Présentation de PYRATatouille

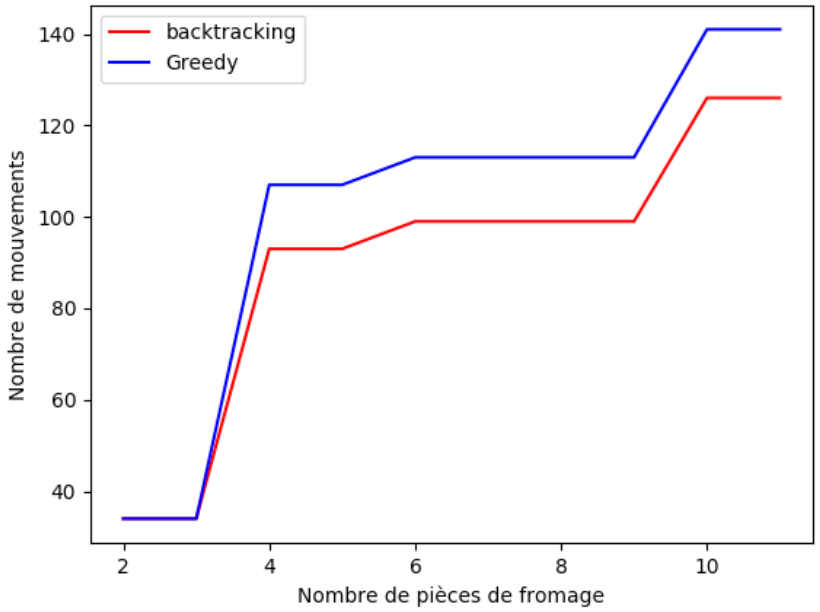
Aymeric Schweitzer & Mokhles Bouzaien

Plan du travail

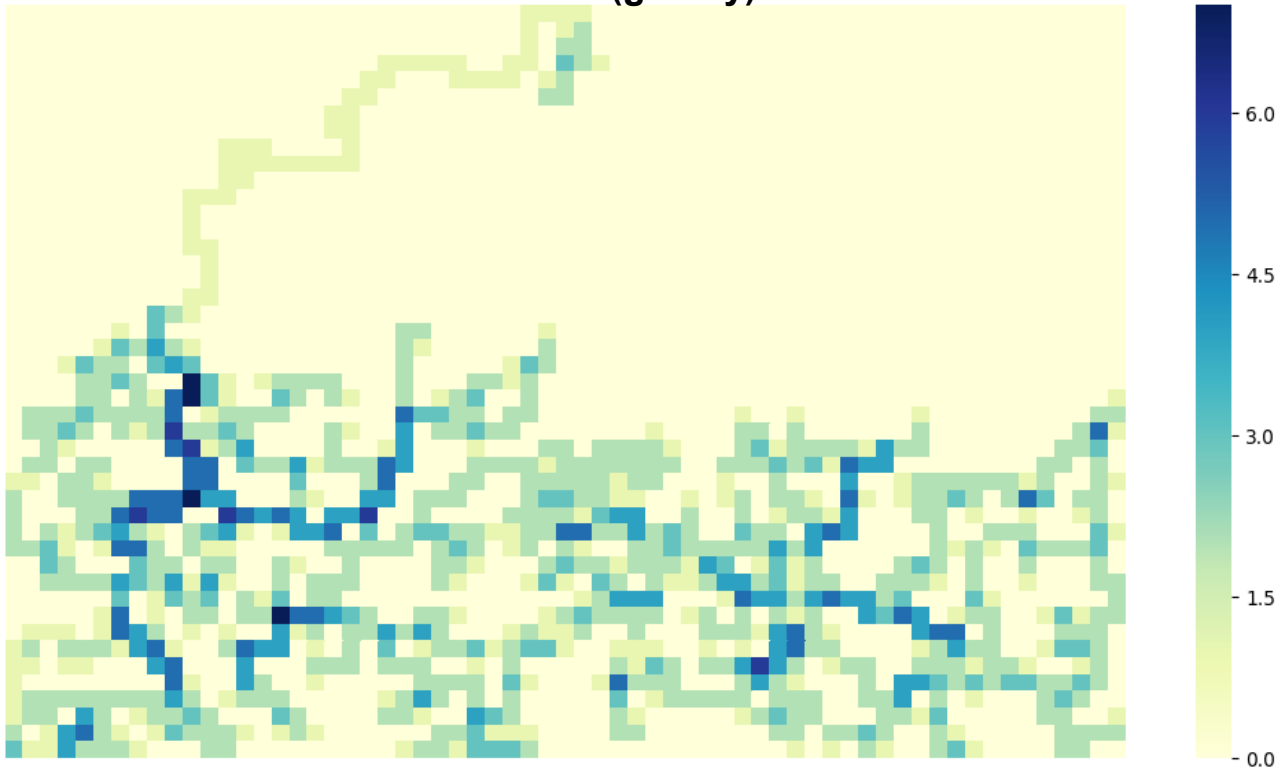
I. **Backtracking et Glouton.**
II. **L'algorithme final $2 - Opt$**
    1. Présentation
    2. Implémentation
    3. Amélioration (2-optsort)
III. **Limites et prochaines idées**

I- Backtracking et Glouton

Pour un même graphe (taille : 21x15)



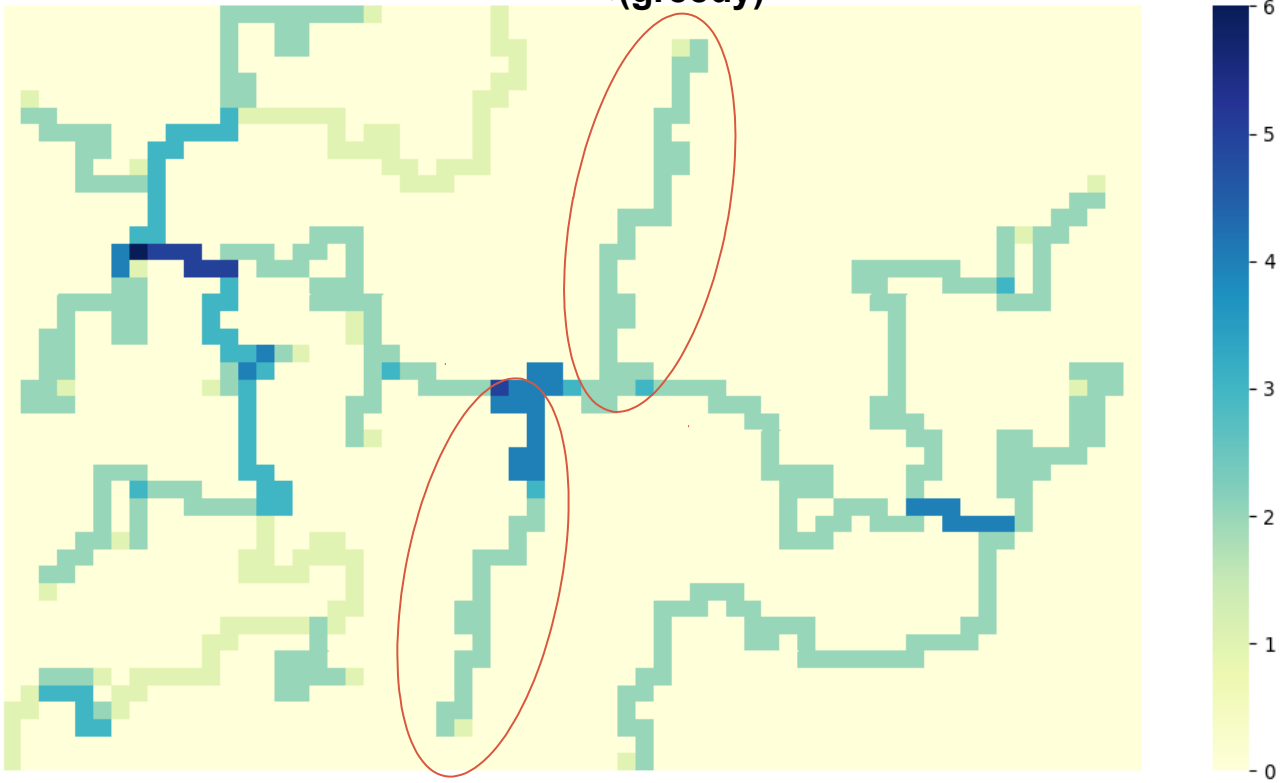
Heatmap des déplacements du Rat (greedy)



Nombre de passages par chaque sommet

$$\textit{Taille}$$
$$63 \times 45$$

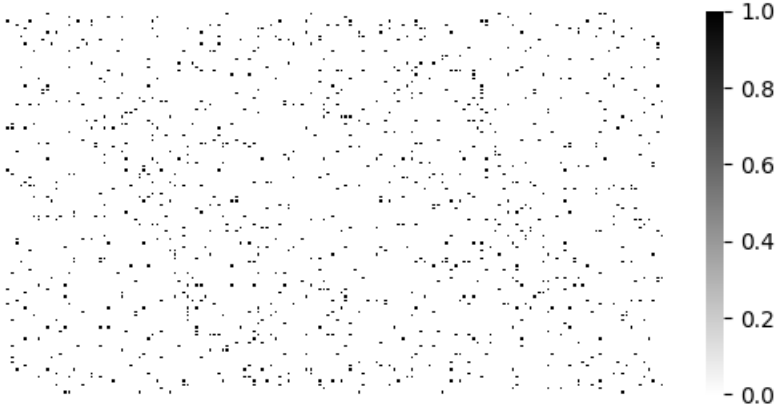
Heatmap des déplacements du Rat (greedy)



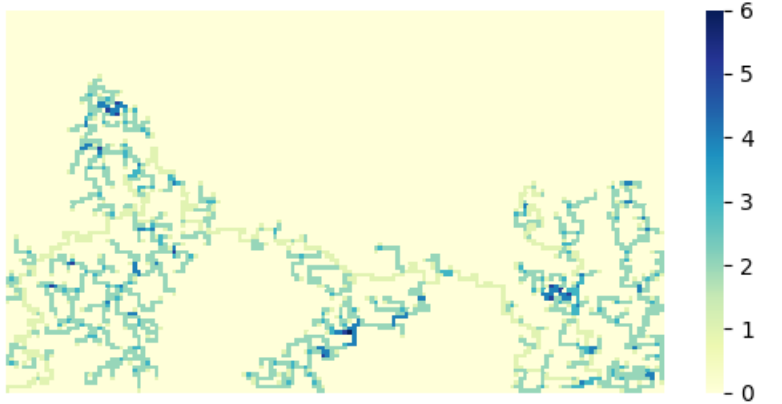
Nombre de passages par chaque sommet

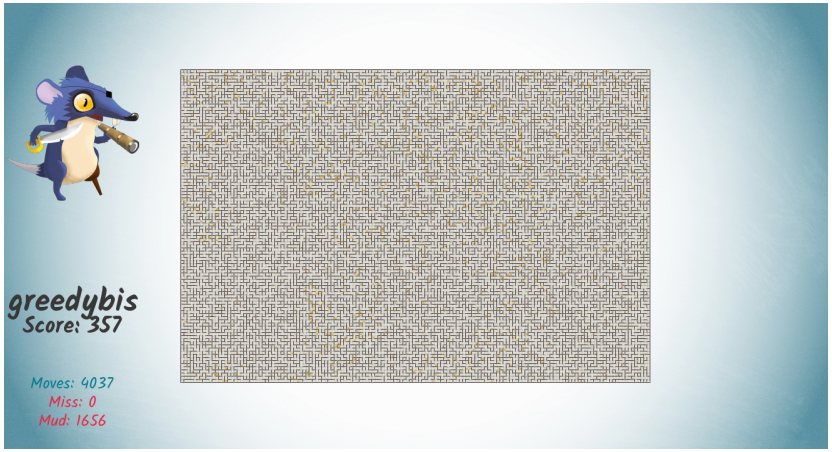
$$Taille$$
$$63 \times 45$$

Pièces de fromage



Heatmap des déplacement du Rat



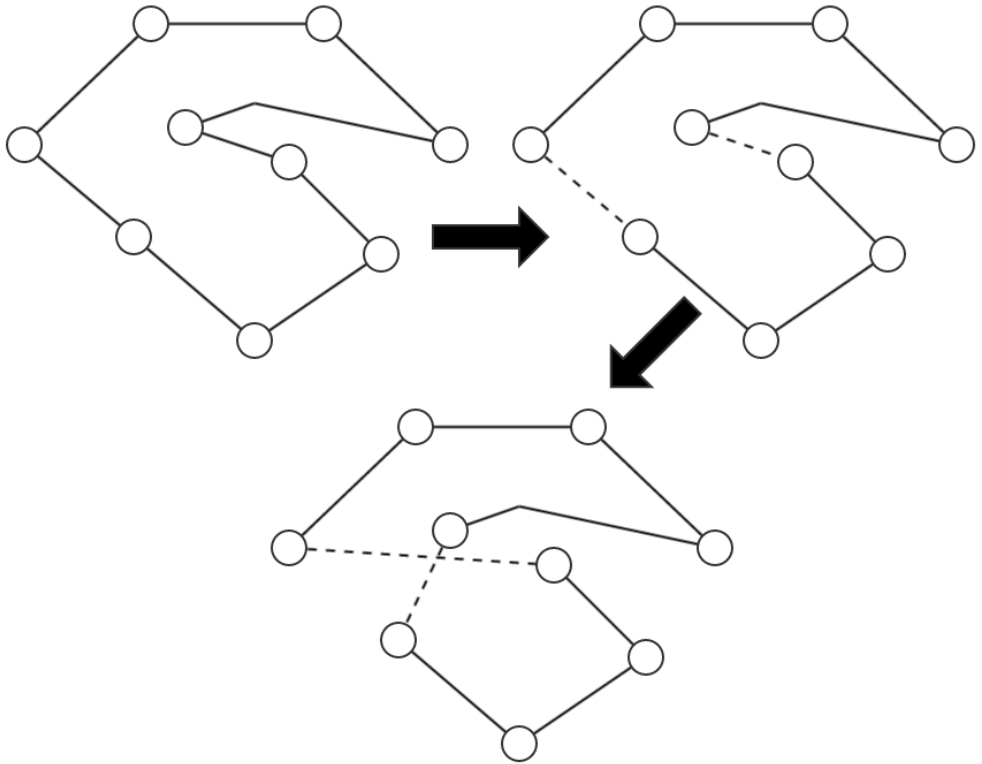


II- L’algorithme 2-opt

1. Présentation

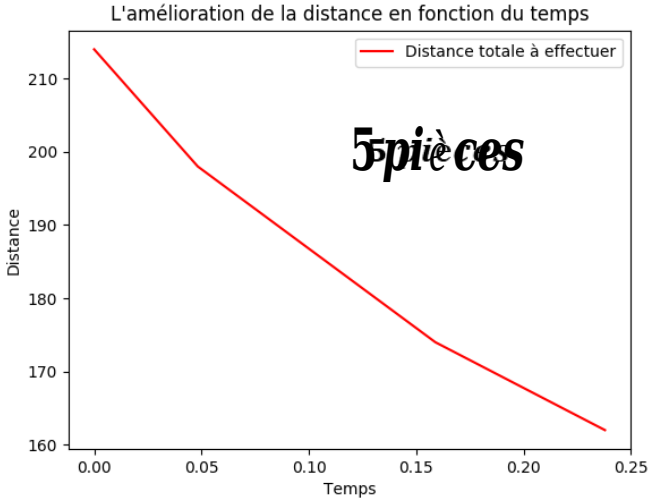
Le principe de l'heuristique de (2- opt Lin-Kernighan) est d’améliorer une solution en faisant un certain nombre de flips successifs et on se restreint à ceux qui sont jugés «prometteurs», dans le sens suivant:

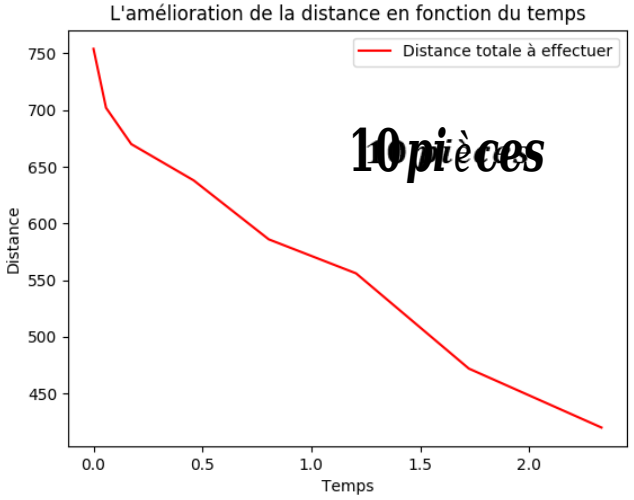
Un flip est « prometteur » si le coût total du parcours est amélioré.

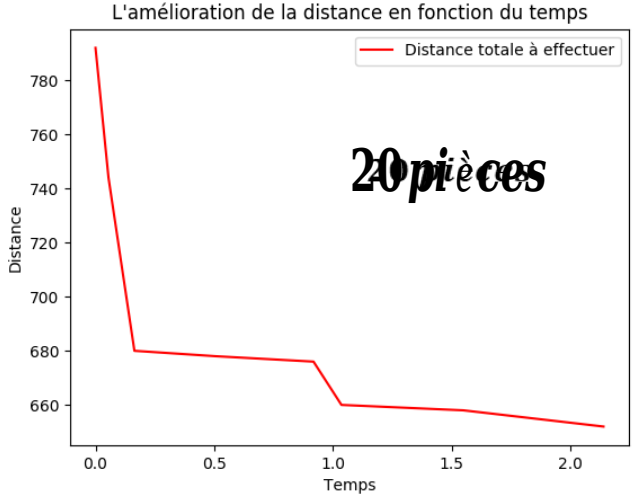


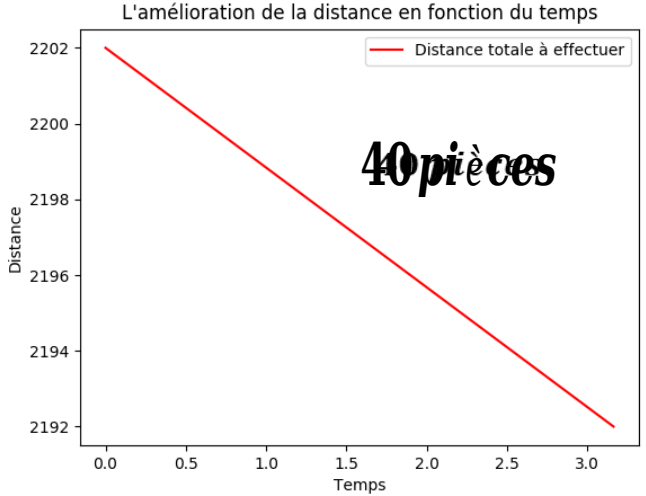
2. Implémentation

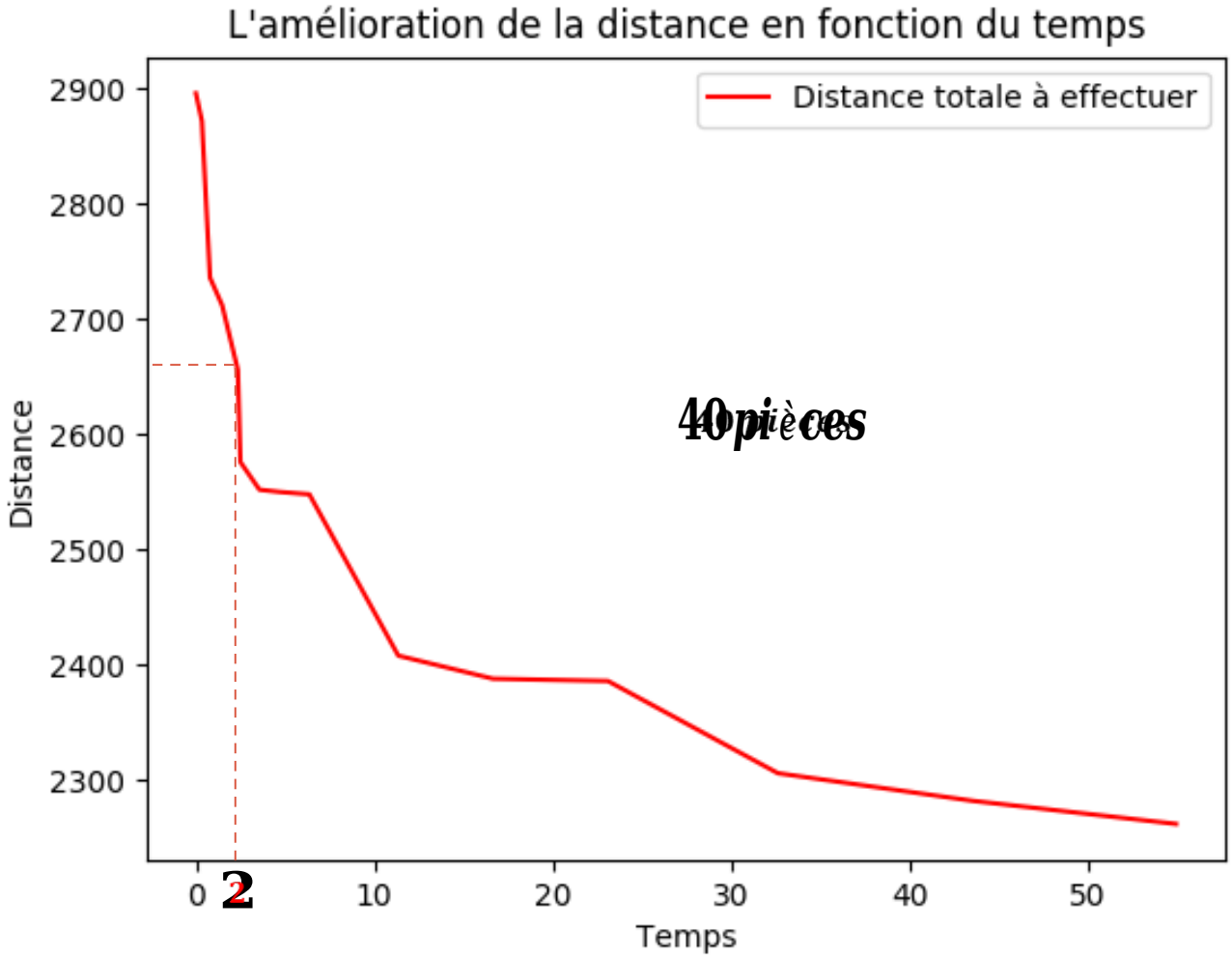
```
def run2opt(mazeMap,playerLocation,piecesOfCheese, timeAllowed):
    improvement = True
    bestRoute = piecesOfCheese
    bestDistance = totalDistance(mazeMap, playerLocation, piecesOfCheese)
    td = [bestDistance]
    te = [0]
    t0 = time.time()
    while improvement and time.time()-t0 < timeAllowed/1000:
        improvement = False
        for i in range(len(bestRoute) - 1):
            for k in range(i+1, len(bestRoute)):
                newRoute = swap2opt(bestRoute, i, k)
                newDistance = totalDistance(mazeMap, playerLocation, newRoute)
                if newDistance < bestDistance:
                    td.append(newDistance)
                    te.append(time.time()-t0)
                    bestDistance = newDistance
                    bestRoute = newRoute
                    improvement = True
                    break #improvement found, return to the top of the while loop
            if improvement:
                break
    plt.plot(te, td, color = 'r', label='Distance totale à effectuer')
    plt.xlabel('Temps')
    plt.ylabel('Distance')
    plt.legend()
    plt.title("L'amélioration de la distance en fonction du temps")
    plt.show()
    return bestRoute
```


L'amélioration de la distance en fonction du temps
Distance totale à effectuer
5 pièces
Distance
Temps
210
200
190
180
170
160
0.00
0.05
0.10
0.15
0.20
0.25


L'amélioration de la distance en fonction du temps
Distance totale à effectuer
10 pièces
Distance
Temps
750
700
650
600
550
500
450
0.0
0.5
1.0
1.5
2.0


L'amélioration de la distance en fonction du temps
Distance totale à effectuer
20 pièces
Distance
Temps
780
760
740
720
700
680
660
0.0
0.5
1.0
1.5
2.0


L'amélioration de la distance en fonction du temps
Distance totale à effectuer
40 pièces
Distance
Temps
2202
2200
2198
2196
2194
2192
0.0
0.5
1.0
1.5
2.0
2.5
3.0


L'amélioration de la distance en fonction du temps
Distance totale à effectuer
Distance
Temps
2900
2800
2700
2600
2500
2400
2300
0
10
20
30
40
50
2
40 pièces

3 – Amélioration 2 opt-sort

Sans boue, sans mur

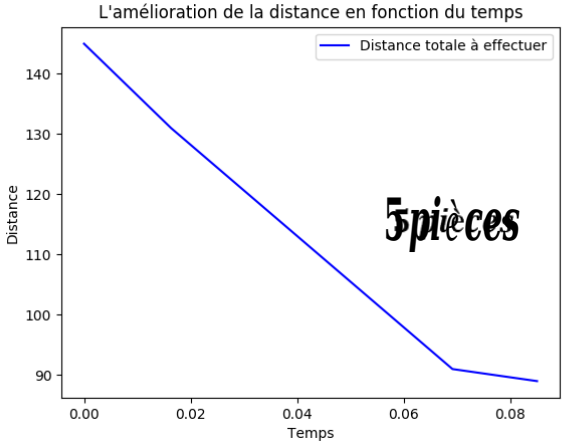
9X9		Random	Imp. Rd	BFS	Imp. BFS	Dijkstra
	Random		99	100	100	100
	Imp. Rd	1		100	100	100
	BFS	0	0		57	0
	Imp. BFS	0	0	30		30
	Dijkstra	0	0	0	60	

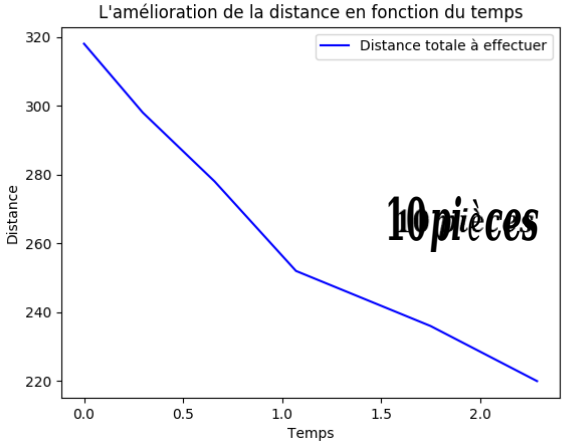
15X15		Random	Imp. Rd	BFS	Imp. BFS	Dijkstra
	Random		100	100	100	100
	Imp. Rd	0		100	100	100
	BFS	0	0		90	0
	Imp. BFS	0	0	8		9
	Dijkstra	0	0	0	89	

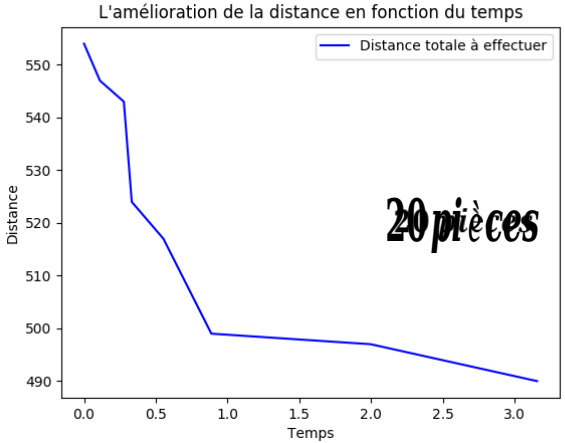
Avec boue, avec mur

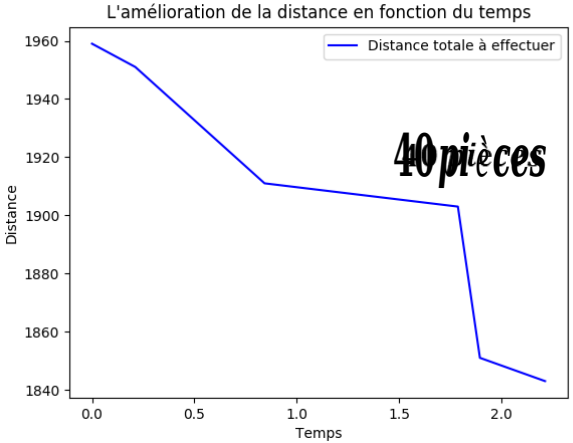
9X9		Random	Imp. Rd	BFS	Imp. BFS	Dijkstra
	Random		90	100	92	100
	Imp. Rd	8		99	95	99
	BFS	0	1		52	15
	Imp. BFS	3	5	36		61
	Dijkstra	0	1	1	25	

15X15		Random	Imp. Rd	BFS	Imp. BFS	Dijkstra
	Random		61	100	75	100
	Imp. Rd	0		100	85	100
	BFS	0	0		53	30
	Imp. BFS	0	5	46		44
	Dijkstra	0	0	3	51	

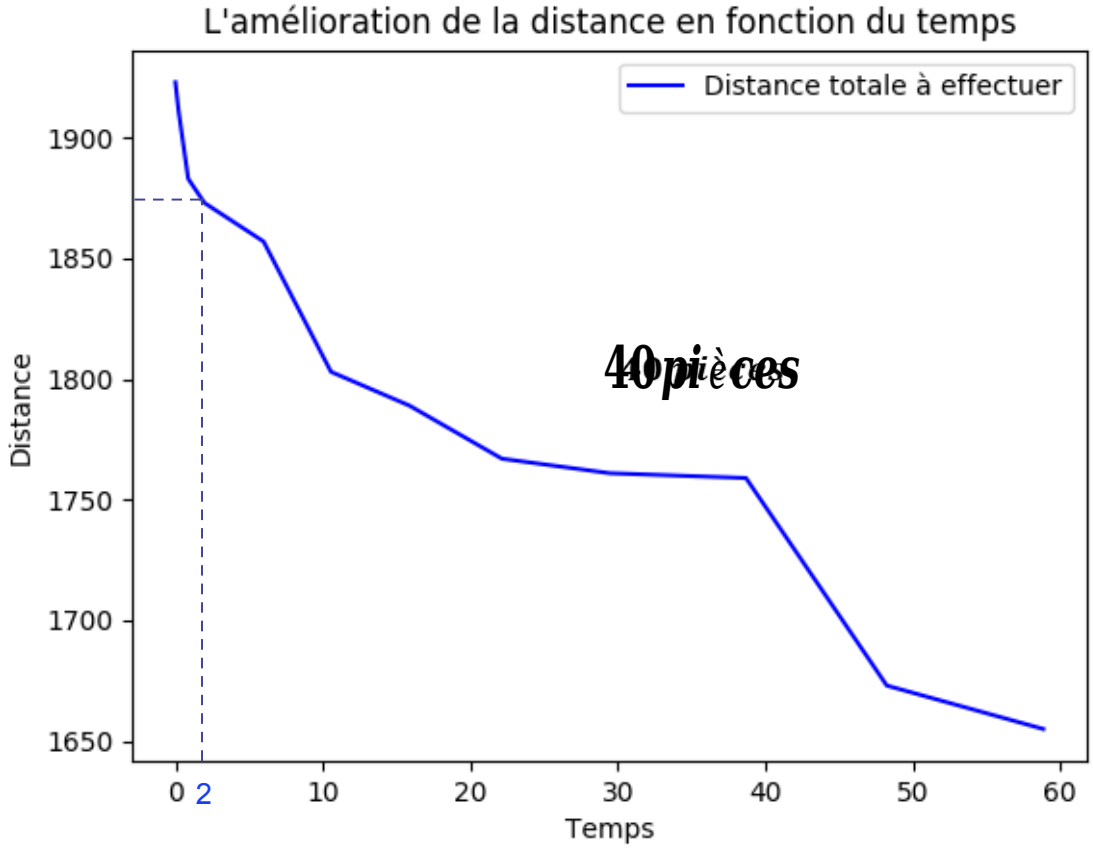

L'amélioration de la distance en fonction du temps
Distance totale à effectuer
Distance
Temps
5 pièces

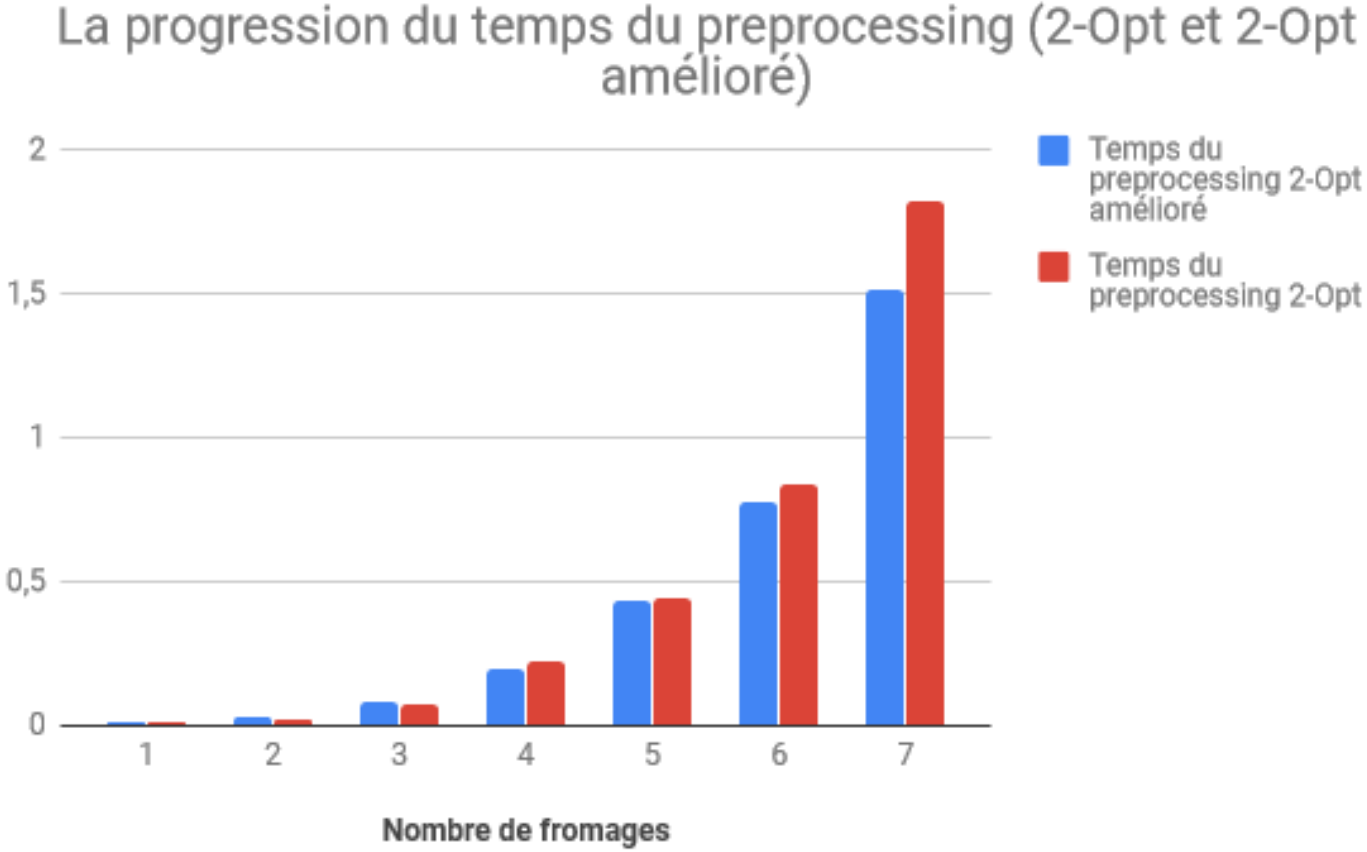

L'amélioration de la distance en fonction du temps
Distance totale à effectuer
Distance
Temps
10 pièces


L'amélioration de la distance en fonction du temps
Distance totale à effectuer
Distance
Temps
20 pièces


L'amélioration de la distance en fonction du temps
Distance totale à effectuer
Distance
Temps
40 pièces

III – Limites et améliorations



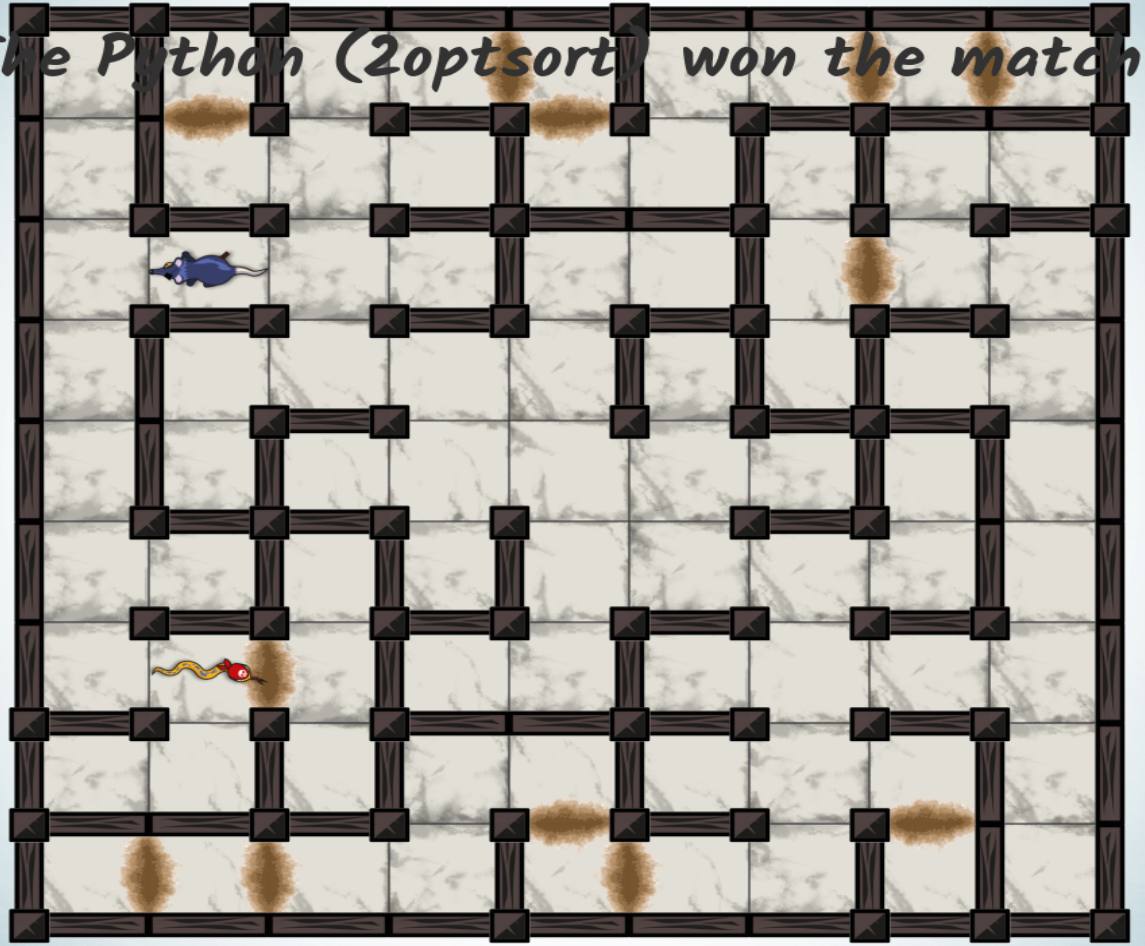

La progression du temps du preprocessing (2-Opt et 2-Opt amélioré)
2
1,5
1
0,5
0
1
2
3
4
5
6
7
Nombre de fromages
Temps du preprocessing 2-Opt amélioré
Temps du preprocessing 2-Opt

PyRat

The Python (2optsort) won the match!

greedy
Score: 4

Moves: 41
Miss: 0
Mud: 22



2optsort
Score: 5

Moves: 41
Miss: 0
Mud: 22