
Licence Informatique
Parcours Mathématiques-Informatique

Projet d'Informatique Scientifique : **PathFinding**

Fait par **AGANZE LWABOSHI Moïse**

Nantes Université

Année académique 2024-2025

Les algorithmes Weighted A* : Étude , Analyse et Conclusions

1. **Première version** : $f(x) = w * g(x) + (1 - w) * h(x)$ avec $0 \leq w \leq 1$

➤ Au regard des points D et A

Cas n°1 : $w = 0$

- $f(x) = h(x)$. On a le même comportement que l'algorithme glouton

Cas n°2 : $w = 1$

- $f(x) = g(x)$. On a le même comportement que l'algorithme Dijkstra

Cas n°3 : w compris entre 0 et 1

- Dans ce cas, on remarque que si w prend des valeurs proche de 0.5 , on obtient le même comportement que A* (production de la solution optimale mais plus rapide).
- Plus on prend des valeur proche de 0 (par exemple 0.09 , 0.11) , on a tendance à tendre à nouveau vers un comportement glouton
- Pour des valeurs aux alentours de 1, il se comporte comme Dijkstra

➤ Expérimentation et comparaison avec A*

1. Didactique : « didactic.map » , D = (12 , 5) et A = (2 , 12)

```
algoAstar
Distance D -> A      : 17.0
Number of states evaluated : 69
Temps : 0.000388 sec
```

```
w = 0.5
Distance D -> A      : 17.0
Number of states evaluated : 69
Temps : 0.006596 sec
```

```
w = 0.35
Distance D -> A      : 17.0
Number of states evaluated : 18
Temps : 0.00062 sec
```

```
w = 0.1
Distance D -> A      : 17.0
Number of states evaluated : 18
Temps : 0.000496 sec
```

2. Instance facile : « test.map » D= (28 , 23) A =(10 , 16)

```
algoAstar
Distance D -> A      : 137.0
Number of states evaluated : 836
Temps : 0.005178 sec
```

```
w = 0.56
Distance D -> A      : 137.0
Number of states evaluated : 848
Temps : 0.004333 sec
```

```
w = 0.32
Distance D -> A      : 137.0
Number of states evaluated : 696
Temps : 0.006582 sec
```

```
w = 0.12
Distance D -> A      : 137.0
Number of states evaluated : 70
Temps : 0.001327 sec
```

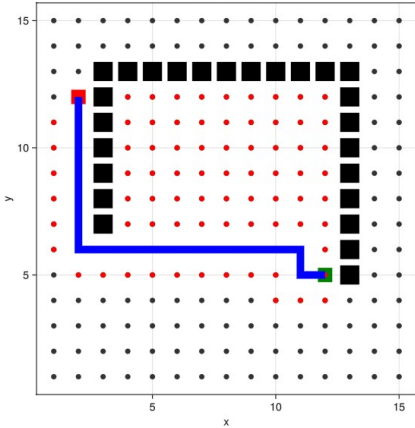
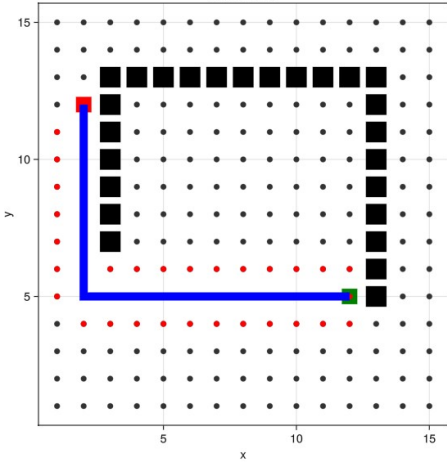
3. Instance difficile : : « theglave.map » D = (189 , 193) A = (226 , 437)

```
algoAstar
Distance D -> A      : 335.0
Number of states evaluated : 10441
Temps : 0.299096 sec
```

w= 0.95 Distance D -> A: 335.0 Number of states evaluated:74937 Temps : 10.163971 sec	w= 0.5 Distance D -> A : 335.0 Number of states evaluated:10441 Temps : 0.427468 sec	w = 0.25 Distance D -> A: 335.0 Number of states evaluated:2871 Temps : 0.302578 sec	w = 0.11 Distance D -> A: 709.0 Number of states evaluated : 282 Temps : 0.300213 sec
--	---	---	--

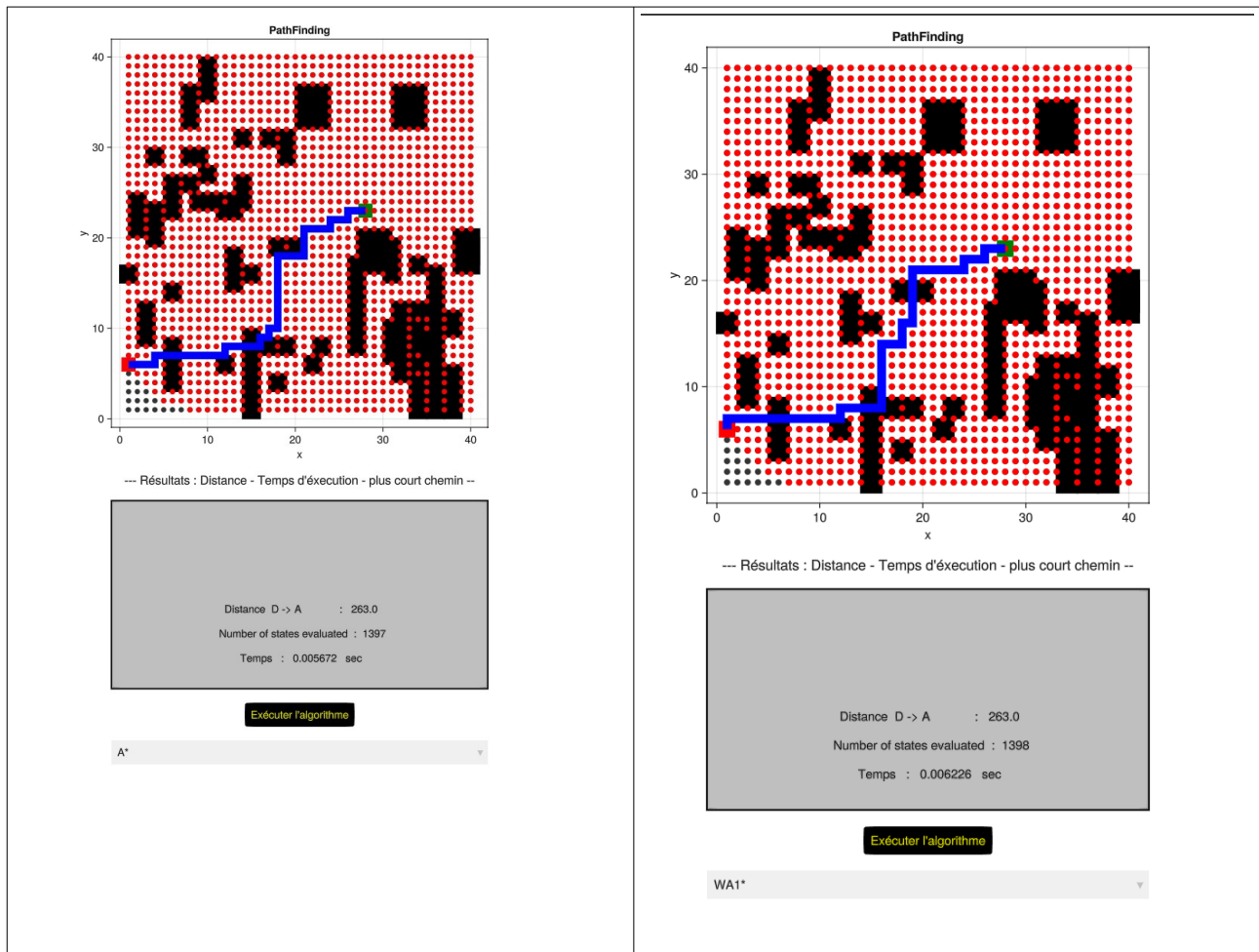
visualisation graphique

Avec « didactic.map » D = (12 , 5) , A = (2 , 12) et w = 0.25

Algorithme A*	Algorithme WA*
<div><p>PathFinding</p><p>--- Résultats : Distance - Temps d'exécution - plus court chemin ---</p><div><p>Distance D -> A : 17.0 Number of states evaluated : 69 Temps : 0.052983 sec</p></div><p>Exécuter l'algorithme</p><p>A*</p></div>	<div><p>PathFinding</p><p>--- Résultats : Distance - Temps d'exécution - plus court chemin ---</p><div><p>Distance D -> A : 17.0 Number of states evaluated : 18 Temps : 0.000257 sec</p></div><p>Exécuter l'algorithme</p><p>WA1*</p></div>

Avec « test.map » D= (28 , 23) , A =(10 , 16) et w = 0.56

Algorithme A*	Algorithme WA*
---------------	----------------



2. Deuxième version : $f(x) = g(x) + w^* h(x)$ avec $w \geq 1$ statique

➤ Au regard des points D et A

Cas n°1 : si $w = 1$, $f(x) = g(x) + h(x)$. On se ramène à A*

Cas n°2 : $w > 1$

Stratégie de réglage de W :

- Plus on prend un W grand, plus on va plus vite
- Plus on prend des grandes valeurs (à partir de 2.0), on constate que notre algorithme tend à avoir un comportement glouton.
- A partir de $w \geq 5.0$, On obtient le même résultat que l'algorithme glouton

➤ Expérimentation et comparaison avec A*

- Didactique : « didactic.map », D = (12, 5) et A = (2, 12)

```
algoAstar
Distance D -> A : 17.0
Number of states evaluated : 69
Temps : 0.000388 sec
```

w = 1.0001 Distance D -> A : 17.0 Number of states evaluated: 18 Temps : 0.000824 sec	w= 1.9 Distance D -> A : 17.0 Number of states evaluated : 18 Temps : 0.001063 sec	w = 9.9 Distance D -> A : 17.0 Number of states evaluated : 18 Temps : 0.001041 sec	w = 1.0 Distance D -> A : 17.0 Number of states evaluated : 69 Temps : 0.001203 sec
--	---	--	--

- Instance facile : « test.map » D = (28 , 23) A = (10 , 16)

algoAstar
Distance D -> A : 137.0
Number of states evaluated : 836
Temps : 0.005178 sec

W= 1.000001 Distance D->A:137.0 Number of states evaluated : 833 Temps : 0.004865 sec	W= 1.5 Distance D->A:137.0 Number of states evaluated : 787 Temps : 0.004367 sec	W=2.5 Distance D->A:137.0 Number of states evaluated : 593 Temps : 0.003917 sec	W= 4.5 Distance D->A:137.0 Number of states evaluated : 215 Temps : 0.002001 sec	W= 1000.5 Distance D->A:137.0 Number of states evaluated : 67 Temps : 0.001266 sec
--	---	--	---	---

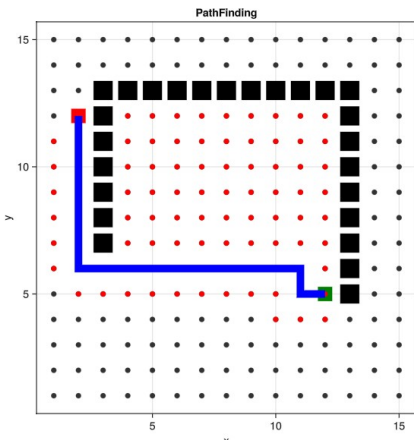
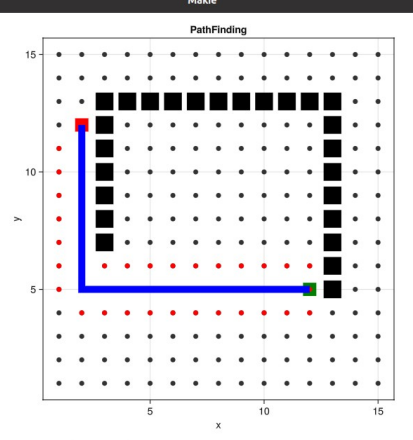
- Instance difficile : « theglave.map » D = (189 , 193) A = (226 , 437)

algoAstar
Distance D -> A : 335.0
Number of states evaluated : 10441
Temps : 0.299096 sec

w = 1.5 Distance D -> A : 335.0 Number of states evaluated:4261 Temps : 0.941897 sec	w = 2.5 Distance D -> A : 335.0 Number of states evaluated:2733 Temps : 0.864114 sec	w= 1.00001 Distance D -> A : 335.0 Number of states evaluated:9143 Temps : 0.360035 sec	w= 78.5 Distance D -> A : 709.0 Number of states evaluated : 282 Temps : 0.259365 sec
---	---	--	--

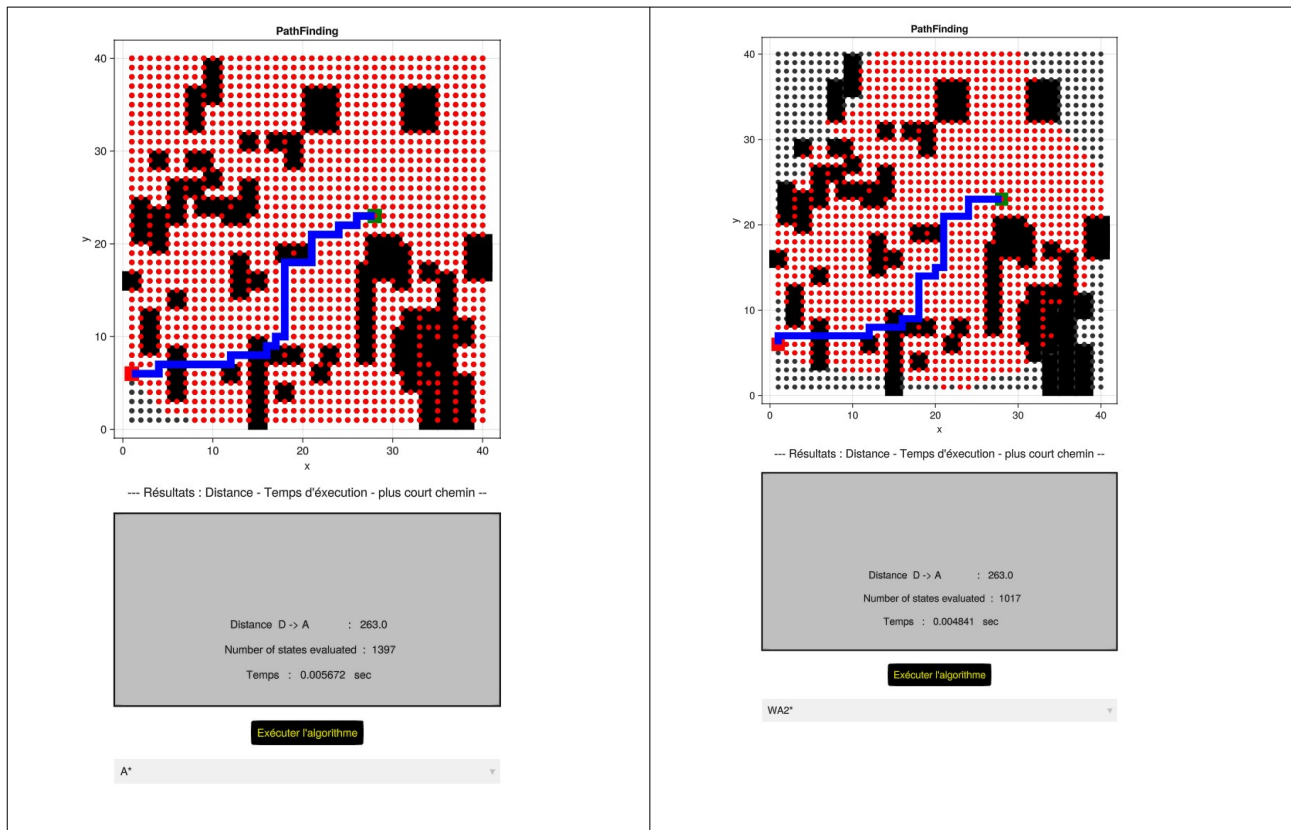
visualisation graphique

Avec « didactic.map » D = (12 , 5) , A = (2 , 12) et w = 1.5

Algorithme A*	Algorithme WA*
 <p>--- Résultats : Distance - Temps d'exécution - plus court chemin ---</p> <p>Distance D->A : 17.0 Number of states evaluated : 69 Temps : 0.052983 sec</p> <p>Exécuter l'algorithme</p> <p>A*</p>	 <p>--- Résultats : Distance - Temps d'exécution - plus court chemin ---</p> <p>Distance D->A : 17.0 Number of states evaluated : 18 Temps : 0.000256 sec</p> <p>Exécuter l'algorithme</p> <p>WA2*</p>

Avec « test.map » D = (28 , 23) , A = (10 , 16) et w = 3.9

Algorithme A*	Algorithme WA*
---------------	----------------



3. Troisième version : $f(x) = g(x) + w^* h(x)$ avec $w \geq 1$ dynamique

➤ Au regard des points D et A

Cas n°1 : si $w = 1$, $f(x) = g(x) + h(x)$. On se ramène à A*

Cas n°2 : $w > 1$

Stratégie de réglage de W :

Dans cette version, je règle W en fonction de la nature de mon graphe :

- en faisant le parcours de mon graphe, je vérifie le nombre d'arêtes que mon sommet actuel possède et je me fixe une limite.
- Si son nombre d'arêtes est ≤ 4 , je suppose que mon graphe est clairsemé et là j'augmente la valeur de w pour aller plus vite : gain en rapidité
- si son nombre d'arêtes > 4 , je suppose que mon graphe est dense et là je diminue la valeur de w pour préserver l'optimalité.

➤ Expérimentation et comparaison avec A*

- Didactique : « didactic.map » , $D = (12, 5)$ et $A = (2, 12)$

algoAstar
Distance D->A : 17.0
Number of states evaluated : 69
Temps : 0.000388 sec

w= 9.9
 Distance D -> A : 17.0
 Number of states evaluated : 18
 Temps : 0.001084 sec

w = 1.0001
 Distance D -> A : 17.0
 Number of states evaluated : 18
 Temps : 0.00088 sec

w= 1.0
 Distance D -> A : 17.0
 Number of states evaluated : 18
 Temps : 0.00054 sec

- Instance facile : « test.map » D = (28 , 23) A = (10 , 16)

algoAstar
 Distance D -> A : 137.0
 Number of states evaluated : 836
 Temps : 0.005178 sec

w = 1.5
 Distance D -> A : 137.0
 Number of states evaluated : 778
 Temps : 0.004408 sec

w = 2.5
 Distance D -> A : 137.0
 Number of states evaluated : 593
 Temps : 0.003376 sec

w = 1000.5
 Distance D -> A : 137.0
 Number of states evaluated : 593
 Temps : 0.003494 sec

- Instance difficile : « theglave.map » D = (189 , 193) A = (226 , 437)

algoAstar
 Distance D -> A : 335.0
 Number of states evaluated : 10441
 Temps : 0.299096 sec

w = 1.00001
 Distance D -> A : 335.0
 Number of states evaluated:8239
 Temps : 0.400199 sec

w= 1.5
 Distance D -> A : 335.0
 Number of states evaluated:3346
 Temps : 0.301308 sec

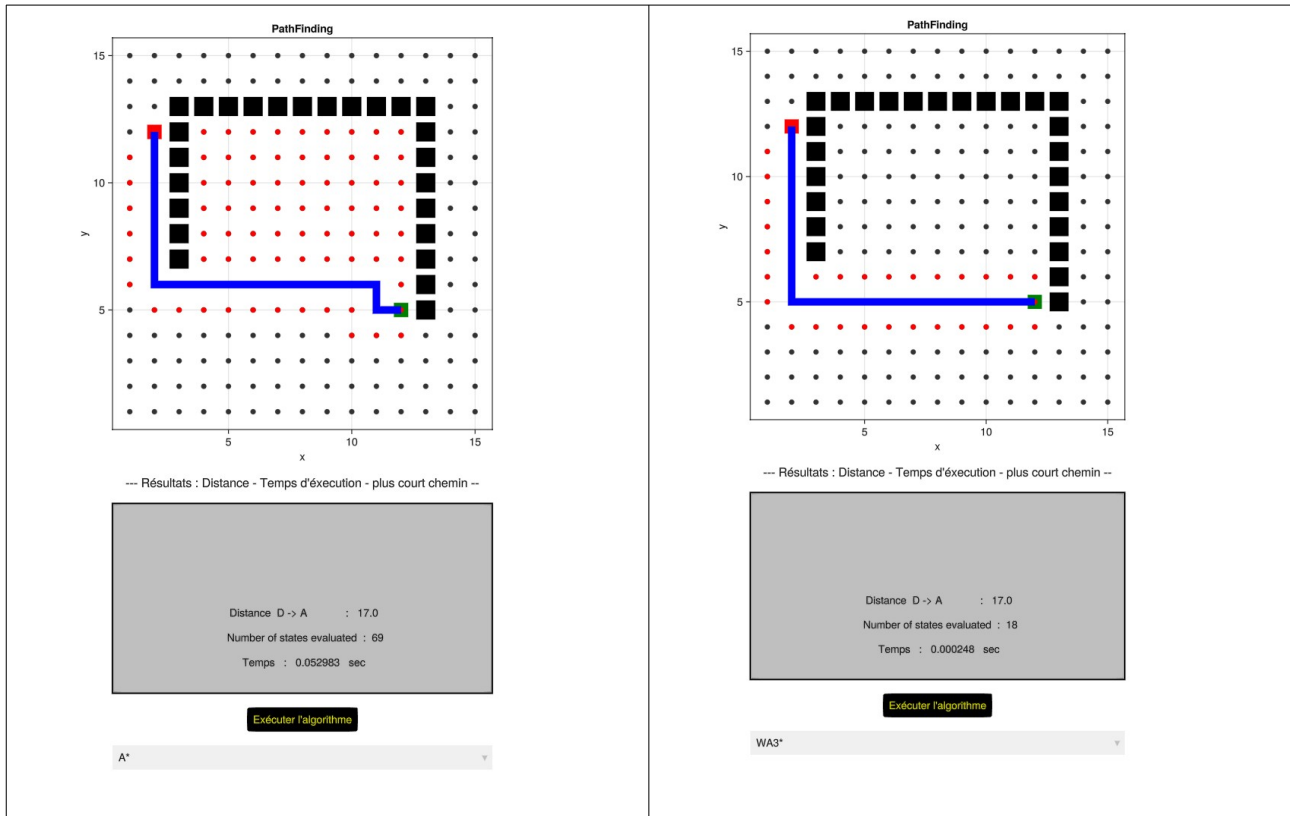
w = 2.5
 Distance D -> A : 335.0
 Number of states evaluated:2733
 Temps : 0.384319 sec

w = 78.5
 Distance D -> A : 335.0
 Number of states evaluated:2733
 Temps : 0.79718 sec

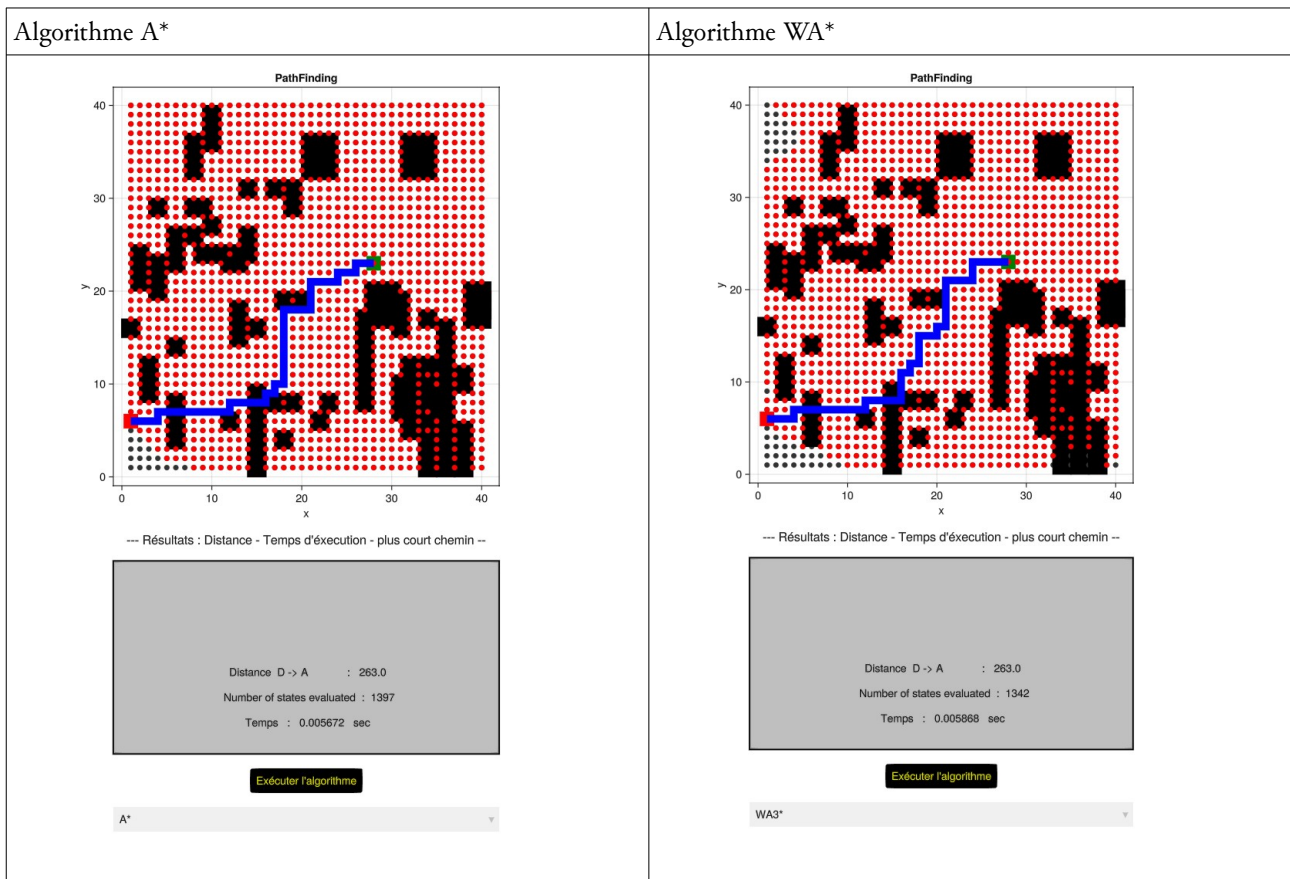
visualisation graphique

Avec « didactic.map » D = (12 , 5) , A = (2 , 12) et w = 4.5

Algorithme A*	Algorithme WA*
---------------	----------------



Avec « test.map » $D = (28, 23)$, $A = (10, 16)$ et $w = 4.5$



4. Conclusion

Les différentes versions d'algorithme de Weighted A* impactent sur la rapidité de la recherche du plus court chemin. Cependant elles ne garantissent pas toujours l'optimalité de ce chemin. A* explore plus de nœuds pour garantir l'optimalité, tandis que les WA* tendent à réduire l'espace de recherche et augmentent le risque d'une solution sous-optimale.

- La première version permet d'ajuster la distance réelle et l'heuristique, ce qui s'avère utile dans certains cas. Comme inconvénient, un mauvais choix de w peut soit ralentir la recherche (si trop proche de 1), soit produire des solutions erronées (si trop proche de 0).
- La deuxième version accélère la recherche et priorise la vitesse à l'optimalité. Pour une valeur w assez grande, on obtient des solutions loin de l'optimalité.
- La troisième version permet d'allier les meilleurs de deux mondes : rapidité + optimalité. Avec cette version nous avons un meilleur équilibre en vitesse et en qualité de la solution produite, cette version est plus adaptable aux différents types de graphe (dense, clairsemé) et elle a aussi tendance à toujours s'approcher de l'optimalité.