ARTIGOUHA Noémy
FICAT Timothée
3MIC D

Année 2015/2016

Livrable
BE Graphe

A l'attention de:
JOZEFOWIEZ Nicolas
CAPELLE Mikaël

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

SOMMAIRE

Présentation générale

Ce document a pour but de confirmer la validité de nos algorithmes ainsi que de prouver son efficacité via des tests de performance. Les deux algorithmes implémentés consistent en la recherche du plus court chemin entre deux nœuds d'une carte. Ces deux algorithmes sont l'algorithme de Dijkstra (1959) et sa version A-Star (1968). Nous vérifierons également que ces algorithmes sont suffisamment rapides pour avoir une application dans le monde réel.

Il faut noter que ces tests ont été réalisés sur un ordinateur comportant les caractéristiques suivantes :

CPU: Intel Core i3-2120 CPU @ 3.30GHz * 4
RAM: 8Go
Système d'exploitation: UBUNTU 14.04 LTS, 64 bits
Nous avons augmenté de 3G la mémoire de la JVM.

Remarque : Tous les tests de validité ont été réalisés avec une sortie graphique en affichant le déroulement des algorithmes. Le temps d'exécution est donc relativement augmenté. Les tests de performances ont été réalisés sans sortie graphique.

I – Tests de validité

Dans cette partie, le but est de prouver que nos algorithmes fonctionnent, c'est à dire, qu'ils trouvent bien le plus court chemin entre une origine et une destination lorsque ces sommets sont connexes. Pour valider la majorité des tests de cette partie, nous nous sommes appuyés des résultats que fourni Google Maps. Nos algorithmes proposent de trouver le plus court chemin en temps et en heure.

1) France

a. Toulouse (566606) – Paris (999145)

Carte : france
Résultats de Google Maps: 5h58 soit 358 min, 679km.

Version	Mode	Coût	Nombre max d'éléments dans le tas	Nombre de sommets explorés	Nombre de sommets marqués	Temps d'exécution (secondes)
Standard	Temps	333.90 min	3265	1432490	1429571	12.56
	Distance	643.52 km	1284	1618888	1617627	8.11
A-Star	Temps	333.90 min	3226	230787	227578	2.345
	Distance	643.52 km	1874	203732	201963	2.927

Nous avons un taux de ressemblance en distance et en temps de 94% avec les résultats annoncés par Google Maps. Ceci est tout à fait raisonnable, puisque Google Maps doit être beaucoup plus précis en termes d'informations par rapport à nos cartes. De plus, il est difficile de sélectionner les mêmes nœuds d'origine et de destination que ceux utilisés par Google Maps. On constate également que l'A-Star est bien mis en valeur ici. En effet, même s'il y a un nombre supérieur d'éléments dans le tas (en distance), on a énormément réduits le nombre de sommets parcourus ainsi que le temps d'exécution. La mémoire est donc beaucoup moins utilisée.

b. Biarritz (2435296) - Strasbourg (1071814)

Carte : france
Résultats de Google Maps: 10h12 soit 612 min, 1161km.

Version	Mode	Coût	Nombre max d'éléments dans le tas	Nombre de sommets explorés	Nombre de sommets marqués	Temps d'exécution (secondes)
Standard	Temps	585.78 min	4895	2414829	2414425	17.01
	Distance	1037.152 km	1824	2413547	2413391	18.15
A-Star	Temps	585.78 min	6510	1419447	1415285	12.73
	Distance	1037.152 km	2447	763616	761671	8.1

Comme pour le test Toulouse – Paris, nos résultats restent très proches des résultats qu'annonce Google Maps (93% de ressemblance). On peut donc penser que nos résultats annoncent bien le plus court chemin.

2) Midi Pyrénées (119963 – 96676)

Carte : midip
Résultat attendu : un peu plus de 200 minutes (Cf. sujet).

Version	Mode	Coût	Nombre max d'éléments dans le tas	Nombre de sommets explorés	Nombre de sommets marqués	Temps d'exécution (secondes)
Standard	Temps	200.69 min	801	142177	141927	0.8
	Distance	291,35 km	464	140216	140077	0.8
A-Star	Temps	200.69 min	1099	123374	122474	0.89
	Distance	291,35 km	926	92637	92193	0.62

Nous retrouvons bien le résultat attendu en temps. Nous sommes donc surs que nos algorithmes déterminent le plus court chemin en temps.

3) INSA (2 – 139)

Carte : insa
Résultats attendus : 2,235 minutes (Cf. sujet)

Version	Mode	Coût	Nombre max d'éléments dans le tas	Nombre de sommets explorés	Nombre de sommets marqués	Temps d'exécution (secondes)
Standard	Temps	2.235 min	10	68	61	0.002
	Distance	1.171 km	11	72	63	0.015
A-Star	Temps	2.235 min	10	66	58	0.008
	Distance	1.171 km	10	51	43	0.007

Nous retrouvons bien le résultat attendu en temps. Nous sommes donc surs que nos algorithmes déterminent le plus court chemin en temps.

4) Partage du plus court chemin en deux

Nous nous sommes servis pour ce test de la carte INSA. Nous avons récupéré un nœud sur le plus court chemin entre le nœud 2 et le nœud 139. Le nœud que nous avons récupéré est le 126
Nous avons ensuite calculé le plus court chemin entre le nœud 2 et 126 (546 m) et le plus court chemin entre le nœud 126 et 139 (625 m). Nous avons additionné ces deux résultats et avons pu constater que leur somme donné le même résultat obtenu pour le plus court chemin entre le nœud 2 et 139, soit 546 + 625 = 1171m = 1.171km.

5) Carré – vérification des distances

Nous avons lancé trois Dijkstra pour déterminer le plus court chemin en distance sur la carte « carre » entre les nœuds suivants :
Le premier du nœud 5 vers le nœud 9 (un côté du carré) : 89.056 km
Le deuxième du nœud 5 vers le nœud 15 (un autre côté du carré)) : 89.056 km
Le dernier du nœud 9 vers le nœud 15) (diagonale) : 125.942 km

Grâce à ces trois résultats, nous avons pu vérifier la longueur de la diagonale : $\sqrt{(89.056^2 * 2)}$= 125.944. Ceci nous permet de conclure sur le fait que nos algorithmes fonctionnent bien.

II – Application sur des cas critiques

1) Sommet inexistant

Lorsque l'utilisateur entre un numéro de nœud (pour l'origine ou la destination), si ce nœud n'est pas trouvé sur la carte, notre programme réitère la demande du nœud à l'aide d'un *do-while*. Et ceci, tant que l'utilisateur ne saisit pas un nœud existant sur la carte.

2) Chemin nul : d'un point A vers ce même point A

Carte : insa
Résultats attendus : 0 s, 0 km
Numéro du noeud A : 139

Version	Mode	Coût	Nombre max d'éléments dans le tas	Nombre de sommets explorés	Nombre de sommets marqués	Temps d'exécution (secondes)
Standard	Temps	0 s	1	2	1	0.001
	Distance	0 km	1	2	1	0.012
A-Star	Temps	0 s	1	2	1	0.001
	Distance	0 km	1	2	1	0.002

Il n'y a pas d'erreurs à l'exécution du programme et il nous donne la bonne solution.

3) Sommets non connexes : Ajaccio (280931) - Paris (999145)

Carte : france
Résultat attendu : « Aucune solution »

Quand deux sommets ne sont pas connexes, l'algorithme va partir de l'origine et explorer tous les sommets et leurs successeurs possibles (tant qu'il y en existe). S'il a parcouru tous les successeurs mais qu'il n'a pas pu parvenir à la destination, l'algorithme s'arrête est nous retourne « Aucune solution ».

Version	Mode	Coût	Nombre max d'éléments dans le tas	Nombre de sommets explorés	Nombre de sommets marqués	Temps d'exécution (secondes)
Standard	Temps	Infinie	147	9442	9443	0.78
	Distance	Infinie	70	9459	9460	0.67
A-Star	Temps	Infinie	140	9442	9443	3.4
	Distance	Infinie	79	9459	9460	0.39

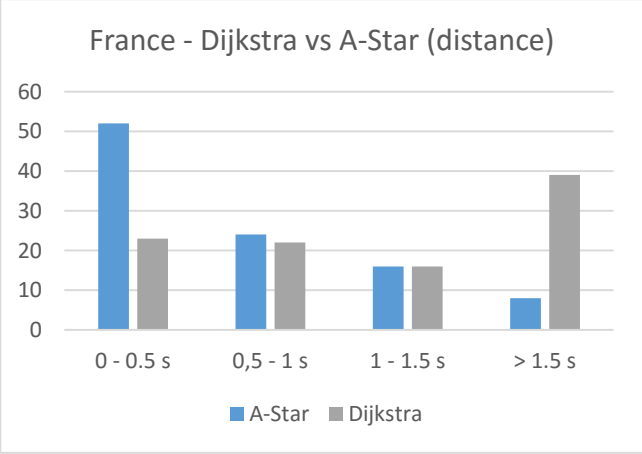
C'est ce qui se passe ici, l'algorithme parcourt tous les sommets de la Corse mais ne trouve pas de connexité avec un sommet en France. Donc le programme s'arrête.

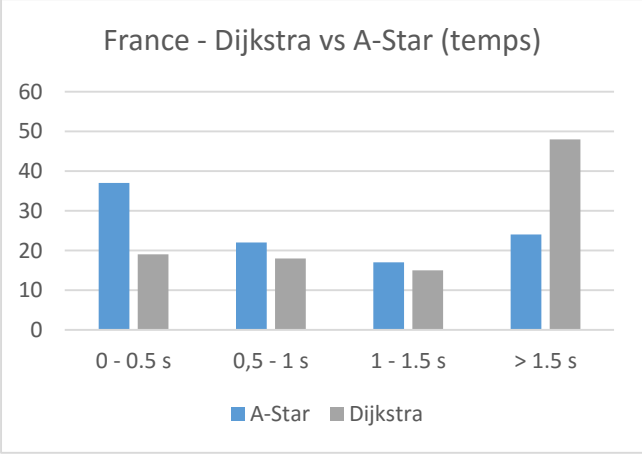
III – Tests de performance

Nous avons lancé l'algorithme de Dijkstra et sa variante A-Star 100 fois sur différentes cartes. Les sommets sont choisis aléatoirement (ces sommets sont bien évidemment compris dans les cartes) afin de tester leurs performances. Ils ont donc retourné 100 chemins différents, si c'était possible (c'est à dire s'il existe un plus court chemin).
Pour nos tests de performance nous avons choisi deux critères : le **nombre d'éléments parcourus** ainsi que le **temps d'exécution**.

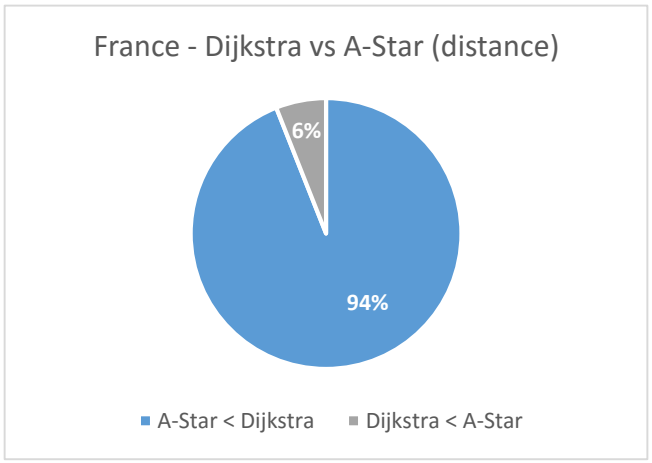
1) France

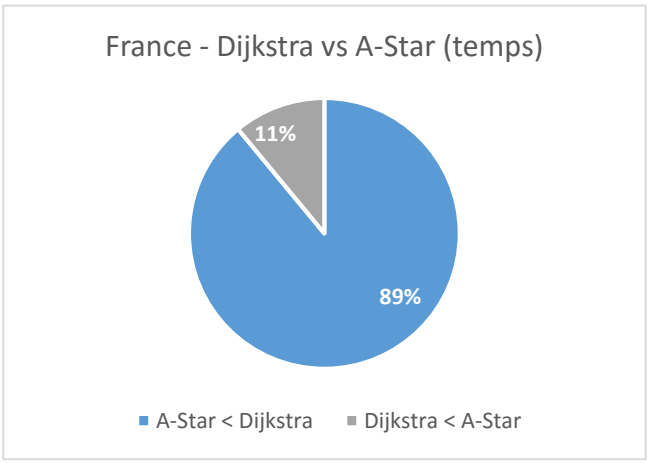
Répartition des temps d'exécution des deux algorithmes:





Rapidité des algorithmes sur les mêmes chemins :



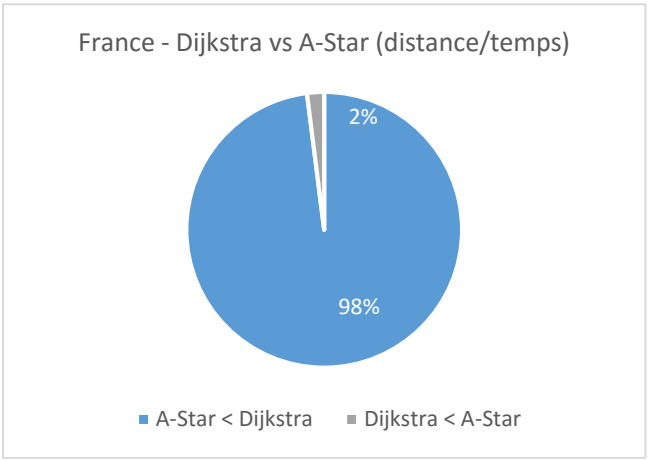


Si nous comparons les deux histogrammes, nous pouvons constater que les algorithmes sont plus lents quand ils sont lancés pour trouver le plus court chemin en temps qu'en distance (ce qui est normal puisqu'il faut faire plus de calculs).
Nous constatons également d'après les diagrammes en secteurs, que l'A-Star est nettement plus rapide que le Dijkstra.

Nombres d'éléments parcourus :

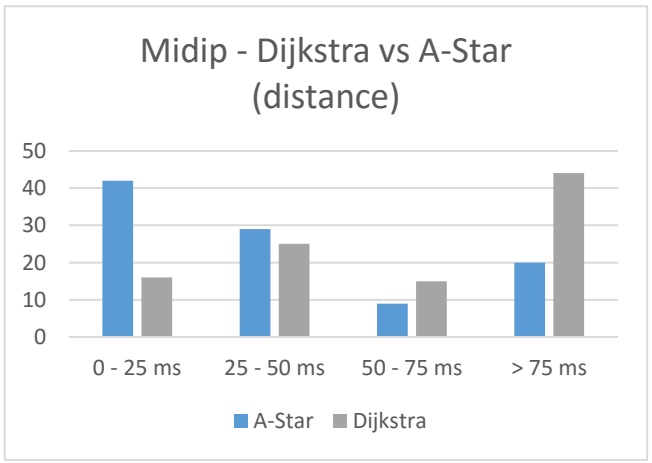
Nous pouvons voir de façon flagrante que l'A-Star parcours à 98% moins de sommets que le Dijkstra.

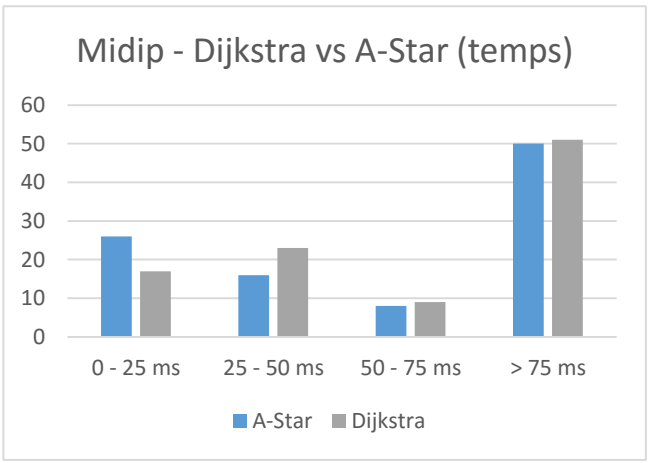
Donc pour la France, l'A-Star est nettement plus intéressant que le Dijkstra standard que ce soit en temps d'exécution ou en nombre d'éléments parcourus. Il utilise donc moins la mémoire.



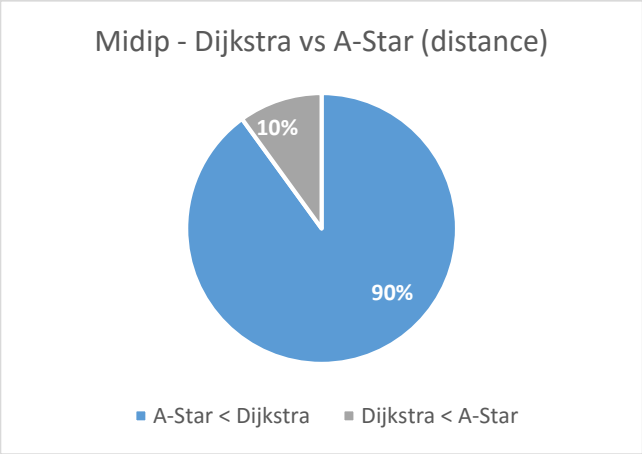
2) Midi Pyrénées

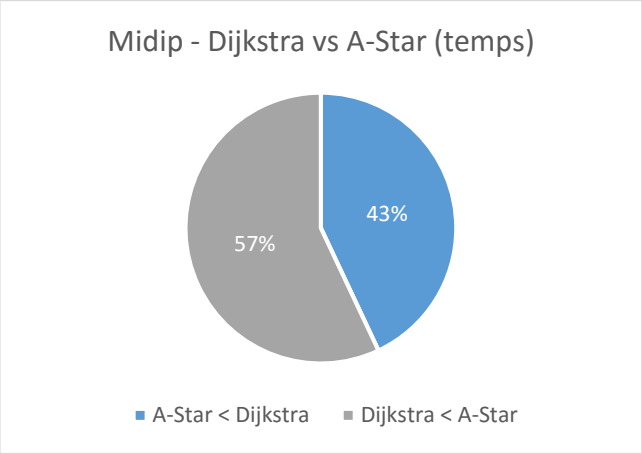
Répartition des temps d'exécution des deux algorithmes:





Rapidité des algorithmes sur les mêmes chemins :



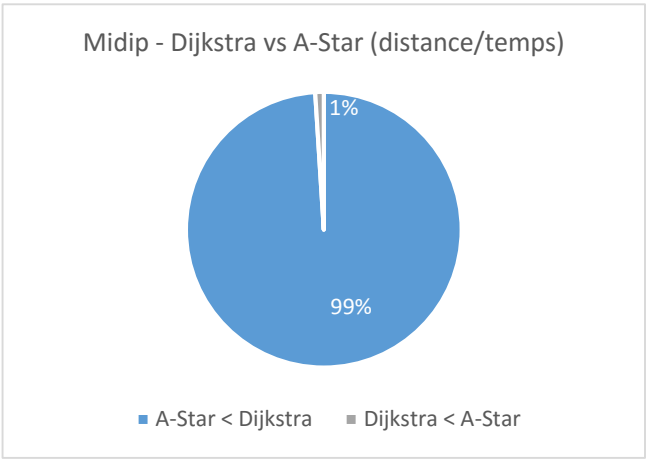


Comme pour la carte de la France, l'A-Star est plus intéressant que le Dijkstra, en distance. Cependant pour le plus court chemin en temps, leurs temps d'exécutions sont proches et nous constatons même que le Dijkstra à 57 % plus rapide que l'A-Star.

Nombres d'éléments parcourus :

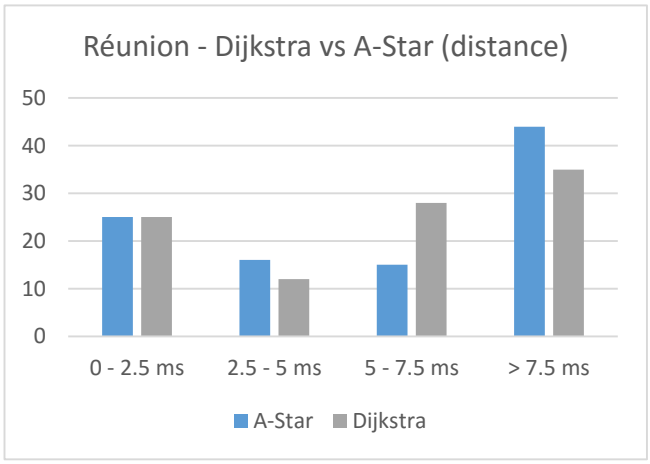
Comme pour la France, l'A-Star parcours beaucoup moins de sommets que le Dijkstra.

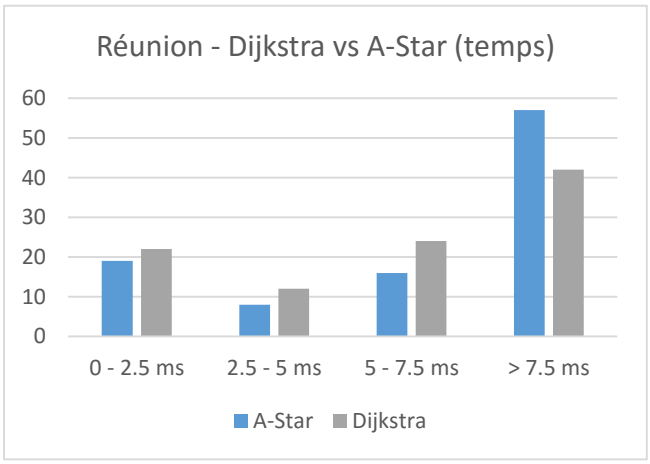
En distance, l'A-Star est donc plus intéressant que le Dijkstra en temps d'exécution sur une carte moyenne. Mais le Dijkstra est plus performant en temps. Malgré tout, l'A-Star parcours toujours considérablement moins de sommets que le Dijkstra.



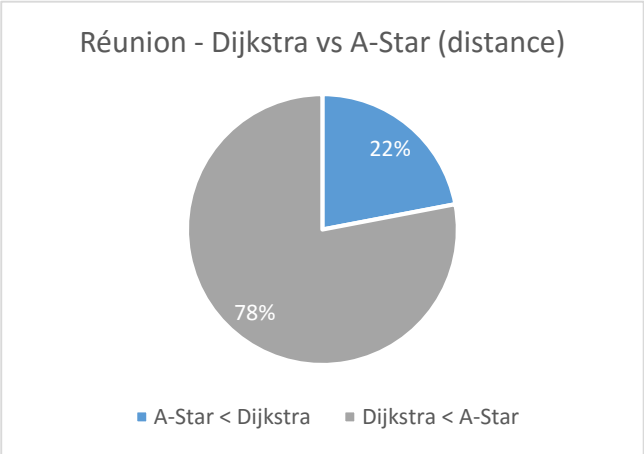
3) La Réunion

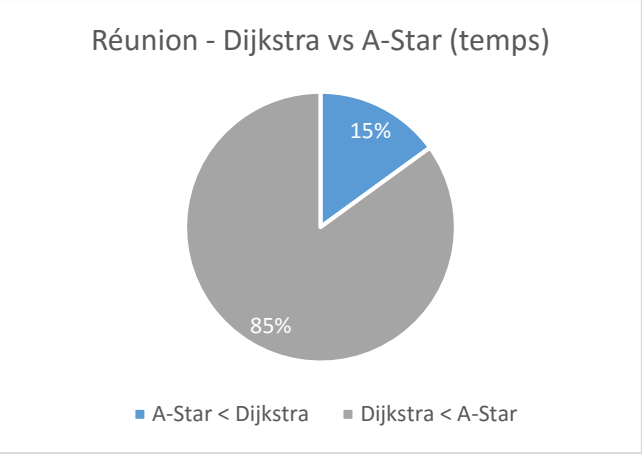
Répartition des temps d'exécution des deux algorithmes:





Rapidité des algorithmes sur les mêmes chemins :

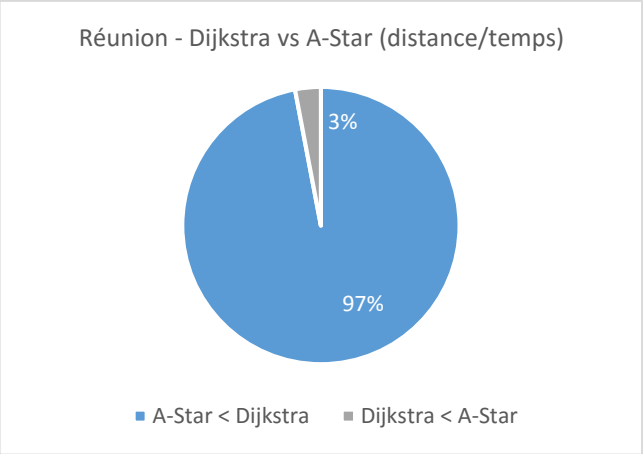




Comme nous pouvons le voir le Dijkstra que ce soit en temps ou en distance est nettement plus rapide que l'A-Star. Ceci vient du fait que la Réunion ne comporte pas de route qui traverse l'île au milieu. Comme les routes ne sont pas bien réparties, L'A-Star n'est pas vraiment utile, ni adapté à ce genre de carte. Avec son calcul à vol d'oiseau, ici inutile, il rallonge le temps de calcul et donc d'exécution.

Nombres d'éléments parcourus :

Néanmoins, l'A-Star reste une fois de plus, plus optimal en termes de nombres d'éléments parcourus.



Conclusion

Grâce à nos tests de validité nous avons pu prouver que nos algorithmes fonctionnent correctement. De plus, nous prenons en compte différents cas critiques, ce qui améliore notre programme et évite de quelconques problèmes lors de l'exécution.
Enfin, les tests de performances ont permis de mettre en exergue les forces et les faiblesses du Dijkstra et de l'A-Star.