Caractérisation et gestion des embouteillages

Des capteurs pour résorber les embouteillages

Dans ce chapitre, nous nous intéressons au cas particulier des réseaux embouteillés. La littérature montre que lorsque la charge dépasse un certain seuil, le comportement du réseau routier est fondamentalement différent du cas fluide. Les métriques que nous utilisons traditionnellement sont moins efficaces. C'est pourquoi nous proposons ici un algorithme alternatif, spécifique à cette situation et tirant parti du réseau de capteurs déployé. Nous baptisons cet algorithme BASALTE (*distriButed and Adaptive Signalization ALgorithm for saTurated intErsections*).

État de l'art des systèmes embouteillés

6. 1. 1 États de trafic et formation des embouteillages

Lors de la mise en place de routes, le concepteur de voirie détermine leur capacité, et notamment leur nombre de voies, en fonction d'un certain niveau de circulation. Afin de considérer l'ensemble des types de véhicules, l'unité UVP (*unité de véhicule particulier*) est utilisée. Tandis qu'un véhicule léger représente 1 UVP, un poids lourd en représente 2 et un cycle 0,3. Ainsi, une route à quatre voies, sans terre plein central, a une capacité moyenne de 10 000 à 13 000 UVP par jour (ceci pouvant varier en fonction des conditions de circulation, de la pente de la route, etc.).

Traditionnellement, les embouteillages se forment lorsque la quantité de trafic subi par une route approche la capacité de cette route, ou la dépasse. Cette hausse de trafic est souvent constatée aux heures de pointes. La capacité de la route peut également, par exemple, être revue à la baisse si des accidents ou des obstacles matériels s'y trouvent. Enfin, le comportement individualiste des conducteurs mène souvent à des ralentissements. Si un véhicule appartenant à une file ralentit ou n'a pas une vitesse constante, des perturbations sont créées et peuvent se propager de véhicule en véhicule, pour finalement ralentir l'ensemble de la file [Sug+08]. Cet effet

est habituellement appelé effet accordéon, un terme à l'origine popularisé par Denis Niquette, un chroniqueur de circulation de la première chaine de Radio-Canada [Wik].

Les embouteillages et leurs formations sont décrits en différents niveaux d'états de trafic par la littérature ([LCP75 ; AM97 ; DHS08 ; HMB11]). [LCP75] donne en particulier de bonnes définitions sur les états de la performance du trafic, orientés autour du mécanisme de formation des files de véhicules et représentés sur la figure 6.1.

Lorsque la demande en trafic sur une intersection est nettement inférieure à la capacité de cette intersection (c.-à-d., le débit qu'une intersection est capable d'absorber), aucune file ne se forme lorsqu'un feu vert passe au rouge. Elle est alors décrite comme étant *non-saturée ou sous-saturée*.

Dans le cas où la demande approche ou dépasse la capacité de l'intersection, des files se forment et il reste des véhicules à l'issue d'un feu vert. Dans ce cas, un embouteillage se forme localement. Ce cas de figure est décrit comme étant *saturé*. Lorsque la file qui se forme reste constante au cours du temps, on parle de *saturation stable*. Lorsque cette file a tendance à augmenter progressivement, on parle de *saturation instable*.

Dans les cas où le trafic est supérieur à la capacité d'accueil d'une direction entrante, cette dernière a tendance à se remplir rapidement au fil des cycles, jusqu'à être totalement pleine et bloquer totalement les arrivées venant de l'intersection voisine commune à cette direction. Les effets provoqués par le délai d'attente ne sont alors plus locaux. Dans le cas où l'encombrement d'une intersection influe sur les intersections lui envoyant des véhicules, on parle de *spillback* (ou *déversement-retour*). Ce type de scénario est qualifié de *sursaturé*. Aucune solution de contrôle efficace n'existe pour ce type de cas, toutes faisant office de moyen de prévention [Yan+13]. À ce moment, l'embouteillage formé a tendance à s'étendre, depuis une ou plusieurs intersections jusqu'à potentiellement bloquer temporairement toute ou partie d'une artère principale ou d'un réseau de plusieurs routes, créant une *fermeture* [AAG11], comme représenté sur la figure 6.2. Dans ce cas, le temps de résolution du problème dépend notamment de la taille de la zone congestionnée et de la capacité des intersections, se situant en bordure de cette zone, à évacuer les véhicules.

Dans certains cas extrêmes, nous pouvons faire face à un *interblocage* [CES71], comme représenté par la figure 6.3. La seule solution pour résoudre le problème est que les véhicules reculent ou fassent des manœuvres de dégagement. La plus simple règle à appliquer pour éviter les interblocages serait que les véhicules n'entrent sur une intersection que si ils sont assurés de ne pas y être stoppés.

Pour une direction entrante, un scénario de saturation instable commence à se manifester à partir du moment où $V/C > 0.9$ [Cj+12], avec V/C le rapport espace occupé (volume) sur capacité (sec. 2. 3. 3). Un scénario de sursaturation se manifeste quant à lui lorsque $V/C > 1$. Dans ce cas, l'infrastructure subit une demande en trafic qu'elle n'est pas capable d'accueillir. Les méthodes de gestion de feux de circulation gérant ce type de situations ne peuvent qu'alléger la situation, en faisant leur possible pour maximiser le débit (en attendant que la charge ne baisse) et envoyer le trafic vers des zones moins embouteillées.

De là, nous voyons toute la différence entre un scénario sous-saturé et un scénario sursaturé. Le premier peut se fier aisément au délai d'attente des véhicules pour déterminer une stratégie de contrôle des feux de circulation. Dans le deuxième cas, ce délai n'est pas nécessairement le plus représentatif car il ne tient pas compte des facteurs propres aux embouteillages. Les auteurs de [DRK04] montrent, en comparant de nombreux modèles d'analyse des délais (p. ex., files d'attente, Webster [Web58]), que les intersections à feux de circulation ne peuvent pas à la fois maximiser leur capacité et minimiser leur délai lorsqu'elles approchent de conditions saturées ou sursaturées. De cette manière, nous identifions clairement deux situations différentes à traiter, l'une maximisant le débit de véhicules, l'autre minimisant la file de la route critique.

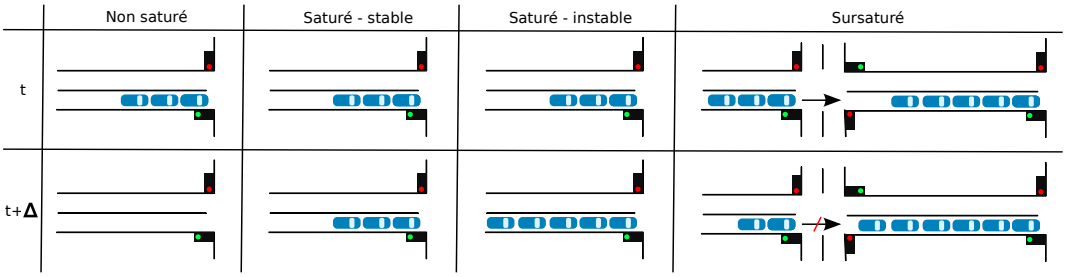


FIGURE 6.1 – Les différents états de trafic.

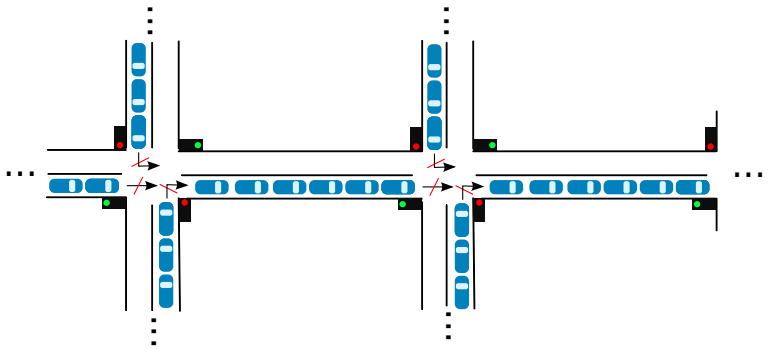


FIGURE 6.2 – Le réseau routier est bloqué, aucune place n'est libre entre les deux intersections. Ce cas de figure étend logiquement l'importance de l'embouteillage.

Dans les précédents chapitres, nous avons étudié une architecture de surveillance capable de comptabiliser le nombre de véhicules et de prendre des décisions locales. Nous pouvons facilement imaginer que cette architecture est capable de détecter les cas où le trafic atteint un certain seuil de saturation, afin d'adapter son comportement en conséquence. Par exemple, en gardant un historique de la taille d'une file et en regardant le nombre de cycles d'augmentation stricte. Dans ce chapitre, nous décrivons un algorithme de gestion des feux de circulation applicable dans les cas où le trafic est embouteillé, en privilégiant les contraintes décrites par la littérature dans la section suivante.

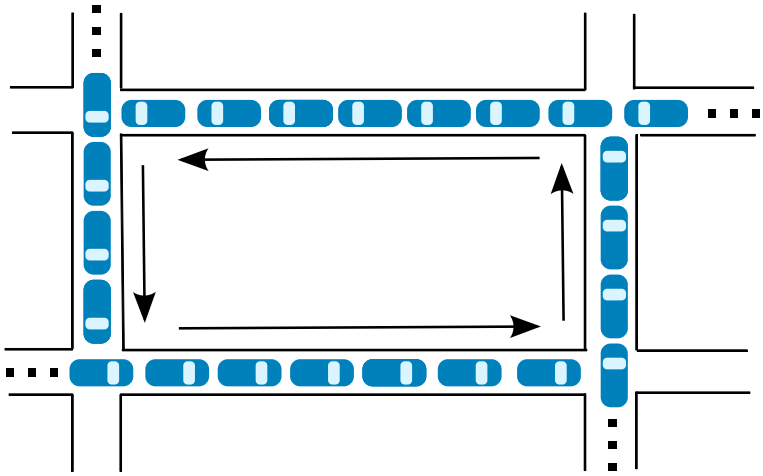


FIGURE 6.3 – Présence d'un interblocage : les véhicules doivent faire marche arrière pour débloquer la situation.

6. 1. 2 Contrôle des feux de circulation dans des conditions saturées

Dans des conditions allant jusqu'à la sursaturation, les techniques d'optimisation doivent prendre en compte des critères ayant directement rapport aux caractéristiques de la saturation, tels que la taille des files comme objectif ou contrainte. Les paragraphes ci-dessous résument les travaux, essentiellement développés durant ces 40 dernières années, touchant à l'optimisation des feux de circulation dans des conditions de saturation ou de sursaturation. D'une manière générale, nous constatons deux axes empruntés par la littérature : certains tentent d'allouer des ressources *temporelles* aux véhicules les plus en difficulté, tandis que d'autres tentent de leur allouer des ressources *spatiales* [Yan+13].

6. 1. 2. 1 Allocation de ressources temporelles

Les premiers travaux sur le contrôle des feux de circulation dans des conditions saturées ont été réalisés sur des feux prédéterminés par Gazis *et al.*. ([Gaz64 ; GP65]). Sur une intersection, leur idée est d'allouer le temps inutilisé par les phases non saturées sur les phases possédant plus de véhicules, afin de favoriser la décharge des files résiduelles. Ainsi, le délai total peut être réduit en donnant le feu au vert aux files possédant un nombre important de véhicules, plutôt que de le donner aux files possédant peu ou pas de véhicules. De cette manière, le débit sur l'intersection est plus important, les files étant gardées à un niveau acceptable. Cette technique est largement utilisée par des stratégies de contrôle adaptatives ou encore par la NEMA en Amérique du nord. Il est en effet possible pour un contrôleur de feux d'étendre ou de terminer une phase plus tôt, et d'allouer le temps restant à une phase plus saturée [Yan+13]. D'Ans et Gazis [DG76] étendent par la suite le travail de Gazis [Gaz64] à n'importe quel nombre de feux sur la base de plans de feux prédéterminés, en tentant de minimiser le temps perdu par les véhicules au sein des files. Ils prouvent que résoudre le problème de sursaturation implique une allocation

optimale des routes aux conducteurs, un choix favorisant les mauvais critères pouvant facilement mener à un aggravation de la situation.

[Lon68] propose une méthode qui gère en temps réel les files de manière à minimiser le nombre d'intersections secondaires bloquées. L'algorithme fonctionne en faisant varier la durée des feux verts entre un minimum et un maximum afin de garder des files de niveaux comparables. Son algorithme fonctionne sur une intersection isolée et sur un réseau de quatre intersections. Sa philosophie de contrôle est basée sur le fait que les feux de circulation ne peuvent pas libérer les files qui sont dans un état de sursaturation, leur fonction est donc prévenir ce cas de figure en maintenant les files à un niveau acceptable, afin qu'elles ne bloquent pas les intersections adjacentes.

Michalopoulos et Stephanopoulos [MS77] proposent une politique de contrôle sur une, puis sur deux intersections sursaturées avec des rues à sens unique. Leur étude considère, en plus de la taille des files d'attente comme contrainte, le temps de traversée entre deux intersections. Dans [MSS81], les auteurs développent une politique de contrôle en temps réel minimisant le délai total des intersections soumises à des contraintes de longueur de files d'attente. Leur stratégie est plus efficace qu'un plan de feu prédéterminé mais est uniquement applicable sur des intersections isolées.

[LRKS86] et [Rat88] suggèrent deux stratégies de contrôle, l'une jouant sur la durée des feux verts et l'autre contrôlant la longueur des files le long des artères, de manière à limiter la probabilité de spillback. D'autres auteurs tentent également de retarder ou d'éliminer le blocage entre intersections. [LCP75] propose une approche donnant le feu vert à une file lorsque sa taille devient supérieure ou égale à une longueur prédéterminée. [AM97] proposent de minimiser le nombre de spillbacks tout en maximisant le nombre de véhicules qui peuvent passer sur une artère de plusieurs intersections sursaturées et dont les rues de croisement possèdent une demande moyenne.

[HHD12] indique que les vagues vertes créées classiquement par les systèmes de contrôle ne sont pas directement applicables à un scénario sursaturé. En effet, la création d'une bande de progression (sec. 2. 2. 3. 5) repose traditionnellement sur le temps de traversée théorique liant les intersections les unes aux autres. En se servant de cette valeur comme décalage dans un scénario embouteillé, un flux de véhicules arrivant se retrouverait systématiquement ralenti, voire bloqué, par des véhicules déjà présents. Si une intersection reste saturée pendant un délai assez long, une telle coordination est susceptible d'aggraver la situation, en créant des spillbacks sur les intersections voisines émettrices ([LWMH09 ; LC11]). Pour cela, plusieurs auteurs se sont penchés sur l'estimation d'une bande de progression variable en fonction des informations de trafic courants (p. ex. [SG96 ; HTYJ11]). Dans ce cas, une bande de progression, habituellement fixe, est déterminée en fonction du trafic. Le problème est que pour s'assurer que beaucoup de véhicules passent cette bande de progression, elle doit durer longtemps, ce qui tend vers un scénario fixe et réduit l'intérêt d'un système adaptatif (une phase avec un temps de feu grand, les autres avec un plus faible).

Certains auteurs se penchent sur des scénarios limités, et utilisent ou génèrent souvent des plans de feux prédéterminés. Les auteurs de [YLC06] utilisent un algorithme génétique et un plan de feux fixe pour déterminer le séquençage d'un réseau

de trois intersections pour une période de sursaturation de trois minutes. Zhang *et al.*. [ZYL10] utilisent un algorithme génétique ainsi qu'un cycle et des feux verts de longueur fixe afin de construire un plan de feux prédéterminé pour une artère traversant 5 intersections surchargées.

Les auteurs de [PMUI00] utilisent un algorithme génétique qu'ils expérimentent sur une artère principale de quatre intersections. Ils utilisent trois fonctions objectifs : minimisation de délai, minimisation de délai modifié avec une fonction de pénalité, et maximisation de débit. Ils prouvent que l'utilisation du premier objectif produit un temps de feu supérieur comparé aux autres stratégies et à la stratégie proposée par TRANSYT-7F [1], un simulateur macroscopique de trafic routier et d'optimisation des temps de feux.

Chang *et al.*. [CBX10] tentent de maximiser le débit tout en gérant les files du système. Leur modèle est testé sur une artère de deux intersections et augmente l'utilisation de la capacité d'une intersection, ce qui génère un délai d'attente plus bas de 22% en comparaison à TRANSYT-7F.

Les auteurs de [CM00] proposent d'ajuster le calcul du temps de cycle formulé par Webster [Web58] à différents niveaux de saturation du trafic.

Certains auteurs s'essayent également à des méthodes moins habituelles. Putha *et al.*. [PQ10] utilisent un algorithme de colonie de fourmis pour résoudre le problème de la coordination de signaux pour un réseau sursaturé. Robertson et Bretherton [RB74] et Gartner [Gar83] utilisent des méthodes par programmation dynamique, décomposant des problèmes complexes en plusieurs problèmes plus simple mais souffrant de problèmes de passage à l'échelle, et demandant de connaitre des données précises et nombreuses au cours du temps.

6. 1. 2. 2 Allocation de ressources spatiales

D'autres auteurs se sont penchés sur l'utilisation de la place disponible sur chaque intersection. [Gor69] propose un algorithme tentant de rendre égal pour chaque phase le rapport entre la taille des files et la capacité maximale. De récentes études [BUB06] montrent que l'utilisation de la place libre comme critère peut retarder la propagation d'un embouteillage, en coupant une phase lorsqu'il y a des restrictions sur les intersections voisines. Ceci peut empêcher les effets spillback [AAG11] : lorsqu'un capteur n'enregistre plus de véhicule pendant le feu vert, c'est que les véhicules n'ont nul part où aller. [Yan+13] montre par simulation qu'une méthode privilégiant des critères de place génère un temps d'attente moins élevé que des méthodes s'appuyant sur une bande de progression classique et de temps.

6. 1. 3 Limites de la littérature

L'une des principales limites de la littérature repose sur le fait que les systèmes proposés soient généralement centralisés, trop complexes ou trop éloignés des systèmes actuels pour être abordés dans la réalité (p. ex., colonies de fourmis). De nom-

1. http://mctrans.ce.ufl.edu/featured/TRANSYT-7F/

breux auteurs cités précédemment considèrent en effet pouvoir agir simultanément sur toutes les intersections d'un réseau, souvent par l'intermédiaire d'algorithmes génétiques ou de méthodes issues de l'optimisation combinatoire. Outre le fait que ces approches seraient difficilement tolérantes aux fautes et implémentables, il nous faut souligner la complexité des calculs et l'importance des données à conserver afin de rendre le système efficace, surtout dans des cas de contrôle adaptatif ou en temps réel ([Lon68 ; Gor69 ; LCP75]). Les algorithmes génétiques imposent par exemple de garder en mémoire une grande quantité d'informations. Les méthodes propres à l'optimisation combinatoire, souvent complexes, nécessiteraient des calculs longs, les paramètres étant nombreux et l'optimisation d'une séquence de feux n'étant pas triviale [HMB11]. De plus, l'échange quasi permanent des informations de trafic et de décisions rendrait un tel système non viable sur une échelle plus large que les scénarios étudiés. Dans le cas de systèmes prédéterminés, si la constitution ou l'utilisation de plans de feux statiques pour une demande connue ne pose pas de problème de dimensionnement, le manque de temps réel et d'adaptation pose problème.

Ensuite, notons pour un certain nombre d'auteurs le manque de généralisation de leur algorithme, qui est conçu pour un scénario particulier ou qui ne considère pas des métriques importantes dans des cas d'embouteillage. Par exemple, Gazis [Gaz64] suppose que le temps de traversée ainsi que le stockage des véhicules entre deux intersections est négligeable. En d'autres termes, il néglige les contraintes spatiales de distance entre les carrefours et la place prise par les files d'attente. Singh et Tamura [ST74] proposent une méthode ne prenant pas en compte les interférences entre les véhicules arrivant et ceux déjà présents sur une intersection. Leur étude est donc limitée, car lorsqu'une intersection déborde sur une autre, il nous faut pouvoir identifier et traiter la situation. [MS77] considère davantage de métriques, mais ne les a appliqué qu'à une route à sens unique avec un séquençage limité. [MSS81] propose une politique de contrôle en temps réel, testée et validée mais uniquement valable pour les intersections isolées.

En résumé, de nombreuses recherches dans le domaine du contrôle du trafic routier pour des environnements saturés reposent sur des hypothèses trop restrictives pour être appliquées dans un système réel [AM97]. Les modèles sont souvent régis par des contraintes statiques, ou à l'inverse sont trop complexes pour passer l'échelle de la réalité. Ils manquent souvent de généralisation et prennent parfois en compte des hypothèses irréalistes de nos jours.

6. 2 BASALTE : *a distriButed and Adaptive Signalization ALgorithm for saTurated intErsections*

Dans cette partie, nous proposons une solution de contrôle des feux de circulation sur des intersections saturées, baptisée BASALTE (*distriButed and Adaptive Signalization ALgorithm for saTurated intErsections*). Cet algorithme est distribué sur chaque intersection et se sert des constats faits par la littérature pour proposer

une méthodologie de prévention et de déconstruction des embouteillages. La distribution des capteurs permet une réaction rapide aux embouteillages, tandis que la résolution du problème peut être géré indépendamment par plusieurs points.

6. 3 Architecture

Cet algorithme se base sur l'architecture entièrement adaptative décrite en section 3. 2. En comparaison aux autres architectures, de nouveaux nœuds sont utilisés sur les voies sortantes de l'intersection : les nœuds *AN*. Ces nœuds permettent notamment de savoir où les véhicules vont une fois sortis de l'intersection, ce qui nous semble essentiel afin de mesurer et agir sur un embouteillage. Nous définissons deux couches de communication nous permettant de mener à bien les différentes opérations de notre algorithme, comme représenté sur la figure 6.4. Bien que le nombre de capteurs nécessaire soit plus important que pour TAPIOCA, cette nouvelle hiérarchie est toutefois plus légère, car nous souhaitons exploiter la capacité de chaque nœud à pouvoir décider du changement des feux. Cette intuition nous permettrait, *a priori*, de gagner en réactivité et en souplesse.

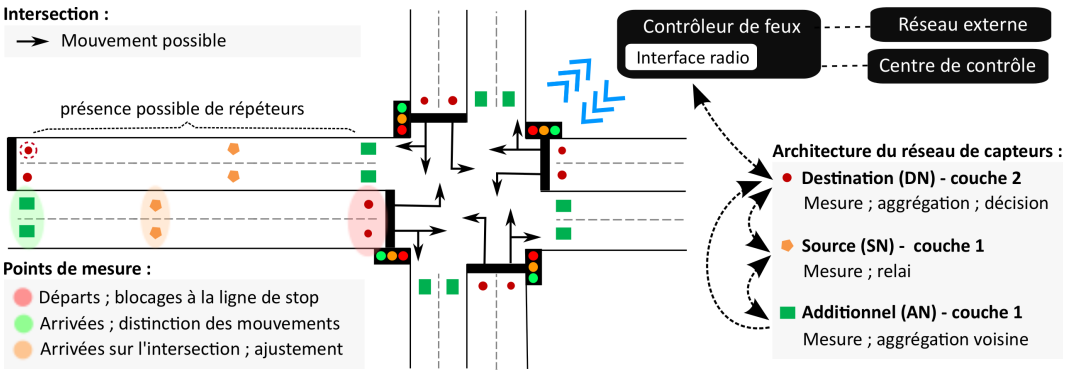


FIGURE 6.4 – Architecture de BASALTE.

La logique que nous souhaitons expérimenter est basée sur le fait que chaque nœud *DN* est responsable des mouvements autorisés par sa voie. Afin de pouvoir prendre une décision, chaque nœud *DN* (couche 2) reçoit les informations des nœuds *SN* et *AN* (couche 1) qui sont pertinentes pour lui. Les nœuds *DN* ont la possibilité de communiquer entre eux afin de se transmettre des informations, ainsi qu'avec un nœud intermédiaire, qui arrange et tri les requêtes reçues. Ce nœud intermédiaire pourrait être un nœud *DN* élu parmi ceux existant, mais nous supposons par la suite qu'il s'agit d'une interface reliée au contrôleur de feux, ou le contrôleur de feux lui-même. Le détail de l'algorithme associé est donné dans la section suivante.

La figure 6.5 représente le diagramme de collaboration de BASALTE, décrivant les relations existantes entre les nœuds et les opérations de l'algorithme, que nous détaillons dans la section suivante.

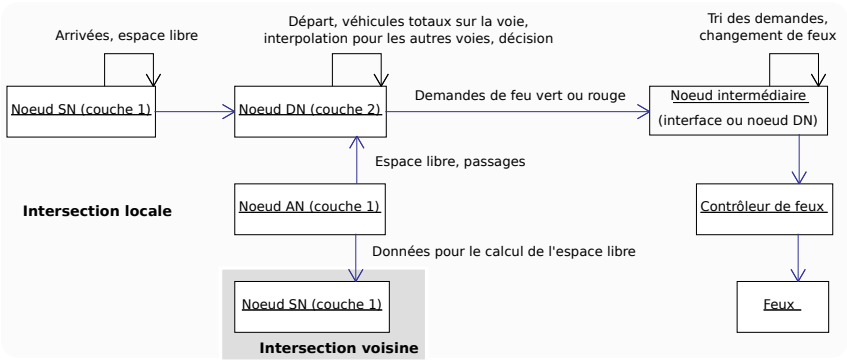


FIGURE 6.5 – Diagramme de collaboration de BASALTE pour le cas d'un réseau multi-sauts.

6. 4 Algorithme de contrôle des feux de circulation

Nous décrivons ici BASALTE en trois étapes. Tout d'abord, les nœuds DN acquièrent une vision de l'état du carrefour au moyen des mesures des différents capteurs (sec. 6. 4. 1). Ils décident alors de demander au contrôleur le passage d'un feu au rouge ou au vert en fonction de leur perception (sec. 6. 4. 2). Le contrôleur reçoit toutes les demandes et prend finalement une décision en fonction du potentiel d'écoulement du trafic (sec. 6. 4. 3).

6. 4. 1 Étape 1 : métriques

Dans une première phase, les nœuds SN et AN collectent le nombre de véhicules qu'ils voient passer. Ils transmettent ensuite, de manière régulière, ces informations à chacun des nœuds DN qui leur est associé. Pour un nœud SN, le nœud DN associé est celui situé sur la même voie que lui. Pour un nœud AN, les nœuds DN associés sont les nœuds dont les voies ont la possibilité de mener à la voie du nœud AN.

Le coût des différentes tâches collaboratives entre les capteurs est évalué dans le tableau 6.1. Par exemple, le nombre de véhicules sur une voie peut être évalué au minimum en comparant les données issues d'un nœud SN et d'un nœud DN, mais peut être évalué plus précisément en ajoutant les données provenant de nœuds AN. En effet, une file peut être plus grande que la distance séparant les nœuds SN des nœuds DN.

L'ensemble des métriques utilisées dans ce chapitre – ainsi que les explications sur la manière de les obtenir – sont décrits dans les paragraphes suivants. En particulier, chaque nœud DN reçoit et stocke, au fil du temps, les valeurs des métriques qui sont propres à sa voie. Ces données sont agrégées puis partagées aux nœuds DN des voies adjacentes qui possèdent des mouvements communs. Cette coopération permet à chacun d'évaluer le nombre de véhicules présents sur les voies de ses mouvements, ainsi que l'espace disponible sur les intersections voisines atteignables. À l'aide de

ces valeurs, le changement des feux peut être géré sur chaque nœud DN au nom des mouvements qui sont possibles sur sa voie.

Métriques de voies	Coopération minimale	Coopération maximale
Nombre de véhicules	1 SN + 1 DN	1 AN + 1 SN + 1 DN
Espace libre	1 SN + 1 DN	1 AN + 1 SN + 1 DN
Taux moyen d'arrivée (EWMA)	1 AN	
Taux de véhicules moyen sur une voie empruntant le mouvement (a, b) (EWMA)	1 DN + nombre de voies en destination de b	
Métriques de mouvements	**Coopération minimale**	**Coopération maximale**
Nombre de véhicules	1 DN	Tous les DN de a
Espace libre	*Idem*	

TABLE 6.1 – Principales métriques utilisées.

6. 4. 1. 1 Taille d'une file d'attente

Sur une voie

Chaque nœud SN transmet de façon régulière (p. ex., toutes les minutes) deux informations à son nœud DN, situé sur une voie commune que nous notons x. D'une part, les véhicules qu'il a comptabilisé, et d'autre part le taux d'arrivée moyen au temps t, noté $\lambda^x(t)$, qui est ajusté sur la base d'un modèle EWMA. À la réception au temps t, un nœud DN peut facilement calculer le nombre de véhicules présents sur sa voie, noté $\eta^x(t)$. Il lui suffit en effet de soustraire le nombre de véhicules qu'il a lui-même enregistré (c.-à-d., départs) au nombre de véhicules transmis par le nœud SN.

Nous nous plaçons ici dans un scénario où les données sont transmises en quasi temps-réel. Toutefois, comme nous le constatons au chapitre précédent, un certain temps peut passer entre le moment où une comptabilisation est mesurée et le moment où elle est transmise, puis utilisée. De plus, les données provenant des capteurs peuvent être perdues, retardées ou tous simplement non disponibles si la fréquence d'acquisition est faible. Les nœuds DN utilisent alors un mécanisme d'interpolation afin de compenser ce manque. À l'aide des informations acquises, il est facile pour un nœud DN d'estimer l'évolution de la file d'attente sur sa voie. Si nous définissons τ^Δ comme étant le temps écoulé depuis la dernière mise à jour de la file d'attente par le nœud, nous obtenons au temps t, quel que soit x :

$$\eta^x(t) = \eta^x(t - \tau^\Delta) + \tau^\Delta \cdot \lambda^x(t - \tau^\Delta) - \eta^x_D \,, \tag{6.1}$$

avec η^x_D le nombre de départs comptabilisés depuis la dernière mise à jour de η^x.

Sur un mouvement

Une fois la file d'attente de la voie courante calculée, l'objectif pour le nœud DN est d'estimer le nombre de véhicules qui veulent emprunter les mouvements que sa voie propose. En effet, tout comme pour TAPIOCA, nous considérons que le changement d'état d'un feu doit être fait par rapport à un ensemble de mouvements, et non par rapport à une file en particulier. Fonctionner par mouvement permet en

particulier de contrôler la destination des véhicules ainsi que d'éviter les interblocages sur l'intersection.

Nous définissons la taille d'une file d'attente pour un mouvement (a, b) comme étant égale à la somme des files d'attente de chaque voie appartenant à la direction entrante a :

$$\eta^{(a,b)}(t) = \sum_{x \in a} \eta^{x}(t) \cdot \alpha_x^{(a,b)} \,, \tag{6.2}$$

où $\alpha_x^{(a,b)}$ représente le taux moyen de véhicules situés sur la voie x qui empruntent le mouvement (a, b). Par défaut, si une voie accueille un seul mouvement, nous notons alors $\alpha_x^{(a,b)} = 1$. Dans le cas où une voie permet aux véhicules d'aller vers $D > 1$ destinations (c.-à-d. plusieurs mouvements possibles), nous appliquons $\alpha_x^{(a,b)} = 1/D$. Si l'architecture le permet, ce taux peut éventuellement être ajusté. Ceci est rendu possible avec l'utilisation des nœuds AN, qui peuvent échanger les signatures électromagnétiques des véhicules avec les nœuds DN, afin de savoir quels chemins sont empruntés et à quelle fréquence.

Étant donné que nous souhaitons rendre chaque nœud DN responsable d'une potentielle décision, l'estimation de $\eta^{(a,b)}(t)$ doit se faire directement sur chaque nœud DN. Pour se faire, nous choisissons, à chaque fin de phase, de faire communiquer les nœuds DN entre eux, sur chaque direction, pour échanger les informations que chacun possède sur sa file.

Le temps de latence entre les différentes communications pouvant être important, chaque nœud DN peut maintenir, de façon indépendante, l'évolution des files représentatives de ses mouvements. Pour cela, il lui suffit de mettre à jour la taille des files d'attente pour chacune des voies connues. Pour sa voie, il lui suffit d'appliquer la méthode décrite plus haut en tenant compte du taux d'arrivée et des départs comptabilisés. Pour les autres voies, cette opération est également possible. D'une part, la mise à jour des nouvelles arrivées peut se faire en ayant connaissance du taux moyen d'arrivée pour chaque voie, communiquée par les nœuds DN voisins. D'autre part, les départs peuvent être pris en compte avec un taux moyen de départ ou en se référant directement à la voie sur laquelle le nœud DN se situe. Nous pouvons par exemple supposer qu'à chaque départ enregistré sur le nœud DN, un départ se produit également sur les autres voies, ou que le débit est similaire.

6. 4. 1. 2 Espace libre

La capacité totale de la voie x est invariante dans le temps, et est notée C^x. Elle est exprimée en nombre de véhicules et est égale à la longueur de la route à laquelle appartient x, divisé par la longueur moyenne d'un véhicule L_{veh}, écarts compris. L_{veh} peut être fixé (p. ex. 6 m estimés [Gor+05]) ou évalué à l'aide de capteurs au sol. Cette manière de procéder est simple, mais nous pouvons également imaginer que les capteurs soient capables de différencier chaque type de véhicule, afin de les compter en UVP (sec. 6. 1), ce qui nous permettrait d'obtenir des valeurs plus justes (approximation selon le type de véhicule).

A partir de cette information, nous pouvons déterminer l'espace libre sur une route comme étant la différence entre sa capacité totale et le nombre total de véhicules qu'elle contient. Cette information peut être transmise de manière régulière aux nœuds AN de cette route, qui comptabilisent les nouveaux véhicules allant l'occuper. Chaque nœud AN envoie, de façon périodique à chaque nœud DN de son intersection, la place disponible, avec d'éventuelles informations supplémentaires telles que le taux de passage. Ceci afin que les nœuds DN puissent interpoler les futures estimations de cette valeur.

Notons que, comme indiqué au chapitre 2, d'autres approches sont également envisageables : une caméra peut être utilisée au niveau d'un feu pour estimer la longueur, en mètres, de la file d'attente, et ainsi déterminer la place restante. Des capteurs peuvent également être positionnés sur la chaussée, séquentiellement ou sur rails, afin de définir des seuils de remplissage.

6. 4. 2 Étape 2 : agrégation et prise de décision distribuée

Un nœud DN peut donc gérer temporairement et de manière autonome l'évolution des métriques dont il a connaissance. Fort de ces données et estimations, chaque nœud DN est alors capable d'évaluer l'impact d'un mouvement (a, b) sur le carrefour.

L'algorithme 6.1 décrit la procédure qui permet à un nœud DN de savoir si le feu auquel il est rattaché est vert ou non, et d'envoyer en conséquence des demandes de changement de feux au contrôleur. Les nœuds DN demandent explicitement le passage de feu au vert lorsque des véhicules sont en attente sur sa voie ou ses mouvements, et que la route de destination n'est pas pleine (lignes 4 à 10). Ils demandent le feu au rouge dans le cas contraire (lignes 11 à 14). Leur demande est assortie de leur évaluation du nombre de véhicules qui pourront traverser le carrefour (non représenté dans l'algorithme 6.1).

Contrairement à la plupart des travaux de la littérature, l'adaptation au trafic se fait de façon dynamique, et ne passe pas par l'estimation d'un temps de feu vert sur un ensemble prédéterminé de phases. Il s'agit également un bon moyen pour détecter les embouteillages. En effet, si aucun véhicule n'est en mouvement sur un nœud DN, alors soit le feu est rouge, soit les véhicules sont bloqués.

Algorithme 6.1: algorithme de prise de décision sur un nœud DN

Paramètres : $mouvements$, liste des mouvements de l'intersection.
Variables : $\eta^{a,b}$, nombre de véhicules représentatifs du mouvement (a, b) ; ϵ^a, espace libre sur la route a ; $booleenVert$, booléen identifiant si le feu est vert (1) ou non (0).

```
1  booleenVert ← 0 // Initialisation de la variable globale
2  Procédure DecisionDN() :
       /* Aucun véhicule n'a été détecté depuis le dernier appel de la procédure.
          Trois possibilités : le feu est rouge, des véhicules sont arrêtés, ou
          aucun véhicule n'est présent.                                          */
3      si derniersVehicules() == 0 alors
           /* Cas 1 : le feu est rouge. Les métriques sont mises à jour et si
              possible, une demande pour passer le feu au vert est envoyée.     */
4          si booleenVert == 0 alors
5              pour chaque (a, b) ∈ mouvements faire
6                  interpolation(a,b);
7                  si ε^b > 0 et η^(a,b) > 0 alors
8                      demandeVert(a,b);
9                  finsi
10             finprch
11         /* Cas 2 : le feu est vert. Deux possibilités : (1) les véhicules sont
              stoppés par un embouteillage, ou (2) il n'y a plus de véhicules.
              Dans tous les cas, le passage du feu au rouge est demandé.         */
12         sinon si booleenVert == 1 alors
13             demandeRouge(a,b);
14         finsi
15         booleenVert ← 0;
16     /* Au moins un véhicule a été détecté depuis le dernier appel de la
          procédure. Le feu est donc encore considéré comme vert. Aucune action
          n'est à effectuer.                                                     */
17     sinon
18         booleenVert ← 1;
19     finsi
20 finproc
```

---

interpolation(a,b) : met à jour les métriques du mouvement (a, b) conformément à la section 6. 4. 1.
derniersVehicules() : retourne le nombre de véhicules qui sont passés depuis la dernier appel de la procédure.
demandeVert(a,b) : le nœud DN envoie une demande de passage au feu vert au contrôleur.
demandeRouge(a,b) : le nœud DN envoie une demande de passage au feu rouge au contrôleur.

6. 4. 3 Étape 3 : mise en place des feux de circulation

Le contrôleur, quant-à-lui, récupère l'ensemble des demandes et prend, à intervalles réguliers, une décision sur les feux qui doivent être verts ou rouges. Si plusieurs demandes concernent le même mouvement, il ne prend en compte que la dernière. Il sélectionne ensuite le mouvement qui représente le plus grand potentiel d'écoulement de véhicules. Ce mouvement doit être compatible avec les mouvements actifs et ne doit pas avoir déjà été sélectionné pour l'itération du cycle en cours de déroulement. Un mouvement avec une grande capacité d'écoulement entrant en conflit avec un mouvement actif devra patienter, ce qui permet de garantir une certaine stabilité. Le feu étant en place, cette procédure est répétée pour chaque nouvelle demande, dans le cas où d'autres mouvements pourraient se greffer à ceux en place. Un feu deviendra rouge lorsque plus aucun véhicule ne sera présent sur sa voie incidente, lorsqu'un message d'arrêt sera reçu, ou alors lorsqu'il sera resté vert durant un temps prédéfini, pris par défaut au temps de cycle maximal en France : 120 secondes. Dans le cas d'un passage au rouge, cette procédure est répétée pour compléter les mouvements restants.

L'algorithme ainsi défini est dynamique et réactif. Il raisonne mouvement par mouvement plutôt que phase par phase et diffère de la littérature qui se base plutôt sur la sélection de la phase prédéterminée la plus adaptée. Cette méthode présente l'avantage de permettre des combinaisons de mouvements plus souples, ne sélectionnant pas les mouvements dont les voies de destination sont saturées. En revanche, sa performance est très dépendante de celle du réseau de communication sous-jacent. En effet, des pertes de paquets ou des retards auront un impact sur la pertinence des données utilisées et donc sur son efficacité.

6. 4. 4 Résumé des messages

Le tableau 6.2 décrit les messages qui sont transmis sur le réseau. Comme nous le constatons, si les nœuds AN et SN sont sollicités régulièrement pour diffuser, entre autres, les nouvelles arrivées, les nœuds DN envoient des informations essentiellement à chaque fin de phase.

Message	Nœud source et destination	Fréquence	Taille
Ordre de changement de feu	DN → Interface	Variable	~15 octets
(1) Ordre d'arrêt de feu et données locales	DN → Interface	Variable	~40 octets
(2) Mise à jour des données sur chaque DN	Interface → tous les DN	A chaque fin de phase, en réponse à (1)	~35 octets
(3) Remontée pour le calcul de l'espace libre	Interface ou DN → AN (intersection voisine)	A la suite de (1) ou de (2)	~15 octets
Espace libre	AN → DN	A la suite de (3)	~15 octets
Arrivées lointaines	AN → SN	Chaque minute	~15 octets
Arrivées proches	SN → DN	Chaque minute	~15 octets
Calcul du coefficient α	tous les AN d'une destination → DN	Chaque minute	Variable

TABLE 6.2 – Messages transmis sur le réseau, fréquence et taille des paquets.

6. 5 Simulations

Nous évaluons notre algorithme par une co-simulation entre le simulateur de trafic routier SUMO 0.19 et le simulateur réseau OMNeT++ 4.2, liés à l'aide du framework Veins 2.1. Les nœuds sont déployés conformément à l'architecture entièrement adaptative décrite en 3. 2 et utilisent le protocole IEEE 802.15.4 *(non-beacon, CSMA)*. La fréquence de transmission des messages est configuré conformément au tableau 6.2.

Nos simulations sont réalisées sur un réseau de trois intersections généré aléatoirement par SUMO et subissant un taux d'arrivée de λ véhicules par seconde pendant *3 600* secondes. Les trajets des véhicules ont été générés aléatoirement par un utilitaire fourni par SUMO. La figure 6.6(a) représente l'évolution du temps d'attente des véhicules en fonction de la fréquence d'échantillonnage des capteurs dans le cas d'une charge faible ($\lambda = 1$ véhicule/s). La figure 6.6(b) représente la même évolution dans un cas de forte charge ($\lambda = 2$ véhicule/s, conduisant à un embouteillage). Notre algorithme avec et sans interpolation est comparé avec un cas idéal dans lequel le réseau de communication est parfait (sans délai ni perte) et au plan de feu prédéterminé par SUMO en guise de référence. La figure 6.6(c) représente le débit d'émission moyen de chaque capteur en fonction de la fréquence d'échantillonnage et les figures 6.7(a) et 6.7(b) représentent les temps de trajet médians des véhicules pour différentes fréquences d'échantillonnage, avec et sans interpolation. Tout comme pour les autres chapitres, des résultats plus complets sont disponibles en ligne [2].

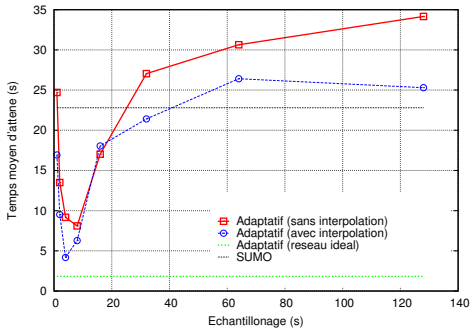
Ces résultats montrent qu'il existe une fréquence d'acquisition optimale. Une fréquence plus rapide provoque des collisions et des pertes de paquets que le mécanisme d'interpolation peut compenser en partie. Une fréquence plus lente conduit à une estimation imprécise de l'état du réseau. Il est aussi intéressant de remarquer que le mécanisme d'interpolation est efficace même lorsque l'algorithme fonctionne à sa fréquence optimale. La comparaison entre les deux niveaux de charge montre que l'intérêt du mécanisme d'interpolation augmente dans les cas embouteillés. Les taux d'arrivées et de départs ayant tendance à être plus stables dans ce derniers cas, ce mécanisme conduit à des valeurs plus proches de la réalité, en comparaison à un scénario plus léger où des files en réalité vides peuvent être sélectionnées.

6. 6 Perspectives

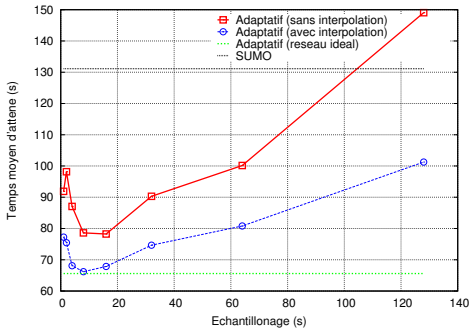
Ce travail étant l'un des derniers travaux effectués au cours de cette thèse, de nombreux autres aspects sont abordables. Toutefois, les principes proposés – nés de l'intuition de vouloir pousser la collaboration entre les capteurs à son maximum – ont le potentiel de pouvoir réagir très finement et rapidement à des situations d'embouteillages.

L'aspect principal qu'il nous faudrait explorer est la collaboration entre intersections. Dans la version présentée de notre algorithme, les intersections collaborent uniquement en vue de calculer la place disponible sur une route. Toutefois, nous

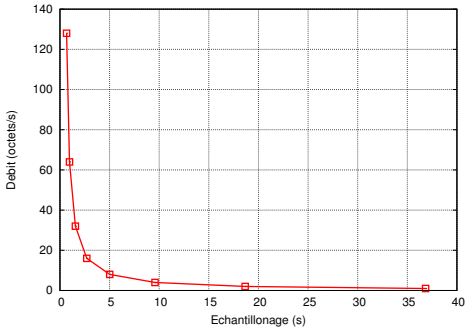
2. http://basalte.sfaye.com/



(a) Temps d'attente vs. fréquence d'échantillonnage ($\lambda = 1$).

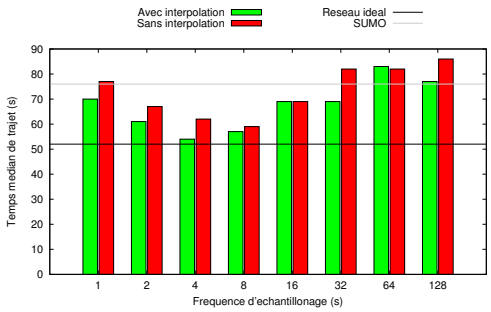


(b) Temps d'attente vs. fréquence d'échantillonnage ($\lambda = 2$).

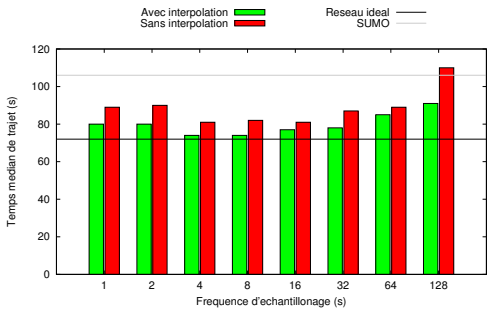


(c) Fréquence d'échantillonnage vs. débit moyen par nœud.

FIGURE 6.6 – Résultats en fonction de la fréquence d'échantillonnage.



(a) Temps médian de trajet ($\lambda = 1$).



(b) Temps médian de trajet ($\lambda = 2$).

FIGURE 6.7 – Performances du système de transport intelligent.

pourrions bénéficier de cette collaboration pour synchroniser les feux entre eux, afin de favoriser l'absorption d'un embouteillage. Nous pouvons tout à fait imaginer un mécanisme asynchrone, similaire à celui qui est proposé sur TAPIOCA et basé sur un score, comme représenté sur la figure 6.8. Un tel mécanisme pourrait, par exemple, hiérarchiser les intersections en fonction de leur charge, de manière à créer des chemins de feux verts menant à la bordure de la zone embouteillée.

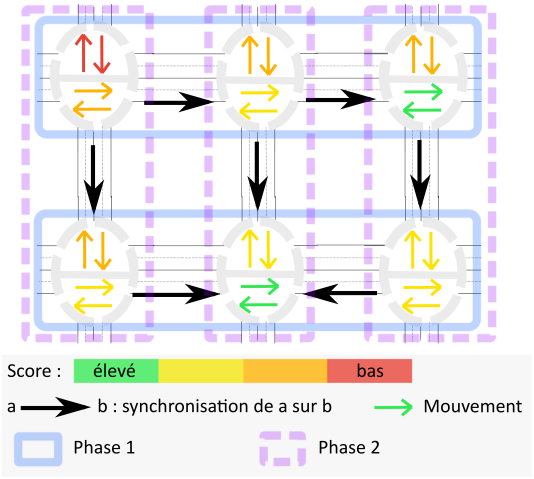


FIGURE 6.8 – Exemple d'une synchronisation des feux verts.

De plus, outre la collaboration avec une intersection voisine, nous pourrions bénéficier de la communication avec des intersections situées à plus grande distance. Dans le cas d'une situation de trafic normale, nous avons le *sentiment* que ceci ne changera rien. Dans une situation où le trafic est embouteillé, nous pouvons imaginer communiquer avec les intersections situées en bordure de la zone congestionnée, de manière à comprendre l'étendue de l'embouteillage et cerner de meilleures stratégies. Nous pouvons également imaginer intégrer des probabilités sur le nombre de véhicules pouvant arriver à plusieurs minutes d'intervalles, afin de préparer leur arrivée.

Ensuite, un point de passage obligatoire serait de présenter des résultats issus de simulations plus étendues que le réseau de trois intersections présenté ici. Ce premier scénario nous montre le potentiel de BASALTE, qui a été conçu pour être généralisable à tout type d'intersection. Toutefois, nous gagnerions à profiter de cette généralisation afin d'étudier des cas de scénarios plus étendus, avec ou sans mécanisme de synchronisation entre les intersections.

Enfin, il nous parait intéressant de coupler l'algorithme présenté à TAPIOCA, afin d'alterner les deux stratégies en fonction du niveau d'encombrement. Nous pourrions notamment bénéficier de meilleures performances. Une idée similaire a déjà été développée par [CS04a]. Ils utilisent une méthode de gestion pour les intersections sous-saturées, par l'intermédiaire de TRANSYT-7F (sec. 2. 3. 4. 1), ainsi qu'une méthode de gestion pour les intersections sursaturées. Leur couplage est testé sur un réseau de 12 intersections sursaturées, entourées de 13 intersections sous-saturées, et fourni de meilleurs résultats que TRANSYT-7F seul.

6. 7 Conclusion

Nous avons présenté dans ce chapitre un algorithme de gestion des feux de circulation traitant le cas des intersections saturées.

Utiliser un tel réseau permet d'augmenter le nombre de points de mesure du trafic et de mettre en place une architecture distribuée plus réactive et plus adaptative que les systèmes traditionnels. En effet, en raisonnant mouvement par mouvement, nous contrôlons finement où les usagers peuvent aller, plutôt que de raisonner par phase et risquer de conduire certains véhicules à une voie saturée.

De plus, afin de gérer les situations d'embouteillage, nous utilisons des métriques d'espace plutôt que de temps. Ceci nous permet de nous focaliser sur le nombre de véhicules qui peuvent se déplacer d'une intersection à une autre, et non sur une stratégie reposant sur des métriques temporelles (p. ex. temps d'attente depuis le dernier feu vert).

Cependant, malgré la faible quantité de trafic générée par les mesures dans un tel environnement, l'impact des pertes de paquets est sensible au niveau de l'application et il existe une fréquence d'acquisition optimale. Les défauts de données peuvent être compensés efficacement par un mécanisme d'interpolation simple. Le déploiement d'un tel réseau au niveau d'une infrastructure routière complète créera, en outre, une diversité de chemins dont il sera possible de tirer profit pour compenser les pertes et pour tirer parti d'informations provenant d'intersection éloignées.