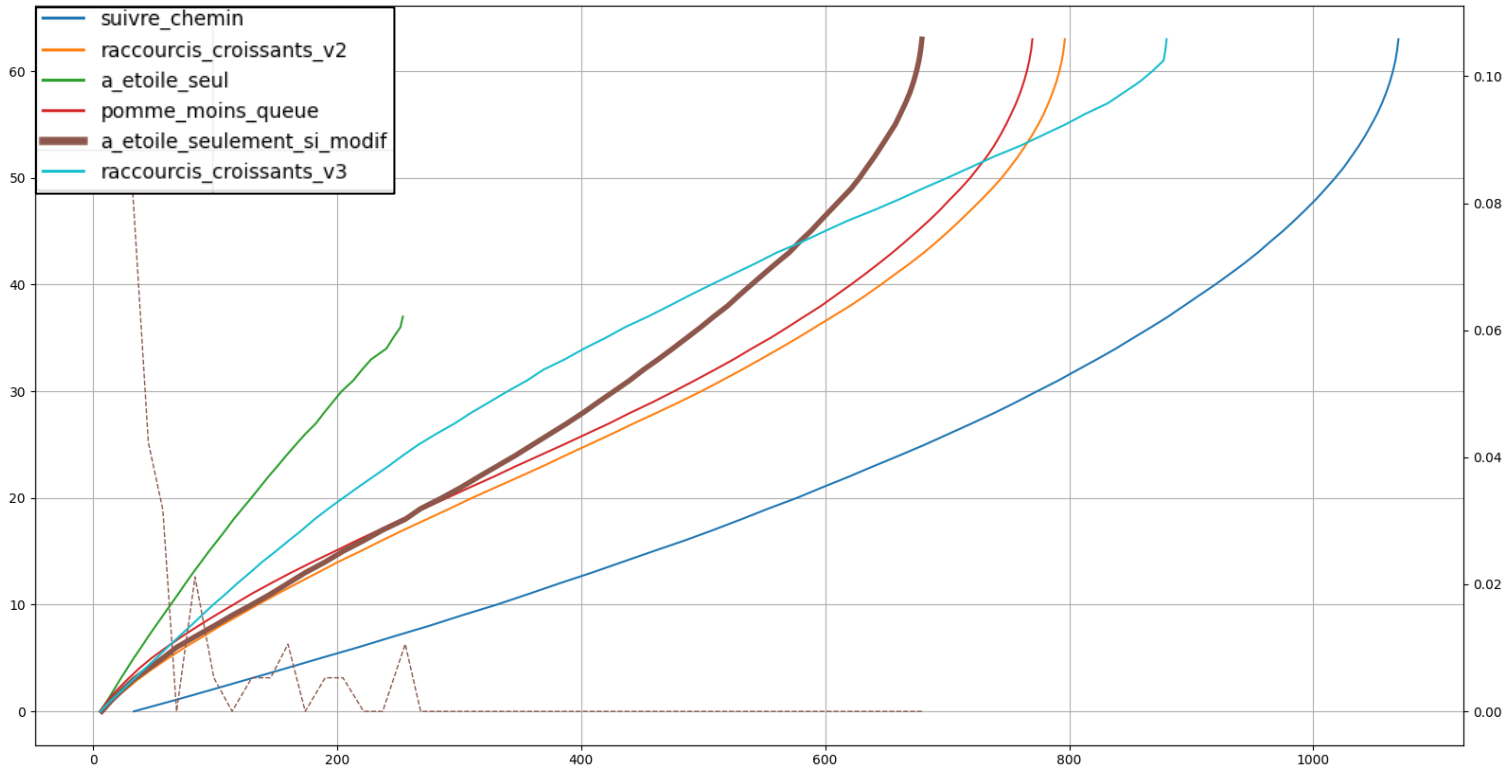
Score

Plus court chemin sur cycle dynamique
Résultats



Plus court chemin sur cycle dynamique
Résultats

Score

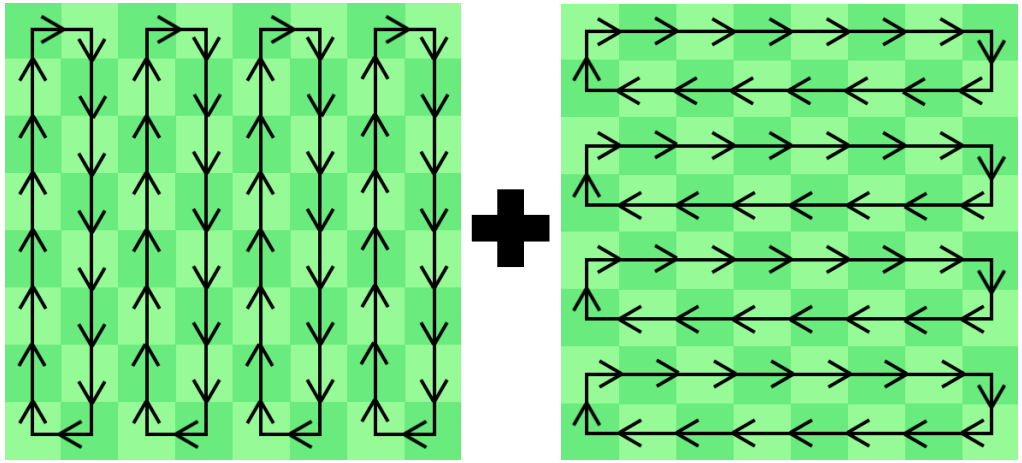


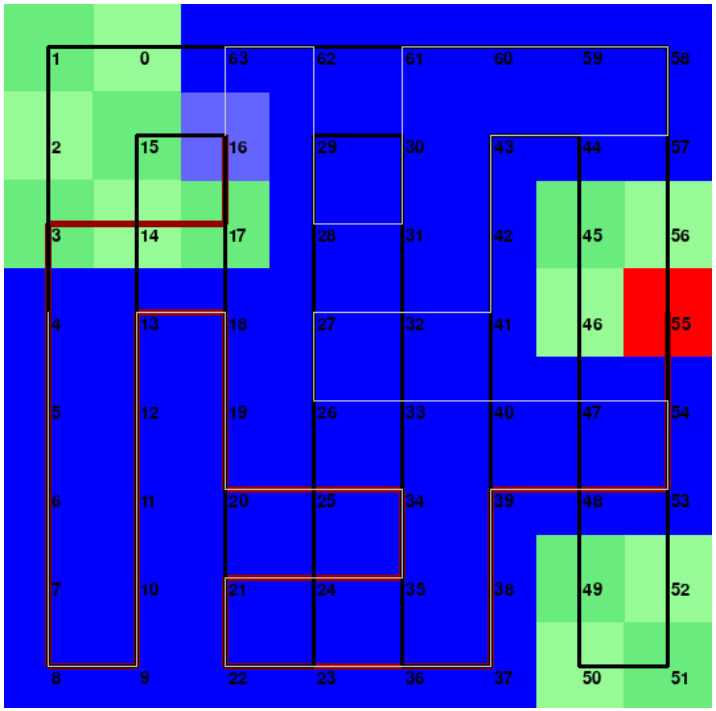
Ratio parties perdues

Nombre de pas total

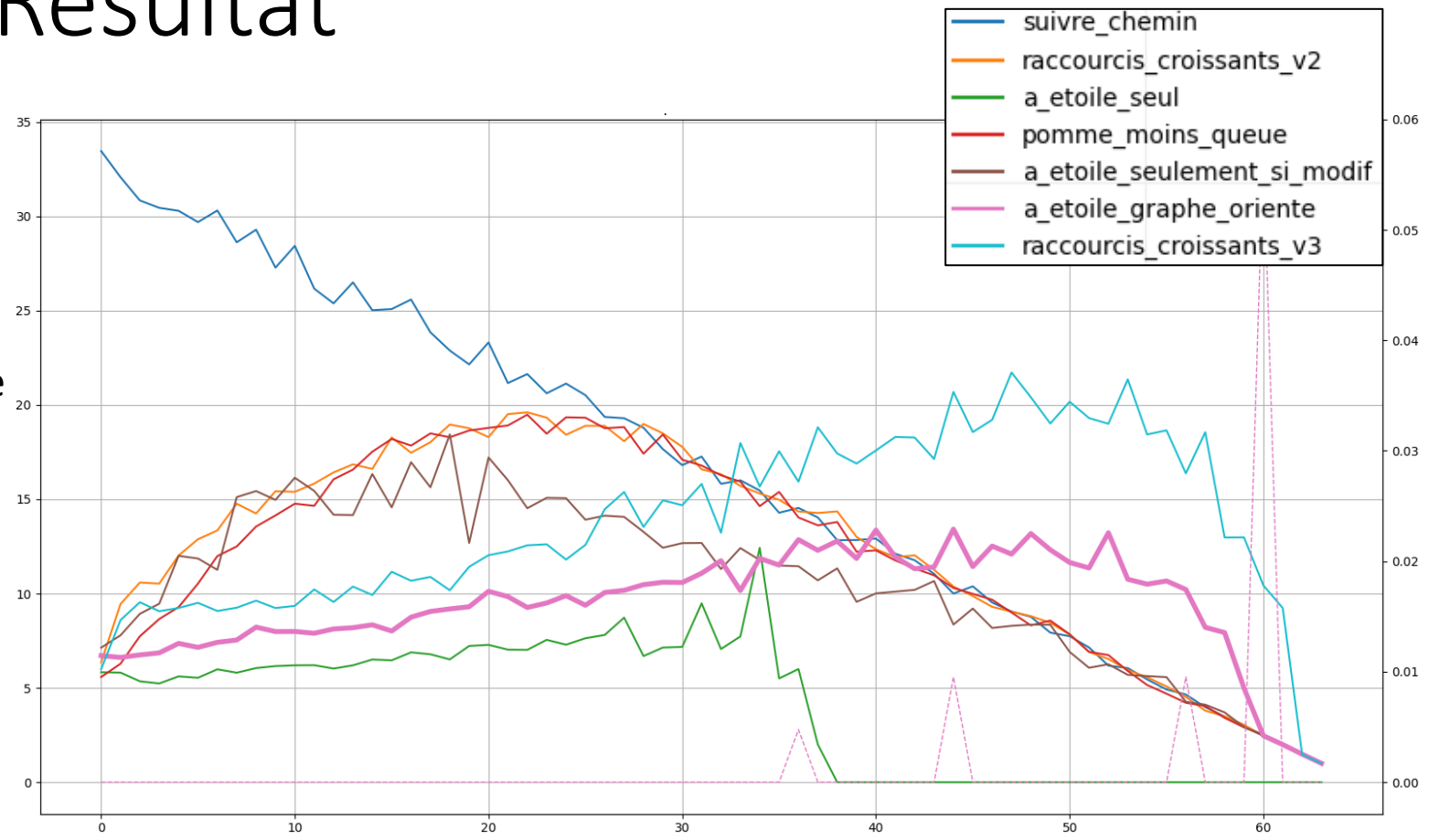
~700 pas

Plus court chemin graphe orienté Présentation

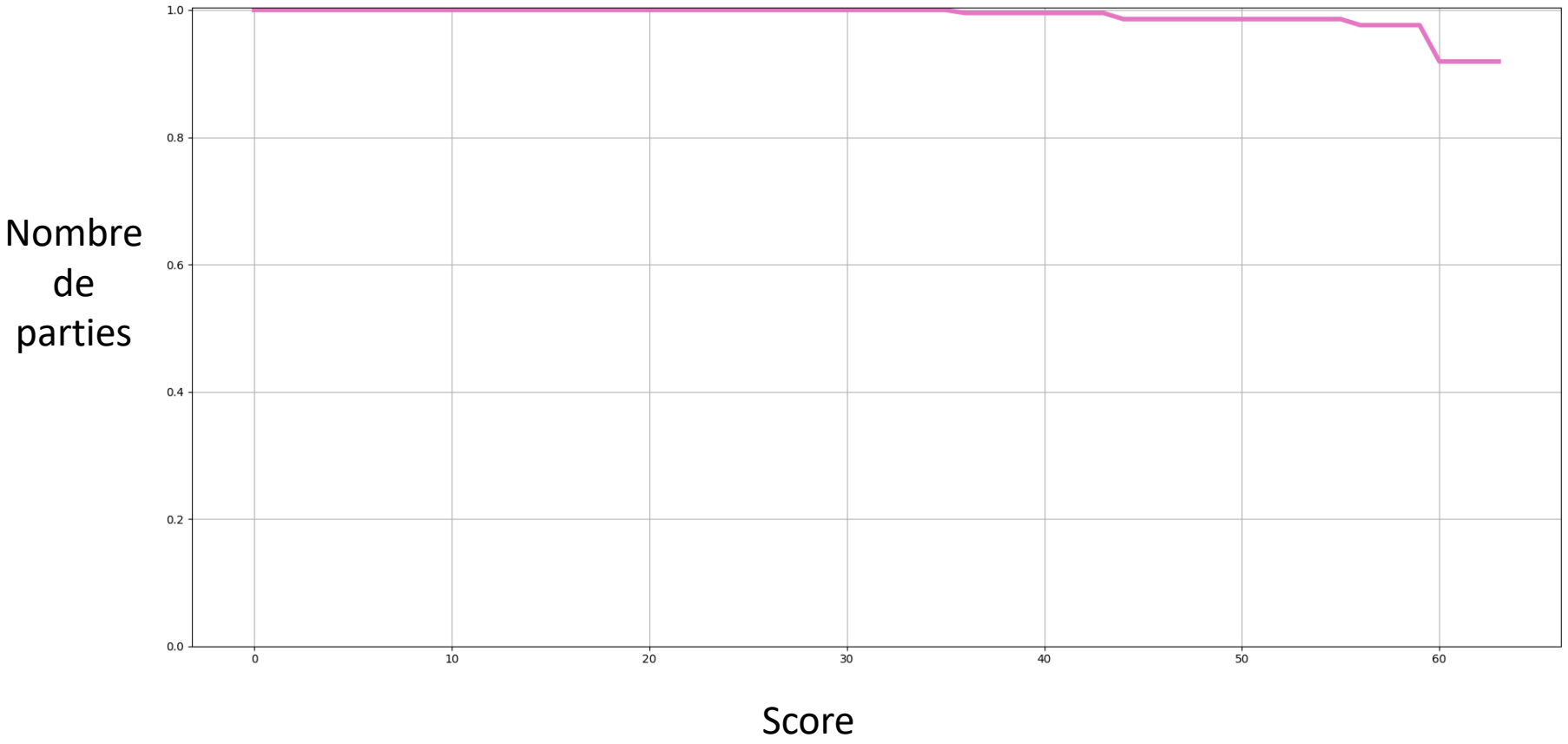




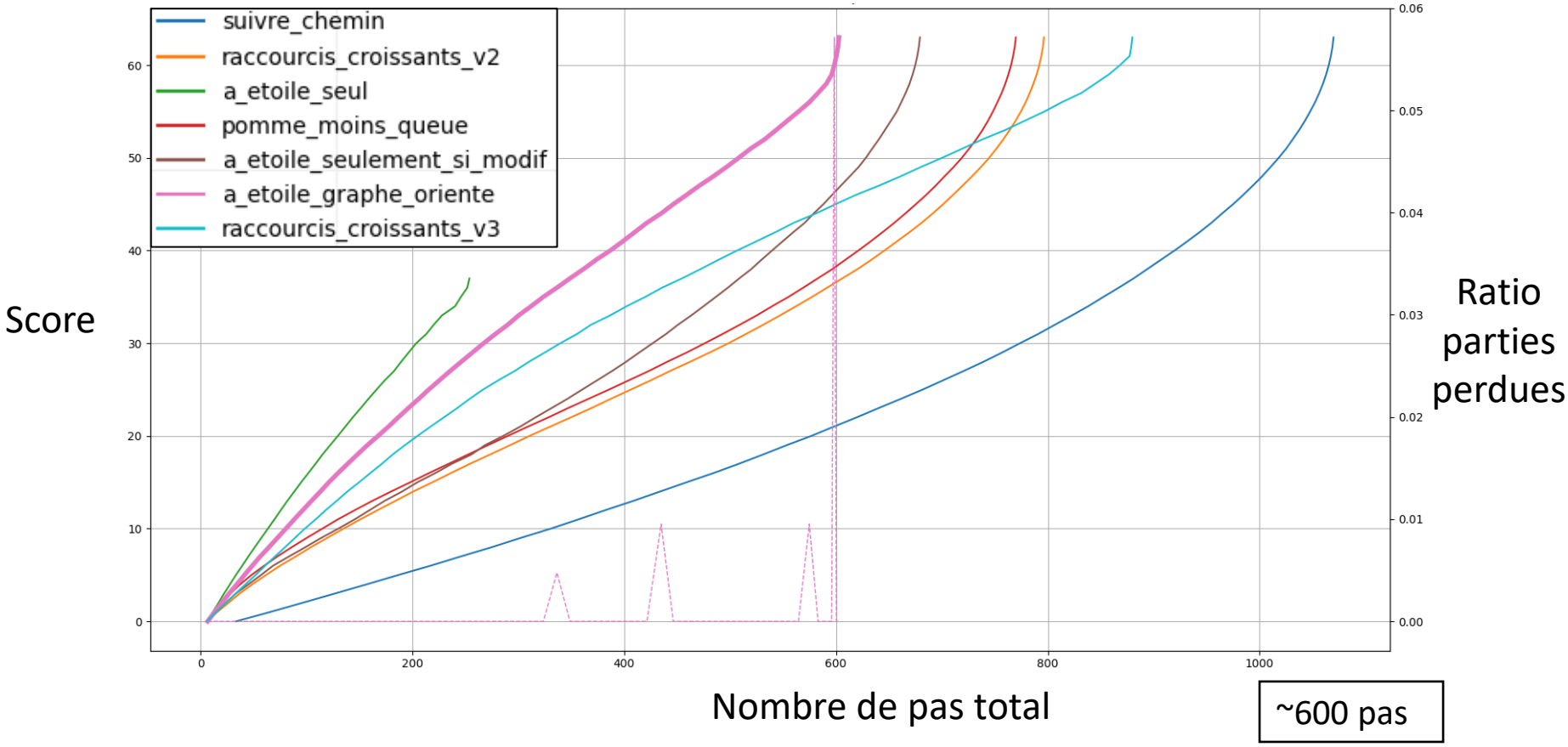
Plus court chemin graphe orienté
Résultat



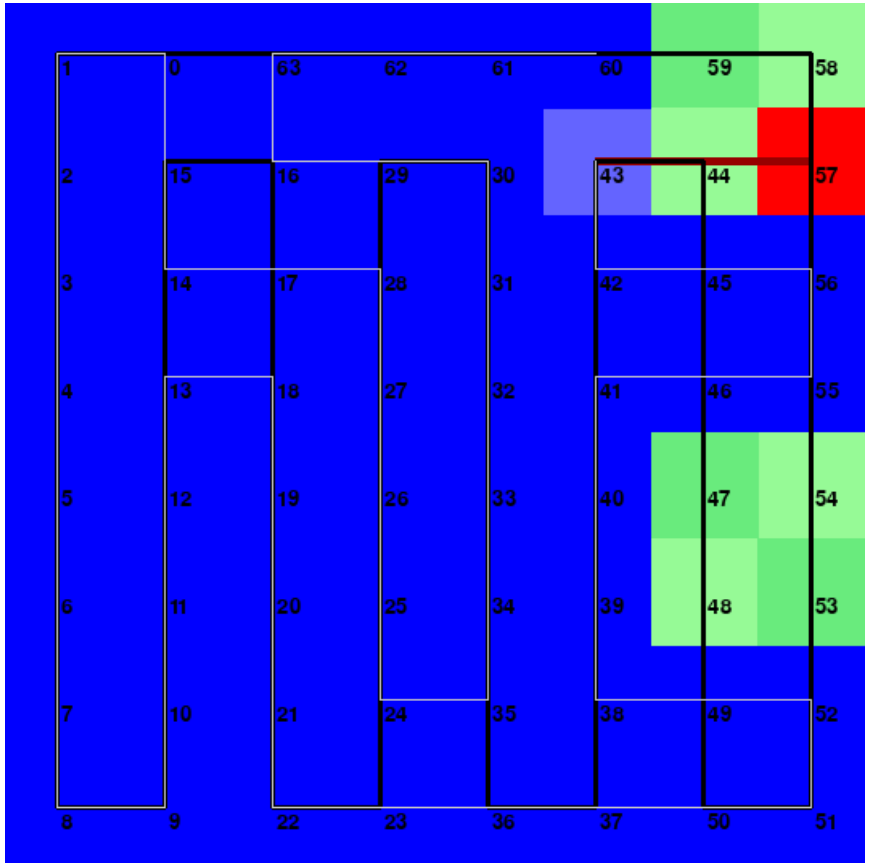
Plus court chemin graphe orienté Résultat



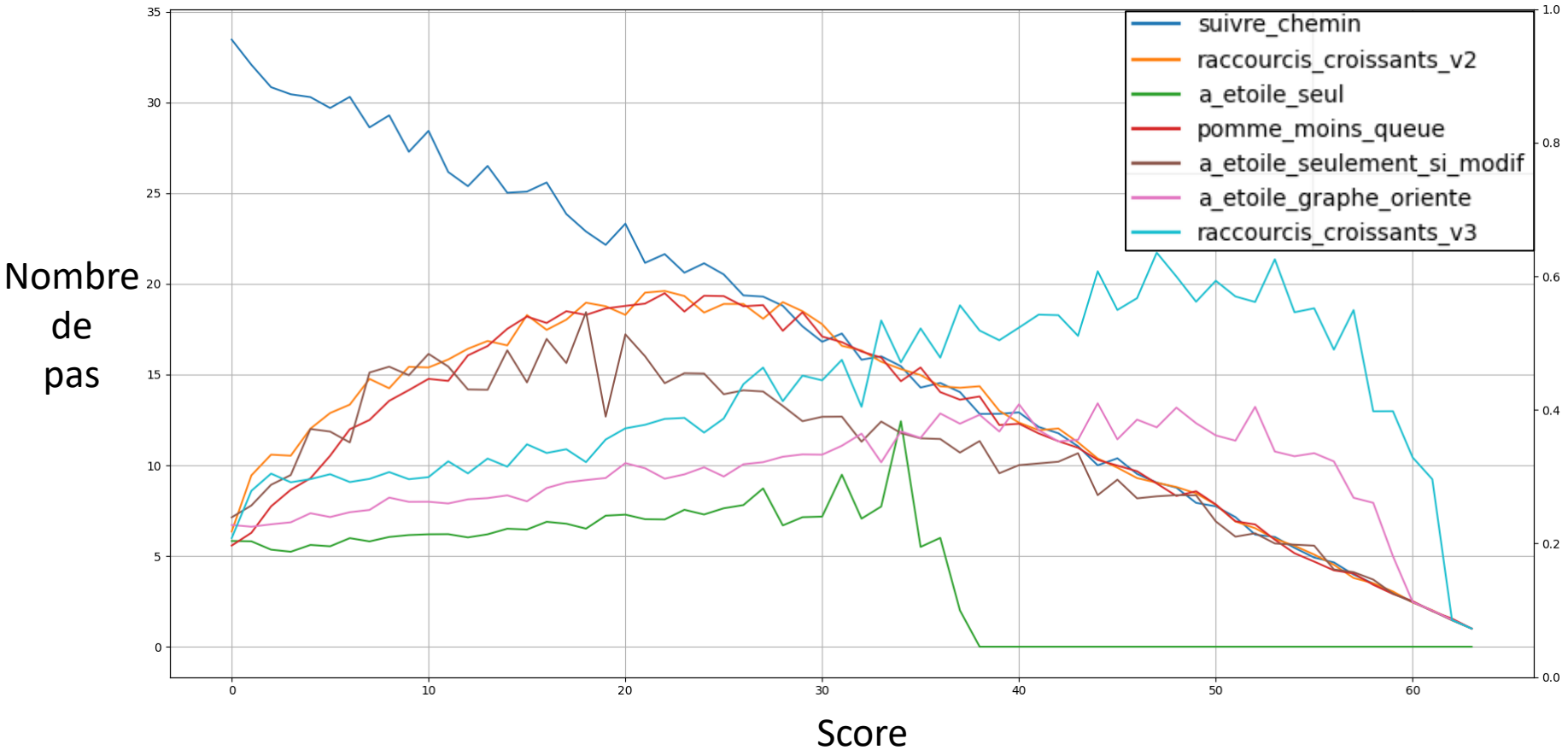
Plus court chemin graphe orienté Résultat



Plus court chemin graphe orienté Résultat



Conclusion


Nombre de pas
Score
suivre_chemin
raccourcis_croissants_v2
a_etoile_seul
pomme_moins_queue
a_etoile_seulement_si_modif
a_etoile_graphe_oriente
raccourcis_croissants_v3
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0

Annexe

Déplacement serpent

```
128     def  mettre_a_jour_serpent(self):
129         """on ne grandit pas au meme moment qu'on mange"""
130         # si on n'a pas de queue
131         if  self.jeu.serpent.taille_queue!=0:
132 
133             if  self.jeu.serpent.en_attente==None:
134                 self.jeu.serpent.pos_queue.popleft()
135                 self.liste_pos_aretes_squelette.popleft()
136             else:
137                 self.jeu.serpent.en_attente=None
138 
139             self.jeu.serpent.pos_queue.append(self.jeu.serpent.pos[:])
140             self.liste_pos_aretes_squelette.append([self.jeu.serpent.pos,[self.jeu.serpent.pos[0]+self.jeu.serpent.direction[0],self.jeu.serpent.pos[1]+self.jeu.serpent.direction[1]]])
141 
142         # si on a une queue
143         else:
144             if  self.jeu.serpent.en_attente!=None:
145 
146                 self.jeu.serpent.pos_queue.append(self.jeu.serpent.pos[:])
147                 self.liste_pos_aretes_squelette.append([self.jeu.serpent.pos,[self.jeu.serpent.pos[0]+self.jeu.serpent.direction[0],self.jeu.serpent.pos[1]+self.jeu.serpent.direction[1]]])
148 
149                 self.jeu.serpent.en_attente=None
150 
151         self.jeu.serpent.taille_queue=len(self.jeu.serpent.pos_queue)
152 
153         # deplacement
154         self.jeu.serpent.pos=[
155         self.jeu.serpent.pos[0]+self.jeu.serpent.direction[0],
156         self.jeu.serpent.pos[1]+self.jeu.serpent.direction[1]
157         ]
```

Annexe

```
3338    def  a_etoile(self,fin,liste_pos_murs,prise_en_compte_deplacement):
3339        debut=self.jeu.serpent.pos
3340        dico_couts={(debut[0],debut[1]):0}
3341        a_traiter=collections.deque([debut])
3342
3343        dico_parents={}
3344        dico_profondeurs={(debut[0],debut[1]):0}
3345
3346        vide=collections.deque([])
3347
3348        while  a_traiter!=vide:
3349            pos=min(a_traiter,key=lambda  p:dico_couts[(p[0],p[1])])
3350            a_traiter.remove(pos)
3351
3352            #  si  c'est  fini
3353            if  pos==fin:
3354                chemin=[[pos[0],pos[1]]]
3355                while  (chemin[-1][0],chemin[-1][1])  in  dico_parents:
3356                    chemin.append(dico_parents[(chemin[-1][0],chemin[-1][1])])
3357                chemin.reverse()
3358                self.plus_court_chemin[:]=chemin[:]
3359
3360                #  print('CHEMIN',chemin)
3361
3362                if  len(chemin)>=2:
3363                    self.jeu.serpent.direction=[chemin[1][0]-debut[0],chemin[1][1]-debut[1]]
3364                else:
3365                    self.jeu.serpent.direction=[fin[0]-debut[0],fin[1]-debut[1]]
3366
3367                if  self.jeu.serpent.direction==[0,0]:
3368                    return  False
3369
3370                #  print(self.jeu.serpent.direction)
3371
3372                return  True
3373
```

Annexe

```
3373
3374            liste_voisins=self.voisins(pos)
3375
3376            profondeur=0
3377            pos_=[pos[0],pos[1]]
3378            while  (pos_[0],pos_[1])  in  dico_parents:
3379                profondeur+=1
3380                pos_=dico_parents[(pos_[0],pos_[1])]
3381
3382
3383            pos_queue=list(self.jeu.serpent.pos_queue)
3384
3385            if prise_en_compte_deplacement:
3386                # si on est sur la pomme
3387                if  self.jeu.pomme_atteinte:
3388                    if  profondeur==0:
3389                        pos_queue_predits=pos_queue
3390                    else:
3391                        pos_queue_predits=pos_queue[(profondeur):]
3392
3393                # si on n'est pas sur la pomme
3394                else:
3395                    pos_queue_predits=pos_queue[(profondeur+1):]
3396
3397                # print(pos_queue_predits)
3398            else:
3399                pos_queue_predits=pos_queue
3400
3401            for  voisin  in  liste_voisins:
3402                if  not  voisin  in  pos_queue_predits  and  not  voisin  in  liste_pos_murs:
3403                    cout_g=abs(voisin[0]-debut[0])+abs(voisin[1]-debut[1])
3404                    cout_h=abs(voisin[0]-fin[0])+abs(voisin[1]-fin[1])
3405                    cout_f=cout_g+cout_h
3406
3407
3408                    if  not  (voisin[0],voisin[1])  in  dico_couts  or  cout_f<dico_couts[(voisin[0],voisin[1])]:
3409                        a_traiter.append(voisin)
3410                        dico_parents[(voisin[0],voisin[1])]=pos[:]
3411                        dico_couts[(voisin[0],voisin[1])]=cout_f
3412
3413        return  False
```