

# Rapport de POM : IA et Pelotons de Véhicule autonome : Gestion automatique de la traversée d'une intersection intelligente

**BONNAVAUD Hedwin - 11407846**

**KLEINBAUER Vincent - 1**

**DALERY Martin - 11602536**

**Enseignant : AKNINE Samir**

Le 29 mai 2020

## 1. Introduction

Dans le cadre de l'évolution de l'intelligence artificielle, et de son utilisation pour le développement de véhicules autonomes, des méthodes de comportements pour ces derniers ont été étudiés pour optimiser la circulation.

En effet, le comportement humain, lorsqu'il s'agit de faire circuler un véhicule, n'est pas optimal. Or, le développement de véhicules autonomes pourrait offrir l'opportunité d'optimiser les comportements des véhicules au sein d'un environnement qui en est saturé.

Pour cela, l'une des solutions est de faire évoluer ces véhicules ensemble lorsqu'ils partagent des itinéraires communs. Il est possible notamment de comparer ce comportement à celui de fourmis se dirigeant vers un même objectif. Il s'agit dans les deux cas de mouvements d'entités évoluant dans un environnement qui en est saturé.

Dans le cadre de la mise en place d'algorithmes permettant ce type de mouvements, notre groupe d'étudiant a travaillé sur un sous-problème du déplacement en groupes de véhicules autonomes, celui de la prise de décision indépendante au sein de chaque véhicule autonome inclus dans groupe qui se présente à l'entrée d'une intersection.

Pour l'ensemble de ce document, toute mention du mot « véhicule » fera allusion à un véhicule autonome inclus dans un groupe d'un ou plusieurs véhicules autonomes.

Pour résoudre ce problème, nous avons admis qu'un véhicule est capable de choisir lui-même la direction qu'il souhaite prendre à une intersection donnée, et que la décision du passages ou de l'attente des différents véhicules peut être réalisé par l'intersection elle-même en tant qu'entité, afin de ne pas effectuer les mêmes calculs plusieurs fois en parallèle.

Afin de présenter notre travail, nous allons dans un premier temps présenter nos choix techniques, ainsi que la manière dont nous avons modélisé ce problème, avant de proposer une méthode permettant de résoudre et d'optimiser ce problème à l'aide d'un programme linéaire.

## 2. Modélisation

Pour résoudre notre problème, nous avons choisis de créer en python, un modèle nous permettant de faire des simulations, et d'observer le comportement des véhicules.

Dans ce but, nous avons créé une mini-ville artificielle représentée par un tableau bidimensionnel d'intersections.

Sur ces intersections vont circuler les véhicules. Ce sont les informations contenus dans les intersections qui leur permettent de choisir leur chemin.

### 1. Intersections

Ces intersections doivent permettre aux véhicules entrants de circuler sur elles selon la direction qu'ils doivent suivre pour rester sur leur itinéraire.

Pour permettre cela, ces intersections ont été remplies de positions de véhicules, répartis sous trois catégories :

- Des positions appartenant à des zones externes <sup>1</sup> ou "external area",
- Des positions appartenant à des zones d'entrées <sup>2</sup> ou "inner area"
- Des positions appartenant à une zone de conflit <sup>3</sup> ou "conflict zone"

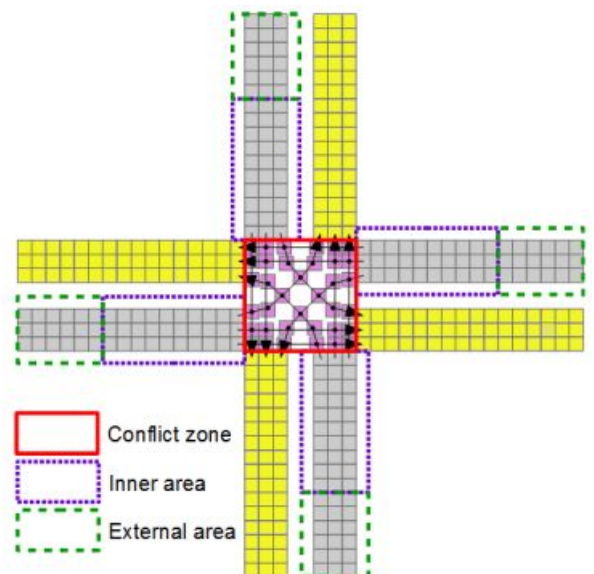
Les zones externes représentent les routes reliant les intersections. Ce sont des zones de passages pour les véhicules, et les véhicules qu'elles contiennent ne sont pas considérés comme faisant parties d'une intersection. Par conséquent, les véhicules s'y trouvant ne sont pas pris en compte lors du choix de l'ordre de traversée de l'intersection par les groupes de véhicules. C'est pour cette raison que nous avons choisis de lui donner une longueur nulle dans les paramètres par défaut.

Les zones d'entrées sont des zones précédant l'intersection. Les positions qui la composent, doivent permettre aux véhicules de se positionner correctement en vue de la traversée de l'intersection. Les véhicules qui s'y trouvent doivent se préparer à leur entrée dans la zone de conflit, c'est à dire à leur traversée de l'intersection. Ils ont donc besoins d'un temps de passage, temps à partir duquel ils vont quitter la zone d'entrée dans laquelle ils se trouvent pour entrer dans la zone de conflit.

De ce fait, dès qu'un véhicule rentre dans la zone d'entrée, il est de la responsabilité de l'intersection de lui attribuer un temps de passage.

Le passage dans cette zone est donc indispensable à la résolution du problème pour tout véhicule, et pour cette raison, sa taille doit-être strictement positive.

La zone de conflit d'une intersection est l'endroit où les groupes de véhicules vont-être amenés à se croiser. C'est dans cet zone que les groupes de



véhicules vont se déplacer en fonction de leurs itinéraires.

figure 1 : découpage d'une intersection en zones, extraite du document annexe

## 2. Positions

On appelle position chaque emplacements sur lesquels peuvent se trouver un véhicule.

Pour permettre aux véhicules de se déplacer dans notre modèle, chaque position a la connaissance des positions accessible par un véhicule depuis elle même.

Ces positions suivantes sont représentées par une liste de quatres positions, correspondant aux quatres directions possible qu'offre une intersection.

La première position de cette liste correspond à la position vers laquelle doit se déplacer un véhicule qui souhaite se diriger vers les vois de sorties en haut de l'intersection.

La seconde est celle menant à droite de l'intersection, et ainsi de suite dans le sens des aiguilles d'une montre.

Si une position ne peut pas mener vers une certaine direction, alors sa liste de fils ne sera pas complète. Cela ne pose pas de problème car si une position n'a pas de fils pour se rendre dans une certaine direction, son père n'en aura pas non plus, et ainsi de suite.

Ainsi, un véhicule entrant dans une intersection ne pourra jamais se tromper de voie et finir dans une impasse.

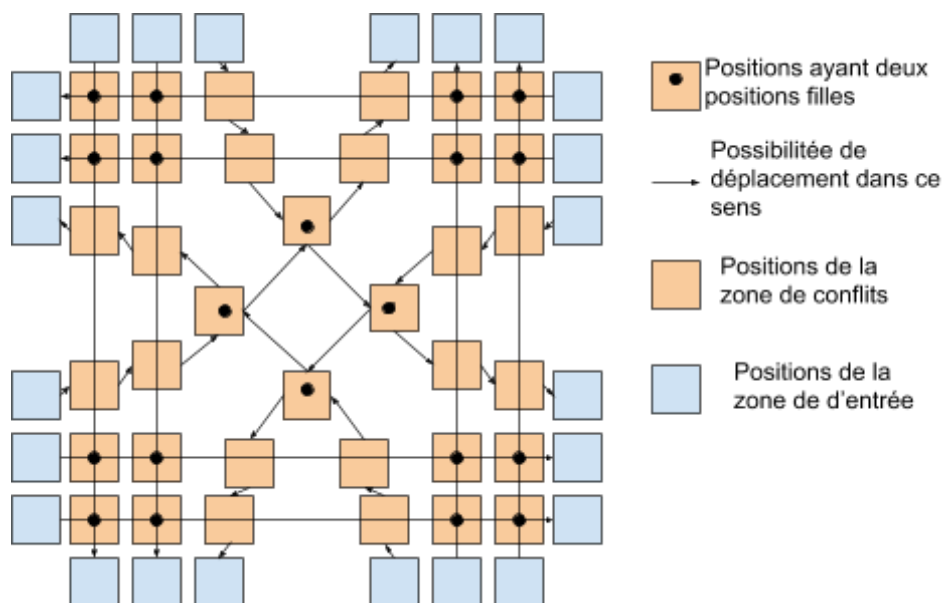


Figure 2 : organisation du graph orienté de positions dans une zone de conflit

Cependant, pour que cela fonctionne il faut n'importe quels positions d'une zone d'entrée puisse mener vers toutes les directions d'une intersection.

Une intersection a 12 voies d'entrées, et 12 voies de sorties, soit 2 fois 3 voies dans chaque directions. Une voie est composée d'une zone externe, suivie d'une zone d'entrée. Les positions qui composent ces voies forment une liste chaînée.

Ainsi, un véhicule peut continuer à se déplacer dans une voie tant cette dernière le mène vers la bonne direction.

Cependant, il peut arriver qu'un véhicule prévoyant de tourner à gauche l'intersection qu'il aborde, se trouve dans l'intersection la plus à droite. Il est donc important qu'une possibilité de déplacement dans les sens vers lesquels ne mène pas la voie courante soit possible depuis ces positions de la zone d'entrée. Ainsi, les positions de voies appartenant les unes aux autres peuvent pointer les unes vers les autres.

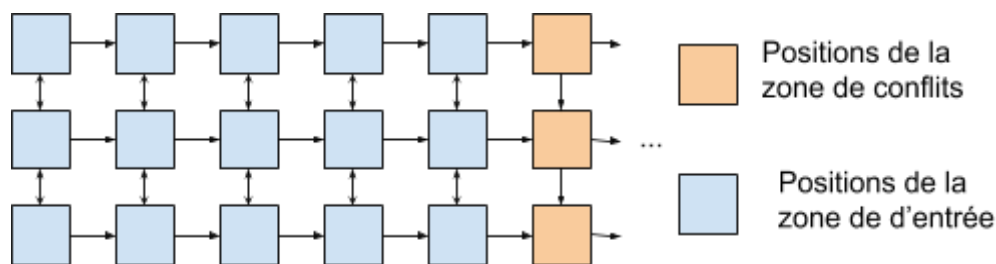


Figure 3 : Organisation des positions dans un groupe de voies

De cette manière, un véhicule entrant dans une intersection peut rejoindre toutes les directions possible, et même faire demis-tour.

Sur les figures 2 et 3, nous pouvons remarquer que notre graphe de positions ne s'arrête pas à une zone, mais relie les positions de toutes les zones d'une intersection. De la même manière, les graphes de positions de deux intersections adjacentes sont reliés l'un à l'autre par leurs voies sortantes/entrantes. Ainsi, notre ville artificielle n'est qu'un seul et même graphe, permettant au véhicules de suivre leur itinéraire.

### 3. Véhicules et groupes

Lors de sa création, un véhicule se voit attribuer un itinéraire aléatoire, sous forme d'une liste d'identifiants d'intersections. Dès qu'il franchit une intersection, ce véhicule met à jour sa connaissance de l'intersections qu'il s'apprête à traverser ensuite.

Ainsi, un véhicule a en permanence conscience de l'identifiant de l'intersections qu'il s'apprête à traverser, ainsi que de l'identifiant de l'intersection vers laquelle il devra se diriger pendant sa traversée.

Grâce à une connaissance globale des dimensions de la grille d'intersections, le véhicule est capable de déduire de ces deux informations la direction qu'il doit prendre pendant la traversée en cours.

Dans un groupe de véhicules, tous suivent le véhicule de tête. Les comportements décrits précédemment pour un véhicule seul s'appliquent aussi à un groupe de véhicules.

En effet, chaque véhicule a connaissance des véhicules qui l'encadrent dans un groupe.

Chaque véhicules a donc connaissance des véhicules qui le suivent. A chaque fois qu'il se déplace, il n'a alors qu'à indiquer aux véhicules qui le suivent ou se déplacer.

Ces véhicules ne font alors aucun calcul, ni pour connaître leur itinéraire, ni pour connaître leur temps de passage.

Inversement, un véhicule qui n'est pas leader de son groupe en a la connaissance car chaque véhicule connaît le véhicule qu'il suit. Un groupe de véhicule est donc une liste doublement chaînée de véhicules, et ainsi, le véhicule leader d'un groupe sait quand il a la responsabilité d'orienter son groupe.

### 3. Résolution

#### 1. Résolution pour des véhicules indépendants

La publication en annexe présente une résolution d'un problème de franchissement d'intersection par des véhicules autonomes sous forme d'un DCOP.

Nous allons dans notre cas utiliser le même type de résolution, en revisitant quelque peu les élément de ce programme linéaire.

Voici le programme linéaire du document en annexe :

Soit

- $t$  le temp courant,
- $\phi_i$  le temps d'acceptation<sup>1</sup> du véhicule  $i$ ,
- $V_t$  l'ensemble des véhicules présents dans l'intersection à l'instant  $t$ ,
- $l_i$  la voie dans laquelle se trouve le véhicule  $i$  avec  $l_i \in L$  l'ensemble des voies,
- $n_i$  la distance entre  $v_i$  et la zone de conflit,
- $s_i$  la vitesse du véhicule  $i$ ,
- $\tau_i$  la trajectoire du véhicule  $i$  dans la zone de conflit,
- $pos(e, \tau_i)$  l'emplacement de la position  $e$  dans la trajectoire du véhicule  $i$ ,

$$\text{Minimiser } \sum_{v_i \in V_t} w_i$$

Sous contrainte de

$$\text{Contrainte de distance : } \forall v_i \in V_t, \phi_i > t + \frac{n_i}{s_i}$$

$$\text{Contrainte d'antériorité : } \forall v_i, v_j \in V_t^2, l_{v_i} = l_{v_j} \wedge n_i < n_j \Rightarrow \phi_i < \phi_j$$

$$\text{Contrainte de conflit simple : } \forall v_i, v_j \in V_t^2, \forall e \in \tau_i, e \in \tau_j \Rightarrow (\phi_i + \frac{pos(e, \tau_i)}{s_i}) \neq (\phi_j + \frac{pos(e, \tau_j)}{s_j})$$

La contrainte de conflit simple pouvant être remplacée par la contrainte de conflit avec intervalle de sécurité :

$$\forall v_i, v_j \in V_t^2, \forall e \in \tau_i, e \in \tau_j \Rightarrow \| (\phi_i + \frac{pos(e, \tau_i)}{s_i}) - (\phi_j + \frac{pos(e, \tau_j)}{s_j}) \| > t_{safe}$$

Avec un tel programme linéaire, suffit à résoudre notre problème pour des véhicules seuls, sans formation de groupes. Et il permet même de résoudre le même problème avec de multiples intersections comme c'est le cas dans notre modèle.

<sup>1</sup> Temp auquel le véhicule  $i$  s'engagera dans la zone de conflit

Il suffirait pour cela, au sein d'une intersection, de recalculer les temps de passages de chaque véhicules se trouvant dans la zone d'entrée lorsqu'un nouveau y rentre, et ce en prenant en compte les véhicules les mouvements des véhicules se trouvant dans la zone de conflits.

Cependant, pour résoudre notre problème contenant des groupes de véhicules, nous allons devoir modifier certains éléments de ce programme linéaire pour qu'il apporte une réponse optimisée à notre problème.

Premièrement, nous pouvons remarquer que dans notre modèle discret, les véhicules ont tous la même vitesse de 1 position / intervalle de temps. Nous pouvons donc en déduire que

$$s_i = 1 \Leftrightarrow \frac{n_i}{s_i} = n_i \text{ et } \frac{pos(e, \tau_i)}{s_i} = pos(e, \tau_i)$$

Nous pouvons remarquer que la contrainte de conflit simple ne prend pas en compte la taille des véhicules. Elle n'est pas adaptée dans le cas où des véhicules ont une taille supérieure à celle d'une position.

Elle semble donc très adaptée à notre modèle très simple où tous les véhicules font la même taille qu'une position.

Cependant, étant donné que nous cherchons à adapter ce DCOP à un modèle manipulant des groupes de véhicules autonomes, nous pouvons utiliser une contrainte de conflit avec intervalle de sécurité pour gérer un groupe de plusieurs véhicules comme étant un véhicule ayant la taille du groupe en question.

## 2. Résolution pour des groupes de véhicules

Dans ce cas,  $t_{safe}$  correspondrait alors à la longueur du groupe s'engageant en premier.

Nous pouvons alors réécrire les contraintes comme suit :

$$\text{Contrainte de distance} : \forall v_i \in V_t, \varphi_i > t + n_i$$

$$\text{Contrainte d'antériorité} : \forall v_i, v_j \in V_t^2, l_{v_i} = l_{v_j} \wedge n_i < n_j \Rightarrow \varphi_i < \varphi_j$$

*Contrainte de conflit avec intervalle de sécurité :*

*Soit  $len(g_i)$  = longueur du groupe  $g_i \in G$  l'ensemble des groupes de l'intersection*

*Soit  $\forall v_i, v_j \in V_t^2, \forall e \in \tau_i, e \in \tau_j, diff(e, v_i, v_j) = (\varphi_i + pos(e, \tau_i)) > (\varphi_j + pos(e, \tau_j))$*

*$\forall v_i, v_j \in V_t^2, \forall e \in \tau_i, e \in \tau_j, diff(e, v_i, v_j) > 0 \Rightarrow diff(e, v_i, v_j) > len(g_i)$*

Cependant, cette nouvelle contrainte remet en question l'efficacité de notre objectif d'optimisation. En effet, si l'un des groupes de l'intersection est trop long, il pourrait créer un bouchon en empêchant tous les véhicules de s'engager.

Il est donc important d'optimiser également la taille des groupes de véhicules.

Pour les raisons énoncées en introduction, la formation de groupes de véhicules au lieu de véhicules autonomes permet d'optimiser l'espace occupé par des véhicules sur une route en limitant la taille

nécessaire pour les distances de sécurité, ou en permettant aux véhicules d'un même groupe d'avancer simultanément lorsqu'ils redémarrent au lieu d'un départ différé dans le cas général, limitant ainsi les bouchons.

Pour ces raisons, il est important d'avoir un maximum de véhicules groupés avec des groupes le plus grand possible au sein d'une même route, afin d'en optimiser l'utilisation.

Cette optimisation revient à minimiser le nombre de groupes de véhicules, en comptant chaque véhicule indépendant comme formant un groupe d'un seul véhicule.

La fonction d'optimisation prendra donc le nombre de groupe en compte. Mais il reste encore à trouver un rapport adéquat entre la somme des temps d'attente et le nombre de groupes pour que les deux aient une importance.

Soit :

- $\phi_i$  le temps d'acceptation du véhicule  $i$
- $\tau_i$  la trajectoire locale du groupe  $i$  dans la zone de conflit suivante,
- $G$  l'ensemble des groupes présents dans l'intersection
- $g_i \in G$  le groupe du véhicule  $v_i$
- $l_i$  la voie dans laquelle se trouve le groupe  $i$  avec  $l_i \in L$  l'ensemble des voies
- $n_{g_i}$  la distance entre le véhicule de tête du groupe  $g_i$  et la zone de conflit,
- $pos(e, \tau_i)$  l'emplacement de la position  $e$  dans la trajectoire du groupe  $g_i$ ,
- $rang(v_i, g_i)$  le rang du véhicule  $v_i$  dans le groupe  $g_i$ , valant 1 pour le véhicule de tête.
- $len(G_i) = g$  le nombre de groupe
- $len(V_i) = v$  le nombre de véhicules
- $V_i$  devient l'ensemble des véhicules présent dans la zone d'entrée et n'appartenant pas à un groupe déjà engagé dans la zone de conflit.

Il est important de chercher un temps de passage pour chaque véhicule. En effet,  $\phi_1, \dots, \phi_v$  étant des variables, nous devons connaître leur nombre au début de la résolution du programme linéaire. Or le nombre de groupes va évoluer au fil de l'algorithme de résolution. Il est donc impossible d'avoir une variable par groupes.

Ces modifications nous permettent de revisiter le programme linéaire précédent comme suit,

$$\text{Variables : } V = \{\phi_1, \dots, \phi_v, g_1, \dots, g_v\}$$

$$\text{Minimiser } \sum_{g_i \in G_i} w_i + g$$

La distance  $n_i$  indiquait la distance entre le véhicule  $i$  et la zone de conflits. Désormais, cette distance se calcule en fonction de la position du véhicule dans son groupe.

$$\text{Contrainte de distance : } \forall v_i \in V_t, \phi_i > t + n_{g_i} + rang(v_i, g_i)$$

$$\text{Contrainte d'antériorité : } \forall v_i, v_j \in V_t^2, l_{v_i} = l_{v_j} \wedge n_{g_i} + rang(v_i, g_i) < n_{g_j} + rang(v_j, g_j) \Rightarrow \phi_i < \phi_j$$

La contrainte de conflits avec intervalle de sécurité ne change pas (C.F. page 6).

Selon les raisons pour lesquels nous formons des groupes de véhicules, il est inutile d'avoir des groupes de véhicules dont les véhicules ne sont pas immédiatement suivis les uns par les autres. On cherche donc à avoir des groupes sans espaces.

De plus, un groupe trop espacé pourrait poser d'importants problèmes par rapport à la contrainte de conflits avec intervalle de sécurité, car elle imposerait une intervalle de sécurité trop importante.

Nous devons donc ajouter une contrainte de densité :

$$\text{Contrainte de densité} : \forall v_i, v_j \in V_t^2, g_i = g_j, \Rightarrow \varphi_i = \varphi_j + (\text{rang}(v_i, g_i) - \text{rang}(v_j, g_j))$$

Nous avons définis les groupes comme étant des ensembles de véhicules suivant un seul véhicule. Il est donc indispensable que les véhicules composant un même groupe aient tous le même itinéraire, au moins pour le franchissement de l'intersection courante :

$$\text{Contrainte de direction} : \forall v_i, v_j \in V_t^2, g_i = g_j, \tau_i = \tau_j$$

Ces nouvelles contraintes nous permettent alors enfin d'obtenir un programme linéaire, permettant de résoudre notre problème de départ.

## 4. Conclusion

Durant ce projet, nous avons pu créer un modèle paramétrable permettant d'observer et de générer le franchissement d'intersection par des groupes de véhicules autonomes.

Malheureusement, nous ne sommes pas parvenu à implémenter le programme linéaire présenté dans notre dernière partie, afin d'optimiser les temps de passages, à la fois pour limiter le temps d'attente des véhicules, que pour limiter les conflits (bouchons) ou les tailles des groupes.

Cependant, notre modèle reste facilement adaptable, et pourrait faire un environnement intéressant pour le tests de multiples méthodes de résolution de notre problème de franchissement d'intersections par des groupes de véhicules autonomes.

De plus, en ayant l'avantage d'offrir, non pas une seul et unique intersection, mais une grille bi-dimensionnelles d'entre elles, notre modèle pourrait mener vers la résolutions d'autres problèmes comme l'optimisation des itinéraires globaux des véhicules, pour désengorger certaines intersections et ainsi, favoriser l'optimisation de notre problème plus local de traversée d'une intersection intelligente.

## 5. Sources

Annexe 1 : A Decentralised Approach to Intersection Traffic Management, Huan Vu, Samir Aknine and Sarvapali Ramchurn