Tests de performance

6 mai 2019

Le but de cette étape est d'analyser les raisons des bonnes ou mauvaises performances des algorithmes Dijkstra et A-star. Ces deux algorithmes sont supposés corrects (cela a été validé lors d'une précédente étape). Les performances de ces deux algorithmes doivent être analysés séparément puis comparativement.

Lire l'ensemble du document avant de commencer.

1 Contexte expérimental

Avant tout, il est nécessaire de déterminer les jeux de test à effectuer. Il faut donc fixer :

- la carte (routières, non routières, de taille variable)
- la nature des coûts (en distance et en temps)
- le nombre de paires (origine, destination) et leurs identifiants respectifs

Il peut également être intéressant de collecter des informations sur les graphes associés au carte (nombre de sommets, d'arcs) ou sur la nature des chemins (comportant plus ou moins d'arcs). On pourrait proposer pour une même carte un ensemble de paires (origine, destination) correspondant à des chemins avec "peu" d'arcs, un nombre "moyen" d'arcs ou un "grand" nombre d'arcs.

Il est également nécessaire de déterminer les résultats attendus de ces tests. Pour un test donné (carte, nature du coût, origine, destination) des éléments de résultats peuvent être :

- la valeur de la solution ;
- le temps de calcul ;
- le nombre de sommets explorés (ie. entrés dans le tas) ;
- le nombre de sommets marqués (ie. sortis du tas) ;
- la taille maximale du tas au cours de l'algorithme ;
- ...

N'oubliez pas de noter les caractéristiques de la machine de calcul.

2 Création des jeux de test

Les tests doivent être en nombre significatif pour permettre une analyse fiable des résultats. Il est donc indispensable de concevoir un programme pour les réaliser.

Dans cette partie, vous allez écrire un programme permettant de créer un jeu de test. Ce programme doit générer deux fichiers appelés : `nom-carte_distance_nombre-test_data.txt` et `nom-carte_temps_nombre-test_data.txt` comportant les mêmes paires origine-destination. Par exemple :

fichier `haute-garonne_distance_100_data.txt`	fichier `haute-garonne_temps_100_data.txt`
haute-garonne	haute-garonne
0 // 0 pour distance et 1 pour temps	1 // 0 pour distance et 1 pour temps
100 // nombre de paires origine-destination	100 // nombre de paires origine-destination
o_1 d_1	o_1 d_1
...	...
o_{100} d_{100}	o_{100} d_{100}

Attention Les paires (origine-destination) doivent toutes permettre d'obtenir un résultat (ie. pas de chemin impossible) afin de garantir que l'on ait bien le bon nombre de tests. Pour cela, lors de la sélection d'une paire origine-destination, il faut s'assurer de l'existence d'un chemin (ou de l'appartenance à une même composante connexe).

Les tests doivent être reproductibles, vous pouvez initialiser de manière déterministe l'utilisation d'un générateur aléatoire afin d'obtenir les mêmes jeux de données.

Pour écrire votre programme, regardez la partie "Création d'un programme from scratch" de l'étape 2. Tentez d'estimer la durée de calcul nécessaire pour le déroulement des jeux de tests que vous prévoyez.

3 Création des fichiers de résultats

Modifiez vos algorithmes pour qu'ils écrivent les résultats attendus dans un fichier de sortie pour chacun des algorithmes.
Par exemple, à partir du fichier de jeux de tests `nom-carte-data_distance_nombre-test.txt` on peut produire les fichiers `nom-carte_distance_nombre-test_dijkstra.txt` et `nom-carte_distance_nombre-test_astar.txt`.
Le contenu d'un de ces fichiers serait (tout autre format est possible dans la mesure où vous pouvez ensuite le lire et l'exploiter facilement) :

Contenu du fichier `haute-garonne_distance_100_dijkstra.txt` :
haute-garonne
0 // 0 pour distance et 1 pour temps
100 // nombre de paires origine-destination
dijkstra
o_1 d_1 valeur_sol temps_cpu nb_explorés nb_marqués max_tas
...
o_{100} d_{100} valeur_sol temps_cpu nb_explorés nb_marqués max_tas

Attention Pour éviter les mauvaises surprises dues à la gestion du ramasse-miette, lancez chaque calcul séparément.

Vous pouvez aussi choisir de produire autant de fichiers de résultats qu'il y a de paires (origine-destination).
Par exemple, le lancement de l'algorithme de Dijkstra sur le fichier `nom-carte_distance_nombre-test_data.txt` produira 100 fichiers de résultat :
`haute-garonne_distance_100_dijkstra_1.txt`
...
`haute-garonne_distance_100_dijkstra_100.txt`

4 Lancement des tests

Tout est prêt, vous pouvez lancer vos tests. Vous pouvez écrire un programme Java ou un script qui fait cela pour vous.

5 Lecture des résultats

Une fois que les calculs sont terminés, il reste à récupérer les résultats pour effectuer les analyses demandées dans le rapport.
Pour que cette étape soit la moins fastidieuse possible, il est important d'avoir des fichiers de résultats facile à exploiter.
Vous pouvez écrire un programme Java ou un script pour vous aider. Ce qu'il faut faire dépend des choix effectués dans la partie 3.

6 Rapport de tests de performances

Afin de synthétiser le travail effectué pour ces tests de performances, vous devez :

- présenter votre analyse de l'algorithme de Disjktra ;
- présenter votre analyse de l'algorithme A-star ;
- comparer les deux algorithmes.

Vous devez fournir une analyse permettant d'expliquer le comportement de chaque algorithme : il s'agit de déterminer quels sont les facteurs ayant une influence sur leur déroulement.

Vous pouvez faire des tableaux ou des graphiques. A vous de déterminer ce qui permettra d'expliquer au mieux le fonctionnement de l'algorithme.
Dans quel cas peut-on estimer que cet algorithme fonctionne bien ?
Quelles sont les limites d'utilisation de cet algorihme ?
N'oubliez pas d'avoir un avis critique, est-ce que vous observez vous semble conforme à ce que vous pensiez ?

Comparez les caractéristiques relatives des algorithmes Dijkstra et A-star pour le calcul de plus court chemin en distance et en temps.
Quel est l'intérêt relatif de chacun d'entre eux ?
Existe-t-il des calculs de plus courts chemins pour lesquels il n'est pas possible d'utiliser l'un ou l'autre des algorithmes ?
De part les algorithmes on s'attend à ce que l'algorithme A-star soit plus "performant" que celui de Dijkstra. Est-ce toujours le cas dans votre analyse ? Justifiez et argumentez.