UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Croisements de trains de véhicules

Travail Supplémentaire - P2015

BECK Florian

Département Génie Informatique
Filière Ingénierie des logiciels et
De la connaissance

Systèmes multi-agents et résolution Distribuée de problèmes

Tuteur du Projet
GECHTER Franck

Responsable de Filière
Amir Hajjam El Hassani

utbm
université de technologie
Belfort-Montbéliard

Table des matières

Introduction

Durant l'automne 2014, j'ai suivi l'unité de valeur IA54, orienté sur l'intelligence artificielle distribuée et les systèmes multi-agents. Dans le cadre de cette UV, nous avions à développer un projet par groupe de deux étudiants. J'avais opté pour un projet visant à simuler le fonctionnement de croisements de véhicules. La problématique était de permettre à des files de véhicules de se croiser au sein d'un carrefour commun, sans heurts et de la façon la plus fluide possible.

A l'issue du projet, le résultat était en demi-teinte et pouvait être amélioré. C'est avec cette conclusion en tête que j'ai opté pour la poursuite du projet en tant que travail supplémentaire, quand cela s'est avéré nécessaire. Le rapport suivant sera composé d'une présentation du sujet et de ces problématiques, puis je présenterai une étude de l'état du projet tel qu'il était à la fin de l'UV IA54 avec ces faiblesses. Enfin je vous présenterai les évolutions et corrections apportées durant ce semestre.

I. Présentation du sujet

L'objectif de ce projet est de développer un modèle à base d'agents réactifs, tel qu'étudié durant l'UV. Ce modèle doit offrir la possibilité de croiser des formations de trains de véhicules sans collisions lors d'une intersection. L'objectif final théorique est de pouvoir éventuellement porter le fonctionnement des agents de la simulation sur des véhicules réels.
Deux trains de véhicules sont formés, se déplacent le long d'un parcours et doivent se croiser à une intersection de la façon la plus fluide possible en adaptant leur vitesse. L'avantage de ce mode de fonctionnement est que l'on peut se passer de règles arbitraires de priorités telles que des feux tricolores, tout en améliorant la fluidité du trafic.

A. Composition du système

Un agent est une entité (logicielle) qui agit de façon autonome, capable de communiquer, disposant d'une connaissance partielle de son entourage et surtout de comportements privés mis en œuvre par sa capacité d'exécution propre.

Un système multi-agent distribué met donc en œuvre un contexte ou plusieurs agents sont implantés et mis en relation, ils suivent leur propre fonctionnement et interagissent entre eux et avec leur environnement.
Dans le système, nous avons prévu 3 fonctions, supportés par un type d'agent à chaque fois. Ce sont des véhicules, des trains de véhicules et l'environnement.

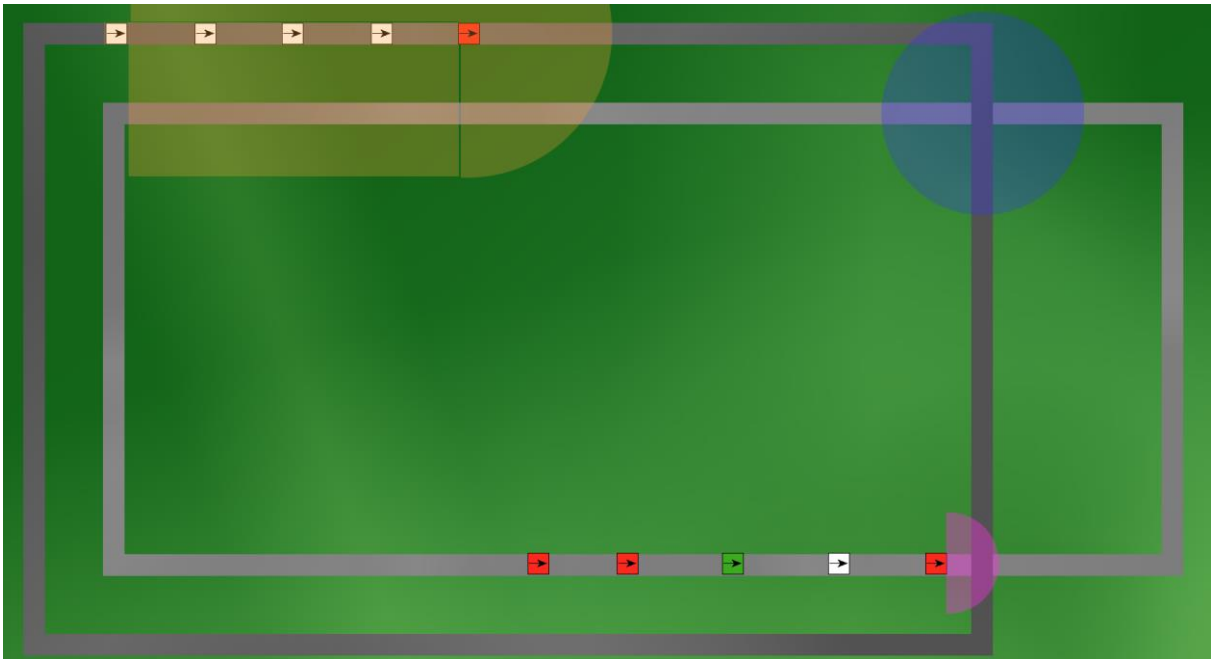

Figure 1 Zones d'émission ou de perception des agents

1. L'agent Véhicule

L'agent véhicule, ou voiture, représente une voiture dans la simulation. Elle fait partie d'un convoi, ou train. Elle évolue dans l'environnement.

Après sa création elle va évoluer le long d'un parcours, comme un véhicule dans un contexte réel. Elle est capable de détecter les autres voitures dans un certain périmètre, et de s'adapter à leur présence. En effet, elle vérifiera constamment si des voisins sont présents devant elle ou non. Si des voisins sont devant et proche d'elle, la voiture va réagir, au besoin ralentir ou même s'arrêter, selon l'évolution de la situation. Sinon elle va accélérer à moins qu'elle ait déjà atteint la vitesse maximale, telle que définie par son train.

2. L'agent train de véhicule

Un train, ou convois de véhicule représente un ensemble de véhicules se suivant et partageant le même parcours. Il communique avec les autres trains à portée, donne des directives à ces voitures, et perçoit quelques informations de son environnement.

Pour simplifier la simulation, on assume que toutes les voitures d'un train suivent le même trajet pendant toute la durée de la simulation. Elles gardent donc la même position, et la même hiérarchie.
Le train est l'entité directrice des véhicules qui le composent. En effet, il définit pour l'ensemble une vitesse ou une distance de croisement adaptée à la

situation. Il détecte de l'environnement les croisements à proximité, et dialogue avec les trains à proximité pour analyser la situation à venir.

3. L'agent environnement

L'agent environnement est un outil nécessaire à la simulation. En effet, il a pour tâche de situer virtuellement tous les véhicules dans l'espace, ainsi que les croisements. Il informe régulièrement les véhicules de la présence d'autres entités dans leur périmètre de détection.
Il informe également les trains de l'existence de croisements à proximité sur leur trajet.

Dans le cas d'un fonctionnement sur véhicules réels, l'environnement serait remplacé par les capteurs inhérents à chaque véhicule, et aux balises des croisements. L'intelligence de cet agent est limitée à la sa réaction aux modifications de position, et à l'émission si nécessaire de messages.

B. Environnement de développement

Le langage utilisé pour programmer ce projet est le Java, couramment utilisé dans ce domaine.

Afin de développer le projet simplement, nous avons utilisé la plate-forme **MadKit** qui nous permet de passer la conception du système multi agents. En tant que débutant dans le domaine des systèmes multi-agents, nous l'avons trouvée légère, bien documentée et facile d'utilisation.
Les points que nous avons trouvés attractifs étaient la gestion automatique d'un ordonnancement des agents (assurance qu'ils aient tous une et une seule exécution par cycle) ainsi que la classification des agents selon un modèle de type communauté/groupe/rôle, rendant ainsi plus simple et plus propre la communication. Elle dispose d'un système de messagerie entre agents, sensible à la classification relative entre ces agents.

Pour le fonctionnement du projet, le projet comporte un programme principal chargé de l'initialisation et du lancement des agents, une bibliothèque de constantes et une de fonctions outils communes.
L'interface graphique est assurée par la librairie **Swing**, simple et suffisante.

Lors du projet supplémentaire, j'ai mis en place le suivi des véhicules par un outil statistique, étudiant l'évolution des vitesses et de la distance entre deux

voitures se suivant pour chaque train. Il est accessible via un bouton de l'interface, et utilise la librairie **JFreeChart** pour dessiner les graphiques.

C. Principales étapes intermédiaires

Afin de mettre le projet sur pied, plusieurs jalons ont été définis lors de la première version du projet. Elles permettent de présenter les principales difficultés rencontrées alors, je les retranscris donc ci-dessous afin de préciser le cadre et les limites de la simulation.

- Le premier cycle de développement consistait à afficher et déplacer une voiture sur un circuit prédéfini dans une fenêtre graphique.

  Afin de permettre le déplacement de ces véhicules, nous avons défini un fichier .xml décrivant, par train de véhicules, un trajet sous forme de coordonnées.
  Le programme créera autant de trains qu'il y a de balises trains dans ce fichier, et cherchera les croisements dans les chemins, que ce soit entre différents trains ou au sein du parcours du train lui-même. Les voitures font donc appel à un ensemble de fonctions leur permettant d'obtenir leur nouvelle position sur leur trajet en fonction de la position actuelle et de leur déplacement.

```xml
<?xml version="1.0" encoding="UTF-8"?>
    <path background="Circuit_7.png">
        <train>
            <point>
                <x>32</x>
                <y>936</y>
                <angle>360</angle>
            </point>
            <point>
                <x>32</x>
                <y>32</y>
                <angle>90</angle>
            </point>
            <point>
                <x>1436</x>
                <y>32</y>
                <angle>180</angle>
            </point>
            <point>
                <x>1436</x>
                <y>936</y>
                <angle>270</angle>
            </point>
            <point>
                <x>32</x>
                <y>936</y>
                <angle>360</angle>
            </point>
        </train>
        <train>
            <point>
                <x>150</x>
                <y>816</y>
                <angle>90</angle>
            </point>
```

Figure 2 : Structure du fichier de parcours

Une difficulté a été rencontrée lors de la nécessité de faire pivoter les véhicules lors de leurs déplacements. Nous avions donc mis au point une couche d'affichage supplémentaire pour la librairie Swing, permettant la rotation simplifiée des images représentant les véhicules.

- Le second cycle consistait à créer les convois de véhicules, ou les véhicules se suivraient sans heurts, et obtiendraient des objectifs (vitesse, écart entre véhicules) du train.

  Nous avons alors mis en place l'environnement, ne pouvant pas décemment se contenter de faire surveiller chaque voiture par sa suivante. L'environnement permet de gérer les positions en un seul endroit, et ainsi de pouvoir envoyer de façon groupée les perceptions des agents, de façon plus réaliste que l'interrogation de chaque autre agent présent, par exemple. En effet, cette méthode permet de simuler le retour d'informations généré par la batterie de capteurs d'un véhicule.

Nous avons également mis en place la prévision des collisions, qui implique deux réactions en fonction de la distance des objets étrangers : la distance de sécurité, qui implique que la voiture pile pour des raisons de sécurité, ainsi que la distance de vision, qui permet à la voiture de s'adapter aux situations avant de devoir piler.

- Le troisième et dernier cycle du projet voyait la mise en place de plusieurs trains de véhicules, de leur croisement et donc de la réaction des agents adaptée. Je vais détailler les résultats de ce dernier jalon dans l'analyse suivante.

II. Etude de l'existant

A l'issue du projet, nous avons présenté nos résultats à notre suiveur M.Gechter. Je les reprends ici, en pointant les difficultés rencontrées, les solutions apportées et les problèmes persistants, notamment ce qui avait été retenu de la soutenance.
Lors de la soutenance, le fonctionnement du programme n'était pas parfait. En effet, plusieurs problèmes se posaient. Les voitures avaient tendance à se heurter lors du croisement, de plus la fluidité n'était pas optimale.

La première difficulté lors de ce projet est la distribution totale de l'intelligence. Le programme a souffert d'une erreur de conception mise en évidence lors de la soutenance, en effet les trains n'étaient pas maîtres de leur réaction, et étaient dirigés par l'environnement. Celui-ci détectait les croisements entre trains, et les informait de la situation, ne leur laissant qu'un rôle de relai. Cette situation, en plus de ne pas correspondre au cahier des charges rendait le fonctionnement des croisements bancals, la gestion de la situation à l'échelle des trains n'étant plus dynamique.

La seconde difficulté se trouve dans la réaction à attribuer aux agents, notamment leurs paramètres et modèles comportementaux. En effet, si l'on peut se baser sur des comportements réels pour définir un freinage et une accélération maximale en définissant une échelle, il est complètement arbitraire de choisir les vitesses par défaut de fonctionnement, ou les distances de détection d'une balise. Il n'est donc pas aisé de faire coller la simulation à la réalité.
De plus le comportement à attribuer au véhicule, le cœur du projet, laisse place à de nombreuses possibilités/ méthodes, et la solution la plus évidente n'est pas toujours la plus efficace au sein de l'ensemble.

Enfin il faut combiner fluidité et sécurité, pour assurer que les véhicules ne se heurtent pas, tout en permettant un passage rapide et fluide du train. De plus, ce point soulève une question fondamentale : est-il judicieux d'envisager le déplacement de véhicules par trains ?
En effet s'il est difficile de combiner fluidité, intégrité du convoi et sécurité des véhicules, en plus de la nécessité arrangée artificiellement du trajet commun, quels sont les avantages du déplacement en files de véhicules ?

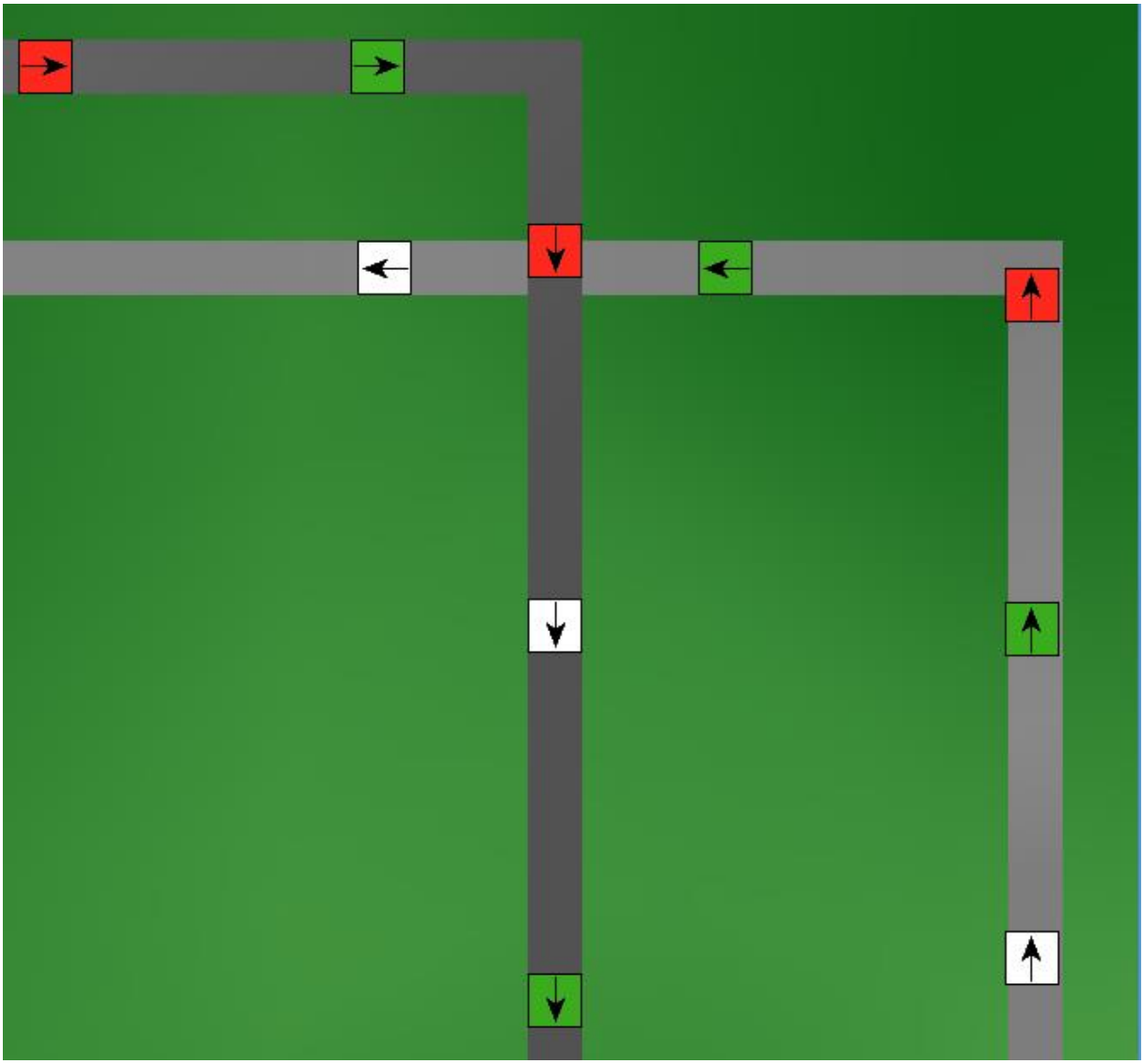

Figure 3 aperçu de croisement de deux trains de véhicules, version actuelle

III. Evolutions apportées

J'ai donc repris le projet dans le cadre de mon travail supplémentaire, afin d'apporter des corrections et évolutions.

A. Correction de la centralisation

Comme décrit plus haut lors de la soutenance nous avons mis à jour une faille à la décentralisation et à l'autonomie des trains, mon premier travail a donc été de corriger la situation.

Tout d'abord j'ai enlevé à l'environnement sa capacité à décider de l'action des trains. En lieu et place des prises de décision, l'agent environnemental se contente de simuler l'émission d'une balise, en informant le train concerné de l'arrivée prochaine d'un croisement. Cette action se produit quand l'environnement détecte que la première voiture du convoi entre dans son rayon d'émission.
Par soucis de simplification et pour maintenir les rôles clairs et distincts dans le schéma de fonctionnement, je n'ai pas envoyé l'évènement de l'environnement à travers les capteurs de la première voiture comme cela se passerait dans la réalité. Dans ce cas, la première voiture agirait comme intermédiaire, en remontant ce signal à son train, toujours par le même type de message.

Ensuite, j'ai mis en place une intelligence basée sur la coopération pour les agents de type train. En effet, ils reçoivent maintenant uniquement des informations de type « balise », et il leur reste encore à déterminer leur comportement. Ils vont tout d'abord stocker cette information, puis s'enquérir de la situation au niveau du carrefour. Pour cela, ils vont émettre un message à l'attention de tous les trains à proximité (un message de type broadcast).
Ce message, du au fonctionnement du système de messagerie, sera reçu par tous les trains. Pour rejoindre un fonctionnement plus réel, les trains qui se considéreront « trop loin » du train émetteur (sur la base d'une distance d'émission propre au train), ignoreront le message. Pour ceux qui sont assez près, ils vérifient que la balise du croisement se trouve également sur leur route. Si c'est le cas, ils savent désormais qu'un croisement risque fort d'avoir lieu, sinon ils ignoreront le message. Ce(s) train(s) susceptibles de se croiser sont forcément plus proches du carrefour, ayant déjà la connaissance de son existence avant le message. Ils indiquent donc leur proximité du carrefour au

train émetteur, ainsi que leur situation de priorité. De plus, ils adaptent leur vitesse et leur distance inter-véhicule, afin de faciliter le croisement.
Dans le cas où le train émetteur ne recevrait aucune réponse, il continue son chemin normalement, mais dans le cas où celui-ci à une réponse il va s'adapter à la présence notifiée en adaptant son allure et sa distance inter-véhicules.

L'entrée dans le croisement et son parcours est ainsi réglée pour les trains, néanmoins une fois la dernière voiture sortie du croisement, il convient de retrouver un couple allure/distance de suivi plus efficace. Dans ce but, le même fonctionnement que l'entrée est utilisé. En effet, les véhicules ne se soucient que des informations au-devant et sur les côtés. Une fois le croisement passé par la dernière voiture du train, le croisement devient « invisible », aucune voiture ne remontant à son train l'existence de la balise du croisement. Pour simuler cette situation, une fois la dernière voiture sortie du croisement, l'environnement envoie au train un message de sortie de croisement.
Une fois ce message reçu, le train réadapte les vitesses et inter-distances, et notifie le cas échéant l'autre train du changement d'état. Celui-ci peut alors changer son comportement de la même manière.

B. Modification du comportement de la voiture

J'ai ensuite estimé, le travail précédent n'apportant pas de résultats suffisants, qu'il était nécessaire de revoir l'algorithme de l'agent élémentaire, le véhicule. Cet algorithme de décision est le point de réflexion et d'amélioration principal, étant donné que la majorité des décisions y sont prises.
Son algorithme précédent est décrit dans l'annexe 1.

Le cycle de décision de la voiture a été modifié.

- Passons la première étape, qui reste inchangée, consistant à traiter les messages reçus, à estimer un déplacement optimal, et de rechercher les voisins.

- Dans le cas ou des voisins sont effectivement présents, j'ai commencé par traiter les voisins présents dans une distance de sécurité. Si des voisins sont recensés dans cette zone, la voiture tente de s'arrêter le plus vite possible, dans analyse plus poussée.

- Dans le cas où la zone de sécurité est libre, on étudie les voisins présents. En effet, on différence le comportement face à une voiture de son propre train de celui à adopter face à une voiture que l'on apprête à

croiser. On tente de repérer (dans le cas où ils existent dans le voisinage visible) le plus proche voisin de son propre train, et le plus proche voisin « à croiser ».

- Une fois cette sélection terminée on détermine, pour chaque type de voisin, si nécessitée d'adaptation il y a :

    - Dans son train, on va tâcher de suivre le véhicule à la distance de suivi (qui varie lors de l'entrée dans un croisement). Pour cela on s'adapte premièrement à sa vitesse de sorte à adopter la même vitesse que le véhicule précédent une fois à bonne distance. On contrôle ensuite le respect de la distance de suivi, au besoin notre véhicule ralenti.
      Si les deux contrôles sont corrects, on ne nécessite pas d'adaptation.
    - Concernant les voitures à croiser, le premier réflexe est de vérifier la priorité. En effet, comme pour le comportement des trains expliqué plus haut, c'est au véhicule n'ayant pas la primeur de s'adapter.
      Si une priorité est définie on l'applique, sinon on la calcule. La priorité est donnée, entre deux voitures, à celle qui théoriquement sortirait la première du carrefour. La première voiture à remarquer l'autre et à définir la priorité informe l'autre véhicule. Une fois la priorité établie, si le véhicule est prioritaire il ne s'adapte pas, sinon il modifie sa vitesse (selon les possibilités) pour entrer dans le croisement, une fois le voisin sorti de celui-ci.
    - Enfin, on fusionne les deux adaptations nécessaires en choisissant la situation la plus prudente.
- On applique la vitesse souhaitée.

J'ai vite remarqué un problème avec cette gestion des urgences (véhicules dans la zone de sécurité), en effet il arrive régulièrement que les deux véhicules arrivent en même temps au carrefour et que cela crée un blocage. En effet, pour traverser le carrefour, un des véhicules doit avancer tandis qu'un véhicule est dans sa zone de sécurité. J'ai donc modifié la gestion des urgences pour outrepasser cette règle.
En conséquence, si une « urgence » ne fait pas partie du train, qu'elle est hors du croisement et que soit notre véhicule est engagé, soit il est prioritaire, et dans ce cas seulement on peut ignorer ce voisin.

Ce sont là la seconde série de modifications importantes que j'ai opéré. J'ai ensuite réalisé des optimisations diverses dans les agents. Ces améliorations ont été mises en place pour améliorer la réaction à un type de situation qui pose problème.

C. Optimisations diverses

Comme demandé lors du premier projet mais qui n'a alors pas pu être réalisé, M.Gechter m'a demandé de mettre en place un outil statistique, permettant de mesurer l'efficacité de mes modèles. Il m'a conseillé la bibliothèque gratuite **JFreeCharts**. J'ai donc mis en place la collecte et l'affichage de données durant le fonctionnement du programme, dans une fenêtre séparée pour augmenter la lisibilité.

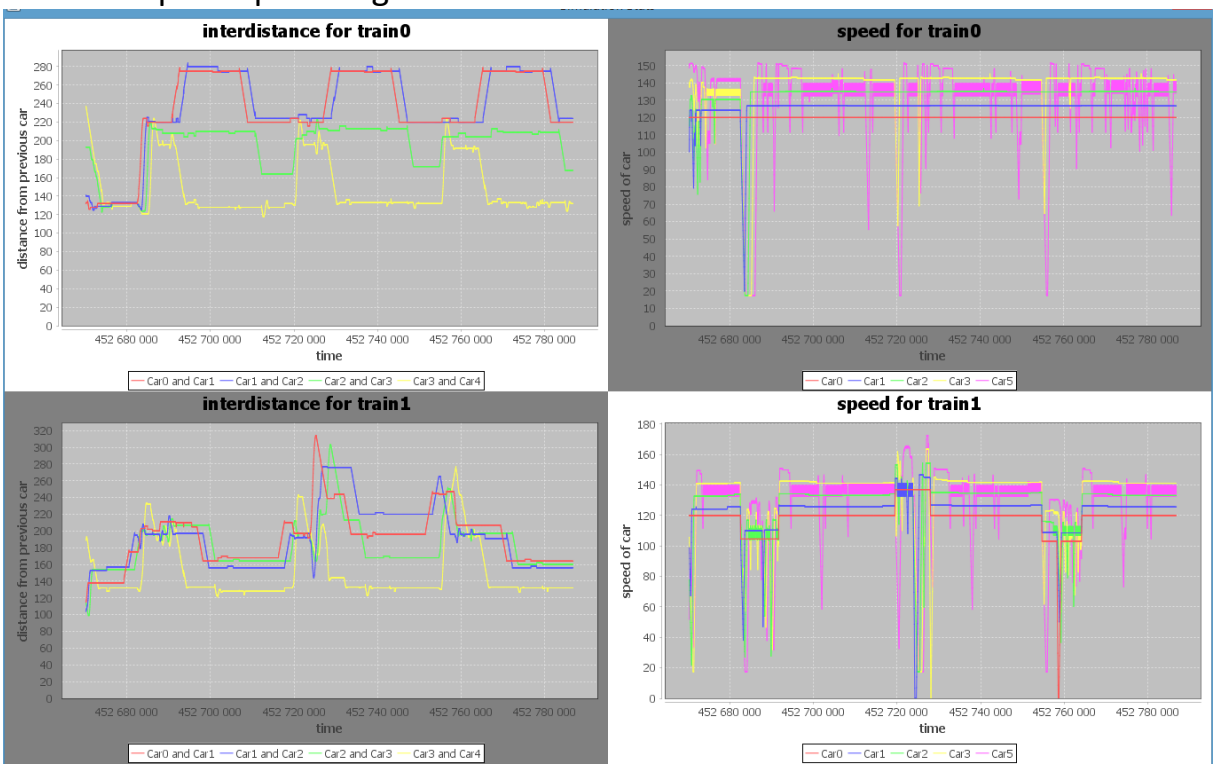

Figure 4 statistiques pour un ensemble de deux trains

On peut observer deux graphiques par train de véhicules, le premier calculant l'inter distance entre véhicules du convoi, et le second mesurant les vitesses des véhicules.

Bien que simples et non optimisés, ces graphiques m'ont permis de détecter des erreurs dans l'application des distances de sécurité, ainsi que de mesurer l'efficacité de certains facteurs. Par exemple, pour consolider l'intégrité du train, j'ai ajouté un coefficient à la vitesse maximale pour chaque voiture en fonction de sa position dans le convoi, ainsi une voiture à l'arrière aura tendance à aller plus vite pour rattraper un retard accumulé durant un

croisement. De la même manière, j'ai modifié les distances de perception afin d'alléger les traitements, gagnant ainsi de la fluidité sans pour autant réduire la qualité des comportements.

En observant un croisement j'ai pu remarquer que lors de ralentissement, les priorités établies allaient à l'encontre de la situation, créant plus d'écarts et de délais. J'ai donc mis en place un système de traitement des priorités, les rendant obsolètes et forçant leur calcul avec de nouvelles données en cas de besoin.

J'ai également travaillé sur le comportement des trains, notamment lors de la détection d'une situation de croisement, en effet le dernier train arrivé va adapter la vitesse de ces véhicules pour les chevaucher le mieux possible avec ceux du train en cour de traversée. Ce comportement avait été envisagé durant la première version, mais en raison de l'irrégularité des distances entre véhicules dans le train, il n'était pas efficace. Il se base sur une modification de la vitesse d'approche, tout en préservant la distance entre les véhicules, et en ajustant la vitesse une fois entré dans le carrefour.

IV. Bilan de fin de projet

Au vu des évolutions apportées, ainsi que du comportement observé des véhicules durant la simulation, on constate que les corrections et améliorations ont eu un effet positif quantifiable. Les véhicules ne se heurtent plus, et les croisements se font de manière relativement fluide. L'inteligence est complètement décentralisée, rendant possible l'utilisation de ce système sur de grands ensembles de véhicules, moyennant bien sur des outils adaptés pour une telle simulation.

On peut dès lors réfléchir sur les questions évoquées précédement, notament sur l'interêt et la faisabilité de convois de véhicules dans un système routier réel.

Bien que le fonctionnement soit perfectible, il est clair qu'il apporte une fluidité et une stabilité par rapport à un fonctionnement actuel de croisement de véhicules avec des priorités établies a priori. Un tel système ne nécesitant pas ou peu d'arrêts peut apporter vitesse de déplacement globale, notament en aglomération, économie d'energie du à la fluidité du traffic et donc mener à un meilleur fonctionnement .

V. Annexes

A. Annexe 1

Algorithme du véhicule à l'issue du premier projet, au sein de l'UV IA54.

- Vérifier la présence de nouveaux messages (de mon train, ou de véhicules que je vais croiser)
- Simuler la vitesse optimale (accélérer jusqu'à atteindre la vitesse maximale)
- Calculer la position après ce pas.
- Demander à l'environnement de m'informer des voisins présents dans cette nouvelle configuration.
- Si des voisins sont présents
  - traiter le cas du voisin le plus proche
  - s'il est avéré que c'est la prochaine voiture que nous allons croiser, on applique la priorité définie
  - si en revanche ce n'est pas l'actuel croisement mais qu'il fait partie de la liste des croisements prévus, on considère que les précédents croisements sont terminés et on met ladite liste à jour en appliquant les priorités établies.
  - finalement, si aucune priorité n'est définie, on vérifie si les voitures sont effectivement sur le point de se croiser,
    - si celles sont sur le point de se croiser, la plus proche du croisement obtient la priorité
    - celle qui détecte le croisement prévient l'autre voiture concernée, avec la distribution des priorités.

// La priorité, si elle a lieu d'être, est maintenant définie

- Dans le cas d'un croisement où notre voiture n'aurait pas la priorité, son objectif sera d'adapter sa vitesse pour passer juste après la voiture qu'elle doit laisser passer. On estime le temps restant avant que l'autre voiture passe, donc le temps dont nous disposons pour arriver à une certaine distance de sécurité du croisement. On en déduit sans mal notre vitesse modérée par les coefficients d'accélération et de freinage si besoins.
- Si la situation étudiée n'est pas un croisement, nous déduisons que nous suivons la voiture en question (le voisin le plus proche). 2 cas sont possibles :