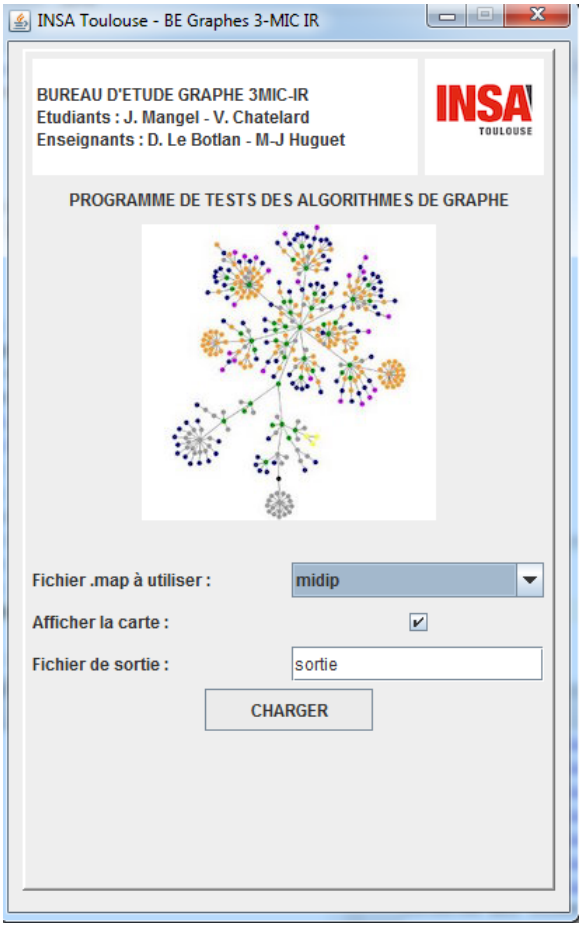
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Marie Jo-Huguet
huguet@laas.fr

Analyse de Performance



Joris Mangel - Valentin Chatelard
Année 2014/2015

Table Des Matières

Table Des Figures

1 Présentation Générale

L'objectif de ce document est de confirmer la validité de notre programme et des algorithmes qu'il implémente, maus aussi de par des tests de performance montrer l'efficacité de celui ci.
Les algorithmes implémentés consiste en la recherche du plus court chemin entre deux noeuds sur la carte.
Deux versions sont disponibles :

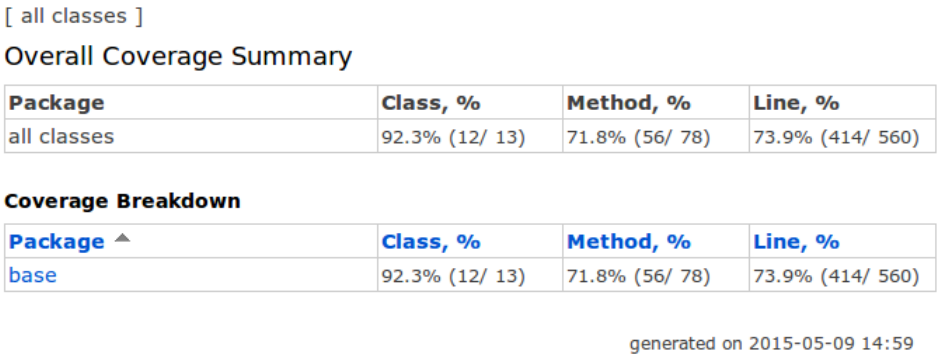
- La première est l'Algorithme de Dijkstra, étudié lors du cours de Graphes, et rédigé par Edsger Dijkstra en 1959.
- La seconde version est l'application de la version A-Star, extension de l'Algorithme de Dijkstra mis au point 9 ans plus tard n 1968 par Peter E. Hart, Nils Nilsson et Bertram Raphael.
  Celui ci présentant l'avantage d'avoir la même complexité dans le pire des cas, mais qui est plus rapide en moyenne. Cet algorithme est basé du l'ajout d'une estimation du coût restant à vol d'oiseau depuis le noeud actuel jusqu'à la destination).

Nous allons mettre en place de tests de performance afin de mesurer l'impact de l'ajout de cette composante. Nous verifierons que ces algorithmes sont suffisament rapide pour avoir une réelle application dans le monde du réel.
Ces tests sont effectués sur ordinateur comportant les caractéristiques suivantes :

- CPU : Intel i7-3610QM - 4 coeurs (2,3 GHz)
- RAM : 6 Go
- Système employé : Linux 64bits : Kubuntu
- Carte : décompressée
- Paramètre de la JVM : Augmentation à 3Go de la valeur maximale mise en ram (permettant l'utilisation rapide de toutes les cartes)

2 Tests de validité

Dans cette section il s'agira de montrer la validité du programme de part son fonctionnement et vérifier qu'il s'exécute correctement.
Pour cela le logiciel que nous utilisons IntelliJ Idea propose après lancement du programme de mesurer la couverture effective du code lors de son utilisation, donnant un gage de confiance sur le programme. Il est bien entendu (quasiment) impossible de tout tester de par certain cas critiques comme les erreurs de lectures de fichier, mais ceci donne une valeur de confiance à notre code.



[all classes]

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	92.3% (12/ 13)	71.8% (56/ 78)	73.9% (414/ 560)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
base	92.3% (12/ 13)	71.8% (56/ 78)	73.9% (414/ 560)

generated on 2015-05-09 14:59

Figure 1: Résultat du coverage après utilisation normale du programme et des fonctionnalités implémentées.

Commentaires :

- 12/13 classes utilisées : la classe dessinInvisible n'a pas été utilisée pour ce test en mode coverage mais testée en dehors
- On a environ 75 % du code qui a été utilisée sans problème, parmi le reste on peut retrouver les tests utilisateurs, cas critiques.

Ce test aura permis de vérifier la fiabilité du programme en tant qu'entité. Par la suite nous allons procédé à sa validation du point de vue des résultats algorithmiques.

3 Tests de validité de la solution

Dans cette section il va s'agir d'une campagne de test pour vérifier la validité de nos algorithmes vis à vis de leur objectif : trouver le plus court chemin d'un point à un autre.

Remarque : Ces temps d'exécution ont été déterminés lors de l'utilisation d'une sortie graphique avec affichage du déroulement des algorithmes ce qui augmente considérablement ces temps.

3.1 Test 1 : Perpignan(329425) - Dunkerque (1880445) (France)

- Carte utilisée : France
- Résultat attendu : 9h41 d'après Google Maps

Version	Mode	Coût	Nombre maximun d'éléments dans le tas	Nombre d'éléments explorés	Temps exécution
Standart	Temps	565.643 min (9h25min)	3210	2417346	35.406s
A-Star	Temps	565.643 min (9h25min)	7127	1252824	18.364s
Standart	Distance	1083.347 km	1244	2432553	30.050s
A-Star	Distance	1083.347 km	4135	959336	14.705s

Commentaires :

- On vérifie que les temps d'éxécution sont respectacles pour de l'usage réel
- Pour le choix du temps : Google maps annonce 9h41 minutes en prenant compte de ralentissements, alors que nos algorithmes (ayant des valeurs moins précises) nous donnent 9h25, soit de 2% d'écart relatif.
- Pour le choix de la distance : Google maps annonce 1115 km, ici nous somme a 1083 ce qui donne une marge d'erreur de là aussi 2% d'erreur relative. Cela reste aussi raisonnable au vu de la précision de nos valeurs.
- Concernant les performances pures : c'est sur des exemples comme ceux ci qu'on voit l'intérêt de A-Star, malgré que le nombre d'éléments max dans le tas est supérieur on a divisé par 2 le nombre d'éléments visités (grand intérêt vu l'ampleur de la carte) et on a aussi divisé par 2 les temps d'éxécution pour des résultats strictement identiques.

3.2 Test 2 : 119963 - 96676 (Midip)

- Carte utilisée : midip
- Résultat Attendu : "Un peu plus de 200 minutes" (Cf. sujet)

Version	Mode	Coût	Nombre maximun d'éléments dans le tas	Nombre d'éléments explorés	Temps exécution
Standart	Temps	200.69min	801	142178	1.146s
A-Star	Temps	200.69min	1100	123374	1.466s
Standart	Distance	291.351 km	464	140217	955ms
A-Star	Distance	291.351 km	926	92638	951ms

Les résultat notamment en temps sont conforment aux attentes notifiées dans le sujet. Nos algorithmes en détermination du chemin le plus court en temps sont donc exacts.

3.3 Test 3 : 2 - 139 (INSA)

- Carte utilisée : INSA
- Résultat Attendu : 2.235 min

Version	Mode	Coût	Nombre maximun d'éléments dans le tas	Nombre d'éléments explorés	Temps exécution
Standart	Temps	2.2348 min	10	69	1ms
A-Star	Temps	2.2348min	10	67	1ms
Standart	Distance	1.171km	11	73	1ms
A-Star	Distance	1.171 km	10	52	1ms

La encore les résultats sont conformes à nos attentes. Les deux algorithmes déterminent tous les deux le même chemin et des temps d'exécutions extrémements rapides.

3.4 Test 4 : Toulouse (140613) - Paris (999145)

- Carte utilisée : France
- Résultat Attendu : Google Maps annonce : 691 km, soit 363 minutes

Version	Mode	Coût	Nombre maximun d'éléments dans le tas	Nombre d'éléments explorés	Temps exécution
Standart	Temps	336.4895 min	3087	1455929	21.410s
A-Star	Temps	336.4895 min	3308	238674	7.436s
Standart	Distance	658.802 km	1267	1624307	14.061s
A-Star	Distance	658.802 km	1848	219933	6.203s

Ces résultats restent toujours très proches des résultats annoncés par Google Maps. Nos chemins paraissent donc être bien ceux du plus court chemin.

4 Tests de performances

Nous avons décidé, de mettre au point une exécution des 2 algorithmes Dijkstra Standart et Dijkstra A-Star sur différentes cartes avec des sommets choisis aléatoirement parmi ceux disponibles, dans le but de tester les performances de ces 2 algorithmes. Les critères choisis sont principalement la durée d'exécution et le nombre d'éléments parcourus.

4.1 Carte Midip

4.1.1 Critère choisi : Durée d'exécution

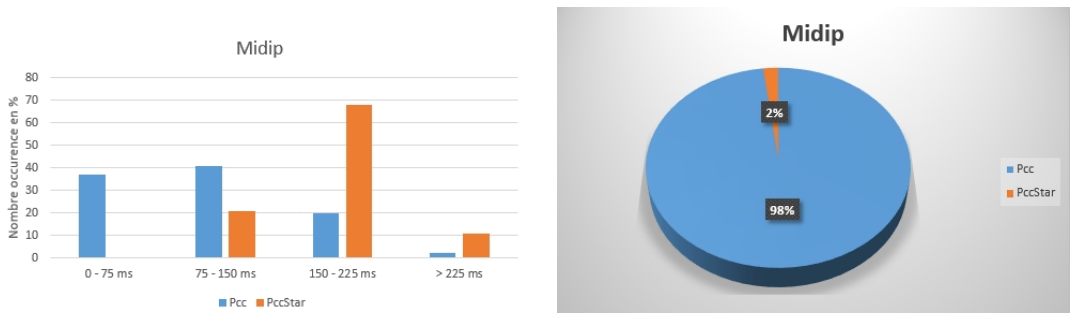


Figure 2: Résultats observés pour la carte midip avec pour choix la distance

4.1.2 Critère choisi : Nombre d'éléments parcourus

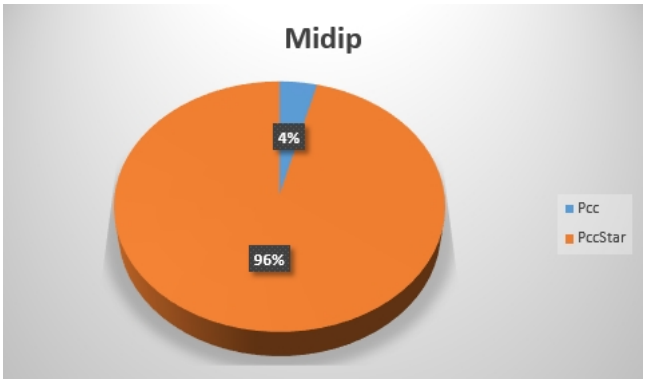


Figure 3: Comparaison du nombre d'éléments parcourus par chaque algorithmes (le minimum des 2)

Commentaires :

- La carte étant de taille moyenne, les deux algorithmes sont en général assez proches en temps d'exécution, on remarque néanmoins que sur un choix de cout en distance Dijkstra Standart est meilleur que Dijsktra A-Star du fait des calculs supplémentaires impactés par l'heuristique dans A-Star.
- Comme le montre la figure 4, le nombre d'éléments parcourus par l'algorithme d'A-Star est beaucoup plus faible et donc impact beaucoup moins la mémoire, ce qui est très important pour des systèmes embarqués par exemple.

4.2 Carte France

4.2.1 Critère choisi : Durée d'exécution

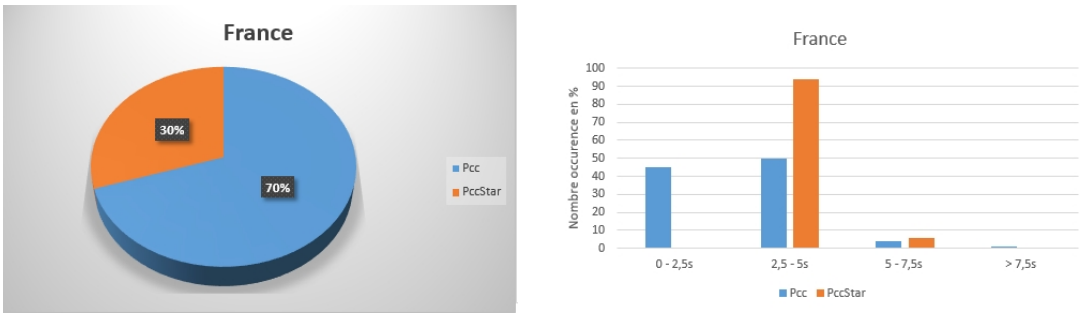


Figure 4: Durée d'exécution pour choix en distance

Commentaires :

- On remarque comme notifié sur la carte midip, que lors du choix de cout en temps, les calculs sont plus nombreux et donc le fait de parcourir moins de sommets réduit le temps d'exécution.
- L'algorithme de Dijkstra A-Star reste donc beaucoup plus intéressant que ce soit pour la durée d'exécution et pour le nombre d'éléments parcourus et donc la mémoire.

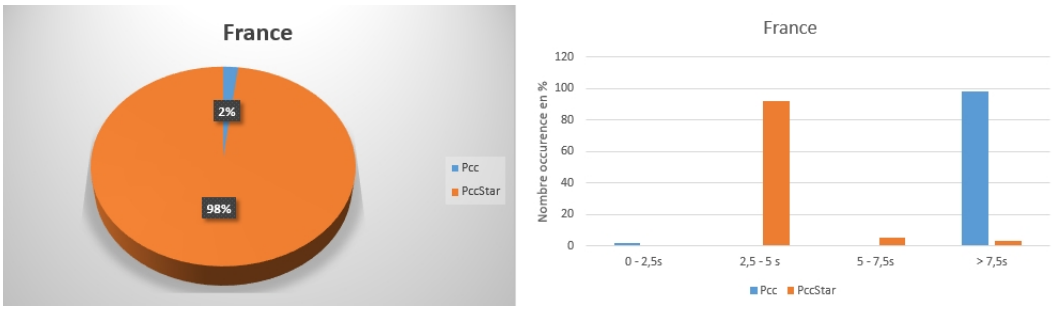


Figure 5: Durée d'exécution pour choix en temps

4.3 Carte Réunion

4.3.1 Critère choisi : Durée d'exécution

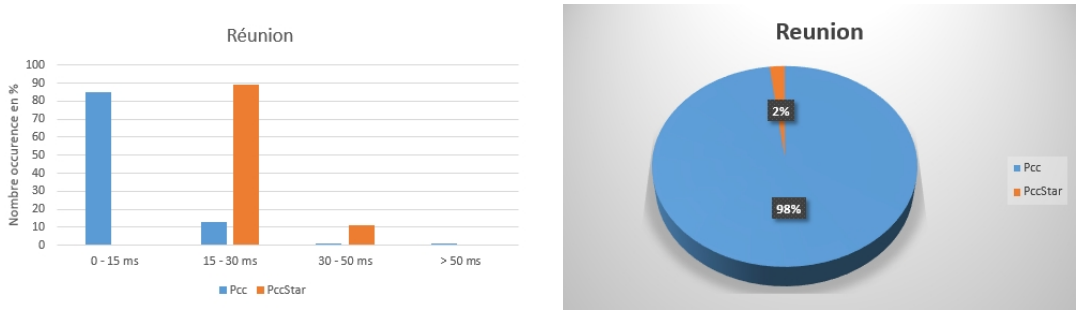


Figure 6: Résultats observés pour la carte france

Commentaires :

- Ici, nous choisissons de mettre en avant le fait que A-Star n'est pas adapté pour une carte à répartition non homogène des routes, du fait qu'aucune route ne traverse la carte. Donc cet algorithme n'apporte aucun intérêt ici et au contraire ralenti le temps de calcul.

5 Application sur cas critiques

5.1 Chemin nul

- Carte utilisée : midip
- Résultat Attendu : 0 s - 0 km

Version	Mode	Coût	Nombre maximun d'éléments dans le tas	Nombre d'éléments explorés	Temps exécution
Standart	Temps	0min	1	2	35ms
A-Star	Temps	0min	1	2	220ms
Standart	Distance	0km	1	2	32ms
A-Star	Distance	0km	1	2	167ms

Le test nous fournit en effet la bonne solution, il n'y a pas d'erreur de programmation.

5.2 Sommets non connexe : Corse (2243690) - Toulouse (140613)

- Carte utilisée : France
- Résultat Attendu : Infinity

Version	Mode	Coût	Nombre maximun d'éléments dans le tas	Nombre d'éléments explorés	Temps exécution
Standart	Temps	Infinity min	137	9460	791ms
A-Star	Temps	Infinity min	147	9460	6050ms
Standart	Distance	Infinity km	85	9460	523ms
A-Star	Distance	Infinity km	98	9460	6117ms

Les algorithmes commencent par explorés tous les sommets de Corse sans arriver à trouver de connexité avec un sommet en France. Le programme s'arrète donc, la réponse est bien celle attendue.

5.3 Sommets Inexistants -1 et -5

- Carte utilisée : midip
- Résultat Attendu : Gestion de l'erreur à l'exécution

Erreur signalée à l'exécution sans problème - pris en compte dans l'implémentation.

6 Conclusion

Ce document a pour but de donner une validité à notre programme. Nous avons à l'aide des tests de validité prouvé que notre programme fonctionne correctement et qu'un gage de confiance peut lui être accordé.
Par la suite les tests de validité de la solution ont prouvés la validité de nos algorithmes.
Enfin les tests de Performances (comparaison des 2 algorithmes) à permis de mettre en avant les forces et faiblesses de chacun d'eux, tout en montrant leur temps d'exécution plus que convenable et une mémoire utilisée convenable. En notant qu'A-Star sera plus adapté lors de système embarqué afin de limité l'utilisation abusive de la mémoire.
Nous avons terminé par montrer les cas critiques bien entendu prévus et traités par notre programme afin d'éviter d'éventuel problème à l'exécution.