Optimisation

Protocole expérimental 2 : comparaison des temps de trajets

Génération d'un plan de navigation à partir d'une scène #PFE23-R-198

Auteurs : Alexis MARIE, David MARCHÈS, Luca BANKOFSKI, Martial BROSTIN, Theo HELLER, Thomas MABILLE

Résumé

Le but de ce protocole est d'effectuer une comparaison des temps de trajets de nos simulations sur différentes villes. Nous prévoyons que, en moyenne, le temps de trajet de notre système sera inférieur ou égal à celui du trajet témoin. Les hypothèses principales sont que les situations S1, S2, S3 et S4 présenteront respectivement des temps de trajet moyens dans l'ordre décroissant.
Après avoir sélectionné un point de départ et un point d'arrivée dans une ville donnée, nous introduirons des perturbations avant de simuler les quatre situations. Par la suite, nous procéderons à l'évaluation des temps de trajet de chaque simulation pour vérifier nos hypothèses. Ce processus sera répété plusieurs fois sur plusieurs villes afin d'obtenir des résultats plus fiables et représentatifs de la réalité.

Objectif :

Une caméra embarquée utilisant de l'intelligence artificielle permettrait d'augmenter la sécurité des voitures faiblement équipés en systèmes d'aide à la conduite et d'optimiser le trajet par rapport à une application de navigation classique en s'appuyant sur l'observation de l'environnement direct du véhicule.

Introduction

Les expériences sont menées sur des graphes des villes réelles dont nous détaillerons la conception dans un autre protocole expérimental [lien du protocole de Théo]. Pour ce protocole, nous considérons que nous avons déjà générer les graphes.
Nous sélectionnons aléatoirement un point de départ et un point d'arrivée à l'intérieur de chaque ville pour chaque série de simulations. Ce processus est répété plusieurs fois afin d'explorer une diversité d'itinéraires. Après avoir réalisé ces expériences de manière itérative, nous évaluons les temps de trajet respectifs de nos quatre situations.
Pour déterminer le trajet le plus court en termes de durée, nous utilisons l'algorithme de Dijkstra, reconnu pour son efficacité dans la recherche du chemin le plus court entre deux points dans un graphe. Contrairement à certaines méthodes heuristiques, Dijkstra garantit de trouver le chemin le plus court, mais il nécessite un temps de calcul plus long. Ceci est expliqué plus en détail dans l'état de l'art qui est disponible ici [lien de l'état de l'art d'Alexis et Thomas].

Nous réalisons chaque simulation sur le graphe d'une seule ville. Nous effectuons un certain nombre d'expérimentations dans lequel nous testons un nombre défini de chemins.

Voici le fonctionnement de nos différentes situations :

- Situation S1 : Il s'agit de notre situation témoin. Elle prend le chemin le plus court sans tenir compte des perturbations. Elle peut représenter différents niveaux. Par exemple, elle peut représenter une carte routière. Dans ce cas, le nombre de perturbations à appliquer devra être important pour être réaliste afin de représenter les bouchons, les accidents, etc. Elle peut également représenter un service de navigation comme Google Maps. Dans ce cas, le nombre de perturbations à appliquer devra être relativement faible afin de représenter les informations que l'application ne connaît pas encore (route bloquée temporairement, accident qui vient de se produire, etc).

- Situation S2 : Il s'agit de la situation qui représente notre système en fonctionnement autonome ; c'est-à-dire qu'il agit sur l'itinéraire de la voiture mais ne le communique pas aux autres voitures. Notre système veut se présenter sous la forme d'une caméra qui permettrait de détecter ce qui se trouve devant la voiture. En termes de choix d'implémentation, à chaque fois que la voiture arrive à un sommet du graphe, elle regarde s'il y a des perturbations sur les arcs adjacents et, dans ce cas, met à jour son graphe puis relance l'algorithme de Dijkstra pour recalculer le chemin le plus court.

- Situation S3 : Il s'agit de la situation qui représente notre système en fonctionnement coopératif partiel ; c'est-à-dire qu'il communique avec les autres voitures proches. De ce fait, il a connaissance des perturbations dans un rayon paramétrable. Lorsqu'un obstacle est rencontré, le graphe est mis à jour et l'algorithme de Dijkstra est réexécuté pour trouver le chemin le plus court. Comme S2, S3 a connaissance des arcs adjacents puisqu'il utilise le même système.

- Situation S4 : Il s'agit de la situation qui représente notre système en fonctionnement coopératif total ; c'est-à-dire qu'il communique avec l'ensemble des véhicules utilisant notre système avec l'équivalent d'un rayon infini. De ce fait, il a connaissance des perturbations de l'ensemble du graphe en temps réel.

A partir de notre objectif, nous définissons deux hypothèses principales :

- Les temps de trajet de chaque simulation utilisant notre système (S2, S3 et S4) sont inférieurs ou égaux à celui du trajet témoin (S1) ;
- Les situations S1, S2 S3, et S4 présenteront respectivement des temps de trajet moyens dans l'ordre décroissant.

Cette expérience vise à optimiser les déplacements urbains, réduire les temps de trajet, réaliser des économies de coûts et diminuer la pollution, ainsi qu'à améliorer la qualité de vie des citoyens.

Matériel et méthodes

Matériel et équipement :

- Un ordinateur portable ;
- Le Jupyter Notebook contenant le code pour effectuer les simulations et récupérer les données: [2] ;
- Les fichiers CSV contenant les données des graphes des différentes villes (obtenus à l'aide du protocol expérimental "analyse comparative des réseaux routiers") ou récupérés directement [3].

Participants ou spécimens :

- Graphes de villes réelles variées dans leur sélection en termes de taille.

Variables :

- Indépendantes : Localisation des perturbations
- Dépendantes : **Aucune**
- Contrôlées :
    - **Nombre d'expérimentations** : le nombre de fois que l'on changera les perturbations sur la ville (*number_of_experiments*) ;
    - **Nombre de chemins** : le nombre de tests que nous effectuons pour un même graphe perturbé avec les mêmes paramètres (*number_of_paths*) ;
    - **Rayon du champ de vision de la simulation S3** : le rayon en mètres dans lequel S3 a connaissance des perturbations (*rayon*) ;
    - **Longueur minimum du chemin à générer** : la taille que doit faire au minimum le chemin entre le point de départ et le point d'arrivée avant perturbation, dans le cas contraire, le point de départ et le point d'arrivée sont changés aléatoirement jusqu'à ce que cette condition soit vérifiées (*minimum_length*) ;
    - **Poids des perturbations** : le temps en secondes que l'on ajoute sur un arc du graphe qui est perturbé (*weight_of_perturbation*) ;
    - **Perturbations aléatoires** : un booléen permettant d'ajouter ou non un temps aléatoire sur un arc perturbé (*random_perturbation*) ;
    - **Pourcentage de perturbations** : le pourcentage d'arcs à perturber par rapport au nombre total d'arcs (*pourcentage*) ;
    - **Graphe de la ville** : les deux fichiers CSV contenant respectivement les sommets et les arcs de la ville que l'on souhaite tester (*nodes_file_path*, *edges_file_path*).

Procédure expérimentale :

- Partie 1 : Préparation de l'environnement

Récupérer le Jupyter Notebook contenant les deux codes Python en suivant ce lien [à ajouter] ;
Installer Python et les modules nécessaires ;
Récupérer les fichiers CSV des villes, soit directement sur le dépôt Git en suivant ce lien [2], soit en utilisant la procédure du protocole expérimental “Analyse comparative des réseaux routiers”.

✓

✓➢ Partie 2 : Sélection des paramètres

Définissez les paramètres de votre simulation en modifiant les variables contrôlées ;

- Les variables associées au nombre de simulations (**Nombre d’expérimentations** et **Nombre de chemins**) sont fixées en fonction de la taille de la ville étudiée. Il est conseillé de ne pas aller trop bas pour que les chiffres soient représentatifs pour une analyse ;
- Le **Pourcentage de perturbations** doit être réaliste. Si l’objectif est de comparer les différentes situations en considérant que S1 est une carte routière, le pourcentage peut varier sans problème entre 0 et 50%. En revanche, en considérant que S1 représente Google Maps, il est inutile de dépasser les 5% de perturbations ;

✓

Choisissez la ville sur laquelle réaliser la simulation.

➢ Partie 3 : Lancement de la simulation

✓ Lancer la cellule contenant le code python ;

- Le temps de simulation peut rapidement dépasser plusieurs heures voire dizaines d’heures, il ne faut donc pas oublier de changer ses paramètres pour éviter que le PC ne se mette en veille et mette en pause la simulation ;
- L’utilisation d’un Jupyter Notebook permet de ne pas couper la simulation si l’ordinateur se met en veille, elle reprendra dès que l’ordinateur ne sera plus en veille.

✓

➢ Partie 4 : Récupération des données

✓

Changer le nom du fichier CSV afin de ne pas écraser les résultats d’une précédente simulation ;
Lancer la cellule contenant le code python.

Collecte de données

Pour chaque test, une situation détermine un itinéraire et retourne le temps en secondes qu'il lui faudrait pour le parcourir en sommant l'ensemble des poids des arcs constituants son chemin. Ces temps sont ajoutés dans une liste durant la simulation. Enfin, lors de la partie 4 du protocole définit auparavant, le deuxième code python va convertir cette liste en un fichier CSV afin de permettre son analyse. Ce fichier CSV contiendra les informations suivantes :

- Le temps de trajet de la situation S1
- Le temps de trajet de la situation S2
- Le temps de trajet de la situation S3
- Le temps de trajet de la situation S4
- La longueur du trajet S1 en nombre d'arcs
- La longueur du trajet S2 en nombre d'arcs
- La longueur du trajet S3 en nombre d'arcs
- La longueur du trajet S4 en nombre d'arcs
- Le numéro d'avancement de la variable **Nombre d'expérimentations**
- Le **Pourcentage de perturbations** appliqué sur notre ville

Analyse des données

À la fin de notre simulation sur une ville donnée, nous calculons la moyenne des temps de trajet de tous les itinéraires pour chaque situation. Puis, nous effectuerons une comparaison de ces résultats et nous les confrontons à nos hypothèses initiales. Afin, de déterminer les paramètres influençant nos résultats, nous pourrons faire varier par exemple le **Rayon du champ de vision de la simulation S3**, mais également, le **Poids des perturbations** ou bien encore le **Pourcentage de perturbations**.

Considérations éthiques et sécurité

Mesures de sécurité :

Sauvegarde des données : Effectuez régulièrement des sauvegardes des données pour éviter toute perte accidentelle ou toute corruption des informations, ce qui garantira la disponibilité continue des données pour l'analyse et la recherche.

Plan d'expérience

Différentes expériences peuvent être menées afin d'étudier l'efficacité du système en fonction des multiples paramètres.
Dans un premier temps, nous pouvons réaliser une expérience sur les performances de nos systèmes en fonction de la taille de la ville. Pour se faire, il faut suivre le protocole expérimental en fixant l'ensemble des paramètres et en réalisant des simulations sur des villes de petites, moyennes et grandes tailles.
Dans un second temps, nous pouvons évaluer les performances de nos systèmes en fonction du type de perturbation (fixe ou aléatoire). De nouveau, il faut suivre le protocole expérimental en utilisant une même ville et en fixant l'ensemble des paramètres. Pour la première simulation, la variable **Perturbations aléatoires** est fixée à False. Pour la seconde, elle est passée à True.
Nous pouvons aussi vérifier qu'elle est l'influence de la proportion de route à sens unique sur nos résultats en fixant les variables de **Rayon du champ de vision de la simulation S3** et **Perturbations aléatoires**. Nous comparons ensuite les résultats en fonctions de la proportion de route aléatoire de chaque ville.
Nous pouvons également étudier la performance de S3 en fonction de son rayon. Pour cela, nous pouvons faire varier **Rayon du champ de vision de la simulation S3** entre 0 et 1000 mètres. Nous pourrons ainsi mettre en évidence la performance de S3, observer à partir de quel rayon nous avons des performances similaires à S4 .
Enfin, nous pouvons étudier la performance de nos systèmes en fonction des tailles d'itinéraires minimum. Il faut utiliser la variable **Longueur minimum du chemin à générer** et réaliser des simulations sur une même ville en faisant varier la longueur de 20 à 200 par exemple.

Les détails des expériences menées avec ce protocole expérimental sont disponibles dans le document *PFE23-R-198_TEM_Optimisation_Simulation-graphes.*

Calendrier prévisionnel

- Étape 1 : 1 jour
- Étape 2 : 1 jour
- Étape 3 : 4 jours
- Étape 4 : 2 jours

Ce calendrier est très sensible au nombre de simulations souhaitées, aux paramètres choisis, au plan d'expérience et au nombre d'ordinateurs capables de fonctionner simultanément.

Bibliographie

[1] Dijkstra : https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/ by Estefania Cassingena Navone

[2] https://github.com/alexismarieece/PFE_RECHERCHE_VOITURE/blob/theo/Parsing_graph/simulation_real_city.ipynb

[3] https://github.com/alexismarieece/PFE_RECHERCHE_VOITURE/tree/theo/Sources/map