Optimisation

Document d'architecture technique : comparaison des temps de trajet

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

Introduction

Ce document a pour objectif de détailler l'architecture technique de la plateforme de test que nous avons mis en place afin de valider ou au contraire invalider la partie optimisation de notre hypothèse de recherche qui s'énonce comme suit : 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. Nous allons donc ici décrire le fonctionnement de chacune des situations comparées

Ce document a pour objectif de décrire le fonctionnement de chacune des simulations permettant la génération d'itinéraires entre un point de départ et un point d'arrivée à partir d'un graphe de ville perturbé. Pour cela, nous avons décidé de mettre en place une plateforme de test qui nous permettra de valider ou non nos hypothèses de recherche. Pour rappel, voici le résumé des 4 situations que nous avons déjà évoqué dans ce protocole expérimental [voir document *PFE23-R-198_PE_Simulation-graphes*] :

- 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 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.

Le système est une plateforme de simulation de déplacement urbain qui utilise des graphes de villes perturbés. Son objectif est de déterminer les itinéraires les plus courts en fonctions des différentes

situations énoncées précédemment tout en tenant compte des perturbations urbaines pour optimiser les temps de trajet.

Objectifs fonctionnels :

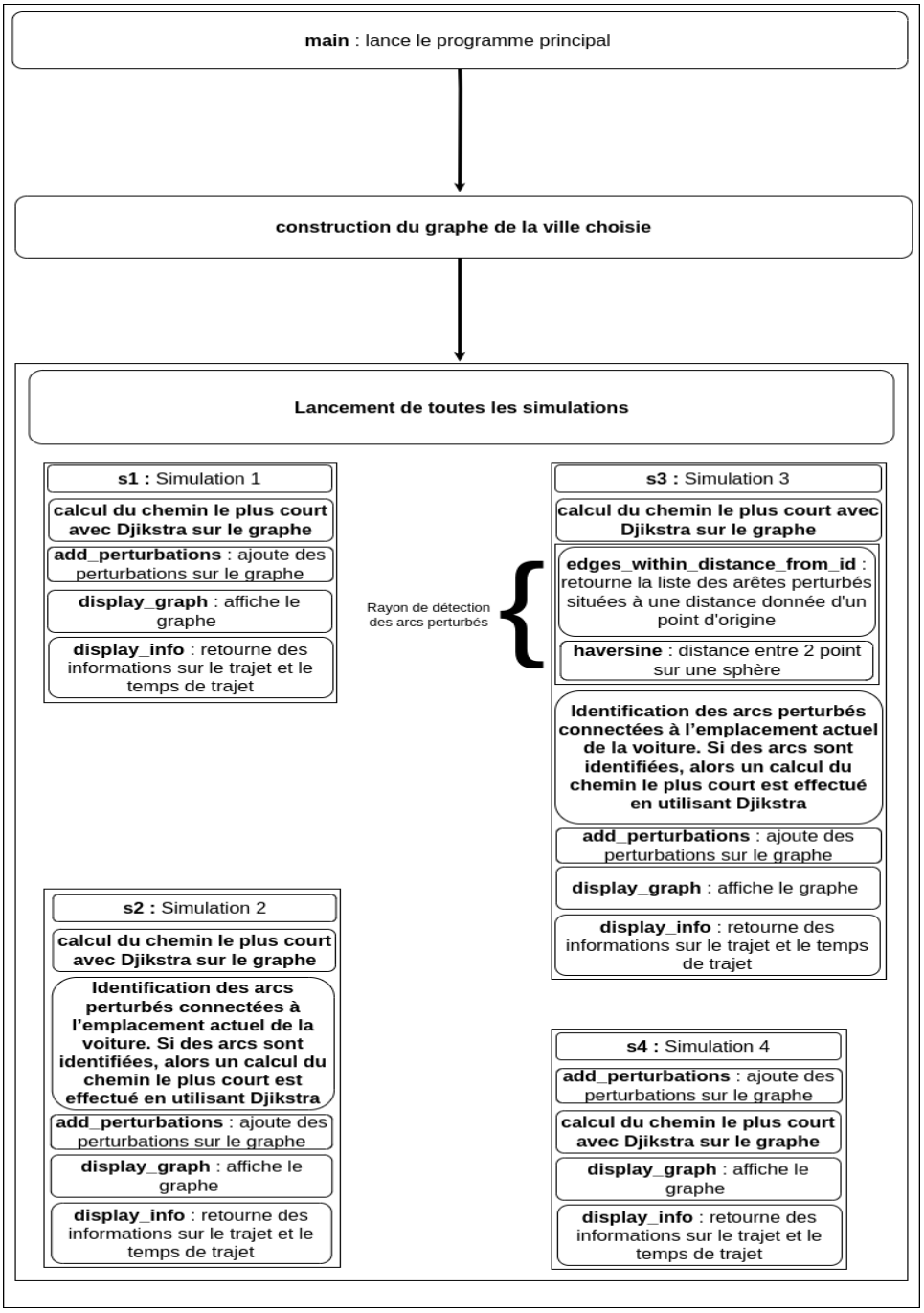
- Optimiser les déplacements urbains,
- Réduire les temps de trajet.

Objectifs non fonctionnels :

- Garantir la fiabilité et la précision des résultats,
- Rendre l'utilisation intuitive de la plateforme,
- Faciliter la mise en place des tests en ne réglant uniquement que les paramètres de nos simulations.
-

Architecture Fonctionnelle

Ce programme prend en entrée deux fichier CSV dont la création est explicitée dans le protocole expérimental ”Analyse comparative des réseaux routiers”. Vous pouvez aussi retrouver tous ces CSV dans le git du projet
[https://github.com/alexismarieece/PFE_RECHERCHE_VOITURE/tree/theo/Sources/map]



Architecture Matérielle et Logicielle

Nous avons utilisé un total de six ordinateurs, avec une combinaison de processeurs Intel Core i7, Intel Core i5 et un Apple MacBook M2, ainsi qu'une mémoire RAM variant de 8 à 16 Go en fonction des divers ordinateurs. Les spécifications individuelles des ordinateurs sont les suivantes :

- Processeurs :
    - 3 ordinateurs avec des processeurs Intel Core i7.
    - 2 ordinateurs avec des processeurs Intel Core i5.
    - 1 Apple MacBook M2.
- Mémoire : 8 à 16 Go de RAM.
- Système d'exploitation :
    - Windows 10 installé sur quatre des ordinateurs,
    - MacOs sur l'Apple MacBook M2,
    - Linux sur un ordinateur comportant un i7.

Cette combinaison de processeurs Intel Core i7, Intel Core i5 et un Apple MacBook M2, associée à une mémoire RAM de 8 à 16 Go, a été utilisée en raison de la disponibilité des ordinateurs et de leurs capacités respectives à exécuter les simulations. Par ailleurs, Nous avons eu la chance de disposer d'une variété de systèmes d'exploitation, notamment Linux, MacOs et Windows, ce qui a contribué à la flexibilité et à la diversité de notre environnement de simulation.

Pour uniformiser notre environnement d'exécution et faciliter le développement des simulations, nous avons installé les logiciels suivants sur tous les ordinateurs :

- Visual Studio Code : Un environnement de développement intégré (IDE).
- Python : Le langage de programmation principal utilisé pour la mise en œuvre des simulations.
- Bibliothèques Python : Nous avons utilisé les bibliothèques Python suivantes pour les graphes, les calculs et l'analyse des données :

➢ **Networkx :** création, manipulation et la visualisation de graphes,
➢ **Matplotlib :** création de graphiques et de visualisations de données,
➢ **Numpy :** offre des structures de données performantes pour manipuler des tableaux multidimensionnels ainsi que de nombreuses fonctions mathématiques pour effectuer des opérations sur ces tableaux,
➢ **Maths :** fonctions mathématiques standard en Python**,**
➢ **Pandas :** manipulation et l'analyse de données, offrant des structures de données et des outils pour effectuer des opérations sur des tableaux de données.

Chaque ordinateur a été configuré pour exécuter des instances des simulations assignées, travaillant en parallèle pour accélérer le processus de modélisation et d'analyse.

En somme, cette infrastructure matérielle, comprenant une combinaison de processeurs Intel Core i7, Intel Core i5 et un Apple MacBook M2, associée à une mémoire RAM de 8 à 16 Go, a été exploitée de manière efficace pour mener à bien les simulations informatiques, garantissant ainsi des résultats précis et fiables dans un délai raisonnable.

Évolutivité et Extensibilité

On pourrait envisager d'intégrer d'autres simulations directement lors du lancement de toutes les simulations, ce qui enrichirait notre étude. De plus afin de réduire le temps d'exécution de nos algorithmes, nous pourrions envisager deux approches :

- Une approche logicielle : Elle consiste à un changement de langage de programmation. En effet, le langage C par exemple nous permettrait d'avoir des performances supérieures de par sa gestion mémoire de bas niveau et de sa proximité avec le processeur. De plus, nous pourrions également ajouter diverses boucles dans notre programme pour tester des villes différentes à la suite et faire varier également de façon automatique divers paramètres.
- Une approche matérielle : Elle consiste à s'équiper de machines plus puissantes pour pouvoir réduire davantage le temps d'exécution de notre programme.

Normes et Conventions

Nous suivons les bonnes pratiques de développement logiciel afin d'assurer la qualité, la fiabilité et la maintenabilité de notre système. Dans cette optique, nous utilisons Git, un système de contrôle de version largement reconnu, qui nous permet de gérer efficacement les modifications du code source, de collaborer de manière transparente et de garantir l'intégrité du code tout au long du cycle de développement. De plus, nous adoptons une approche systématique en ce qui concerne les commentaires du code, assurant ainsi une compréhension claire pour faciliter la maintenance et l'évolution du logiciel. En intégrant Git dans notre processus de développement, et en veillant à la qualité de nos commentaires, nous nous assurons que notre équipe travaille de manière cohérente et collaborative, ce qui contribue à la robustesse et à la pérennité de notre solution logicielle. De plus, nous privilégions l'utilisation exclusive d'outils open source, rendant ainsi notre solution pratique à reproduire, tout en tirant parti de modules existants pour faciliter la compréhension générale du code. Cette pratique nous permet également d’assurer une maintenabilité du code.

Conclusion

Notre étude met en avant l'importance de l'optimisation des déplacements urbains via l'évaluation de diverses simulations prenant en compte diverses perturbations. Pour cela, il était nécessaire d'avoir une architecture matérielle solide que nous avons mise en place avec six ordinateurs portables combinant des processeurs Intel Core i7, Intel Core i5 et un Apple MacBook M2, associée à une mémoire RAM de 8 à 16 Go. Par ailleurs, nous avons uniformiser notre environnement d'exécution en utilisant Visual Studio Code avec le langage Python.

Nous avons également envisagé des améliorations pour optimiser les performances de nos algorithmes. Une approche logicielle a été envisagée, notamment en considérant l'utilisation du langage C pour des performances supérieures grâce à sa gestion mémoire de bas niveau et sa proximité avec le processeur. De plus, l'ajout de boucles dans notre programme pour tester différentes villes et faire varier automatiquement les paramètres a été envisagé pour enrichir notre étude.

En parallèle, une approche matérielle a également été envisagée, avec la possibilité d'investir dans des machines plus puissantes pour réduire davantage le temps d'exécution de nos simulations.

Par ailleurs, le contrôle de version qui a été effectué grâce à Git combiné avec l'usage d'outils open source, nous a permis d'obtenir une solution aisément reproductible et compréhensive, tout en favorisant la collaboration et la pérennité de notre projet.

Nous envisageons d'étendre notre étude en incluant plus de simulations et en explorant d'autres facteurs influençant les déplacements urbains. Ces efforts contribueront à améliorer la mobilité urbaine et la qualité de vie des citoyens.