Auto-structuration de trafic temps-réel multi-objectif et multi-critère dans un monde virtuel

Augustin Degas

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Augustin DEGAS

Le 3 juin 2020

Auto-structuration de trafic temps-réel multi-objectif et multi-critère dans un monde virtuel

---

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Marie-Pierre GLEIZES et Elsy KADDOUM

Jury

Mme Giovanna DI MARZO SERUGENDO, Rapporteure
M. Guillaume HUTZLER, Examinateur
M. Arcady RANTRUA, Examinateur
Mme Marie-Pierre GLEIZES, Directrice de thèse
Mme Elsy KADDOUM, Co-directrice de thèse
M. Rene MANDIAU, Président

Augustin Degas

AUTO-STRUCTURATION DE TRAFIC TEMPS-RÉEL MULTI-OBJECTIF ET MULTI-CRITÈRE DANS UN MONDE VIRTUEL

Directrice : Marie-Pierre Gleizes, Professeur, UPS

Co-Directrice : Elsy Kaddoum, Maître de Conférences, UT2J

Résumé

Dans de nombreux domaines, la simulation est un outil puissant pour apprendre, visualiser, et comprendre l'impact d'une décision à un temps donné sur l'ensemble du système. Le domaine de la navigation aérienne ne fait pas exception. Les outils de simulation de trafic aérien sont essentiels dans la gestion du trafic aérien, et doivent être capables de générer une large variété de scénarios réalistes tout en prenant en compte différentes contraintes observables par l'utilisateur de simulations, appelées situations, telle qu'une densité de trafic, une typologie de flux, des collisions, un évènement météorologique, ou tout autre évènement émergeant.

Structurer une simulation de trafic pour obtenir le réalisme et différentes situations est une tâche complexe, de par les nombreux objectifs et les nombreux critères à respecter, la diversité des entités mobiles et leurs multiples interactions, ainsi que la dynamique de l'environnement. Dans le domaine de la navigation aérienne, cette complexité est très souvent gérée par des humains, que ce soit l'expert scénariste qui génère le scénario de trafic au prix de nombreuses heures d'essais-erreurs, ou par les acteurs humains lors de la simulation qui gèrent l'adaptation temps-réel du trafic si celle-ci est requise. Les approches classiques de résolution ont montré leurs limites pour faire face à la complexité de ces applications. L'objectif de cette thèse est de résoudre la structuration temps-réel d'une simulation de trafic multi-objectif et multi-critère par l'utilisation de la théorie des *AMAS (Adaptive Multi-Agent Systems)*. Dans ces systèmes, les agents poursuivent des buts locaux et interagissent d'une manière coopérative. Au travers de leurs interactions locales, le système est rendu plus robuste et s'auto-adapte face à la dynamique de l'environnement, permettant une émergence de la fonction globale. Suite à plusieurs études, cette théorie a montré son adéquation pour la résolution de problèmes complexes et dynamiques.

L'objectif de ce travail est de modéliser et de spécialiser cette théorie pour la structuration de simulation de trafic temps-réel, multi-objectif et multi-critère. Pour cela, le modèle d'agents *AGATS* avec des comportements et des interactions coopératifs et locaux a été défini. Ce modèle est composé de deux sous-modèles, *AGEAS*, pour la structuration de la simulation en fonction d'un scénario, et *CAAMAS*, pour l'adaptation des entités mobiles aux scénarios et à la dynamique de la simulation. Les résultats de l'instantiation de ces deux modèles pour les simulations de trafic aérien montrent l'adéquation de l'approche proposée pour la génération autonome de scénarios, et l'adaptation lors de la simulation.

Institut de Recherche en Informatique de Toulouse - UMR 5505
Université Paul Sabatier, 118 route de Narbonne, 31062 TOULOUSE cedex 4

Remerciements

Au contraire de ce qui semble être la tradition, j'écris ces derniers mots après ma soutenance (l'un des quelques privilèges d'avoir soutenu pendant le coronavirus) [1], et ceux-ci sont donc sûrement libérés [2] des quelques doutes qui pouvaient subsister. Ceux qui me connaissent savent à quel point il est difficile pour moi d'écrire ces mots, sans phrase infinissables, pages annexes, 36 notes de bas pages pour faire des blagues internes [3] [4], et peut-être, il faut l'avouer, quelques larmes.

Je voudrais tout d'abord remercier les membres du jury. Merci à mes rapporteurs, Réné Mandiau et Giovanna Di Marzo Serugendo, pour leur travail de relecture, malgré les conditions. Merci à mes examinateurs, Guillaume Hutzler et Arcady Rantrua pour avoir accepté d'évaluer mon travail de thèse. Merci à tous de vous être intéressés à mon travail, d'avoir pris le temps de lire mon (volumineux) manuscrit, et d'avoir participé à cette soutenance. J'espère pouvoir travailler avec vous dans un futur proche.

Ce qui m'amène aux autres membres du jurys, mes encadrantes [5], qui eux aussi ont lu les différentes versions du manuscrit, plusieurs fois, notamment les versions plus obscures. Vous avez grandement contribué à ce travail, que ce soit dans les réflexions ou bien cette synthèse, et c'est finalement aussi votre travail. Merci à Marie-Pierre et son mari au nom (et prénom) presque pareil, mais différent, pour leur enthousiasme contagieux pour les AMAS, et les différentes réflexions qu'ils ont pu faire au cours de ce travail. Merci à Elsy pour ton énergie, tes conseils et de tout le temps que tu m'as accordé, je ne te connaissais pas en commençant ma thèse, et je suis extrêmement content de t'avoir eu comme encadrante ! Merci à Françoise, pour ta bonne humeur constante. Merci aussi à Eric, Emmanuelle et Aurélie, pour votre encadrement, nos différentes discutions et mon intégration au sein de Sopra Steria, je n'étais pas un grand parleur en arrivant et vous avez sûrement contribué à ça aussi. J'ai aimé réaliser cette thèse, et c'est grâce à vous tous.

Vient ensuite les remerciements plus hasardeux, confus, puisque de nombreuses sphères d'influences inter-connectées, remplie de rétroactions, non linéarités et de différentes ouvertures [6], ont eu un impact sur ce travail, de près, de loin, dans les derniers moments, ou dans les débuts [7]. Merci donc à l'équipe SMAC [8]. J'ai fait mon arrivée dans cette équipe pendant mon stage, et je pense que si je n'avais pas fait mon stage ici, je n'aurais pas fait de thèse. Merci donc à mes encadrants de l'époque Sylvie et Jean-Paul (et les autres officieux), qui m'ont donné goût au travail de recherche et m'ont montré que la thèse ce n'était pas juste des études en plus. Merci aux permanents de cette équipe formidable, qui rendent cet environnement si agréable.

1. Je voulais aussi garder ces pages pour le pot de thèse, parce que les traditions c'est important !
2. Délivrés !
3. Mais 35 ça passe, il parait. Sachez que vous n'avez pas besoin de les lire hein. Donc pas de "gnagnagna, c'est difficile à lire avec toutes ces notes en bas de pages".
4. Bon celle-là c'est cadeau (ça c'est cadeau c'est cadeau) parce que c'est quand même les remerciements de ma thèse et je fais ce que je veux. Enfin je sais pas, enfin peut-être.
5. Cette faute-là, est intentionnelle, vous étiez majoritaires :D
6. Vous êtes un système complexe
7. Et désolé à ceux que j'aurais oublié, mais vous êtes nombreux !
8. Voilà, c'est fait.

Merci aux doctorants, et PostDoc de l'époque [9], qui rendaient [10] cette équipe formidable eux aussi. Sans ordre de préférence (Ou alors, adaptatif !) donc... Merci à Jérémy (pour tes remarques cinglantes et ta capacité à venir discuter au café quand tu entendais les mots clefs du moment), à Jupy (pour Siegfried et tes réfutations déguisées, "T'as sans doute raison, mais..."), à Bob (et ses inepties en économie [11]), notamment pour vos discussions si animées le midi, puisque vous permettiez à certains d'entre nous de manger tranquillement en observant. Merci à Alex (pour m'avoir accompagné dans ces observations, et d'avoir lancé le Dubli du Jeudi), à John, à Sameh, à Teddy ("Normalement, si tout va bien, dans le pire du pire des cas"), à David (t'as vu, je t'ai pas oublié), à Dorian, à Estelle, à Marjorie, à Nicolas Brax (et sa vision véritable des méchants à Avalon), à Sébas (et ses précieux conseils, parce qu'il est vieux), à Flo (pour son calme dans le 353 et "herbes et potion, sagacité, parle avec les morts !"), à El Senior Hadi (et son amour pour le japon et Assassin's Creed), à Arca (pour son Jdr aussi !) et à Valou (pour son énergie à préparer les nombreuses activités d'équipe). Merci aussi aux "vieux du Mulli du Dredi", ou autres anciens non cités précédemment, Tom, Victor, Fanf, Raja (merci pour nos nombreuses discussions au bar), Boulb, Luc, Maroo.

Merci aux doctorants, PostDoc et stagiaires de la nouvelle époque, parce que vous êtes pas si mal que ça en fait [12] : Tanguy (PAS pour ses percussions du midi), Guilhermo (d'avoir repris le flambeau des cadeaux), Davide (et ses blagues nulles), à ceux de l'ordre 363 (Bruno et ses conneries, Walid et sa gentillesse, et à mon Petit Coeur bien sûr), Maroun (et son égocentrisme), Tim (le prodige de Magic), Max (et sa logorrhée), Fabrice, Damien, JB (le véritable Merlin), Vincent, Doryan, Grégory, et Patri (et ses aprem jeux). Je souhaite à ceux que ça concerne de réussir leur thèse ! Merci à Yvan, Nico, Delphine, Maxime, David, Didier, François, et à l'équipe SVS (Sebastien, Hugo, Fred, Romain, Pierre, Aida, Oliver...).

Merci à ceux de la chorale, ou assimilés, Mel & Didou pour leur connerie, Bensou pour sa connerie d'un autre niveau, Leslie ("J'vais t'arracher les yeux tête de cul"), Florent, Marine, Laure, Robert, Marie.

Merci à mes anciens amis de l'ENAC, Jérém, Tonio, Nico, Ruixin, les années d'ingénieur ont été aussi bien grâce à vous, que ce soit pour LOL, Borderland, les sorties resto, casino, les barbecues, les aprem jeux, les breack, les pots, et soirées improvisées... A très bientôt !

Merci à mes amis d'enfance [13], Flo, Dim, Manu, Thibaut, Elo, Vincent, leurs différents frères et sœurs et leurs parents, et finalement leurs amis à eux aussi... qui forment finalement une bonne grosse bande de gens qui ont fait des grandes études avec des petits cartables, pas venus là pour équeuter les haricots, avec qui j'ai de super souvenirs. Désolé de pas passer dernièrement, promis, je vais passer plus souvent !

Merci à ma famille, mes parents, mes soeurs, mon beau-frère, mes grand-parents, cousins/cousines, neveux. Finalement, si je suis qui je suis aujourd'hui c'est aussi grâce à vous (ou à cause !). Merci de m'avoir toujours soutenu dans les différentes épreuves de mes quelques 26 années (ou 16 selon certains) et d'avoir toujours répondu présent. Je suis très heureux de vous avoir dans ma vie.

9. Voilà, c'est fait.
10. Parce que c'est plus ce que c'était...
11. Il était resté sur l'économie avant Bretton Woods I, c'est dire
12. Vous rendez aussi cette équipe formidable.
13. Voilà, c'est fait. Le comique de répétition c'est important.

Enfin, merci à toi, Kristell. Il est difficile de trouver les mots pour qualifier et quantifier tout ton soutien, mais je vais m'y risquer, quand même... Merci de ton sourire, de ta patience, de ta bonne humeur. Merci de ton rire, ta confiance, mon cœur. Merci d'avoir su me remonter le moral quand j'avais mes moments de mou, d'avoir su supporter mes mal-êtres, dans la fin de thèse, surtout. Tu as été mon phare dans cette mer agitée, et je n'aurais sûrement pas trouvé bon port sans ta lumière. Merci d'être toujours là à mes côtés, simplement. Je t'aime de tout mon cœur.

Table des matières

Introduction

Dans de nombreux domaines, la simulation est un outil puissant pour apprendre, visualiser, et comprendre l'impact d'une décision à un temps donné sur l'ensemble du système. Le domaine de la navigation aérienne ne fait pas exception. Les outils de simulation de trafic aérien sont essentiels dans la gestion du trafic aérien, et doivent être capables de générer une large variété de scénarios réalistes tout en prenant en compte différentes contraintes observables par l'utilisateur de simulations, appelées situations, telle qu'une densité de trafic, une typologie de flux, des collisions, un évènement météorologique, ou tout autre évènement émergeant.

Néanmoins, la mise au point d'un scénario réaliste de trafic virtuel respectant des *situations* impose des délais longs (typiquement plusieurs semaines) et la mobilisation d'une équipe conséquente pour contrôler son déroulement, adapter et préserver la structuration du trafic qui en découle lors de la simulation. De plus, les simulations actuelles sont peu ou pas interactives, et, si elles existent, les interactions sont souvent effectuées par des humains.

Contribution de la thèse

Dans cette thèse, nous explorons le problème de structuration d'une simulation de trafic, c'est-à-dire l'organisation des entités mobiles dans une simulation de trafic afin que le trafic soit réaliste, du point de vue des trajectoires et du point de vue du trafic, et qu'il fasse émerger un ensemble d'évènements observables (par exemple des densités ou des collisions) requis par un scénariste, que nous nommons *situations*. Ce problème, de par les multiples interactions entre les différentes entités mobiles, la contradiction entre leurs différents objectifs locaux, l'ouverture de la simulation (que ce soit par la modification de l'environnement de simulation, ou les actions d'humains dans la simulation), les différentes contraintes requises pour le réalisme par le scénariste, est un problème complexe. En appliquant l'approche des systèmes multi-agents adaptatifs (*Adaptative Multi-Agents Systems (AMAS)*), nous avons développé le système *Autonomous GenerAtion of Traffic Simulations (AGATS)*, un AMAS dédié à la structuration de simulations de trafic, capable de structurer le trafic pendant la génération du scénario (phase de création), mais aussi la simulation (phase de simulation), afin de structurer pleinement un trafic réaliste, correspondant aux attentes, et pouvant être rejoué.

Nous évaluons notre approche selon ces deux phases. Les différents agents d'*AGATS* se divisent en deux ensembles, que nous testons séparément. Le premier ensemble, *Collision*

Avoidance Adaptive Multi-Agent System (CAAMAS) permet aux entités mobiles de s'adapter au scénario et aux différentes interactions lors de la phase de simulation. Le deuxième ensemble, *Autonomous GEneration of trAffic Scenarios (AGEAS)*, permet la génération et l'adaptation du scénario lors de la phase de création et de simulation.

Organisation du document

La suite de ce document est structuré en 4 parties qui regroupent respectivement le contexte applicatif, l'état de l'art, le modèle développé, et les expérimentations. La première partie comporte deux chapitres et décrit le contexte applicatif de cette thèse, le trafic aérien, sa structure, et les défis qui émergent de sa structure actuelle :

— Le chapitre 1 introduit le trafic aérien et sa gestion. Il décrit brièvement l'historique du contrôle aérien, l'organisation de l'espace aérien en secteurs et en d'autres espaces, le réseau de routes utilisé par les avions commerciaux pour voler d'un aéroport à un autre, la gestion de l'espace aérien, et en particulier le rôle des contrôleurs aériens.

— Le chapitre 2 décrit les limites actuelles de la gestion du trafic aérien, en particulier l'augmentation du trafic et la nécessité de modifier cette gestion, et donc la nécessité d'étudier de nouvelles structures, de nouveaux outils, de nouvelles techniques, et par conséquent, la nécessité de tester celles-ci dans un environnement de simulation réaliste, pouvant répondre à une diversité de contraintes.

La deuxième partie de ce manuscrit est dédiée à l'état de l'art sur les techniques utilisées pour la structuration de trafic. Elle comprend deux chapitres :

— Le chapitre 3 décrit et analyse les techniques de structuration de trafic aérien, souligne leurs limites, avant d'analyser les techniques utilisées pour le trafic routier et finir par un premier positionnement de cette thèse.

— Le chapitre 4 décrit et analyse les techniques d'évitement de collisions, leurs généralités, puis les techniques utilisées dans le trafic aérien.

— Le chapitre 5 résume les différentes analyses des états de l'art sur la structuration de trafic et l'évitement de collisions et décrit le positionnement de cette thèse.

Les solutions les plus prometteuses dans ces deux états de l'art laissent à penser que la solution se trouve, en partie, dans la décentralisation. Ainsi, les systèmes multi-agents, naturellement décentralisés, et en particulier les systèmes multi-agents adaptatifs, représentent une alternative séduisante. La troisième partie présente ainsi les systèmes multi-agents adaptatifs, et décrit notre système multi-agent pour la structuration de trafic :

— Le chapitre 6 présente les modèles et systèmes utilisés dans cette thèse, à savoir les systèmes multi-agents, en particulier les systèmes multi-agents adaptatifs, ainsi que le pattern AMAS4Opt et le système Environnement Virtuel Auto-Adaptatif (EVAA).

— Le chapitre 7 décrit le système multi-agent *AGATS* développé dans cette thèse pour la structuration de trafic. Ce système est décrit en commençant par son fonctionnement général, et les deux phases durant lesquelles il s'exécute ; la phase de création de scénario et la phase de simulation, puis le comportement nominal de chaque type d'agent est

détaillé. Le chapitre se termine par la description de la détection et la résolution des différentes situations de non coopération (situations dans lesquelles les agents n'arrivent pas à atteindre leurs buts locaux).

La quatrième partie présente les expérimentations qui ont été effectuées sur le système multi-agent *AGATS* :

— Le chapitre 8 décrit les expérimentations effectuées sur les comportements locaux des entités mobiles, et se concentre sur la phase de simulation des agents d'*AGATS*, et présente le sous-système *CAAMAS*.

— Le chapitre 9 détaille les expérimentations effectuées sur la génération de scénarios, et présente plus particulièrement la phase de création de scénario des agents d'*AGATS*, et présente le sous-système *AGEAS*.

Enfin, nous concluons avec une synthèse du travail effectué dans le contexte de cette thèse, ainsi que différentes perspectives à plus ou moins long terme pour ce travail.

Première partie

Présentation du contexte

1 La Gestion du trafic aérien

Dans ce chapitre, la façon dont est géré le trafic aérien est présentée, en commençant par un très bref historique de la gestion du trafic aérien (section 1.1), suivi de la description de l'organisation de l'espace aérien (section 1.2), et enfin de la présentation de la gestion du trafic aérien telle qu'elle est effectuée actuellement (section 1.3). Ce chapitre se termine par un état des lieux du domaine et des défis à surmonter.

1.1 Bref historique et héritages

Nous ne ferons pas dans ce manuscrit une présentation exhaustive de l'histoire de la gestion du trafic aérien, mais une présentation permettant de comprendre pourquoi le trafic aérien d'aujourd'hui est organisé ainsi. Nous dirigeons le lecteur vers d'autres références s'il souhaite plus de détails sur l'histoire du trafic aérien : [DGAC, 2017], ou (le plus complet) [DGAC, 2013].

Une brève histoire de la gestion du trafic aérien. Les débuts de l'aviation civile organisée commencent après la première guerre mondiale et le surplus de pilotes, qui ont donné naissance à l'aéropostale. Une première organisation de l'espace autour de routes aériennes se matérialise par des phares aéronautiques que les pilotes suivent à vue, comme le décrit Antoine de Saint-Exupéry dans *Vol de nuit* [de Saint-Exupéry et al., 1931].

Après la guerre, ces phares aéronautiques, qui sont des balises visuelles, furent une première base pour la gestion du trafic aérien d'aujourd'hui. Afin de contrôler les avions pour augmenter le trafic et la sécurité, des Centres de Contrôle Régional (CCR) ou Area Control Center (ACC) en anglais, sont créés (en France, ils sont au nombre de trois en 1947 fig. 1.1a) et des techniques issues de la guerre sont dans un premier temps utilisées (fig. 1.1b).

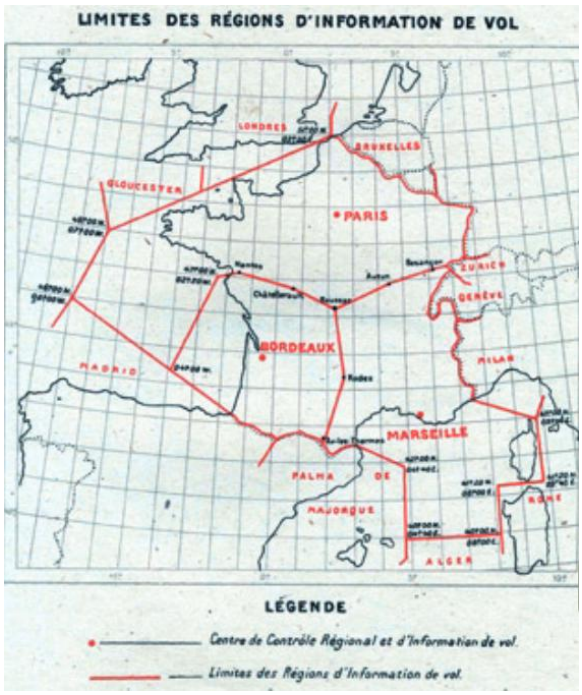
Au fil du temps, pour prendre en compte une demande croissante de trafic aérien, de nouveaux moyens ont été employés (listés sans ordre établi, ni exhaustivité) comme :

— augmentation du nombre de balises, de leur fonctionnement et de leur qualité (phare aéronautique, balises radios non directionnelles, balises radios directionnelles...),

— et par conséquent, augmentation du nombre de routes aéronautiques,

— apparition (ou plutôt utilisation pour le civil) des radars primaires, puis secondaires,

— augmentation du nombre de contrôleurs pour surveiller les avions,



(a) Les trois centres de contrôle régional (CCR) (Paris Bordeaux, Marseille), août 1947 [DGAC, 2017]

(b) Le système plotting, 1939 [DGAC, 2017]. La dernière position annoncée à la radio par les pilotes est matérialisée par un pion posé sur une carte. Le pion est équipé d'un grande flèche qui représente sa trajectoire estimée pendant les prochaines minutes

Figure 1.1 – Les débuts du contrôle aérien

— augmentation du nombre d'ACC et subdivision de l'espace de ces centres en espaces plus petits, et spécification de ces espaces,

— amélioration des techniques de contrôle (liaison radio, strips, image radar...),

Pour arriver à l'état actuel du trafic aérien, que nous décrivons dans la suite.

Unités impériales. Dans une majorité des domaines aériens, en particulier dans le domaine de la navigation aérienne, le système de mesure impérial est encore beaucoup utilisé. En particulier sont utilisés (liste non exhaustive) :

— Le pied (ft), unité de mesure de distance ($1ft = 0,3048m$).

— Le mille nautique (NM), simplifié en nautique, unité de mesure de distance ($1NM = 1852m$). Cette mesure correspond à la valeur moyenne d'une minute d'arc de méridien.

— Le noeud (knt), est une unité de mesure de la vitesse utilisée en navigation maritime et aérienne ($1knt = 1,852km.h^{-1}$).

Notations anglaises Les notations anglaises sont beaucoup employées dans le domaine aéronautique. Par exemple, la gestion du trafic aérien ne possède pas d'acronyme et seul

l'acronyme anglais ATM pour Air Traffic Management est employé. D'autres acronymes seront introduits dans la suite de la présentation.

1.2 L'organisation de l'espace aérien

Nous distinguons dans la suite deux types de vols :

— les vols obéissant aux Visual Flight Rules (VFR), que nous appellerons vols VFR. Les VFR définissent un régime de vol à vue, soumis par conséquent à de bonnes conditions météorologiques.

— les vols obéissants aux Instrument Flight Rules (IFR), que nous appellerons vols IFR. Les IFR définissent un régime de vol par instruments, qui n'a pas besoin, ou peu besoin, de repères visuels.

Dans ce qui suit, nous nous intéressons majoritairement aux vols IFR, qui constituent la grande majorité du trafic aérien commercial et cargo de nos jours.

1.2.1 Le découpage de l'espace aérien

Différentes catégories d'espaces. L'ensemble de l'espace aérien français, en règle générale, est découpé en portions d'espace aérien où les services fournis sont déterminés par une classe dont l'appellation uniformisée par l'International Civil Aviation Organization (ICAO) va de A à G. Il est possible de résumer ces classes par deux espaces :

— *l'espace contrôlé :* il regroupe les portions d'espace de classes A,B,C,D et E ; les règles ne sont pas uniformes et diffèrent notamment si les vols sont des VFR ou des IFR.

— *l'espace non contrôlé :* il regroupe les portions d'espace restant, à savoir les portions d'espace de classes F et G, dans lesquels les avions auront seulement des informations de vol (par exemple la météorologie), si elles existent.

L'ensemble de ces portions d'espace est représenté par la figure 1.2, sauf l'espace F qui n'est pas utilisé en France.

Ces portions d'espace elles-mêmes sont découpées et attribuées à des organismes ; en France ce sont :

— Les Centre en Route de la Navigation Aérienne (CRNA), qui gèrent les espaces dits *en route*, c'est-à-dire les espace aériens supérieurs (figure 1.3), en anglais Upper Traffic Area (UTA).

— Les Services de la Navigation Aérienne (SNA), qui gèrent les espaces d'approche et les aérodromes.

La figure 1.3 représente le découpage et l'attribution des portions d'espace entre CRNA et SNA. Les zones en gris clair sont des zones dont le contrôle est délégué aux pays frontaliers (ou non contrôlées).

Les portions d'espace qui sont gérées par les CRNA et SNA sont découpées elles-mêmes en secteurs qui sont gérés par des contrôleurs aériens (si le service de contrôle est assuré). Ce découpage en secteurs est représenté à la figure 1.4.

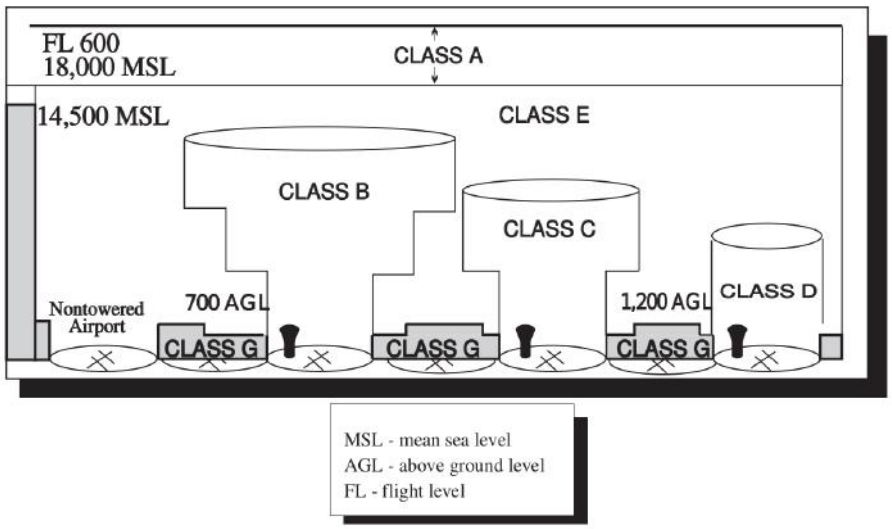


Figure 1.2 – Les différentes classes de portion d'espace aérien

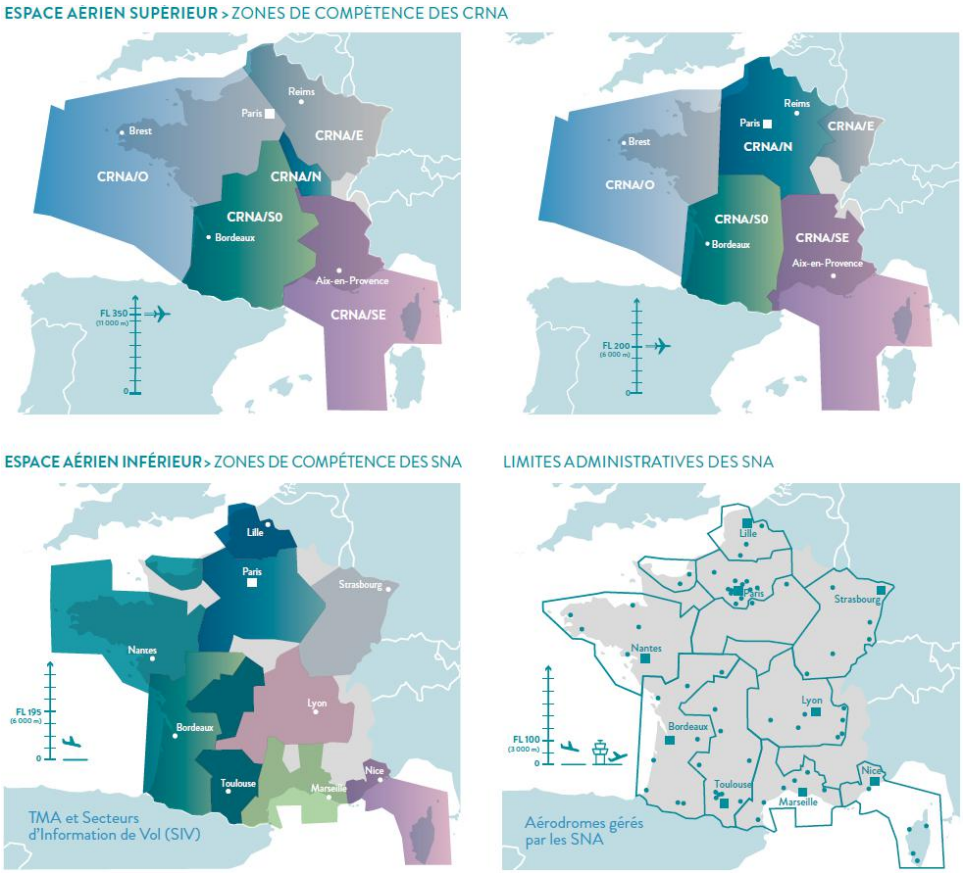


Figure 1.3 – Découpage de l'espace aérien français en fonction des différents CRNA et SNA [DGAC, 2018]

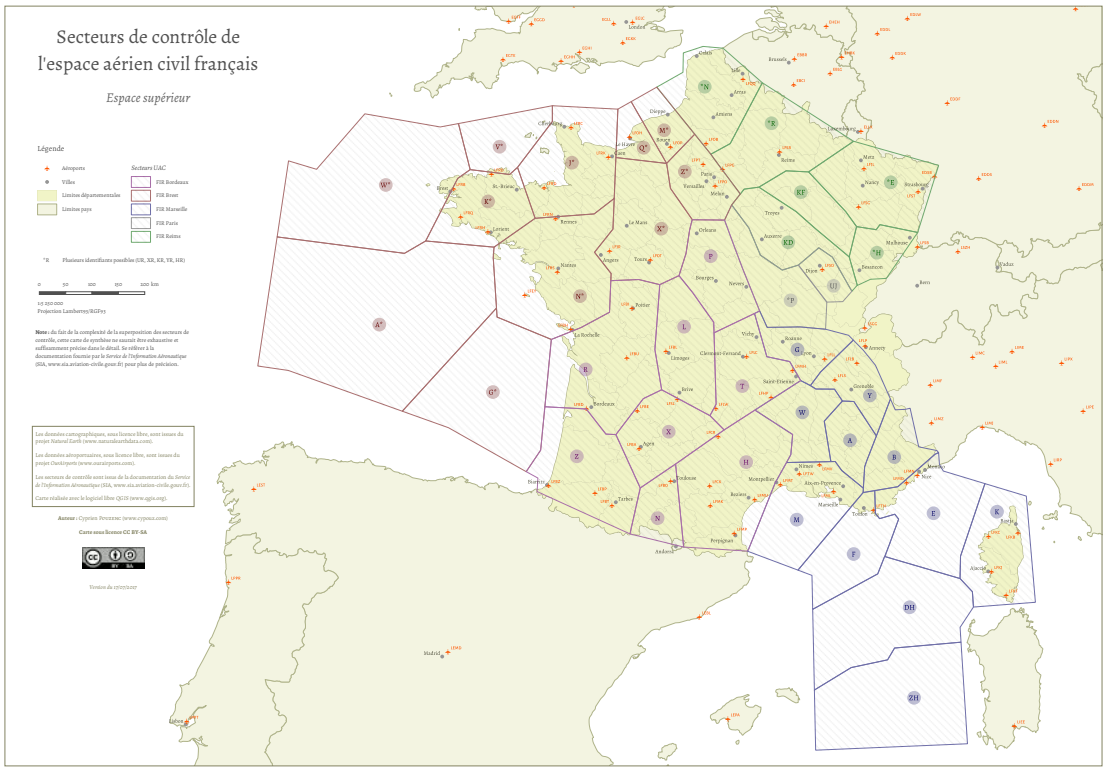


Figure 1.4 – Découpage des CRNA en secteurs dans l'UTA [DGAC, 2018]. Les CRNA de Reims, Paris, Marseille, Bordeaux et Brest sont respectivement en vert, gris, bleu, violet, et rouge.

L'espace aérien est aussi partagé entre les militaires et les civils. Certaines zones spécifiques (voir figure 1.5) de l'espace aérien peuvent être ouvertes (c'est-à-dire qu'elles peuvent être utilisées par les avions civils) ou être fermées (c'est-à-dire qu'elles ne peuvent plus être utilisées par les avions civils) en fonction de demandes militaires.

En fonction de la nature et de la densité du trafic aérien, les secteurs peuvent être groupés ou dégroupés. Par exemple, lorsque le trafic est faible comme en pleine nuit, les secteurs qui sont dégroupés peuvent être groupés pour surveiller le trafic. La figure 1.6 représente deux configurations sur un secteur de contrôle fictif en fonction des flux d'avions qui le parcourent, chaque teinte de gris représentant un espace géré par un même groupe de contrôleurs.

1.2.2 Réseau de routes

Ces différents espaces aériens sont parcourus par des routes aéronautiques qu'empruntent les avions pour voler de leurs aéroports de départ à leurs aéroports de destination. Ces routes sont définies par un ensemble de points de cheminement, en anglais *waypoints*, qui définissent géographiquement des points par lesquels doivent passer les avions. La figure 1.7a représente le réseau de routes en France dans l'espace supérieur. Ces *waypoints*,

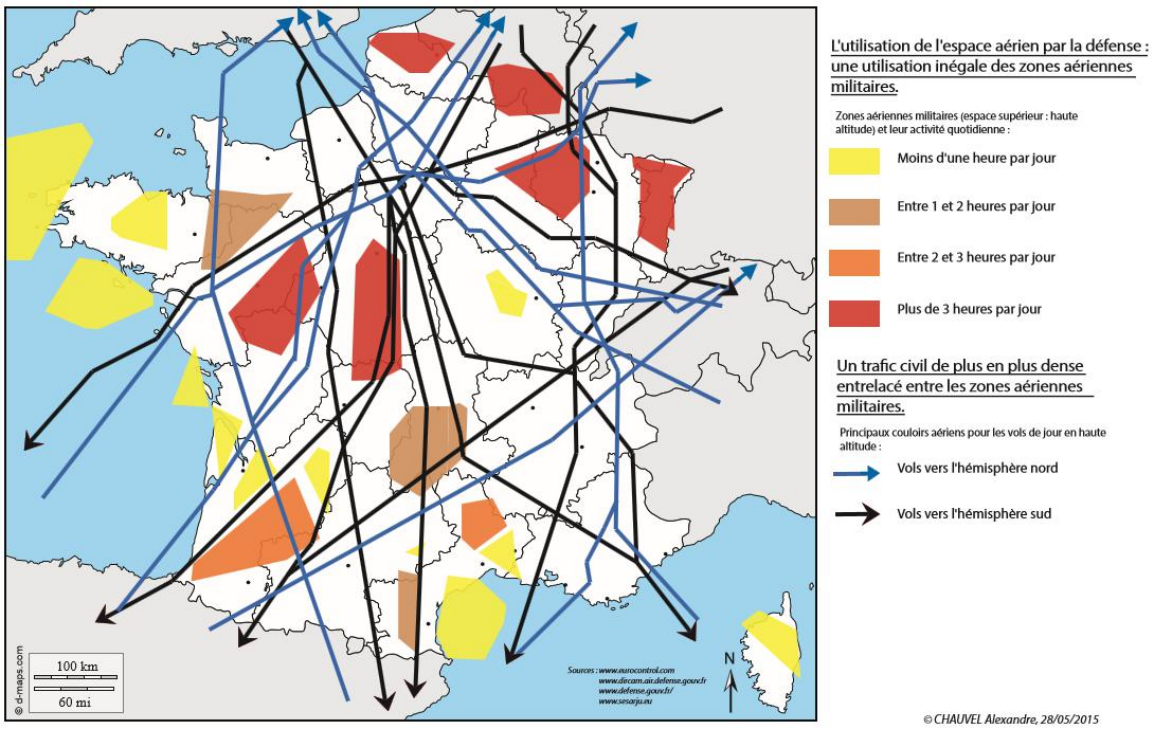
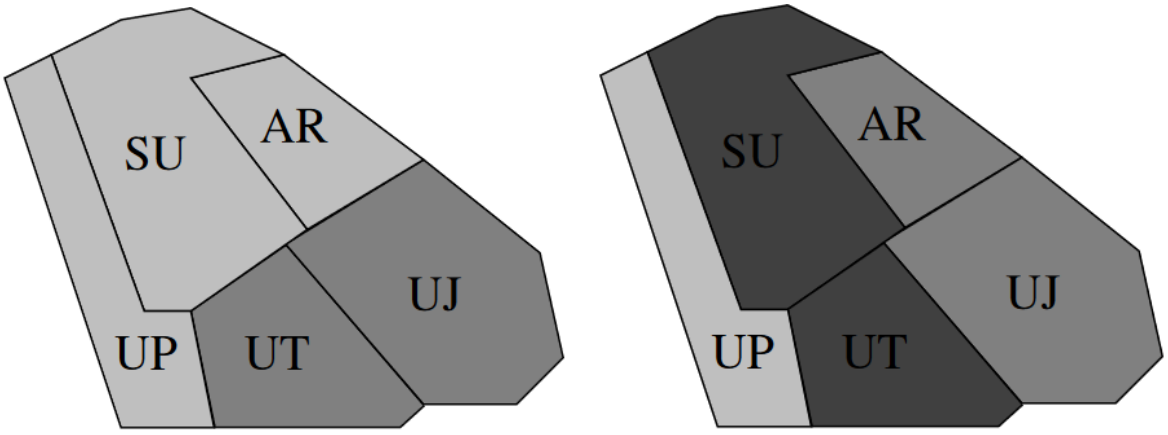


Figure 1.5 – Zones militaires dans l'espace aérien français qui peuvent être ouvertes ou fermées à l'utilisation pour les avions civils

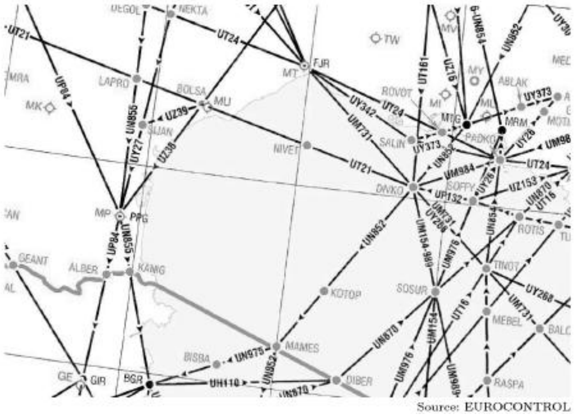


(a) Configuration adaptée pour des flux (Sud-Ouest)-(Nord-Est)

(b) Configuration adaptée pour des flux (Sud-Est)-(Nord-Ouest)

Figure 1.6 – Deux configurations possibles sur un centre de contrôle *en route* fictif [Allignol et al., 2012]. Chaque teinte de gris représente un regroupement d'espaces.

(a) Réseau de routes de France dans l'espace supérieur



(b) Une partie de la carte du réseau de routes dans le sud de la France. Les points noirs sont des vrais *waypoints* associés à une balise radio, et les points gris sont des *waypoints* fictifs

Figure 1.7 – Les réseaux de routes et les *waypoints*

dont certains étaient des phares aéronautiques à l'origine, sont désormais des balises radio-électroniques, ou des balises fictives (qui représentent des coordonnées géographiques), comme le montre la figure 1.7b.

Sur le réseau de routes, les avions sont aussi organisés en altitude, afin de séparer au maximum les flux. Quand les avions sont suffisamment loin des reliefs, c'est-à-dire quand ils ne sont pas en phase d'approche et sur les aérodromes, la mesure de l'altitude se fait à partir d'une référence commune à tous qui est la pression atmosphérique au niveau de la mer, appelée pression standard, équivalente à un isobare de $1013,25hPa$. L'utilisation d'une référence commune (même si cette référence est inexacte car la pression au niveau de la mer fluctue) permet de réduire les écarts de mesure d'altitude aux erreurs des instruments de mesure.

Il existe des routes aériennes toutes les dizaines de centaine de pieds, qu'on appelle des niveaux de vols, en anglais *Flight Level (FL)* . Ainsi le $FL300$ correspond à l'altitude $30000ft$. Dans le but de séparer les flux, les avions vont pouvoir utiliser ces routes aéronautiques en fonction de leur cap c'est-à-dire leur orientation géographique. Si leur cap est dans l'intervalle $[0°, 179°]$, ils occupent des niveaux de vol de dizaines impaires (par exemple le $FL210$), sinon ils occupent des niveaux de vols de dizaines paires (par exemple le $FL220$), comme le montre la figure 1.8.

Cette organisation de l'espace aérien a pour but principal de permettre, ou de faciliter, l'*Air Space Management (ASM)* , et la tâche du contrôleur aérien, que nous présentons dans la section 1.3.1.

Le **plan de vol** déposé pour un avion, *Flight Plan* en anglais, est un ensemble de renseignements décrivant le vol. Il contient en particulier le vol projeté, c'est-à-dire l'ensemble des *waypoints* par lesquels l'avion passe et l'horaire estimé de passage, les indicatifs des aéroports de départ et d'arrivée, l'indicatif de l'avion. Il est par exemple utilisé pour prévoir la demande (c'est-à-dire le nombre d'avions qui veulent utiliser des secteurs, des aéroports)

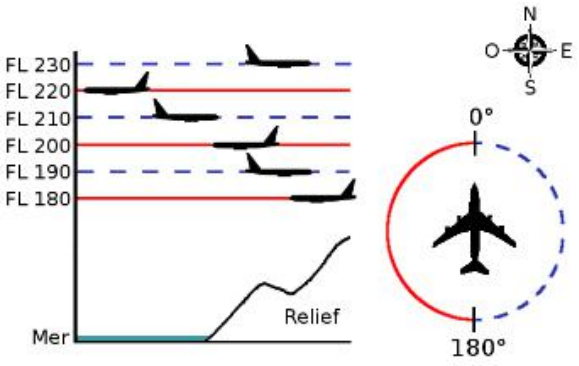


Figure 1.8 – Séparation des avions en niveaux de vol selon la règle semi-circulaire : les altitudes auxquelles les avions peuvent voler sont déterminées par leur cap (fig. par [Breil, 2017])

mais il est aussi utilisé en cas d'alerte (en cas d'accident par exemple).

1.3 La gestion du trafic aérien

La gestion du trafic aérien, en anglais *Air Traffic Management (ATM)* , se découpe en trois sous parties [OACI, 2012] :

— L'*Air Traffic Services (ATS)* , qui correspond à l'ensemble des services disponibles pour les avions une fois qu'ils sont en vol.

— L'*Air Space Management (ASM)* , qui correspond à la gestion de l'espace aérien, c'est-à-dire sa structure, que nous verrons plus en détail dans la suite.

— L'*Air Traffic Flow Management (ATFM)* , qui correspond à la gestion des flux d'avions, dont le but est d'optimiser les flux du trafic, en amont des phases de vol, c'est-à-dire avant que l'avion ne décolle, par exemple en modifiant les heures de départ des avions, mais aussi lors des phases de vols, notamment par des régulations.

Nous présentons dans cette section les différentes facettes de l'*ATM* : l'*ATS* , puis l'*ASM* et enfin l'*ATFM* .

1.3.1 Air Traffic Services

Les *ATS* correspondent à l'ensemble des services disponibles pour les avions une fois qu'ils sont en vol. Cette tâche incombe en grande partie aux **contrôleurs aériens**. Afin de maintenir la sécurité de l'espace aérien, cet espace est surveillé par des contrôleurs aériens. Ici, nous faisons la différence entre sécurité aérienne et sûreté aérienne. La sécurité traite de la gestion des risques involontaires ; A l'inverse de la sûreté, elle ne traite pas d'actes malveillants, comme le terrorisme. Nous traitons dans la suite uniquement de la sécurité. Le contrôleur aérien assume dans l'espace qui lui est attribué, une partie ou l'ensemble des services suivants (en fonction de la catégorie d'espace) :

— Le *service de contrôle*, rôle primaire du contrôleur qui est d'assurer la séparation entre les avions.
— Le *service d'information de vol*, rôle d'informateur du contrôleur qui est de transmettre aux avions les différents renseignements nécessaires à la conduite de vol, tels que la météo pour les vols VFR.
— Le *service d'alerte*, rôle d'aide et d'assistance aux avions en difficulté.

En général en Europe, chaque secteur "en route" de l'espace aérien est surveillé par un groupe de deux contrôleurs (un contrôleur dit "Radariste" et un contrôleur dit "Organique") qui assurent ces différents services dans le secteur et les liaisons avec les secteurs adjacents. Ainsi l'organisation de l'espace est faite de telle sorte qu'un contrôleur puisse effectuer sa tâche le plus aisément possible.

Évitements entre avions. L'évitement de collision entre avions, ou "abordage", est bien sûr essentiel dans le trafic aérien. Pour cela, de nombreux moyens sont mis en place. Ainsi, à chaque avion est associé une bulle de protection, dans laquelle aucun autre avion ne doit se trouver. Si un avion est dans la bulle de protection d'un autre, on dit que les deux avions sont en perte de séparation, et si deux avions sont en perte de séparation dans un futur proche, on dit qu'ils sont en conflit. Cette bulle de protection est construite pour pallier les différentes incertitudes et délais, en particulier (liste non exhaustive) :

— Incertitude sur la vitesse, due aux instruments de mesure.
— Incertitude sur la position, due aux instruments de mesure.
— Temps de réaction du contrôleur, en particulier le temps qu'il met pour repérer un conflit.
— Temps de réaction du pilote, par exemple le temps qu'il met pour réagir à un ordre du contrôleur.
— Temps de communication, la durée des échanges verbaux entre les contrôleurs et les pilotes.

Cette bulle de protection est une forme géométrique englobant l'avion, qui est nommée "zone de séparation". La forme de la **zone de séparation** d'un avion correspond à un cylindre orienté selon l'axe de l'altitude (axe des z). Dans la majorité des cas, cette zone est d'un rayon $r_{conflit} = 5NM(= 9,26km)$, et d'une hauteur $h_{conflit} = 1000ft = 304,8m$, comme le montre la figure 1.9.

Nous pouvons observer trois **types de conflits** dans le monde aérien [Alam et al., 2007] :
— un croisement : deux avions sur des routes différentes convergent vers un *waypoint* commun à leurs routes,
— un rattrapage : un avion rattrape un autre avion sur la même route parce qu'il a une plus grande vitesse,
— un face à face : deux avions sur la même route sont l'un en face de l'autre. Cette situation ne doit normalement pas arriver grâce à la règle semi-circulaire (1.2.2), mais peut arriver lorsqu'un avion change de niveau de vol, *FL* et qu'un autre avion est au-dessus ou au-dessous de lui.

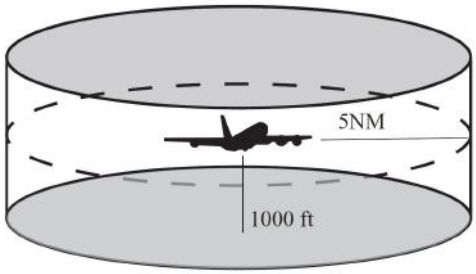


Figure 1.9 – Zone de séparation d'un avion

La **résolution des conflits** est différente en fonction de la typologie du trafic ("en route", de "transition" c'est-à-dire en montée, ou en descente, ou aux abords des aérodromes). En route, les contrôleurs français préfèrent, dans l'ordre (qui correspond à l'ordre de l'action la moins consommatrice en carburant à la plus consommatrice) :

1. Modifier la vitesse des avions.
2. Modifier le cap des avions.
3. Modifier le niveau de vol des avions.

Cet ordre est bien sûr indicatif, et peut varier, par exemple, lorsqu'un avion est proche de sa destination, il arrive plus souvent de le faire descendre pour éviter un conflit. De plus, cet ordre n'est pas le même partout, en particulier aux Etats-Unis où les contrôleurs préfèrent modifier le niveau de vol ou la vitesse des avions, avant de modifier leur cap. La raison avancée est de préserver les routes comme elles ont été définies [Erzberger, 2006][Rantanen and Wickens, 2012].

La **capacité d'un secteur** dépend directement de la capacité d'une équipe de contrôleurs à gérer un certain nombre d'avions. Celle-ci varie en fonction de nombreux autres facteurs, en particulier de la typologie du secteur (en route/ en approche, la forme et la taille du secteurs, entre autres). Cette notion de complexité du trafic aérien est très développée dans la littérature (du point de vue de la complexité [Breil, 2017],[Prandini et al., 2012], du point de vue de la charge mentale du contrôleur [Martin, 2013],[Borghini et al., 2014][Hilburn, 2004]) mais nous ne l'aborderons pas ici. Dans [Flynn et al., 2003], la limite de capacité d'un secteur est estimée entre 45 et 50 vols par heure, mais cette valeur reste très spécifique aux secteurs étudiés, et restent des chiffres élevés.

1.3.2 Air Space Management et Air Traffic Flow Management

L'*ASM* et l'*ATFM* sont deux facettes de la gestion du trafic aérien qui, au contraire de l'*ATS*, sont prises en compte avant la phase tactique (voir ci-dessous). Dans l'ensemble, ces deux services ont pour but de distribuer la demande (le nombre d'avions qui veulent voler) en fonction de la capacité des secteurs (dans lesquels vont passer les avions) pour que la capacité des secteurs ne soit pas dépassée.

En aéronautique, trois phases de gestion du trafic aérien sont distinguées [ECTL, 2018] :

— la phase **Stratégique**, allant de un an à sept jours avant le jour d'opération (c'est-à-dire le jour même où les avions volent),

— la phase **Pré-Tactique**, les six jours avant le jour d'opération,

— la phase **Tactique**, le jour d'opération.

Stratégique. Entre un an et sept jours avant le jour d'opération, la phase stratégique consiste à se concentrer sur la collecte de toutes les activités liées à l'*ASM* , c'est-à-dire la définition des différents secteurs, de leur géométrie, des routes, des différentes procédures de vol, dans le but de les améliorer, mais aussi de les adapter aux flux qui sont prévus. Elle consiste en particulier à étudier les données de vols passés (comme les *Flight Plans*), mais aussi prévoir les futures demandes, afin d'être capable d'accepter au mieux cette demande, et de prévoir des possibles dépassements de la capacité des différents secteurs. Dans ces flux prévus, nous pouvons notamment citer les flux saisonniers, tels que les flux d'été s'effectuant du Nord de l'Europe vers le Sud de l'Europe, ou des évènements majeurs (sportifs par exemple).

Pré-Tactique. Durant les six jours avant le jour d'opération, la phase pré-tactique consiste à étudier la demande du jour d'opération, et à la comparer avec les capacités qui ont été calculées. La demande étant mieux connue (les plans de vols par exemple sont affinés), de même que certaines conditions météorologiques, des ajustements peuvent être effectués pour mieux prédire le besoin à chaque instant. Le but global de cette phase est d'équilibrer au mieux la demande et la capacité (comme les configurations des secteurs, des propositions de modifications de *Flight Plans* ...).

Tactique. La phase tactique se déroule pendant le jour d'opération. Le but est de prendre des décisions en temps réel concernant la demande telle qu'elle est observée, ou est prédite, avec un granularité bien plus fine. Ici, il faut prendre en compte des évènements non prévus tels que des changements de météorologie, des grèves, des problèmes dans un aéroport (sur une piste, des problèmes au départ...), mais aussi les opportunités qui se présentent pour faire les changements nécessaires dans la demande pour équilibrer celle-ci avec la capacité. Une des mesures applicables pendant cette phase est de retarder ou d'avancer les heures de départ des avions.

1.4 Conclusion

Nous avons fait dans ce premier chapitre une présentation de la gestion du trafic aérien dans son ensemble.

En résumé, la grande majorité du trafic aérien est structurée autour d'un réseau de routes. Afin de surveiller le trafic, et de gérer l'évitement entre les différents avions, l'espace aérien a été subdivisé en sous-espaces, dont le volume a été défini pour permettre à des contrôleurs aériens d'effectuer cette tâche. Dans le but d'assurer une plus grande sécurité du trafic aérien, une zone de séparation correspondant à une portion de cylindre de

$5NM$ de rayon et de $1000ft$ de hauteur est établie autour de chaque avion. En fonction de la demande, les sous-espaces peuvent être regroupés ou dégroupés dans le but de mieux gérer la demande pendant le contrôle. Afin de ne pas dépasser la capacité des espaces aériens, ou des aéroports, le trafic aérien est aussi géré avant leur phase de vol, dans le but de réguler la demande et de réduire la charge des contrôleurs.

Cette gestion du trafic aérien fait face à différents défis, que nous présentons dans le chapitre suivant.

2 Défis et objectifs de la thèse

2.1 État des lieux de la gestion du trafic aérien (ATM)

A part quelques années comme celle de 2008, depuis 1970 le trafic aérien mondial n'a pas cessé d'augmenter comme le montre la figure 2.1. Cette augmentation a été possible grâce à de nombreuses avancées techniques et organisationnelles.

Parmi les avancées techniques, l'amélioration de la précision des radars a permis en 50 ans de réduire les normes de séparation de $20NM$ à $5NM$ horizontalement et de $2000ft$ à $1000ft$ verticalement. Ce qui a permis d'avoir non seulement le double de niveaux de vols, mais aussi d'avoir plus d'avions à une même altitude [DGAC, 2013].

Un contrôleur aérien ne pouvant pas contrôler un nombre infini d'avions à la fois (1.3.1), cette augmentation a été compensée par l'augmentation du nombre de contrôleurs et du nombre de secteurs, comme le montre la figure 2.2.

Cependant, la subdivision de l'espace aérien en secteurs ne peut pas se faire indéfini-

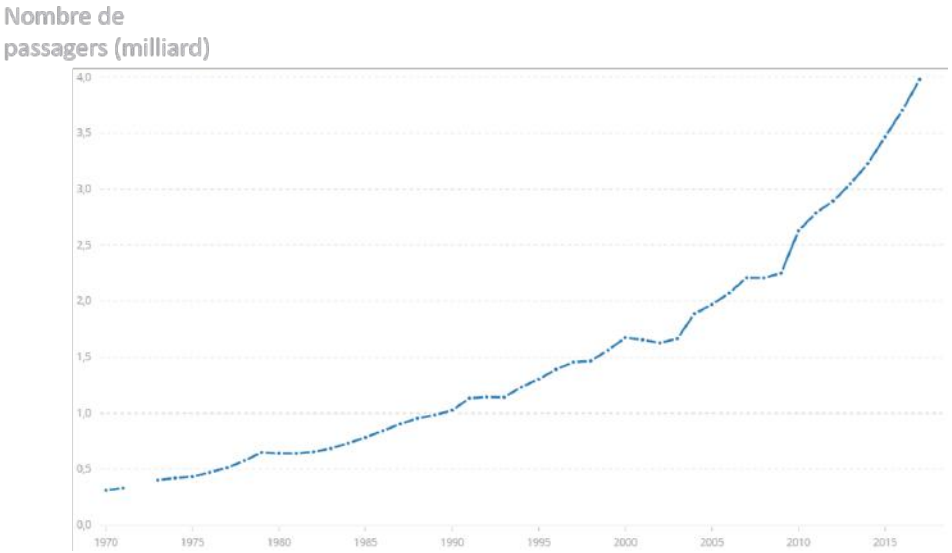


Figure 2.1 – Évolution dans le monde du nombre de passagers transportés par l'aviation civile (en milliard) par an

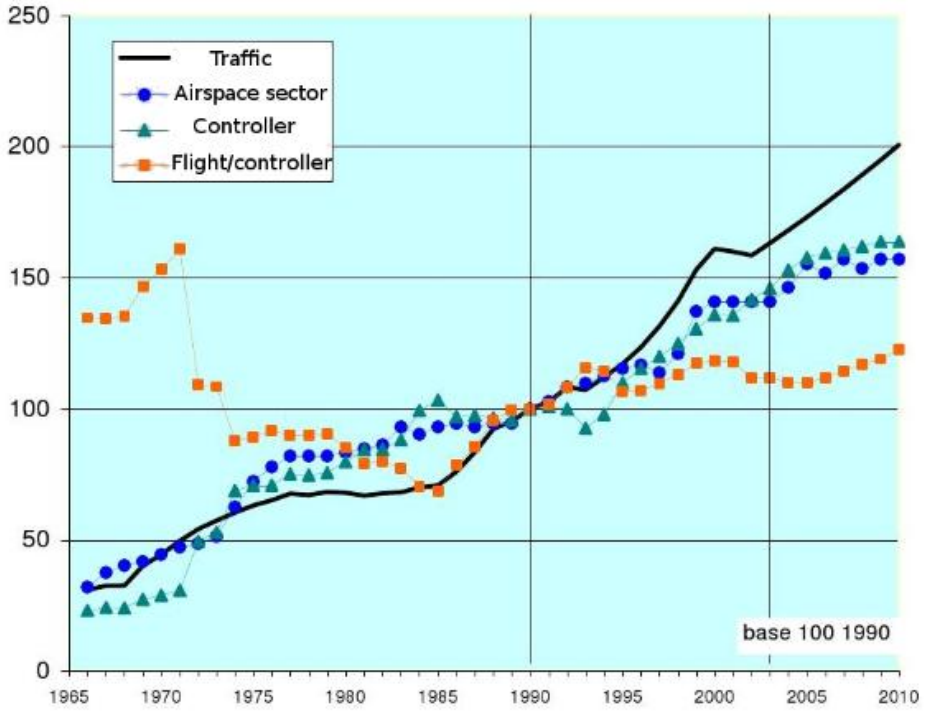


Figure 2.2 – Évolution en France entre les années 1966 et 2010 du nombre d'avions, de secteurs, de contrôleurs, et d'avions par contrôleur [Alliot and de Verdière, 2003]

ment. En effet, un contrôleur a besoin de temps pour interagir avec les avions dans son secteur, tout d'abord pour les communications réglementaires d'entrée et de sortie du secteur, mais aussi pour gérer les conflits. Pour cela des modifications de la forme des secteurs ont été proposées [Tran Dac, 2004] ; néanmoins, cette remodélisation des secteurs est sujette aux mêmes contraintes de taille.

Ainsi, même si la capacité au fil des années a augmenté, elle n'a pas totalement suivi la croissance du trafic ; ce qui amène aux congestions que connaissent aujourd'hui l'Europe et les Etats-Unis. Cette congestion se situe surtout au niveau des aéroports aux Etats-Unis et en Europe.

Ces limites, qui sont déjà atteintes dans certains cas, risquent d'être dépassées dans quelques années puisque les prévisions d'évolution du trafic sont toujours à la hausse. En effet, les estimations prévoient une augmentation du trafic aérien d'environ 2,5% par an comme le montre la figure 2.3.

2.2 Le défi du trafic aérien

La gestion du trafic aérien actuel a atteint différentes limites qui sont d'autant plus difficiles à surmonter que de nouvelles contraintes et de nouveaux besoins sont apparus et apparaissent encore pour la gestion du trafic aérien.

Par exemple, l'utilisation des drones pourrait considérablement augmenter le nombre

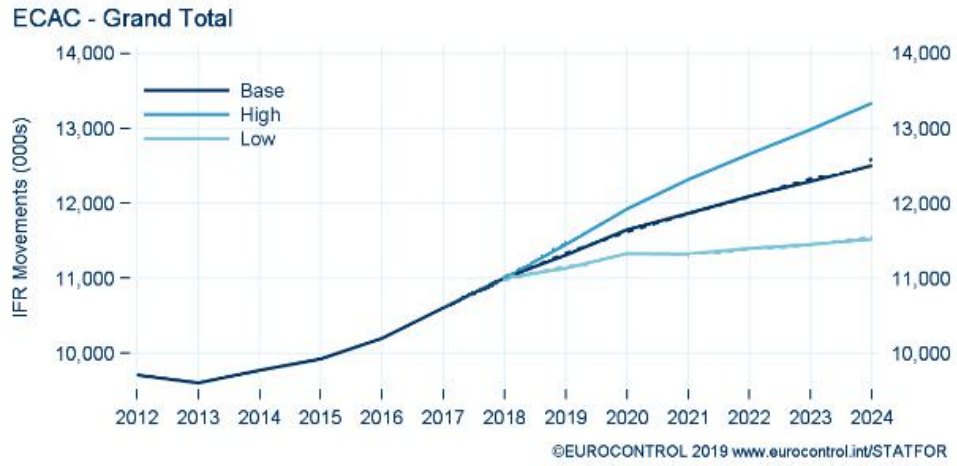


Figure 2.3 – Prévisions de l'évolution du trafic aérien Européen (en milliers de vols IFR), EUROCONTROL, mai 2019 [ECTL, 2019]

d'objets volants, ajouterait une grande hétérogénéité d'objets volants, avec différentes altitudes et vitesses. D'autres défis apparaissent aussi, comme la réduction de l'impact environnemental, ou encore et toujours l'augmentation de la sécurité.Le problème n'est plus mono-critère comme il pouvait l'être au début de l'aviation où seule la sécurité importait. Il est à présent non seulement *multi-critère*, mais aussi *multi-objectif* de par les différents acteurs qui sont impliqués. En effet, alors que les passagers veulent avoir un transport rapide, efficace et peu onéreux, un aéroport veut augmenter au maximum la sûreté, le contrôle aérien la sécurité, et les compagnies aériennes leurs profits.

Cette problématique de congestion ouvre la voie à de nombreuses propositions, comme une meilleure gestion des ressources actuelles, une redéfinition des espaces ou de leur utilisation, ou une gestion complètement différente du trafic aérien. Cette problématique fait donc l'objet de nombreuses recherches. Dans le monde de l'aviation, il existe notamment deux grands programmes de modernisation de l'espace aérien, les programme NextGen (États-Unis) et le programme SESAR (Europe). Afin de montrer l'efficacité de ces nouvelles structures, méthodes ou outils, il est nécessaire de les tester. Dans le contexte aérien où la sécurité prime, cela implique dans la quasi-majorité des cas de les utiliser dans un premier temps en simulation.

2.3 Le besoin de simulations interactives réalistes

Dans de nombreux domaines, la simulation est un outil puissant pour apprendre, visualiser, comprendre l'impact d'une décision à un moment donné sur tout un système. Le monde de l'aviation ne fait pas exception, et les simulations sont utilisées dans de nombreux contextes, que ce soit pour la formation des contrôleurs, pour l'intégration ou la validation de nouveaux outils, dans les phases opérationnelles pour prédire le trafic, ou encore comprendre un incident. Ces simulations ont besoin d'abord d'être réalistes, du point de vue des trajectoires et du trafic. Elles nécessitent aussi de correspondre aux besoins d'un utilisateur,

par exemple un expert scénariste qui construit une simulation pour un apprenti contrôleur, peut vouloir que la simulation contienne trois collisions et vingt avions. La simulation devra donc satisfaire ces objectifs. Pour la suite, nous appelons ces objectifs, qui correspondent à des contraintes voulues par un scénariste, des *situations*.

La mise au point d'un scénario réaliste de trafic virtuel respectant des *situations* impose des délais longs (typiquement plusieurs semaines) [Signor et al., 2004] et la mobilisation d'une équipe conséquente pour contrôler son déroulement. De plus, les simulations actuelles sont peu ou pas interactives, et, si elles existent, les interactions sont souvent effectuées par des humains. Par exemple, un contrôleur aérien en apprentissage nécessite, au début, au moins une personne pour gérer l'interactivité de sa simulation, et peut nécessiter jusqu'à quatre personnes pour des simulations avec beaucoup d'avions en approche. L'évaluation de nouvelles politiques de gestion de trafic est en conséquence onéreuse et limitée.

En conclusion, une automatisation de la génération de scénarios, mais aussi une interactivité accrue, et assistée par un système, seraient des apports intéressants.

2.4 Objectifs de la thèse

L'objectif de cette thèse est la structuration multi-niveau et multi-critère d'une simulation. Nous définissons la **structuration d'un simulation** comme étant l'organisation des entités mobiles dans le but de générer un trafic qui soit **réaliste**, aussi bien du point de vue du comportement des entités, c'est-à-dire le respect de leurs contraintes physiques et comportementales, que du point de vue du trafic général, le tout en produisant des *situations* voulues par un scénariste. Cette structuration doit permettre la génération d'un scénario reproductible, mais aussi l'interaction, et donc l'adaptation en temps réel.

Nous décomposons l'objectif de cette thèse en plusieurs sous-objectifs :

1. Être capable de générer un scénario réaliste du point de vue macroscopique et microscopique, c'est-à-dire un scénario comportant des **trajectoires réalistes**, le tout dans un **trafic réaliste**,
2. Une composante importante du réalisme de trafic est l'**évitement de collisions**, le système devra donc permettre l'évitement de collisions entre les différentes entités de la simulation,
3. Lors de la simulation, l'utilisation du scénario devra permettre l'apparition des différentes **situations** voulues par le scénariste,
4. Cette simulation devra être **interactive**, et permettre à des acteurs humains d'agir sur la simulation et que celle-ci réagisse à ses actions, en s'adaptant éventuellement pour continuer de générer les *situations* désirées.
5. Enfin, si la thèse s'inscrit dans le trafic aérien, le modèle devra être **générique**, afin d'être applicable à d'autres trafics, tel que le trafic routier.

Les chapitres suivants font un état de l'art des méthodes de structuration de simulation de trafic aérien et routier, ainsi que des méthodes de planification de trajectoires et d'évitement de collisions, afin de positionner cette thèse en accord avec ces objectifs.

Deuxième partie

État de l'art

3 État de l'art sur la structuration de simulations de trafic

L'objectif de cette thèse est de structurer les entités mobiles d'une simulation de trafic afin de générer un trafic qui soit réaliste au niveau macroscopique et microscopique, dans lequel les entités mobiles sont capable de s'éviter, mais aussi de générer un ensemble de situations observables requises par un scénariste, et de s'adapter aux modifications de leur environnement. Ces modifications sont notamment dues à des interactions avec un acteur humain dans la simulation.

Nous divisons le réalisme de la simulation en deux parties. La première concerne le *réalisme des trajectoires* (niveau microscopique). Cette partie se divise en deux sous-niveaux : le premier sous-niveau concerne la *physique* de la trajectoire c'est-à-dire la faisabilité de la trajectoire par une entité mobile. Le deuxième sous-niveau concerne le *comportement* d'une entité mobile, c'est-à-dire si l'entité mobile effectue une trajectoire qu'il effectuerait en temps normal. Par exemple, un avion peut sortir le train d'atterrissage, c'est-à-dire ses roues, quand il est en haute altitude, mais cette action n'est pas réaliste du point de vue du comportement, puisque les avions n'effectuent pas cette action habituellement. Le deuxième niveau du réalisme de la simulation concerne le *réalisme du trafic*, c'est-à-dire un trafic qui contient des caractéristiques réelles, comme l'évitement entre les entités, ou une charge habituelle de l'espace aérien.

Les *situations* sont des caractéristiques qu'un scénariste veut voir apparaître dans sa simulation, par exemple des densités de trafic, un panne moteur, une mauvaise attribution de niveau de vol, ou encore un flux d'avions particulier.

Nous explorons dans un premier temps les méthodes utilisées pour générer des scénarios et structurer des simulations de trafic aérien (section 3.1). Cette section propose une analyse et une discussion des avantages et des limites de ces différentes méthodes avant de conclure sur le manque d'outils pour la génération de trafic aérien réaliste. La section (3.2) explore les outils pour la génération de trafic routier.

3.1 Structuration de simulations de trafic dans le monde aérien

Le domaine de l'aviation possède de nombreux outils pour générer des trajectoires réalistes à partir de plans de vol (voir section 1.2.2). Nous dirigeons le lecteur vers d'autres références pour plus de détails [Rantrua, 2017]. Nous notons néanmoins que la quasi totalité

des méthodes utilisent un modèle mathématique afin de générer la trajectoire. Ces modèles décrivent la physique de l'entité mobile, comme des taux de montée, des taux de virage, des vitesses d'un avion.

En revanche, le domaine de l'aviation possède peu d'outils pour structurer une simulation, encore moins pour des simulations interactives avec des *situations* (à part avec des acteurs humains), et cette structuration se fait quasi uniquement par la génération de scénarios contenant les *situations*.

Dans cette section, nous donnons un aperçu des méthodologies utilisées pour générer des scénarios (section 3.1.1), puis nous faisons un état de l'art des outils pour la génération de trafic aérien réaliste (section 3.1.2), et ensuite des outils de génération de scénarios avec des conflits, qui sont les seules *situations* non liées à la densité générées automatiquement (section 3.1.3). Tous les systèmes et méthodes présentés dans ces deux dernières parties sont ensuite discutés dans la dernière sous-section (section 3.1.4).

3.1.1 Catégories de méthodologie de génération de scénarios dans le monde aérien

Dans le domaine de l'aviation, deux catégories de méthodologies sont principalement utilisées pour générer des scénarios à des fins de simulation [Alam et al., 2007] [Signor et al., 2004][Kupfer et al., 2013]. Nous décrivons dans ce qui suit ces deux catégories :

— Les méthodologies "à partir de zéro" ("*from scratch*" dans la littérature) : l'expert scénariste ajoute plan de vol après plan de vol jusqu'à atteindre le trafic qui correspond aux *situations* définies (figure 3.1a).

— Les méthodologies "à partir d'un enregistrement d'un trafic" : l'expert scénariste applique des modifications jusqu'à ce que les *situations* souhaitées soient obtenues (figure 3.1b).

Dans la majorité des cas, ces méthodologies se basent sur des plans de vols et des modèles mathématiques représentant la physique des avions pour générer des trajectoires, ce qui génère deux problèmes : d'une part, les plans de vols ne sont que des chemins que les avions suivent selon leurs capacités, mais aussi en fonction des pilotes (qui obéissent souvent à des directives de leur compagnie), et l'utilisation des modèles physiques ne suffit pas pour obtenir le réalisme souhaité [Rantrua, 2017]. D'autre part, travailler sur un plan de vol implique de le simuler pour obtenir une trajectoire qui puisse être vérifiée par un humain. Par conséquent, la majorité des méthodes nécessitent un expert scénariste pour créer le scénario.

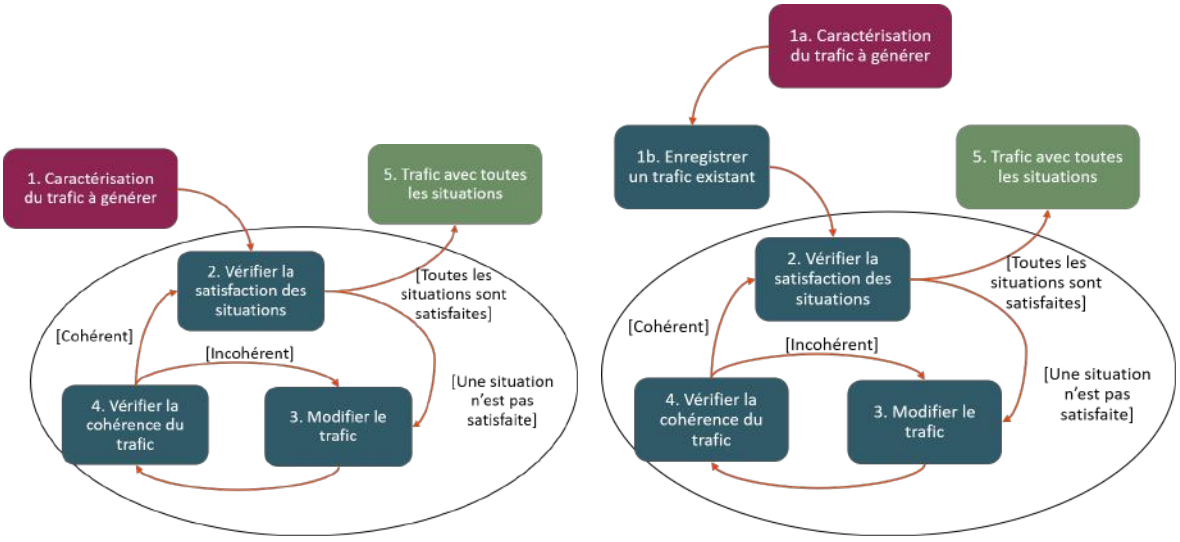
Les deux catégories de méthodologies utilisent des plans de vol pour définir une trajectoire. Dans ce cadre, la simulation est nécessaire pour vérifier que les *situations* sont satisfaites, et que les modifications apportées (par exemple une modification du plan de vol ou l'ajout d'un autre avion) n'interagissent pas de manière négative avec le trafic déjà généré. Cette vérification correspond à la phase 4 dans la figure 3.1.

La principale différence dans ces deux catégories de méthodologies réside dans la phase de modification du trafic (phase 3) :

— Dans la catégorie à partir de zéro, une modification du trafic correspond majoritairement à l'ajout d'avions et leur modification jusqu'à obtenir une *situation*, et de façon plus secondaire à vérifier que les avions n'interagissent pas entre eux.

— Dans la catégorie à partir d'un enregistrement, une modification concerne majoritairement la modification d'un plan de vol (par exemple ajouter un *waypoint*, voir section 1.2.2), et la vérification de la cohérence du trafic. De manière minoritaire, la phase 3 peut éventuellement correspondre à l'ajout ou la suppression de plans de vols, par exemple pour modifier la densité.

Dans les deux cas, générer un scénario prend du temps, nécessite un expert scénariste, le scénario peut manquer de réalisme par rapport à un trafic réel, ne peut pas être modifié en temps réel et manque ainsi de flexibilité et d'adaptabilité.



(a) A partir de zéro (b) A partir d'un enregistrement

Figure 3.1 – Les deux catégories de méthodologies de génération de scénario de trafic aérien

Historiquement, le choix de l'une ou l'autre de ces deux catégories vient surtout de l'objectif de la simulation. En général, les simulations qui nécessitent plusieurs *situations* variées sont construites à partir de zéro, alors que la construction du scénario à partir d'un enregistrement se fait plutôt quand le réalisme est l'élément principal, ou que les *situations* ne sont pas nombreuses ni variées, comme le suggère [Signor et al., 2004].

Les différents systèmes de génération de trafic aérien se basent sur ces deux catégories. Dans ce qui suit, la première section (section 3.1.2) présente des systèmes se préoccupant majoritairement de générer des trajectoires ou des trafics réalistes, la deuxième (section 3.1.3) se focalise sur la génération de situations de conflits.

3.1.2 Génération de scénarios de trafic aérien réaliste

La définition d'un trafic réaliste commence souvent par la génération de trajectoires réalistes. Cette génération de trajectoires réalistes suit le principe des deux catégories de méthodologies présentées précédemment, c'est-à-dire soit en créant une trajectoire à partir de zéro,

la plupart de temps à partir de *Flight Plan* et d'un modèle mathématique comme le montre [Rantrua, 2017], soit en modifiant des trajectoires réelles, à partir d'un enregistrement.

Génération de scénarios de trafic aérien réalistes de zéro à partir de *Flight Plan*. Nous présentons deux systèmes, le premier [Heesbeen et al., 2003] permet de générer des trajectoires réalistes tout en respectant une densité de trafic (qui peut donc être une densité réelle observée), le second [Prats et al., 2017] génère des trajectoires réalistes avec différents comportements possibles.

[Heesbeen et al., 2003] présente le "Traffic Manager" (TMX), un environnement de simulation complet incluant un générateur de scénarios. Ce générateur de scénarios utilise en entrée un environnement contenant une zone simulée et des *waypoints* reliés par un réseau de routes. Un utilisateur peut définir un nombre d'aéroports ou des points d'arrivée/départ fictifs (qui correspondent à des entrées/sorties dans la zone simulée) et des intervalles de temps moyens entre chaque départ sur les points de départ. Le système crée ensuite automatiquement des avions qui partent de ces points, et attribue à chacun un point d'arrivée. Pour rejoindre son point d'arrivée depuis son point de départ, l'avion suit soit une trajectoire directe entre les deux points, soit suit un plan de vol prédéfini par un utilisateur. Ainsi, la génération de scénarios permet dans ce système de générer une densité stable du trafic. Les trajectoires sont ensuite générées dans la simulation en utilisant la table *Base of Aircraft DAta (BADA)* [Mouillet, 2017]. Cette table correspond à des modèles mathématiques décrivant les performances des avions, et leur comportement pour suivre une route. Notons que d'autres modèles mathématiques existent, par exemple la table utilisée dans les simulateurs *Aircraft Simulation for Traffic Operations Research (ASTOR)*.

[Prats et al., 2017] présente le projet APACHE, projet de simulation de trafic aérien comportant différents sous-systèmes, en particulier APACHE-TAP. Ce dernier prend en entrée les demandes de trafic (des paires aéroport de départ et aéroport d'arrivée) et des données météorologiques. Les plans de vols générés peuvent prendre en compte le réseau de routes et des préférences utilisateurs (*cost index*, distance minimale, temps minimal), ou d'autres stratégies, telles que le *free route* (c'est-à-dire laisser l'avion décider de sa route, sans utiliser de réseau de route, donc en général le laisser effectuer le plus court chemin). Les trajectoires sont ensuite générées en fonction d'un modèle de performance des avions. Dans leur cas d'utilisation, les auteurs utilisent le modèle *BADA* [Mouillet, 2017], et se limitent à la définition de trajectoires dans l'espace en-route, c'est-à-dire au dessus du *FL*180 (voir section 1.2.2).

Génération de scénarios de trafic aérien réaliste à partir de données réelles Les auteurs de [Signor et al., 2004] décrivent la génération d'un scénario comme étant fastidieuse, et proposent en conséquence AvScenario, une interface utilisateur pour accélérer le développement de scénarios. Le logiciel AvScenario est un outil générique de manipulation de données de scénarios aéronautiques (données météorologiques, les *Flight Plans*, le réseau de routes...), qui peut être utilisé par différents simulateurs. Il est capable de lire automatiquement diverses données de vols (qui sont hétérogènes dans le format), et de modifier celles-ci afin de générer le scénario tel que spécifié dans les phases amont. Cette modification se fait

majoritairement grâce à une interface graphique ; les *Flight Plans* peuvent être modifiés par simple glisser-déposer. D'autres modifications sont possibles avec la même interface comme modifier le réseau de routes, ou d'autres caractéristiques de l'espace aérien comme les zones d'espace aérien à usage spécial. Un fichier de trafic AvScenario peut être exporté dans le format XML ou dans des formats spécifiques de simulation.

[Kupfer et al., 2013] propose un système comparable, spécialisé dans la phase d'approche des avions, avec des outils de détection d'erreur et de modification de *Flight Plans* pour normaliser un ensemble de données sélectionnées. Le système facilite donc la génération en fournissant des interactions et des indicateurs sur les données à un expert scénariste.

3.1.3 Génération de scénarios avec conflits

Dans le cadre de l'automatisation de la structuration de scénarios avec des *situations*, la génération de *scénarios* avec conflits est la plus étudiée. Ces systèmes et méthodes sont majoritairement faits dans le but de tester des systèmes de détection de conflits et/ou d'évitement. Nous retrouvons les deux catégories de méthodologies précédentes, à la différence que la catégorie "à partir de zéro" génère des trajectoires réalistes sans tenir compte du réseau de routes (ou de données).

Génération de scénarios avec conflits à partir de données réelles. Les auteurs de [Bilimoria, 2001] proposent une méthode pour tester les systèmes de détection de conflits. Il s'agit d'utiliser un trafic réel enregistré, qui par conséquent contient peu de conflits puisqu'il est normalement régulé par les contrôleurs, mais a l'avantage de contenir toutes les imperfections des données réelles afin de tester les systèmes de détection de conflits. Pour augmenter le nombre de conflits potentiels, la méthode consiste à modifier la définition de la zone de séparation afin d'obtenir des pseudo-conflits. La zone de séparation peut alors être augmentée en taille, verticalement ou horizontalement, ou bien être décalée par rapport à l'avion. En particulier, il est possible d'effectuer des décalages différents sur les différents avions, notamment en altitude, ce qui correspond à translater les trajectoires en altitude.

[Paglione et al., 2003] reprend l'idée d'utiliser des trajectoires réelles afin de tester les systèmes de détection de conflits. En revanche le système développé ne prend pas les trajectoires réelles, mais les conditions initiales réelles et le plan de vol volé (c'est-à-dire les véritables temps de passage à chaque *waypoint* et les véritables altitudes). Cette méthode permet de générer des trajectoires réalistes qui correspondent approximativement aux trajectoires si les contrôleurs n'avaient pas eu d'influence sur le trafic. L'approche pour obtenir des conflits est différente, puisqu'elle consiste à décaler dans le temps (quelques minutes) les différentes trajectoires afin de créer des conflits. Afin d'explorer l'espace de recherche, et trouver les bons décalages, un algorithme génétique est utilisé. Dans cet algorithme, chaque gène correspond à un décalage dans le temps d'une trajectoire et un chromosome correspond à un scénario. La fonction de fitness évalue négativement le décalage total dans le temps de toutes les trajectoires c'est-à-dire à quel point les données ont été modifiées, et elle évalue positivement le nombre de conflits créés.

Génération de scénarios avec conflits à partir de zéro. Le système de [Alam et al., 2007] crée des scénarios de conflits difficiles à résoudre en utilisant un algorithme génétique. La génération de conflits se fait à partir de la définition du *Closests Points of Approach (CPA)*. Le *CPA* correspond au point où les deux avions (ou entités mobiles dans le cas le plus général) passent le plus proche l'un de l'autre. Le système crée uniquement des conflits entre deux avions respectant un ensemble de caractéristiques et calcule ensuite des trajectoires linéaires. Les caractéristiques à définir sont :

1. La position en longitude et latitude et aussi l'altitude de l'un des avions lors du *CPA*.
2. La séparation horizontale : la distance horizontale séparant les deux avions lors du *CPA*.
3. La séparation verticale : la distance verticale séparant les deux avions lors du *CPA*.
4. L'angle d'approche entre les deux avions.
5. La géométrie de chaque avion : les auteurs ont défini la géométrie comme étant la phase de vol (montée, en route, descente) de chaque avion.

Le système prend en entrée une définition de l'espace aérien, la densité d'avions souhaitée et des intervalles de valeurs pour les caractéristiques de conflits. L'espace aérien est un pavé défini par une longitude, une latitude, une altitude maximale et minimale, un intervalle de temps pour la simulation. Le système génère un nombre de conflits en accord avec la densité souhaitée. Chaque conflit (l'ensemble des caractéristiques) représente un gène dans un chromosome, qui lui-même définit un scénario. Le système génère une population de scénarios de départ, qu'il donne à un simulateur pour générer les trajectoires, sur lesquelles il teste ensuite un algorithme de détection de conflits. Afin de maximiser la difficulté des scénarios pour l'algorithme testé, la fonction de *fitness* favorise les erreurs des systèmes de détection de conflits (faux négatifs et faux positifs).

Encounters from Actual Trajectories (EnAcT) [Ritchie III, 2019] utilise la même méthode pour la génération de scénarios avec des conflits, en partant de la définition de *CPAs* avec les caractéristiques précédentes, en ajoutant le type d'avion et en utilisant la table *BADA*. Le système est ensuite capable de générer un ensemble de conflits, soit à partir d'une définition des caractéristiques de chaque conflit voulu, soit à partir d'un nombre de conflits et d'une distribution probabiliste pour chaque caractéristique.

3.1.4 Conclusion sur la structuration de simulation dans le monde aérien

Nous avons évalué les différents systèmes et méthodes présentés précédemment selon huit critères principaux :

1. **Structuration de la simulation :** quand s'effectue la structuration de la simulation, lors de la construction d'un *scénario* ou lors de la *simulation*,
2. **Catégorie de méthodologie :** quelle catégorie de méthodologie est utilisée pour structurer la simulation, soit en générant un scénario *à partir de zéro*, soit *à partir d'un enregistrement* du trafic,
3. **Automatisation :** quel est le niveau d'automatisation du système, si le système est autonome, son automatisation est *forte*, sinon l'automatisation est *faible*, ou *moyenne*,

4. **Utilisation de l'environnement :** est-ce que le système s'appuie sur l'environnement pour développer son scénario ? Est-ce qu'il ne fait *aucune* utilisation de l'environnement, ou bien utilise-t-il un *réseau de routes*, ou encore la *météorologie* ? Ou encore est-ce que l'utilisation est *indirecte* et vient de l'utilisation d'un enregistrement ?,
5. **Situations :** quelles *situations* sont générées ? Comme une *densité de trafic* (niveau **macroscopique**), des *conflits* ou des *densités de départ* (niveau **microscopique**),
6. **Réalisme :** le *réalisme de la simulation*, est divisé en fonction du *réalisme du trafic* et du *réalisme de la trajectoire*.
   — **Le réalisme de trajectoire :** nous divisons le réalisme de trajectoire en deux, la *physique* de la trajectoire c'est-à-dire si la trajectoire respecte les règles physiques qui régissent l'entité mobile et le *comportement* de l'entité mobile, c'est-à-dire si l'entité mobile se comporte comme une entité mobile dans le monde réel. Le réalisme de la trajectoire du point de vue **physique** vient soit d'un *enregistrement*, soit d'un *modèle mathématique* comme les tables *BADA* ou *ASTOR* en utilisant un réseau de routes, soit encore d'un modèle mathématique utilisé sur trajectoires *linéaires*. Le réalisme de la trajectoire du point du vue du **comportement** est soit *complexe*, soit *simple*, soit n'est pris en compte d'*aucune* manière.
   — **Le réalisme du trafic :** soit il vient de l'utilisation d'un *enregistrement*, soit il est *à la charge de l'expert scénariste*, soit il vient de mesures de *densité* que le système essaye d'obtenir dans le scénario, ou alors le système ne présente *aucun* réalisme de trafic.
7. **Adaptation du trafic :** est-ce que la simulation est adaptative, c'est-à-dire est-ce que les avions réagissent aux modifications de leur environnement de *simulation* ou de leur *scénario* ? L'adaptation du trafic à l'évolution de la **simulation** correspond par exemple à l'évitement entre des avions qui n'étaient pas en conflit mais qui sont devenus en conflit à cause de la modification de la trajectoire d'un avion. L'adaptation du trafic à l'environnement de simulation est soit effectuée *par l'humain*, soit il n'y a *aucune* adaptation. Quant à elle, l'adaptation du trafic au scénario correspond par exemple à modifier la trajectoire d'un des deux avions impliqués dans un conflit parce que l'autre avion a eu une modification de sa trajectoire. Cette adaptation est soit effectuée *par l'humain*, soit il n'y a *aucune* adaptation.
8. **Nécessité d'un plan de vol :** un dernier critère propre à l'aviation, si la méthode ou le système nécessite un plan de vol de manière, *obligatoire*, *non obligatoire*, ou au contraire ne nécessite *aucun* plan de vol.

Critères / Systèmes	Structuration simulation	Catégorie de méthodologie	Automatisation	Utilisation de l'environnement	Situations		Réalisme			Adaptation du trafic		Nécessite plan de vol
					Micro	Macro	Trajectoire		Trafic	Scénario	Simulation	
							Physique	Comportement				
[Bilimoria, 2001]	Scénario	Enregistrement	Faible	Indirectement	Conflits	Aucun	Enregistrement	Enregistrement	Enregistrement	Aucune	Aucune	Aucun
[Signor et al., 2004]	Scénario	Enregistrement	Faible	Réseau de routes	Non précisé	Non précisé	Dépend du simulateur	Dépend du simulateur	Enregistrement et à la charge du scénariste	Humaine	Humaine	Obligatoire
[Kupfer et al., 2013]	Scénario	Enregistrement	Faible	Réseau de routes	Non précisé	Non précisé	Modèle Mathématique (ASTOR)	Aucun	Enregistrement et à la charge du scénariste	Humaine	Humaine	Obligatoire
[Paglione et al., 2003]	Scénario	Enregistrement	Moyenne	Indirectement	Conflits	Aucun	Enregistrement	Enregistrement	Enregistrement	Aucune	Aucune	Aucun
[Heesbeen et al., 2003]	Scénario	De zéro	Faible	Réseau de routes	Densité de départ	Densité trafic	Modèle Mathématique (BADA)	Aucun	Densité possible par fitting	Humaine	Humaine	Non obligatoire
[Prats et al., 2017]	Scénario	De zéro	Moyenne	Réseau de route, Météorologie	Dépend de l'expert (Densité)	Dépend de l'expert (Densité)	Modèle Mathématique (BADA)	Simple	Enregistrement et à la charge de l'expert	Humaine	Humaine	Non obligatoire
[Alam et al., 2007]	Scénario	De zéro	Moyenne	Aucune	Conflits	Densité trafic	Linéaire avec Modèle mathématique (BADA)	Aucun	Aucun	Aucune	Aucune	Aucun
[Ritchie III, 2019]	Scénario	De zéro	Moyenne	Aucune	Conflits	Densité trafic	Linéaire avec Modèle mathématique (BADA)	Aucun	Aucun	Aucune	Aucune	Aucun

Table 3.1 – Comparaison des systèmes de structuration de simulation dans le monde aérien

Tous les systèmes et méthodes présentés **structurent la simulation** au cours de la génération du scénario pour jouer les différentes *situations* souhaitées. La structuration au cours de la simulation, si elle existe, se fait ensuite à travers des interfaces utilisateurs, ce qui implique donc des humains lors de la simulation [Signor et al., 2004][Kupfer et al., 2013]. Parmi les 8 systèmes et méthodes présentés, seulement la moitié permettent l'interaction lors de la simulation, et s'adaptent à des modifications, et ce, seulement par l'intervention d'humains [Heesbeen et al., 2003] [Prats et al., 2017] [Signor et al., 2004] [Kupfer et al., 2013]. En structurant la simulation uniquement par la génération du scénario, la simulation s'**adapte** très faiblement, voire pas, à des modifications de l'environnement, des trajectoires, ou d'un quelconque paramètre. La seule adaptation constatée vient de l'intervention d'un acteur humain, ce qui est un procédé coûteux, et limité pour le **passage à l'échelle**.

Parmi les 8 systèmes et méthodes présentés, la moitié utilise des **enregistrements** de trafic et l'autre moitié **part de zéro** pour générer les scénarios. D'un côté, l'utilisation d'un enregistrement permet d'obtenir un réalisme de départ accru, puisque l'enregistrement contient à la fois le *réalisme de la trajectoire*, c'est-à-dire un respect de la physique du mobile, mais aussi des comportements que peut avoir un mobile, et le *réalisme de trafic* [Heesbeen et al., 2003] [Prats et al., 2017] [Signor et al., 2004] [Kupfer et al., 2013]. Ce réalisme est altéré pour correspondre aux attentes de l'expert scénariste. Cette altération de réalisme est plus forte dans [Signor et al., 2004] [Kupfer et al., 2013] ; en revanche ces outils permettent de générer des *situations* plus variées et fines, à la charge des experts scénaristes. *A contrario*, partir de zéro pour construire des trajectoires permet de construire incrémentalement un scénario qui correspond aux différentes *situations* désirées, et donc facilite leur création [Bilimoria, 2001] [Paglione et al., 2003] [Alam et al., 2007] [Ritchie III, 2019]. En revanche, partir de zéro altère fortement le réalisme qui peut se regagner en partie par l'utilisation de modèles mathématiques (réalisme de trajectoire) et en utilisant en entrée des indicateurs, par exemple des départs réels ou des densités de trafic (réalisme de trafic) [Heesbeen et al., 2003] [Prats et al., 2017].

Sur les 8 systèmes et méthodes et systèmes présentés, 5 sont **automatisés**, avec des entrées diverses [Heesbeen et al., 2003] [Prats et al., 2017] [Paglione et al., 2003] [Alam et al., 2007] [Ritchie III, 2019]. Parmi ces 5 systèmes, seulement [Paglione et al., 2003] utilise des enregistrements, et uniquement pour la génération de situations simples de conflit avec comme seule caractéristique la perte de séparation. Ceci requiert peu de modifications du trafic (dans sa méthode, uniquement des translations dans le temps). Les *situations* générées par les systèmes et méthodes automatiques sont soit relativement simples (des conflits avec peu de caractéristiques) [Paglione et al., 2003], soit manquent de réalisme [Heesbeen et al., 2003] [Alam et al., 2007] [Ritchie III, 2019], soit demandent un travail de l'expert scénariste pour obtenir des *situations* plus complexes qui ne concernent que la densité pour les méthodes citées ici [Heesbeen et al., 2003] [Prats et al., 2017]. Pour conclure, l'automatisation présentée dans ces méthodes apporte peu de flexibilité, et repose majoritairement sur le scénariste pour obtenir des situations fines et variées.

L'adaptation aux modifications est faible, et l'**utilisation de l'environnement** est relativement faible. En effet, seuls [Prats et al., 2017] utilisent plus que le réseau de routes dans la génération de scénarios. Par exemple, la définition d'évènements comme l'ouverture ou la fermeture d'un zone militaire peut être possible dans ces simulations, mais c'est à l'opéra-

teur humain de le spécifier, et pas au système qui génère le trafic et le structure.

Les ***situations*** générées sont uniquement des *situations* de densité de trafic, densité de départ, ou des conflits. Les méthodes et systèmes manquent de généricité pour générer des *situations* différentes en particulier [Paglione et al., 2003][Bilimoria, 2001][Alam et al., 2007][Ritchie III, 2019]. Aucun système n'est capable de générer des *situations* variées et potentiellement contradictoires.

Le **réalisme des trajectoires** du point de vue **physique** est obtenu dans 5 méthodes par l'utilisation d'un modèle mathématique (*BADA* ou *ASTOR*) [Heesbeen et al., 2003] [Alam et al., 2007] [Prats et al., 2017] [Kupfer et al., 2013] [Ritchie III, 2019]. Nous ajoutons [Signor et al., 2004] à cette liste car la majorité des simulateurs utilisent ces tables ou des tables équivalentes [Rantrua, 2017]. Dans [Bilimoria, 2001] [Paglione et al., 2003] le réalisme de trajectoire est obtenu par l'utilisation d'enregistrements. Du point de vue du **comportement**, le réalisme vient aussi de l'utilisation d'enregistrements pour ces deux systèmes. Dans le cas des méthodes qui utilisent un modèle mathématique, il n'y a pas de réalisme de trajectoire du point de vue du comportement [Heesbeen et al., 2003] [Alam et al., 2007] [Prats et al., 2017] [Kupfer et al., 2013] [Ritchie III, 2019], sauf [Prats et al., 2017] qui inclut dans le choix du plan de vol des critères que peuvent utiliser les compagnies aériennes (cost index, free flight, temps le plus court).

Le **réalisme de trafic** vient en général d'enregistrements [Bilimoria, 2001] [Paglione et al., 2003], souvent en le laissant à charge de l'expert scénariste [Prats et al., 2017] [Kupfer et al., 2013] [Signor et al., 2004]. Dans les autres cas, il peut venir de l'utilisation de densités si l'expert scénariste le souhaite [Heesbeen et al., 2003], ou il n'y a aucun réalisme de trafic [Alam et al., 2007] [Ritchie III, 2019].

En **conclusion**, les outils pour la structuration de simulation dans le domaine aérien manquent d'*automatisation*, et ont besoin d'un modèle plus générique pour la génération de *situations* diverses et variées. La structuration de la simulation à partir du scénario mais aussi dans la simulation est une piste à développer, qui apporterait certainement plus de flexibilité et d'*adaptation* aux simulations. Enfin, un système capable de partir de zéro pour générer les *situations*, mais aussi capable d'utiliser des enregistrements pourrait amener à un compromis entre le réalisme de la simulation, et l'obtention des *situations*. Globalement, le monde de l'aviation manque d'outils pour la structuration de simulations, nous nous sommes donc intéressé à d'autres domaines afin d'étudier des systèmes et outils qui complèteraient ces différents travaux.

3.2 Structuration de simulations de trafic routier

Le domaine de l'aviation manque d'outils pour accélérer la génération de scénarios et de simulations réalistes. Cependant, dans d'autres domaines utilisant des simulations de trafic, des outils plus variés existent. Dans le domaine du trafic routier par exemple, plusieurs systèmes existent. Comme pour le trafic aérien, ces systèmes se divisent en deux familles, celles qui structurent la simulation à partir de données réelles, et celles qui partent de zéro. Nous présentons dans un premier temps des méthodes appartenant à ces deux familles, avant de faire une synthèse sur la structuration de simulation de trafic routier (section 3.2.3).

3.2.1 Structuration de simulation de trafic routier à partir d'enregistrements

Le monde routier possède de nombreux simulateurs microscopiques où les différentes entités choisissent de manière plus ou moins autonomes comment évoluer dans le réseau routier, en fonction de l'environnement, pour arriver à leur destination. Ces simulateurs comportent de nombreux paramètres, dont la calibration est essentielle pour obtenir un trafic routier réaliste. Au contraire de l'aviation, le monde routier ne possède pas (dans la quasi-totalité des cas) d'informations sur les trajectoires des véhicules, ou sur l'origine et la destination des véhicules, mais seulement des indicateurs locaux, par exemple le nombre de véhicules qui passent par un tronçon de route en particulier. Les experts scénaristes estiment alors à la fois les points de départ et d'arrivée des véhicules (matrice d'origine-destination) et le modèle de comportement des véhicules. Dans ce qui suit, nous présentons certaines des méthodes utilisées, et nous renvoyons les lecteurs vers d'autres références [Hollander and Liu, 2008] [Barceló et al., 2010] pour plus de détails.

Les auteurs de [Kim and Rilett, 2003] proposent un système pour calibrer des modèles de comportements microscopiques (c'est-à-dire de comportements des véhicules), en utilisant des données d'observation de volume sur des intersections et un algorithme simplexe séquentiel. L'algorithme améliore séquentiellement une fonction de fitness basée sur la différence entre l'observation et les résultats de la simulation en modifiant les paramètres du modèle microscopique. Le système est testé pour optimiser les paramètres des simulateurs CORSIM et TRANSIMS, sur un petit réseau composé d'une avenue principale traversée par cinq autres rues. Les auteurs évaluent leur calibrage sur des mesures de volume.

Les auteurs de [Hourdakis et al., 2003] proposent une méthodologie pour le calibrage de modèles microscopiques en 3 étapes. La première étape consiste à calibrer les paramètres globaux qui affectent la performance globale du système (comme la vitesse maximale, le taux d'accélération maximum, le taux de décélération maximum) afin d'obtenir un trafic globalement correct, avec une marge d'erreur permissive. La deuxième consiste à calibrer les paramètres de manière plus locale, c'est-à-dire sur des sections de route (par exemple la vitesse désirée sur cette section). Une troisième étape, optionnelle, calibre le système afin d'obtenir des *situations* particulières observées. Les auteurs ont appliqué le système uniquement à une autoroute, et conseillent de le calibrer en suivant le sens du flux de véhicule à chaque étape. La méthodologie peut s'utiliser avec un système d'optimisation ; dans leur implémentation, les auteurs utilisent le système d'optimisation non linéaire MINOS. Celui-ci combine 4 méthodes : la méthode du simplexe, une méthode quasi-Newton, la méthode du gradient réduit et une méthode du gradient projeté. La méthodologie est appliquée pour calibrer le simulateur microscopique AIMSUN. Le calibrage est évalué sur des mesures de volumes, de temps de trajet et de tailles de file d'attente.

Les auteurs de [Toledo et al., 2003] proposent une approche itérative pour calibrer une matrice d'origines-destinations de véhicules sur tout le réseau, le choix de routes et le modèle de comportement du conducteur. Une première estimation des paramètres des différents modèles est faite à partir des données, puis elle est utilisée dans le reste de leur méthodologie. Celle-ci consiste en différentes itérations qui suivent l'ordre suivant :

1. Estimer une matrice d'origines-destinations et calibrer le modèle de choix de route.
2. Calibrer le modèle de comportement.

A la fin de la deuxième étape, si les calibrations ne satisfont pas les données, on reprend à nouveau les calibrations. Afin de calibrer plus facilement les modèles, on les calibre dans un premier temps sur une portion de route simple dont on dispose des données d'entrée et de sortie, puis on affine la calibration du modèle sur le reste du réseau. L'évaluation de la calibration se fait sur des mesures de volumes, ainsi que de temps de trajet, et de tailles de file d'attente.

La méthodologie est utilisée dans [Toledo et al., 2003] [Ben-Akiva et al., 2000] sur un réseau de taille moyenne avec une calibration majoritairement itérative avec des choix de l'expert, en utilisant les outils de MITSIMLab (matrice d'Origines-Destinations), et l'outil d'optimisation stochastique Boss/Quattro (paramètres comportementaux). La méthodologie a été reprise et automatisée en partie dans [Jha et al., 2004] pour être utilisée sur un réseau de grande taille, en évaluant sur des mesures de volume et de temps de trajet. Ces 3 applications utilisent et calibrent le simulateur MITSIMLab. Les auteurs de [Chu et al., 2003] présentent une méthodologie semblable à [Toledo et al., 2003], en inversant l'ordre des étapes précédentes, pour optimiser les paramètres du simulateur Paramics. Ils effectuent leur calibration avec des outils stochastiques de Paramics (matrice d'Origines-Destinations), et par essais-erreurs (paramètres comportementaux). Ils évaluent leur calibrage sur des mesures de volumes.

Les auteurs de [Ma and Abdulhai, 2002] proposent GENOSIM, un algorithme génétique pour calibrer les paramètres du simulateur Paramics. Dans GENOSIM, chaque gène correspond à un ensemble de paramètres à calibrer. Afin de réduire l'espace de recherche, les auteurs conseillent de choisir un sous-ensemble de paramètres en se focalisant sur les paramètres qui influencent le plus les résultats observés, mais aussi sur les paramètres qui sont le plus liés au modèle de comportement du véhicule (conduite et choix de route). La méthode est appliquée sur un réseau de taille moyenne, dont les auteurs génèrent la matrice d'origines-destinations à partir d'une étude auxiliaire, sur laquelle ils se basent pour optimiser les paramètres. L'évaluation se base sur des observations de volumes sur des intersections.

Les auteurs de [El Esawey and Sayed, 2011] proposent une méthodologie pour le calibrage des simulations, qu'ils appliquent sur un réseau de taille moyenne avec le simulateur VISSIM. Ils divisent les critères d'évaluation du calibrage en deux groupes, le premier concerne les mesures de la demande de trafic (volume de trafic dans une rue par exemple), et le deuxième concerne les performances du système (vitesse de parcours d'une rue par exemple). Pour cela, leur méthodologie consiste dans un premier temps à estimer la matrice d'origines-destinations (avec un outil propre à VISSIM), pour ensuite calibrer itérativement le modèle de choix de routes, et enfin calibrer itérativement les différents paramètres de comportement des véhicules. Les auteurs calibrent donc dans un premier temps les critères liés à la demande, c'est-à-dire la matrice d'Origines-Destinations (en utilisant un outil de VISSIM), pour ensuite calibrer en fonction des autres critères d'évaluation. Ils préconisent de calibrer un sous-ensemble de paramètres qui influence le plus la simulation dans le but de limiter le temps de calibration. Ils effectuent cette deuxième calibration par essais-erreurs pour deux paramètres, et en échantillonnage par hypercube latin (échantillonage quasi-aléatoire qui s'assure que les échantillons n'ont pas de coordonnées en commun) pour les valeurs des autres paramètres sélectionnés, pour ensuite sélectionner l'échantillon donnant les meilleurs

résultats. Leur évaluation se fait sur des observations de volumes de trafic et de temps de trajet.

3.2.2 Structuration de simulation de trafic routier à partir de zéro

Il existe aussi dans le monde du trafic routier des systèmes qui permettent la génération de situations plus variées que des volumes de passages, des temps de trajet et des tailles de file d'attente. A la différence des systèmes précédents qui veulent générer une simulation de trafic réaliste sur différentes échelles de réseaux, ces systèmes se focalisent sur la génération de trafic avec des situations autour d'un véhicule particulier, *trafic ambiant*, que ce soit pour l'apprentissage d'un conducteur, ou le test de systèmes à bord du véhicule.

Les auteurs de [Abdelgawad et al., 2016] présentent un système qui génère automatiquement un trafic d'une certaine densité et d'une certaine vitesse autour du véhicule simulé. Le trafic est simulé dans un espace autour du véhicule, appelé région d'intérêt (region of interest (ROI)), afin d'éviter de générer un trafic trop important et donc de minimiser les calculs.

Les auteur de [Tönnis, 2007][Tönnis and Klinker, 2009] présentent un outil de génération de scénarios qui utilise des voitures miniatures. L'outil est composé de voitures miniatures équipées de capteurs passifs qui sont repérés par des caméras pour suivre les mouvements donnés à celles-ci. Les utilisateurs déplacent chaque voiture sur une table équipée d'un écran projetant l'environnement dans lequel ils les déplacent. L'outil permet d'accélérer la génération de trajectoires de voitures en permettant une interaction tangible, avec comme désavantage de requérir un utilisateur pour générer une trajectoire à chaque itération. L'adaptation d'un scénario lors de la simulation vient alors d'humains qui interagissent avec les voitures miniatures. Pour pallier le besoin de plusieurs utilisateurs, l'outil permet d'enregistrer les trajectoires afin d'en générer plusieurs dans une simulation qui alors ne s'adapte plus.

[Bonakdarian et al., 1998] propose un système d'adaptation de trafic ambiant pour le simulateur Hank dans le but de suivre un scénario donné et ainsi mettre un conducteur dans certaines conditions. Cette adaptation se fait par l'intermédiaire d'une attribution dynamique de rôles aux véhicules en utilisant une métaphore du cinéma. Les auteurs décrivent un scénario ou scène, comme un triplet composé du *set* (l'environnement et l'ensemble des entités mobiles), du *cast* (les rôles principaux décrits par des caractéristiques et des capacités), et d'un *script* (la description des actions à effectuer). Leur système est composé de différents types d'entités, des *actors* qui sont alors majoritairement des entités mobiles, un *stage manager* dont le but est de générer des entités mobiles avec certaines caractéristiques spécifiées par l'expert scénariste, et un *casting director* dont le but est de chercher le casting d'*actors* parmi les entités mobiles et d'orchestrer le bon déroulement du scénario. Pour orchestrer le scénario, le *casting director* donne des instructions aux *actors* qui adaptent leur comportement pour les prendre en compte. Ces directives sont des changements de couleur d'un feu tricolore, ou des changements de comportement d'un véhicule (ajuster la vitesse, ignorer un feu, s'arrêter, changer de voie, tourner). Le trafic ambiant est généré dans une ROI autour du véhicule.

Nous retrouvons cette métaphore dans le système de [Wassink et al., 2006] de manière étendue, en utilisant le paradigme multi-agent. Dans ce système, nous retrouvons les *ac-*

tors qui sont les entités mobiles, le *casting director* dont le rôle est uniquement de choisir les bons *actors*, l'orchestration étant assurée par les *actors* sélectionnés. Contrairement à [Bonakdarian et al., 1998], ce système ne place pas le véhicule de l'élève directement sur le lieu de la *situation*, mais dans un environnement ouvert. Le système est un système d'apprentissage de conduite, qui génère des *situations* à la volée en fonction du contexte et des capacités de l'élève. Il détecte de manière décentralisée des lieux qui sont propices à la création d'une *situation* grâce à des *location scouts*. Ces derniers en informent des *assistant directors* qui proposent leur *situation* au *director* qui décide alors des *situations* à jouer. L'attribution des rôles se fait de manière décentralisée par des *assistant casting directors*. Les différents *actors* se coordonnent alors pour créer les situations décidées par le *director*, potentiellement en échangeant leurs rôles avec d'autres entités s'ils ne sont pas ou plus capables de réaliser leur rôle dans la situation.

Les auteurs de [Olstam et al., 2011] [Olstam and Espié, 2007] utilisent une métaphore équivalente venant du théâtre, avec un scénario composé de *plays* (*situations*), un *manuscript* (*set*), une *role list* (*casting*), et des *roles* (*actors*). Le système met en scène les différentes *situations* les unes après les autres pour un conducteur dans un trafic ambiant. Les véhicules sont soit autonomes (les véhicules qui n'ont pas de rôle), soit contrôlés (ceux qui ont un rôle). Ces générations de *situations* sont entre-coupées de phases de trafic ambiant sans *situation*, phases appelées *every day life*, et de préparation de la *situation*. La préparation d'une *situation* est découpée en 3 sous-problèmes : l'estimation des conditions de départ, l'attribution des rôles et la transition entre le simple trafic ambiant et le trafic ambiant qui crée une *situation*. La transition se fait par adaptation des comportements des différentes entités pour qu'il y ait le bon nombre d'entités mobiles, et que les véhicules avec un *role* soient en place au début de la *situation*. Cette mise en place peut donner lieu à de l'entraide entre entités mobiles pour arriver à la mise en place des acteurs. Le système est utilisé pour générer deux *situations* : une dans laquelle le véhicule qui est devant est lent et un véhicule passe rapidement sur sa gauche, et une autre où le véhicule de devant freine brusquement pour sortir de la route avec trois véhicules qui le dépassent en même temps.

Les auteurs de [Gajananan et al., 2013] proposent le *Scenario Markup Language* (SML), un langage XML pour définir des scénarios qu'ils utilisent ensuite dans leur système. L'utilisateur doit spécifier l'ensemble des entités dans le scénario, et assigner les rôles dans les différentes situations qu'il a spécifiées. Les entités mobiles qui ont un rôle se déplacent alors de manière autonome pour correspondre aux caractéristiques initiales voulues, et sont ensuite dirigées par un superviseur qui orchestre leur comportement pour exécuter la *situation*. Les auteurs testent leur système avec une collision par rattrapage.

3.2.3 Conclusion sur la structuration de simulations de trafic routier

Nous avons évalué les différents systèmes et méthodes présentés précédemment selon les 7 premiers critères utilisés pour la structuration de simulation de trafic aérien adapté pour la simulation de trafic routier (sous-section 3.1.4). Nous ajoutons 3 critères, qui sont la possibilité d'**application à l'ATM**, la **taille du réseau** sur lequel se fait la structuration, et la **méthode de calibration utilisée**. L'apparition de ce dernier critère vient d'une automatisation accrue dans les systèmes du domaine routier.

— **Catégorie de méthodologie :** comme pour le trafic aérien, les méthodes de structuration de simulation appartiennent à la catégorie *à partir de zéro*, ou à la catégorie *à partir d'un enregistrement*,

— **Taille de réseau :** le réseau est soit de *petite* taille (portion de route, autoroute), soit de taille *moyenne* (partie d'une ville), soit de *grande* taille (ville), soit *non spécifié*.

— **Automatisation :** le niveau d'automatisation du système est soit *fort*, soit *moyen*, soit *faible*,

— **Utilisation de l'environnement :** les systèmes peuvent utiliser un *réseau de route* simple, soit l'utilisation de l'environnement *dépend du simulateur* (c'est-à-dire que les simulateurs utilisent par exemple les feux tricolores, des panneaux de signalisation, etc.).

— **Réalisme :**

  — **Le réalisme de trafic :** le réalisme de trafic vient de mesures locales de *volumes* de passage, de *temps de trajet*, de tailles de *file d'attente*, que le système essaye d'obtenir dans le scénario ou dans la simulation. Il peut *dépendre de l'expert scénariste*, ou alors le système ne présente *aucun* réalisme de trafic.

  — **Le réalisme de trajectoire :**

    — Le réalisme de trajectoires du point de vue **physique** vient d'un *modèle complexe*, *varié* ou *simple*, ou *dépend de l'expert*. Par modèle simple, nous entendons que les déplacements des véhicules sont régis par quelques règles de comportements. Au contraire, les modèles variés contiennent un nombre plus important de ces règles. Enfin, les modèles complexes peuvent être appris et dépendent majoritairement du contexte.

    — Le réalisme de trajectoires du point du vue du **comportement** est soit *simple*, soit *varié*, soit *complexe*, selon la même échelle que le réalisme physique.

— **Adaptation du trafic :**

  — **Simulation :** l'adaptation du trafic à l'environnement de simulation est soit effectuée par *l'humain*, soit par des *comportements* des entités mobiles.

  — **Scénario :** l'adaptation du trafic au scénario est soit effectuée par *l'humain*, soit par le comportement *simple* ou *avancé* des entités mobiles, soit d'*aucune* manière.

— **Structuration de la simulation :** comme pour le trafic aérien, la structuration de la simulation se fait lors de la construction d'un *scénario* ou lors de la *simulation*,

— **Méthode de calibration :** la méthode de calibration utilisée pour calibrer le trafic combine généralement un *outil* du simulateur (*MITSIMLab, VISSIM, Paramics*), une méthode d'optimisation (*algorithme génétique*, *Boss/Quattro*, *MINOS*), et parfois des mécanismes d'*essais-erreurs* ou l'utilisation d'échantillonages par hypercube latin, ou alors la calibration *n'est pas mentionnée*.

— **Situations :** au niveau microscopique, les situations générées sont des situations de *volume* local de passage, de *temps de trajet*, de *taille de file d'attente*, de *collision*, ou des situations *variées* dont la description est plus complète (comme le dépassement d'une voiture par 3 autres, une voiture qui grille un feu devant une autre...). Au niveau macroscopique, on retrouve des situations de *volume route*, *vitesse route*, *typologie trafic route* (*TTR*) sur une route, ou encore de *typologie de trafic* du niveau macroscopique.

— **Application à l'ATM :** la possibilité d'adapter le système ou la méthode à l'aviation est soit *faible*, soit *moyenne*, soit *forte*.

Critères / Systèmes	Catégorie de méthodologie	Taille du réseau	Automatisation	Utilisation de l'environnement	Réalisme			Adaptation du trafic		Structuration simulation	Méthode de Calibrage	Situations		ATM
					Trafic	Trajectoire						Micro	Macro	
						Physique	Comportement	Scénario	Simulation					
[Kim and Rilett, 2003]	Enregistrement	Moyenne	Fort	Dépend du simulateur	Volume	Modèle varié	Modèle varié	Comportement Simple	Comportement	Scénario et Simulation	Simplexe Séquentiel	Volume	Aucune	Faible
[Hourdakis et al., 2003]	Enregistrement	Petite	Moyen	Dépend du simulateur	Volume, Temps de trajet, File d'attente	Modèle varié	Modèle varié	Aucune	Comportement	Scénario et Simulation	MINOS (simplexe, quasi-Newton, gradient réduit, gradient projeté)	Volume, Temps de trajet, File d'attente	Aucune	Faible
[Toledo et al., 2003]	Enregistrement	Moyenne	Moyen	Dépend du simulateur	Volume, Temps de trajet, File d'attente	Modèle varié	Modèle varié	Comportement Simple	Comportement	Scénario et Simulation	Outil MITSIMLab, Boss/Quattro (optimisation stochastique)	Volume, Temps de trajet, File d'attente	TT	Faible
[Jha et al., 2004]	Enregistrement	Grande	Moyen	Dépend du simulateur	Volume, Temps de trajet	Modèle varié	Modèle varié	Comportement Simple	Comportement	Scénario et Simulation	Outil MITSIMLab, Boss/Quattro (optimisation stochastique)	Volume, Temps de trajet	Aucune	Faible
[Chu et al., 2003]	Enregistrement	Petite	Moyen	Dépend du simulateur	Volume	Modèle varié	Modèle varié	Aucune	Comportement	Scénario et Simulation	Outil Paramics, Essais-Erreur	Volume	Aucune	Faible
[Ma and Abdulhai, 2002]	Enregistrement	Moyenne	Fort	Dépend du simulateur	Volume	Modèle varié	Modèle varié	Comportement Simple	Comportement	Scénario et Simulation	Algorithme Génétique	Volume	Aucune	Faible
[El Esawey and Sayed, 2011]	Enregistrement	Moyenne	Moyen	Dépend du simulateur	Volume, Temps de trajet	Modèle varié	Modèle varié	Comportement Simple	Comportement	Scénario et Simulation	Outil VISSIM, Essais-Erreur, Échantillonnage par Hypercube Latin	Volume, Temps de trajet	Aucune	Faible
[Abdelgawad et al., 2016]	De zéro	Petite	Fort	Réseau de route	Dépend de l'expert scénariste (Volume Route, Vitesse Route)	Modèle simple	Modèle simple	Aucune	Comportement	Simulation	Non mentionné	Aucune	Volume Route, Vitesse Route	Faible
[Tönnis, 2007]	De zéro	Petite	Faible	Réseau de route	Aucun	Dépend de l'expert	Dépend de l'expert	Humaine	Humaine	Scénario et Simulation	Non mentionné	Variés	Aucune	Faible
[Bonakdarian et al., 1998]	De zéro	Petite	Fort	Dépend du simulateur	Dépend de l'expert scénariste (Volume Route, Vitesse Route, TTR)	Modèle varié	Modèle varié	Comportement Avancé	Comportement	Simulation	Non mentionné	Variés	Volume Route, Vitesse Route, TTR	Moyenne
[Wassink et al., 2006]	De zéro	Non spécifié (Moyenne ou Grande)	Fort	Dépend du simulateur	Sans indications	Modèle varié	Modèle varié	Comportement Avancé	Comportement	Simulation	Non mentionné	Variés	Non spécifié	Forte
[Olstam et al., 2011]	De zéro	Petite	Fort	Réseau de route	Dépend de l'expert scénariste (Volume Route, Vitesse Route, TTR)	Modèle varié	Modèle varié	Comportement Avancé	Comportement	Simulation	Non mentionné	Variés	Aucune	Moyenne
[Gajananan et al., 2013]	De zéro	Petite	Fort	Réseau de route	Dépend de l'expert scénariste	Modèle simple	Modèle simple	Comportement Avancé	Comportement	Scénario et Simulation	Non mentionné	Collision	Aucune	Faible / Moyenne

Table 3.2 – Comparaison des différents systèmes et méthodes de structuration de simulation de trafic routier

Parmi les 13 systèmes présentés, 7 systèmes appartiennent à la **catégorie de méthodologie** qui utilise des enregistrements de manière explicite pour structurer leur simulation. Ils utilisent ces enregistrements dans le but de générer un trafic réaliste. Néanmoins, les données utilisées sont des données locales, situées principalement à des intersections. Il s'agit de données de volume de passage en un point, de temps de trajet sur une portion de route, de taille de file d'attente, ou encore de typologie de trafic de route. De manière générale, ces données sont des données statistiques agrégées sur des intervalles de temps. Les 6 autres systèmes n'utilisent pas d'enregistrements de manière explicite, et sont donc considérés comme partant de zéro. Les 7 systèmes qui utilisent des données locales présentent une approche mixte entre les deux catégories. En effet, ces méthodes génèrent une matrice d'origine-destination (en partant donc de zéro), pour obtenir les données locales. La génération du scénario se focalise alors sur la génération des trajectoires, qui est une approche intéressante. Les méthodes qui partent de zéro présentent d'autres avantages détaillés dans la suite.

Cette structuration de la simulation se fait sur des **tailles de réseau** différentes. La moitié (7 systèmes) des structurations se fait sur des réseaux de petite taille, qui correspondent généralement à une portion de route simple. Sur la quasi totalité du reste, elle se fait sur un réseau de moyenne taille, l'exception étant [Jha et al., 2004] qui effectue une structuration sur un réseau de grande taille. Dans notre cas, nous voulons générer du trafic sur des réseaux de grandes tailles, ce qui pourrait rendre difficile le passage à l'échelle de certaines de ces méthodes.

A l'exception de [Tönnis, 2007], le **niveau d'automatisation** des systèmes est au moins moyen (5 systèmes), et fort dans le reste des systèmes. Le niveau moyen est constitué de méthodologies qui possèdent des outils pour effectuer certaines ou la totalité des tâches (génération de la matrice d'origine destination par exemple), mais n'ont pas explicitement de transitions automatiques entre les différentes tâches, et nécessitent donc un humain pour faire les transitions. L'automatisation est un critère important dans notre modèle puisque nous cherchons à contrôler un système complexe, qui peut fortement évoluer en fonction des interactions. Certains systèmes apportent des réponses intéressantes, en particulier lors de la simulation.

L'**utilisation de l'environnement** dépend fortement du simulateur. 9 systèmes utilisent différents éléments de l'environnement pour faire fonctionner les véhicules (feux tricolores, signalisation, parfois météorologie...). Au contraire de l'ATM, ces systèmes utilisent donc de manière autonome de nombreux critères dépendant de l'environnement pour faire évoluer les différentes entités mobiles. L'adaptation est alors dynamique, et pas forcément effectuée par un humain. Les autres méthodes utilisent au moins le réseau de route. De manière générale, cette utilisation de l'environnement est importante pour l'adaptation du trafic aux éléments de la simulation, et est donc un aspect positif des systèmes présentés.

Le **réalisme de trafic** est obtenu dans la majorité des cas par des mesures de volume de passage local (7 systèmes) ou de volume de route (3 systèmes à condition que l'expert scénariste utilise des données réelles). Il est aussi obtenu par des temps de trajet (4 systèmes) et sont équivalent sur une route, la vitesse de route (3 systèmes, encore à condition que l'expert scénariste utilise des données réelles). Les temps de file d'attente sont utilisés de manière

minoritaire (2 systèmes), pour obtenir un réalisme de trafic. En somme, le réalisme de trafic est concentré sur l'obtention de situations locales (correspondant à des données) dans une grande majorité des cas. Les systèmes n'essayent pas d'obtenir des comportements s'ils ne sont pas observés ailleurs. Par exemple, l'obtention d'un ralentissement sur une rampe d'accès à un périphérique n'est garanti que si des données locales l'attestent. Une généralisation des comportements pourrait avoir un intérêt, en particulier pour le passage d'un réseau à un autre.

Le **réalisme de trajectoire** est obtenu dans 10 des systèmes par l'utilisation d'un modèle du comportement du véhicule varié. Le modèle est soit issu d'un simulateur externe utilisé par le système [Kim and Rilett, 2003] [Hourdakis et al., 2003] [Toledo et al., 2003] [Jha et al., 2004] [Chu et al., 2003] [Ma and Abdulhai, 2002] [El Esawey and Sayed, 2011] [Bonakdarian et al., 1998] [Wassink et al., 2006], soit a été inclus dans le système en utilisant les mêmes comportements [Olstam et al., 2011]. Dans les autres systèmes, soit un sous-ensemble de ces comportements est implémenté [Gajananan et al., 2013] [Abdelgawad et al., 2016], soit le comportement dépend totalement de l'expert scénariste [Tönnis, 2007]. Nous ne détaillerons pas ces comportements, qui traitent principalement (mais pas seulement) de l'adaptation de la vitesse à l'infrastructure, le suivi de voiture, le changement de voie, le dépassement, l'évitement de conflit, le mouvement latéral, la signalisation de virage, et les feux de signalisation. Nous dirigeons le lecteur vers [Ratrout and Rahman, 2009] [Barceló et al., 2010] [Passos et al., 2011] [Lopez et al., 2018] pour plus de détails sur les principes et leurs différentes implémentations (analytique, probabiliste, logique floue...). Parmi ces systèmes avec des comportements variés, ceux qui utilisent un réseau de route moyen ou grand ont aussi des comportements pour décider du chemin à suivre, qui dépendent par exemple de la taille des files d'attente [Kim and Rilett, 2003] [Toledo et al., 2003] [Jha et al., 2004] [Ma and Abdulhai, 2002] [El Esawey and Sayed, 2011]. Pour les 3 autres systèmes, le comportement est soit simple [Abdelgawad et al., 2016] [Gajananan et al., 2013] et vient d'un sous-ensemble des comportements cités précédemment, soit est entièrement à la charge de l'expert scénariste [Tönnis, 2007]. Cette utilisation de comportements pour définir les trajectoires permet de générer des trajectoires réalistes du point de vue de la physique et du comportement. Néanmoins, ces comportements sont longs à paramétrer sur de grand réseaux (voir paragraphe sur les méthodes de calibration), et sont souvent peu ou pas adaptés au contexte, ce qui est préjudiciable lorsque le réseau est grand et qu'il présente des typologies différentes.

L'adaptation du trafic aux éléments de **simulation** vient de cette modélisation des entités mobiles par des comportements dans tous les systèmes, sauf dans [Tönnis, 2007] où elle est humaine, et dans [Hourdakis et al., 2003] et [Abdelgawad et al., 2016] où aucune adaptation au scénario n'est requise. L'adaptation du trafic au **scénario** est soit simple et consiste uniquement à respecter la matrice d'origine-destination [Kim and Rilett, 2003] [Toledo et al., 2003] [Jha et al., 2004] [Ma and Abdulhai, 2002] [El Esawey and Sayed, 2011], soit avancée, et contient des mécanismes dynamiques et distribués d'attribution de rôle et d'orchestration de la mise en scène du scénario à travers une métaphore du cinéma [Bonakdarian et al., 1998] [Wassink et al., 2006], ou du théâtre [Olstam et al., 2011]. De manière générale, l'utilisation de comportements locaux présente une adaptation très intéressante lors de la simulation, très proche du paradigme des systèmes multi-agents.

Des 13 systèmes présentés, 4 **structurent la simulation** uniquement lors de la simulation [Abdelgawad et al., 2016] [Bonakdarian et al., 1998] [Wassink et al., 2006] [Olstam et al., 2011], et les 9 autres structurent celle-ci lors de la construction du scénario et lors de la simulation. Parmi ces 9, 7 sont orientés uniquement dans la génération d'un trafic réaliste [Kim and Rilett, 2003] [Hourdakis et al., 2003] [Toledo et al., 2003] [Jha et al., 2004] [Chu et al., 2003] [Ma and Abdulhai, 2002] [El Esawey and Sayed, 2011]. Les 2 autres ont pour but de générer des situations, dans un trafic réaliste [Abdelgawad et al., 2016] [Tönnis, 2007]. Pour ces 2 derniers, [Tönnis, 2007] permet d'enregistrer des trajectoires générées par l'outil pour les ajouter à un scénario, et [Gajananan et al., 2013] permet d'écrire un scénario dans un langage de haut niveau, que le système interprète pour générer les *situations* désirées par l'expert scénariste. Dans les 4 systèmes qui structurent la simulation lors de celle-ci, [Abdelgawad et al., 2016] génère en continu dans une zone d'intérêt un trafic correspondant à des statistiques entrées en amont, et on retrouve les 3 systèmes de la métaphore du cinéma (ou du théâtre) et du trafic qui génèrent dynamiquement les situations [Bonakdarian et al., 1998] [Wassink et al., 2006] [Olstam et al., 2011]. Les 7 systèmes restant, dont le but est de générer un trafic réaliste procèdent par calibrage explicite du modèle d'un simulateur. Pour cela, ils effectuent deux opérations principales :

1. Générer une matrice d'origines-destinations.
2. Calibrer les paramètres de comportement des véhicules.

Leur principale différence est la manière avec laquelle ils effectuent ces opérations, c'est-à-dire les méthodes d'optimisation employées pour la **calibration des modèles**, mais aussi l'ordonnancement de ces opérateurs : successivement, en parallèle, de manière itérative sur tous les paramètres, de manière itérative sur certaines tâches ou sous-tâches... Dans ces méthodes, la structuration lors de la construction du scénario se résume à la construction de la matrice d'origines-destinations, la structuration lors de la simulation vient du comportement des entités mobiles. Ces méthodes d'optimisation sont souvent longues et difficiles à utiliser. Afin de calibrer plus rapidement, les auteurs utilisent souvent un sous-ensemble des paramètres de leur modèle. Et pour faciliter d'autant plus cette calibration lorsque le réseau est important, ils procèdent pas à pas en essayant de trouver des portions du réseau pour calibrer un sous-ensemble de paramètres (par exemple, une portion de route droite sans sortie, dont ils possèdent les volumes de passages à chaque bout). Cette calibration est d'autant plus difficile que le réseau est grand et que la typologie de trafic peut beaucoup changer d'une partie à l'autre du réseau [Jha et al., 2004]. Ainsi, une auto-adaptation des paramètres ou de comportements apporterait une simplification de la calibration des modèles à de nouveaux réseaux, ou sur les sous-réseaux.

Nous retrouvons les distinctions précédentes dans les **situations** qui sont générées. Les 7 méthodes qui génèrent un trafic réaliste génèrent des situations microscopiques liées aux données qu'elles possèdent (volume de passage, temps de file d'attente, temps de trajet) [Kim and Rilett, 2003] [Hourdakis et al., 2003] [Toledo et al., 2003] [Jha et al., 2004] [Chu et al., 2003] [Ma and Abdulhai, 2002] [El Esawey and Sayed, 2011], avec l'exception de [Toledo et al., 2003] qui génèrent aussi une typologie de trafic. Dans les méthodes restantes, les 3 méthodes utilisant la métaphore du cinéma (ou du théâtre) génèrent des situations microscopiques variées, sans situation macroscopique, à l'exception de [Bonakdarian et al., 1998] qui génère des typologies de trafic sur des routes, des volumes

de route et des vitesses de route. [Gajananan et al., 2013] quant à eux ne génèrent que des situations de collisions, et [Abdelgawad et al., 2016] ne génèrent que des situations de volume de route et de vitesse de route. De manière générale, les systèmes présentent peu de moyens pour obtenir des situations macroscopiques. En résumé, les méthodes utilisant la métaphore du cinéma (ou du théâtre) permettent une décentralisation naturelle de la réalisation des situations, et permettent par conséquent de générer des situations plus variées que les autres systèmes.

L'**applicabilité à l'ATM** dépend du niveau de généricité des systèmes. La totalité des systèmes qui calibrent les paramètres du modèle de comportement utilisent une méthodologie basée sur la structure de trafic routier ; par conséquent, ils ont une faible généricité, même si les méthodes d'optimisation utilisées sont génériques. [Tönnis, 2007] est peu applicable dans le monde aérien puisqu'un changement de plan sur une surface plane reste difficile, et l'obtention de réalisme physique par cette méthode reste relativement difficile. L'idée de [Abdelgawad et al., 2016], même si sa transposition n'est pas directe, existe déjà en plus complet dans l'aéronautique [Prats et al., 2017], et ne permet pas de générer des *situations* autres que des densités. [Gajananan et al., 2013] présente une description de plus haut niveau du scénario, qui reste peu générique dans son exécution. Enfin, les 3 systèmes utilisant la métaphore du cinéma (ou du théâtre) présentent des degrés plus avancés de généricité et donc d'applicabilité, en particulier [Wassink et al., 2006].

En conclusion, la moitié des systèmes présentés structurent la simulation en partant de la catégorie de méthodologie dite de zéro pour structurer la simulation. La génération de trajectoires du trafic routier dans ces systèmes se fait par un modèle physique, mais surtout par un système de comportements des entités mobiles, souvent variés, qui permet d'obtenir un trafic **flexible** qui s'**adapte** à des modifications dans l'environnement.

En revanche, le couplage entre le scénario et la simulation n'existe que dans la génération d'un trafic réaliste. Ces trafics réalistes sont sensibles à l'environnement pour lequel ils ont été calibrés. En effet, lors d'une perturbation les entités mobiles, qui n'ont pas conscience des *situations* qu'elles doivent générer à part leur destination, ne sont pas préservées les *situations* du scénario (*situations* de volume de passage par exemple).

Ce lien entre les objectifs macroscopique et le comportement microscopique a été étudié de manière approfondie dans des domaines autre que la calibration de simulateurs de trafic à travers la calibration dynamique de systèmes [Cheng et al., 2009], en particulier pour l'auto-adaptation dynamique de systèmes [Brun et al., 2009]. Cette auto-organisation est aussi présente dans les systèmes multi-agents [Serugendo et al., 2006]. Les systèmes multi-agents coopératifs ont montrés leur efficacité et leur adéquation pour de telles tâches [Boes, 2014], mais aussi pour de l'optimisation [Kaddoum, 2011]. Il sont aussi largement utilisés dans les simulations, notamment les simulations de trafic (maritime, routier, pédestre, ect.), en raison de leur localité et de leur passage à l'échelle.

Dans le cas où le système est capable d'adapter la simulation aux *situations* [Bonakdarian et al., 1998] [Wassink et al., 2006] [Olstam et al., 2011], il n'y a pas de conception de scénario en avance. La principale conséquence de l'absence de scénario est l'absence de garantie d'avoir les conditions nécessaires pour générer la situation voulue si le "*casting*" n'est pas possible, c'est-à-dire si des entités mobiles ayant les caractéristiques requises pour

la génération de la *situation* ne sont pas disponibles. L'expert scénariste doit intervenir dans le mécanisme de génération du trafic pour que celui-ci soit propice au casting et donc à la génération des situations. En revanche, les auteurs de [Wassink et al., 2006] présentent une distribution et une généricité très avantageuses pour s'adapter à d'autres trafics.

Enfin, les situations sont rarement des *situations* macroscopiques, et les mécanismes pour générer les situations globales sont souvent limités [Bonakdarian et al., 1998][Abdelgawad et al., 2016].

3.3 Conclusion sur la structuration de simulation de trafic et positionnement intermédiaire

Nous faisons dans cette section une synthèse des systèmes et méthodes présentés dans ce chapitre, en faisant une comparaison de la structuration de simulation de trafic dans le domaine de trafic aérien et dans celui de trafic routier (section 3.3.1), pour faire un premier positionnement de cette thèse (section 3.3.2).

3.3.1 Comparaison de la structuration de simulation dans le trafic aérien et dans le trafic routier

Au contraire du monde de l'aviation, le monde routier n'utilise pas d'enregistrements complets pour des raisons non indiquées. La principale raison que nous avançons est la difficulté d'obtenir des données complètes sur les réseaux routiers, notamment parce que les voitures ne diffusent pas d'informations, au contraire des avions qui diffusent en clair de nombreuses données dans leur environnement. La protection de la vie privée est aussi une hypothèse pour expliquer que de telles données ne soient pas accessibles : au contraire des avions qui transportent un ensemble de passagers (non identifiables à moins d'avoir accès aux bases de données des compagnies), une voiture est souvent personnelle, et identifiable (par l'endroit où se gare le conducteur le soir par exemple).

Cette absence de données a eu, au vu des systèmes présentés, un effet radical sur le développement des simulateurs et des différents systèmes et outils issus du monde routier. Afin de générer des trafics réalistes dans le monde routier, de nombreux modèles de comportements des différentes entités ont été développés, au contraire du monde de l'aviation qui a surtout développé des modèles physiques des différents types d'avions.

Ainsi, pour obtenir un trafic réaliste, le monde routier a besoin de calibrer les différents modèles de comportements, tandis que le monde de l'aviation peut se contenter d'utiliser des enregistrements.

Néanmoins, les systèmes de structuration de simulation de trafic aérien montrent différentes limites :

1. Les systèmes de l'aviation utilisent souvent un plan de vol pour générer les trajectoires des avions, alors que ce plan de vol n'est pas toujours respecté [Rantrua, 2017].
2. L'expert scénariste est souvent très impliqué dans la génération d'un trafic aérien car l'automatisation fait souvent défaut.

3. Les systèmes qui génèrent automatiquement des scénarios avec des *situations* manquent de généricité.
4. Les systèmes ne sont pas flexibles, et ne s'adaptent pas lors de la simulation, autrement que par l'intervention d'un humain.

De l'autre côté, les systèmes de structuration de simulation de trafic routier sont plus développés par différents aspects et apportent des réponses aux problèmes sus-cités :

1. Les simulateurs, qui utilisent des comportements locaux pour le choix des routes, nécessitent uniquement une matrice d'origines-destinations.
2. L'automatisation est bien plus forte, que ce soit pour générer des trafics réalistes en calibrant les modèles de comportement et les matrices d'origines-destinations, ou pour la génération de situations variées.
3. Certains systèmes ont été développés pour générer des situations lors de la simulation, et présentent une généricité intéressante [Wassink et al., 2006].
4. Les simulateurs à base de comportements présentent, lors de la simulation, des adaptations au scénario ou à d'autres aspects de la simulation très intéressants.

Néanmoins, les systèmes de structuration de simulation de trafic routier présentent eux mêmes différentes limites supplémentaires :

— Dans les systèmes présentés, le couplage entre la simulation et le scénario, n'existe pas (sauf avec un manque de résilience pour générer un trafic réaliste), ce qui est problématique pour rejouer des scénarios.
— Les situations générées sont rarement des situations macroscopiques, et les mécanismes pour générer ces situations sont souvent limités.

3.3.2 Positionnement intermédiaire

Pour rappel, l'objectif de la thèse est de concevoir un système capable de structurer une simulation de trafic, c'est-à-dire générer un scénario et une simulation avec des trajectoires et un trafic réaliste, notamment avec un évitement de collisions entre les entités, structurées autour de *situations* voulues par un scénariste. Cet objectif se décline en plusieurs sous-objectifs :

1. Permettre l'apparition d'un ensemble de *situations* dans la simulation,
2. Adapter la simulation à des interactions extérieures,
3. Générer un trafic et des trajectoires réalistes,
4. Permettre l'évitement de collisions entre les différentes entités mobiles,
5. Enfin, le système doit être générique.

Apparition de *situations* dans la simulation. Parmi les différents systèmes que nous avons étudiés, ceux utilisant la métaphore du cinéma (ou du théâtre) [Bonakdarian et al., 1998] [Wassink et al., 2006] [Olstam et al., 2011] apportent la réponse la plus générique pour générer de manière autonome des *situations* variées, en particulier les auteurs de [Wassink et al., 2006].

Néanmoins, ces derniers présentent le désavantage de créer les situations uniquement à la volée lors de la simulation. Par conséquent, nous nous inspirerons de la **décentralisation** observée dans ces systèmes et de la **généricité** qu'ils apportent, en prévoyant une adaptation lors de la simulation, mais aussi lors de la construction des scénarios.

Adapter la simulation à des interactions extérieures. Nous prévoyons que le système puisse s'adapter aux interactions extérieures, pour que le trafic s'adapte à des modifications de l'environnement sans l'intervention d'un humain, mais aussi pour maintenir les *situations* prévues.

Afin que le trafic s'adapte aux modifications de l'environnement, nous nous inspirerons des simulateurs de trafic routier. Le monde du trafic routier présente de nombreux simulateurs qui font évoluer les différentes entités mobiles en fonction de leur comportements locaux en utilisant des systèmes multi-agents [Mandiau et al., 2008], ou un paradigme proche [Passos et al., 2011]. Définir les différentes entités mobiles comme des agents, leur permet (à condition de leur donner les comportements nécessaires) de s'adapter de manière autonome à des modifications de leur environnement, par exemple dues à des interactions de plusieurs acteurs humains [Badeig et al., 2016], et donc d'obtenir une structuration lors de la simulation. Les différentes entités mobiles de la simulation devront donc bénéficier de comportements autonomes afin de s'adapter à des modifications de leur environnement, en particulier d'un comportement d'évitement.

Afin que le trafic s'adapte au scénario, nous prévoyons que les entités mobiles aient un comportement local, qu'elles peuvent modifier pour maintenir les situations, et que le générateur de scénarios puisse faire évoluer dynamiquement le scénario, et indiquer aux entités leur nouveau comportement si nécessaire.

Générer un trafic et des trajectoires réalistes. Comme nous avons pu le voir, le calibrage de modèles pour générer des trajectoires et des trafics réalistes est long et difficile [Ma and Abdulhai, 2002]. Afin de générer des trajectoires réalistes, nous proposons d'utiliser des comportements contextuels, c'est-à-dire des comportements qui varient en fonction du contexte. Ces comportements contextuels peuvent être des comportements d'adaptation donnés aux entités mobiles qui varient en fonction de l'environnement [Ksontini et al., 2015], ou issus de l'apprentissage (par exemple sur des enregistrements de trajectoires).

Dans le but de générer un trafic réaliste, nous utiliserons à nouveau le paradigme des systèmes multi-agents. En utilisant des données, comme des enregistrements de trajectoires ou des données utilisées dans le domaine routier, un ensemble de points de passages devra être satisfait en volume de passage.

Permettre l'évitement de collisions entre les différentes entités mobiles. Une composante non négligeable de la génération d'un trafic et de trajectoires réalistes, est l'évitement de collisions entre entités. Cet évitement est étudié en détail dans l'état de l'art du chapitre suivant, mais l'utilisation de comportements locaux comme dans les simulateurs de trafic routier semble être une proposition intéressante en terme de généricité et de passage à

l'échelle.

Généricité. Un trafic est composé d'entités en mouvement, allant d'un point de départ à un point d'arrivée suivant une trajectoire. C'est particulièrement vrai dans le trafic aérien, où les avions se déplacent d'un point à l'autre, évitant les autres avions grâce au contrôleur. C'est pourquoi, comme le suggèrent Esawey et Sayed dans [El Esawey and Sayed, 2011], notre travail se concentrera sur la génération de trajectoires afin de générer un scénario.

Notre modèle utilisera également l'attribution de rôles comme dans [Bonakdarian et al., 1998] [Wassink et al., 2006] [Olstam et al., 2011] pour générer les situations requises. En simulation, la représentation d'entités mobiles par des agents permet une description naturelle et décentralisée du problème, que l'on veut étendre à l'attribution des rôles et à la génération de scénarios. Enfin, notre génération d'un scénario se focalisera sur la génération de trajectoires.

Conclusion. En conclusion, le système présenté devra :

— Générer un scénario, c'est-à-dire un ensemble de trajectoires,

— Apporter des comportements d'adaptation aux différentes entités mobiles, pour leur permettre de s'ajuster à des modifications de l'environnement, notamment l'évitement entre entités mobiles, mais aussi à des modifications du scénario,

— S'inspirer de la décentralisation observée dans [Bonakdarian et al., 1998] [Wassink et al., 2006] [Olstam et al., 2011], en prévoyant une adaptation lors de la simulation mais aussi lors de la génération du scénario,

— Effectuer un couplage entre la structuration lors de la simulation et lors de la génération du scénario [Signor et al., 2004], notamment pour améliorer l'adaptation,

— Utiliser des données pour générer des densités réalistes et des comportements contextuels, dans le but d'apporter du réalisme à la simulation, aussi bien en terme de trajectoires que de trafic.

Dans la suite, nous cherchons à donner des comportements d'adaptation au entités mobiles, pour qu'elles s'adaptent à un ensemble d'objectifs multi-niveaux et multi-critères. L'un des comportements les plus importants est l'évitement de collisions, et il sera le cœur de l'état de l'art du chapitre suivant (chapitre 4).

4 État de l'art sur l'évitement de collisions

Nous généralisons la génération d'un scénario et son adaptation au cours de la simulation en considérant qu'un scénario de trafic est un ensemble de trajectoires. Générer un scénario de trafic correspond alors à planifier un ensemble de trajectoires. Ainsi, l'obtention de *situations* dans un trafic se fait par la construction de trajectoires qui respectent des contraintes données par les *situations*.

Dans la suite, nous cherchons à donner des comportements d'adaptation aux entités mobiles, pour qu'elles s'adaptent à un ensemble d'objectifs multi-niveaux et multi-critères. L'un des comportements le plus important est l'évitement de collisions. Le but est d'ensuite généraliser ces comportements d'évitement pour que l'entité mobile puisse s'adapter à d'autres objectifs.

Nous faisons dans un premier temps une présentation du domaine, en particulier des notations et des concepts utilisés dans la planification de trajectoire(s) (section 4.1). Nous les utiliserons pour présenter un état de l'art historique de la planification d'une trajectoire pour des entités mobiles de manière générale (section 4.2), et ensuite présenter l'évitement de collisions dans le domaine aérien (section 4.3).

4.1 La planification de trajectoire(s)

Nous présentons dans un premier temps les concepts qui seront utilisés dans la suite de cet état de l'art, ainsi que leurs notations associées. La majorité des concepts sont tirés de [Latombe, 2012], qui a unifié la plupart des concepts de planification de trajectoires autour du concept d'**espace des configurations**. Ce concept d'espace des configurations vient de la physique Lagrangienne, dans la planification de trajectoires. Ce concept se trouve notamment dans le travail de Lozano-Perez [Lozano-Pérez and Wesley, 1979] [Lozano-Perez, 1983], et plus avant dans celui d'Udapa [Udupa, 1977].

4.1.1 Concepts et notations

Mobiles. Les entités mobiles sont notées M_i. L'ensemble des n entités mobiles est noté M, tel que $M = \left\{M_i\right\}_{i \leq n.}$.

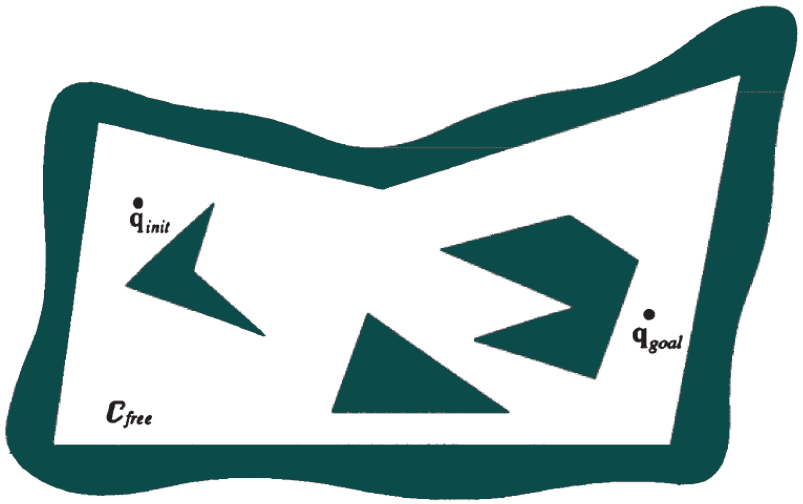


Figure 4.1 – Représentation de l'espace des configurations C, de l'espace des configurations occupé $CObs$ en couleur, et des configurations initiale et objective (q_{init} et q_{goal}) [Latombe, 2012]

Espace. L'espace dans lequel peut évoluer l'entité mobile : son environnement, est noté W pour *Workspace* en anglais. En général, $W = \Re^n, n \in \{2,3\}$.

Obstacle. L'ensemble des m obstacles dans W est noté O. Chaque obstacle sera noté O_j, de telle sorte que $O = \{O_j\}_{j \leq m.}$

Espace des configurations. L'espace des configurations d'une entité mobile, est un espace d'états pour la planification de chemins ou de trajectoires, habituellement noté C. Il correspond à l'ensemble des configurations q_i que peut prendre une entité mobile. Dans notre cas où les mobiles sont rigides, c'est-à-dire que leur structure ne change pas, q_i est un triplet contenant la position du mobile, son vecteur vitesse et son orientation. La notation est la même dans la littérature pour la planification de trajectoires de n mobiles. Une configuration représente alors le n-uplet des différentes positions et orientations des n mobiles.

Nous notons $CObs$ l'espace des configurations occupé par des obstacles O_j, et de manière complémentaire nous notons $Cfree$ l'espace des configurations sans obstacles. Ainsi $C = Cfree \cup CObs$, avec bien entendu $Cfree \cap CObs = \emptyset$

Dans un problème de planification, la configuration de départ est notée q_{init} et celle d'arrivée est notée q_{goal}.

Planification de chemins et planification de trajectoires. Nous distinguons dans ce qui suit la planification de chemins, *path planning* de celle de trajectoires, *trajectory planning*.

— La planification de chemins, *path planning*, correspond à chercher un chemin dans l'espace des configurations reliant q_{init} et q_{goal}. Cette recherche ne tient pas compte de la vitesse à laquelle le mobile doit suivre ce chemin, ni des lois de contrôle qui lui permet-

tront de le suivre. Un chemin peut être représenté comme une fonction c de $\left[0,1\right]$ dans $Cfree$ avec $c(0) = q_{init}$ et $c(1) = q_{goal}$

— La planification de trajectoires, *trajectory planning*, consiste à chercher une trajectoire reliant q_{init} et q_{goal}. Elle revient à chercher une loi de contrôle (simple ou complexe) permettant au mobile de rejoindre sa destination, en respectant éventuellement ses contraintes différentielles. De manière plus formelle, la planification devient une planification de trajectoires quand la paramétrisation de la fonction c est interprétée comme le temps : nous avons alors $c(s(t)) : \left[0,1\right] \rightarrow Cfree$, avec $s(0) = t_{start}$ et $s(1) = t_{end}$.

Contraintes différentielles. Les contraintes différentielles sont toutes les contraintes qui s'appliquent aux mouvements d'un mobile M_i, comme une restriction sur la vitesse, ou sur la vitesse de rotation. Par exemple, un avion de ligne n'est pas capable d'effectuer une montée verticale de manière soutenue comme une fusée.

Métrique. Dans la majorité des algorithmes qui suivent, les notions de métrique et de pseudo-métrique sont beaucoup utilisées. De manière simplifiée, une métrique est une fonction qui donne une distance entre deux éléments d'un ensemble.

Plus formellement, une métrique d sur un ensemble E est une fonction définie de ExE dans l'ensemble des réels positifs.

$$d : ExE \rightarrow \Re^+$$

Cette fonction doit vérifier trois conditions pour tout x, y, z dans E :

1. $d(x,y) = 0 \Leftrightarrow x = y$ (identité des indiscernables)
2. $d(x,y) = d(y,x)$ (symétrie)
3. $d(x,z) \leq d(x,y) + d(y,z)$ (inégalité triangulaire)

Une métrique usuelle est la distance euclidienne sur $\Re^2$ qui à toute paire de couples $(x_1,y_1),(x_2,y_2) \in \Re^2$ associe le réel $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$.

Pseudo-Métrique. Définir une métrique dans un espace E s'avère parfois difficile. Dans ce cas, il est possible de définir une pseudo-métrique, qui ne respecte pas toutes les conditions d'une métrique. Ces pseudo-métriques sont souvent utilisées pour définir le coût de différentes actions, afin de choisir la meilleure.

Par exemple, prenons une voiture dont la configuration correspond à sa position (x,y) et sa vitesse v. Partant d'une configuration $q_{init} = (A, 0km.h^{-1})$, la voiture doit atteindre la configuration $q_{goal} = (B, 0km.h^{-1})$. La voiture peut effectuer le voyage à $100km.h^{-1}$ ou $10km.h^{-1}$. Dans le premier cas, elle consomme beaucoup plus, mais dans l'autre elle met beaucoup plus de temps. Les algorithmes peuvent alors définir une pseudo-métrique pour concilier les deux aspects, et cette pseudo-métrique ne respecte pas la symétrie.

Autre exemple, l'espace des configurations contient des paramètres que le concepteur ne veut pas prendre en compte (parce qu'ils ne sont pas utiles pour la planification, ou par souci de simplification). Par exemple, l'espace des configurations contient la température du

moteur de la voiture. Il est possible alors d'utiliser une pseudo-métrique qui ne tienne pas compte de ce paramètre, et ne respecte donc pas l'identité des indiscernables.

Roadmap. Une *roadmap* est un graphe topologique extrait de l'espace des configurations. Chaque nœud du graphe est une configuration $q_i \in Cfree$. Deux configurations du graphe sont liées par un arc si un chemin ou une trajectoire entre les deux configurations a été trouvé. Dans le cas des trajectoires, le graphe est orienté [Siméon et al., 2000].

Nous présentons succinctement dans la suite les débuts de la planification de chemins et de trajectoires, pour ensuite présenter la planification de la trajectoire d'une entité mobile de manière générale.

4.1.2 Du discret au continu

La planification de chemins a majoritairement commencé sur des espaces discrets. Le déplacement d'un seul mobile dans un environnement statique où l'ensemble des configurations se résume à un ensemble fini dénombrable n'est pas le cœur de cet état de l'art. Néanmoins, ces méthodes ont aujourd'hui encore une forte influence sur les méthodes développées. Nous en faisons donc un rapide tour d'horizon.

Un tel environnement correspond dans la plupart des cas à une approximation de l'environnement comme étant un graphe. Par exemple, un labyrinthe s'apparente à ce type d'environnement si l'on cherche uniquement à en trouver la sortie. Le labyrinthe est alors équivalent à un graphe où chaque nœud est une intersection.

Les algorithmes les plus connus de planification de chemins sont sur des graphes orientés, comme les algorithmes de Bellman-Ford-Moore [Bellman, 1958] et celui de Dijkstra [Dijkstra, 1959], ou encore les algorithmes A* [Hart et al., 1968] et D* [Stentz, 1994].

Le problème de planification change beaucoup lorsque nous passons d'un monde W discret à un monde W continu, comme $\Re^2$ ou $\Re^3$. Néanmoins l'efficacité des algorithmes dans le monde discret a beaucoup orienté certaines solutions dans le continu. En effet, beaucoup de travaux se sont focalisés sur la création d'une *roadmap*, notamment pour réutiliser ces algorithmes.

4.1.3 La complexité de l'espace continu

Il est intéressant de noter que la distinction entre domaine continu et domaine discret n'implique pas, dans le cas général, que résoudre un problème continu est plus difficile que résoudre un problème discret. Certains problèmes discrets peuvent être NP-complet dans le cas général, tandis que certains problèmes continus peuvent être résolus en un temps polynomial. Cette étonnante propriété vient du fait que la difficulté d'un problème discret est d'explorer de manière efficace l'espace de recherche, alors que pour les problèmes continus, la difficulté vient plus de la forme de l'espace de recherche. Dans des problèmes d'optimisation continue simples, l'espace de recherche est très régulier et l'optimum facile à trouver.

Néanmoins, que ce soit par l'expérience des différents auteurs qui s'y sont confrontés, ou par des démonstrations dans des cas spécifiques, le problème de planification de trajec-

toires est généralement accepté comme $NP - Hard$. Le problème général de planification de trajectoires a été prouvé en 1979 comme étant $PSPACE - hard$ [Reif, 1979].

L'introduction de la continuité, si elle ne change pas nécessairement la complexité du problème, requiert un changement dans la façon d'aborder le problème. Deux philosophies majoritaires s'adressent au problème de planification de trajectoires dans un espace continu. La première famille crée une *roadmap* et utilise celle-ci pour planifier le chemin ou la trajectoire, tandis que la deuxième famille génère la trajectoire de manière itérative en utilisant une fonction de navigation.

Nous présentons dans la suite de ce chapitre les différentes méthodes de planification de trajectoires pour des entités mobiles de manière générale.

4.2 Planification d'une trajectoire pour une entité mobile

Les systèmes de planification d'une trajectoire peuvent globalement se diviser en deux familles. La première famille construit l'espace des configurations sans obstacles $Cfree$ en utilisant la connaissance de l'ensemble des obstacles O, pour ensuite construire une *roadmap* sur cet espace et enfin planifier une trajectoire. La deuxième famille ne construit pas l'espace $Cfree$, mais soit construit une *roadmap* par échantillonage, soit navigue itérativement dans l'espace des configurations C. Dans la figure 4.2, nous proposons une classification plus détaillée.

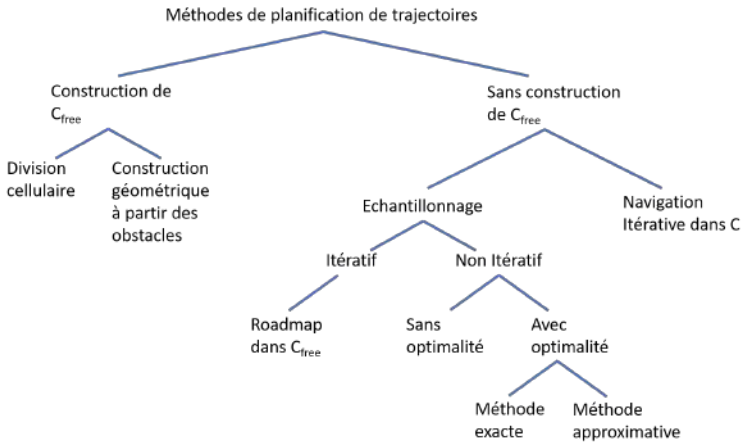


Figure 4.2 – Classification des méthodes de planification de trajectoires

Cette section permet de présenter les principes de la planification de trajectoires, en illustrant certains concepts vus précédemment, mais surtout de souligner des algorithmes que l'on ne retrouve pas ou peu dans l'aviation, mais qui pourraient servir dans un cas plus générique. La suite de cette section concerne dans un premier temps les algorithmes qui construisent l'espace $Cfree$ en utilisant la connaissance de l'ensemble O (sous-section 4.2.1), puis les algorithmes qui planifient une trajectoire suite à la construction d'une *roadmap* de $Cfree$ par échantillonage (sous-section 4.2.2), et enfin les algorithmes qui naviguent itérativement dans l'espace des configurations C pour la planification des trajectoires (sous-section

4.2.3).

Les méthodes qui construisent une *roadmap*, planifient par la suite simplement la trajectoire en faisant une requête sur cette *roadmap*.

4.2.1 Planification par construction d'une *roadmap* en connaissant Cfree

Les algorithmes qui suivent, s'appuient sur la connaissance de l'espace des configurations C et de l'ensemble des obstacles O pour construire l'espace des configurations libres $Cfree$ et une *roadmap* associée. La planification d'une trajectoire à partir de la *roadmap* étant simple, nous nous focalisons dans la suite uniquement sur la construction de la *roadmap*.

Nous pouvons globalement diviser ces algorithmes en deux catégories. La première catégorie divise l'espace des configurations C en cellules (*Cell Decomposition* dans la littérature), et ensuite utilise ces cellules pour obtenir une *roadmap*. La seconde catégorie utilise la connaissance de l'ensemble des obstacles O pour créer directement la *roadmap*.

Ces algorithmes traitent majoritairement de la planification de chemins ou de trajectoires à travers des obstacles statiques, nous éloignant de notre sujet principal de structuration d'un trafic, qui évolue dans un espace ou les obstacles sont majoritairement les autres entités mobiles, et donc mobiles. Par conséquent, nous présentons que très succinctement les principes de ces approches dans les deux paragraphes suivant et indiquons d'autres références plus complètes au lecteur : [Souissi et al., 2013] [Preparata and Shamos, 2012].

Découpage de l'espace en cellules - Cell Decomposition. D'une manière générale, l'algorithme cherche à décomposer l'espace des configurations libres $Cfree$ en cellules de certaines tailles et de certaines formes, éventuellement en modifiant la taille ou la forme en fonction des algorithmes ; le découpage est ensuite utilisé pour créer une *roadmap*. Pour effectuer ce découpage en cellules, il est par exemple possible de découper l'espace des configurations C verticalement (ou horizontalement) en traçant une droite verticale (ou horizontale) à partir de chaque sommet (les obstacles), qui s'arrête lorsqu'elle rencontre un autre obstacle, comme dans la figure 4.3. Il est aussi possible d'utiliser le principe des *quadtree*, de diviser l'espace en quatre cellules, et de procéder ainsi récursivement sur chaque cellule tant que la précision n'a pas été atteinte ou que la cellule ne contient pas uniquement une portion d'obstacle ou une portion d'espace libre, comme le montre la figure 4.4.

La *roadmap* est ensuite créée en mettant des noeuds au centre de ces cellules, et/ou au centre des côtés des cellules, et en reliant ces noeuds.

Création d'une roadmap à partir de l'ensemble des obstacles O. Nous retrouvons dans cette catégorie la *Visibility Roadmap*, cette fois sans échantillonnage [Lozano-Pérez and Wesley, 1979]. Le principe est de considérer chaque extrémité des obstacles (éventuellement un point légèrement éloigné de l'obstacle pour considérer des entités mobiles qui ne sont pas seulement de simples points) comme un point de vue, qui est un noeud du graphe, et de connecter les noeuds entre eux si un chemin ou une trajectoire directe existe.

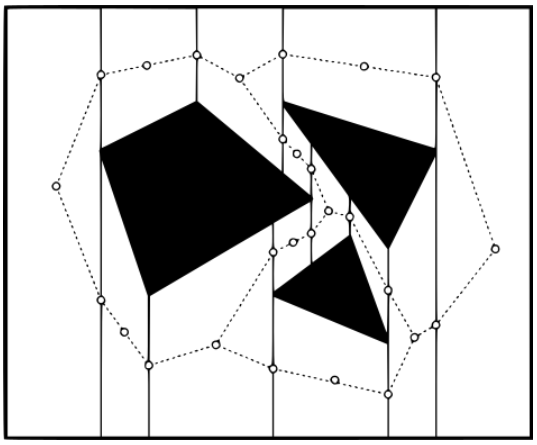


Figure 4.3 – Découpage vertical de l'espace des configurations libre par des trapèzes et la *roadmap* qui en résulte [De Berg et al., 2008]

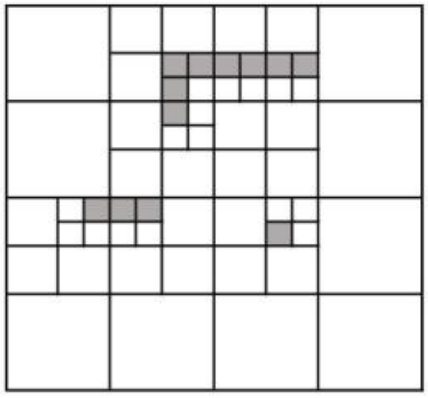


Figure 4.4 – Découpage de l'espace des configurations libre par un *quadtree* de hauteur 3 [Goerzen et al., 2009] (les cellules grises sont des cellules avec obstacles)

Nous retrouvons aussi une utilisation des diagrammes de Voronoï dont le principe est de découper l'espace par des lignes situées à égale distance de chaque obstacle. Les noeuds de ce diagramme peuvent constituer alors des noeuds pour une *roadmap* [Aurenhammer, 1991].

4.2.2 Planification de trajectoires par construction d'une *roadmap* par échantillonnage

Nous présentons les principaux algorithmes qui utilisent l'échantillonnage pour construire une *roadmap*. Nous renvoyons le lecteur vers [Elbanhawi and Simic, 2014] pour une présentation plus complète des différents algorithmes existants.

L'échantillonnage correspond à la première idée pour pallier la problématique du passage d'un espace des configurations C discret à un espace des configurations C continu. En effet, il suffirait d'échantillonner l'espace des configurations C avec une grille ré-

gulière afin d'adapter les algorithmes de planification de chemins (Bellman-Ford-Moore [Bellman, 1958], Dijkstra [Dijkstra, 1959], A* [Hart et al., 1968], D* [Stentz, 1994]...) sur l'espace discret échantillonné.

Étant donné que l'espace des configurations sans obstacles $Cfree$ n'est pas construit à l'avance, il est possible que la grille d'échantillonnage soit trop petite et qu'aucune solution ne soit trouvée. Une solution est d'augmenter la taille de la grille tant qu'une solution n'a pas été trouvée. Mais, si l'échantillonnage est trop important, bien que l'algorithme trouve théoriquement la solution, il met en pratique trop de temps pour que son utilisation soit intéressante.

Il est possible de calculer la taille minimale de la grille d'échantillonnage pour trouver une solution, mais ce calcul nécessite de connaître à l'avance l'espace des configurations sans obstacles $Cfree$, ce qui est une opération coûteuse en temps. Par conséquent, les algorithmes que nous présentons dans la suite construisent des échantillonnages qui ne sont pas basés sur des grilles.

4.2.2.1 Planification de trajectoires par construction d'une *roadmap* par échantillonnage itératif

Principe général des algorithmes. Les algorithmes par échantillonage itératif fonctionnent dans la grande majorité de la même façon, que nous résumons dans l'algorithme 4.1. Dans cet algorithme, la *roadmap* est considérée comme un graphe $G(V, E)$, construit itérativement en ajoutant des configurations à son ensemble de noeud V (vertices) et en connectant les noeuds par des arrêtes souvent orientées qui représentent des trajectoires dans l'ensemble E (edges).

Cette construction se fait en prenant un échantillon, c'est-à-dire une configuration q_i de l'espace des configurations C, pour ensuite tester, le plus souvent par un algorithme de détection de collisions externe au système, si $q_i \in Cfree$, et si c'est le cas tenter de connecter cette configuration au reste du graphe. Ces algorithmes planifient en fait des parties de trajectoires entre des configurations temporaires de départ $q_{temp,init}$ et d'arrivée $q_{temp,goal}$, et utilisent ensuite ces parties de trajectoire pour planifier une trajectoire complète entre les véritables configurations de départ q_{init} et d'arrivée q_{goal}. Autrement dit, l'algorithme ne construit pas l'espace des configurations sans obstacles $Cfree$, mais teste si chaque configuration échantillonnée et chaque trajectoire planifiée est sans collision, sans pour autant calculer tout l'espace (ce qui est considéré comme bien plus coûteux).

La *roadmap* ainsi construite est par la suite utilisée pour planifier les trajectoires. En général, cette planification est simple, et elle est la condition d'arrêt de beaucoup des algorithmes présentés ici.

La différence dans les algorithmes réside majoritairement dans le choix des configurations $q_{temp,goal}$ et $q_{temp,init}$, c'est-à-dire dans les étapes 3 et 4 de l'algorithme 4.1, qui sont parfois inversées.

Présentation des algorithmes. Les auteurs de [Overmars, 1992][Bohlin and Kavraki, 2000] présentent l'algorithme appelé *(Lazy) Probabilistic RoadMap (PRM)*. Dans cet algorithme, le

Algorithme 4.1 : Échantillonnage itératif

1: Le graphe G est initialisé avec l'ensemble des noeuds V contenant les noeuds q_{init} et q_{goal} et l'ensemble des arrêtes E vide
2: **tant que** Pas de trajectoire trouvée entre q_{init} et q_{goal} **faire**
3: Choisir une configuration $q_{temp,goal}$ dans $Cfree$ (éventuellement en tirant aléatoirement dans l'espace des configurations C tant que le tirage n'est pas déjà dans un obstacle), qui appartient ou n'appartient pas à V.
4: Choisir une configuration $q_{temp,init}$ dans V.
5: Essayer de construire une trajectoire T entre $q_{temp,init}$ et $q_{temp,goal}$ contenue dans $Cfree$.
6: **si** La planification a réussi **alors**
6: Ajouter la configuration $q_{temp,goal}$ à V et la trajectoire liant les deux configurations à E.
7: **fin si**
8: **fin tant que**

tirage de la configuration $q_{temp,goal}$ se fait de manière aléatoire. L'algorithme vérifie ensuite si cette configuration est dans $Cfree$ en vérifiant l'absence de collisions. Si la configuration $q_{temp,goal}$ est dans $Cfree$, il cherche la configuration $q_{temp,init} \in V$ la plus proche et tente de lier les deux configurations par une trajectoire T. Dans tous les cas, il ajoute la configuration $q_{temp,goal}$ dans V, et s'il réussit à planifier T, il place T liant $q_{temp,init}$ et $q_{temp,goal}$ dans E.

Plusieurs variantes du tirage aléatoire sont possibles :

1. tirer aléatoirement une configuration $q_{temp,goal}$ tant que celle-ci n'est pas dans $Cfree$ (tant que celle-ci n'est pas en collision avec un obstacle),
2. tenter de déplacer la configuration $q_{temp,goal}$ dans $Cfree$. Par exemple, dans une direction aléatoire (utile quand l'espace contient beaucoup d'obstacles),
3. diriger le tirage aléatoire par les configurations déjà tirées (pour s'en éloigner, ou s'en rapprocher).

La méthode est étendue dans [Kavraki, 1994] [Kavraki et al., 1996], pour permettre plusieurs planifications de trajectoires. La méthode explore alors plus l'espace des configurations C que le juste nécessaire pour planifier une trajectoire entre les configurations q_{init} et q_{goal} (PRM). De plus, elle permet aussi de reprendre l'exploration de l'espace des configurations C si celle-ci n'est pas suffisante. La méthode est étendue dans [Svestka and Overmars, 1995] pour planifier des trajectoires de plusieurs entités mobiles.

Les auteurs de [Hsu et al., 1997] présentent l'algorithme appelé *Expensive Space Planner*. L'idée de l'algorithme est de construire deux sous-graphes (SG_1 et SG_2), en partant de q_{init} et q_{goal}, et de tenter de joindre régulièrement ces sous-graphes. Focalisons-nous sur la construction d'un graphe (SG_i) :

1. à chaque étape, l'algorithme choisit la configuration $q_{temp,init} \in SG_i$ avec une probabilité inversement proportionnelle au nombre de noeuds proches de la configuration $q_{temp,init}$ dans SG_i

2. l'algorithme échantillonne ensuite le voisinage de la configuration $q_{temp,init}$, et sélectionne la configuration $q_{temp,goal}$ parmi cet échantillonnage, avec un probabilité inversement proportionnelle au nombre de nœuds dans son voisinage,
3. l'algorithme tente ensuite de connecter les deux configurations $q_{temp,init}$ et $q_{temp,goal}$ par une trajectoire, et s'il réussit, ajoute la nouvelle configuration au graphe SG_i.

Les auteurs de [LaValle, 1998] présentent l'algorithme appelé *Rapidly exploring Random Tree (RRT)*. Comme pour *PRM*, l'algorithme *RRT* tire aléatoirement une configuration $q_{temp,goal} \in Cfree$, mais n'ajoute celle-ci que si un chemin n'est pas trouvé. D'autre part, la configuration q_{goal} n'est pas ajoutée à l'initialisation ; l'algorithme tente régulièrement de la lier avec l'une des configurations de son graphe G. Les auteurs de [Kuffner and LaValle, 2000] proposent d'explorer plusieurs arbres en même temps, et d'essayer de joindre ceux-ci régulièrement, comme pour les *Expensive Space Planner*.

Les auteurs de [Siméon et al., 2000] présentent un algorithme pour construire une *Visibility Roadmap* par échantillonnage. Une *Visibility Roadmap* est une représentation simplifiée de la topologie de l'espace des configurations qui s'inspire du principe simple et réel du point de vue. Ainsi, un noeud (une configuration) de la *Visibility Roadmap* est lié à une autre configuration s'il peut "voir" cette configuration. Dans cet algorithme, il existe deux sortes de configurations retenues par l'algorithme pour construire la *roadmap* :

— les *Guards* : ce sont les points de vue de l'algorithme. Lorsque l'algorithme se termine, l'ensemble des *Guards*, noté E_{Guards}, doit normalement couvrir l'espace de configurations sans obstacles $Cfree$,

— les *Connections* : ce sont des configurations qui lient des *Guards* entre elles. L'ensemble des *Connections* est noté $E_{Connections}$.

L'algorithme tire de façon aléatoire une configuration $q_{temp,goal} \in Cfree$. Il tente ensuite de lier la configuration $q_{temp,goal}$ avec les configurations de E_{Guards}. Si l'algorithme arrive à connecter la configuration $q_{temp,goal}$ à une seule configuration de E_{Guards}, il ajoute la configuration $q_{temp,goal}$ à E_{Guards}. S'il arrive à connecter deux configurations de E_{Guards} non connectées par une configuration $q \in E_{Connections}$, il ajoute la configuration $q_{temp,goal}$ à $E_{Connections}$. Sinon il ne conserve pas la configuration $q_{temp,goal}$.

Les auteurs de [Mazer et al., 1998] présentent l'algorithme du fil d'Ariane. L'algorithme construit le graphe en explorant en priorité les configurations les plus éloignées d'un ensemble d'autres configurations déjà explorées. Pour sélectionner une configuration, ils utilisent un algorithme génétique.

4.2.2.2 Planification de trajectoires par construction d'une *roadmap* par échantillonnage non itératif

Une autre façon de procéder pour l'échantillonnage, est de considérer les configurations de départ q_{init} et d'arrivée q_{goal}, et de relier les deux par différentes trajectoires dans l'espace des configurations C, pour ensuite tester si ces trajectoires sont dans l'espace des configurations sans obstacles $Cfree$ comme le montre l'algorithme 4.2.

La planification d'une trajectoire est alors immédiate à partir de la *roadmap*.

Algorithme 4.2 : Échantillonnage non itératif

1: Le graphe G est initialisé avec l'ensemble des noeuds V contenant les noeuds q_{init} et q_{goal} et l'ensemble des arrêtes E vide
2: Construire un graphe $G(V, E)$ sans considérer les collisions
3: **tant que** Condition d'arrêt non atteinte **faire**
4: Explorer le graphe $G(V, E)$ en testant les arêtes pour vérifier les collisions
5: **fin tant que**

La condition d'arrêt dépend de l'objectif de l'algorithme. Si l'objectif est d'obtenir la meilleure solution, alors l'exploration sera exhaustive (pour prouver l'optimalité), sinon l'algorithme s'arrête lorsqu'il trouve une solution acceptable (dont le coût est suffisamment bas), ou encore dès qu'il trouve une solution.

Cette famille d'algorithmes est particulièrement représentée dans l'évitement de collisions dans le domaine aérien, puisqu'elle se prête à des preuves d'optimalité dans l'ensemble discret, et parce que le monde aérien possède relativement peu d'obstacles.

4.2.3 Planification d'une trajectoire par navigation itérative dans C

Nous présentons dans cette sous-section différentes méthodes qui font naviguer l'entité mobile dans l'espace des configurations C. L'entité mobile possède une trajectoire créée itérativement à partir de sa position actuelle, en utilisant des fonctions et des comportements qui dirigent l'entité vers sa destination en évitant les obstacles.

Les auteurs de [Carpin and Pillonetto, 2005] présentent une méthode entièrement basée sur l'aléatoire. L'algorithme tire à chaque pas une configuration de l'espace des configurations C aléatoirement dans le voisinage de la configuration actuelle, et tente de planifier une trajectoire vers cette configuration. Si la planification réussit, l'entité est déplacée vers la nouvelle configuration. Sinon, une autre configuration est tirée aléatoirement. Le tirage aléatoire est guidé par les k (nombre paramétrable) dernières configurations dont il tente d'éloigner la nouvelle configuration.

Les auteurs de [Khatib and Le Maitre, 1978] [Khatib, 1986] présentent l'algorithme basé sur des *potential fields*. L'algorithme consiste à faire évoluer l'entité mobile dans un champ de potentiel U artificiel. Le principe est de voir l'entité mobile comme une particule chargée négativement, sa destination comme une particule chargée positivement, et les obstacles comme des particules chargées négativement, et de faire ainsi évoluer l'entité mobile en suivant une descente de gradient. Les forces qui s'appliquent sur le mobile sont généralement les trois suivantes [Zeghal, 1994] :

— force répulsive : elle permet au mobile de s'éloigner des obstacles,

— force attractive : elle permet au mobile de se rapprocher de sa destination,

— force glissante : parfois ajoutée aux deux autres forces, cette dernière force correspond au glissement tangentiel du mobile sur les zones équipotentielles. Elle permet de contourner les obstacles.

Les auteurs de [Barraquand and Latombe, 1990] utilisent un algorithme se rapprochant de l'algorithme précédent, nommé *Randomized Potential Field*. Le principe de l'algorithme est d'utiliser une pseudo-métrique pour diriger l'entité mobile vers sa destination, sans lui donner d'autre pseudo-métrique (en particulier pas de force répulsive). L'algorithme fonctionne selon deux modes, le premier consiste à suivre la descente de gradient, l'autre est utilisé pour sortir des minima locaux. La descente de gradient se fait soit en tirant aléatoirement une configuration dans le voisinage, et en s'y déplaçant si le gradient y est plus faible, soit en évaluant un échantillonnage du voisinage et en se déplaçant vers le gradient le plus faible atteignable. L'algorithme peut se retrouver dans un minimum local ; dans ce cas, il effectue des changements de configurations aléatoires.

Dans le but d'éviter les minima locaux des *potential fields*, les auteurs de [Rimon and Koditschek, 1992] ont défini une fonction de navigation, utilisant aussi une fonction de répulsion des obstacles β et une fonction d'attraction γ_k, mais en les composant dans une fonction de conditionnement σ_λ. La formule de la fonction de navigation étant :

$$\phi_k(q) = \frac{||q - q_{goal}||^2}{||q - q_{goal}||^{2k} + \beta(q)^2}$$

L'algorithme descend lui aussi le gradient de cette fonction.

4.2.4 Conclusion sur la planification d'une trajectoire

Nous avons présenté dans cette section des systèmes de planification d'une trajectoire, principalement dans le but de présenter le principe général des approches. La présentation n'a pas montré les apports d'autres méthodes notamment pour gérer la dynamique, puisque ces méthodes, à part la planification d'une trajectoire par navigation itérative dans l'espace des configurations C et par échantillonage non itératif, sont peu adaptées à la planification lorsque le nombre de degrés de liberté devient trop important, ce qui est le cas lors de la planification de nombreuses trajectoires.

En effet, les systèmes de construction de *roadmap* par échantillonnage n'exploitent pas les connaissances sur les obstacles (ou les autres avions) et leur trajectoire, ce qui est préjudiciable lors de la recherche de solutions.

De plus, les systèmes qui créent des *roadmaps* à partir de l'espace des configurations sans obstacles $Cfree$ sont quant à eux peu adaptés à des obstacles mobiles, autrement que par des modifications de vitesse dans certains cas.

Enfin, les systèmes à base de création de champs de potentiels sont peu adaptés à de nombreuses entités mobiles car le calcul des champs est coûteux.

Nous présentons dans la section suivante les systèmes de planification de trajectoires, appliqués aux trafics d'avion, qui ont été construits pour répondre à la problématique du nombre d'entités mobiles et des contraintes liées à l'aviation. Dans cette section, chaque avion a une trajectoire prévue, qui est souvent considérée comme une ligne droite, mais cette trajectoire ne considère pas l'ensemble des autres avions. Par conséquent, la section suivante parle d'évitement de collisions entre avions, avec suivi de la trajectoire prévue.

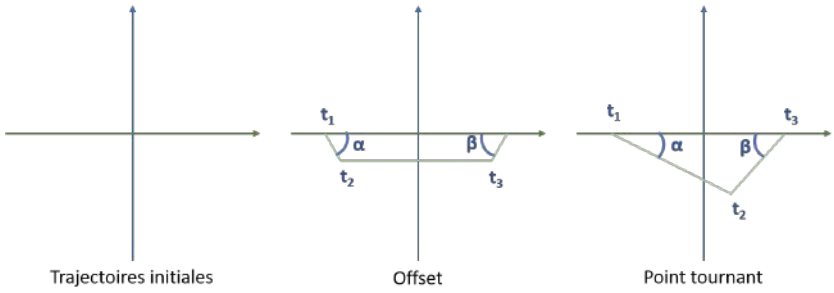


Figure 4.5 – Manœuvres d'offset et de point tournant

4.3 Évitement de collision dans le domaine aérien

Nous présentons dans cette section différentes approches pour la planification de trajectoires d'avions. Nous pouvons diviser les algorithmes présentés en deux familles. La première famille effectue un échantillonnage non itératif de l'espace des configurations C pour ensuite explorer cet échantillonnage par différentes méthodologies, et en déduire une planification de trajectoires (section 4.3.1). La deuxième famille fait évoluer les entités mobiles en suivant une heuristique pour éviter les collisions et diriger l'entité mobile vers sa destination (section 4.3.2).

Vocabulaire préliminaire Plusieurs de ces méthodes utilisent des modifications de trajectoires proches de celles effectuées par des contrôleurs, dites proches du milieu opérationnel, qu'ils qualifient de manœuvres (d'évitement). Dans le plan, ces manœuvres peuvent être :

— un "*offset*", qui correspond à un décalage d'un avion parallèlement à sa trajectoire prévue pendant un certain temps (ou une certaine distance),

— un point tournant, qui correspond à un éloignement de la trajectoire en suivant un certain cap pendant un certain temps, puis un retour à la trajectoire en suivant un nouveau cap.

Ces deux manœuvres d'évitement sont représentées dans la figure 4.5. Dans la manoeuvre d'"*offset*", l'avion commence à s'éloigner de sa trajectoire initiale au bout d'un temps t_1, en suivant une déviation par rapport à sa trajectoire initiale d'un angle α. Il tourne puis commence à avancer en parallèle par rapport à sa trajectoire initiale à partir de t_2. Puis, à t_3, il rejoint sa trajectoire avec un angle β.

Dans la manoeuvre de point tournant, l'avion effectue une manoeuvre semblable, sans se mettre en parallèle et en retournant directement à sa trajectoire initiale.

4.3.1 Échantillonnage non itératif de l'espace des configurations C

Contrairement à la planification de trajectoires présentée dans la section précédente, nous trouvons de nombreux algorithmes qui échantillonnent l'espace des configurations C non itérativement. Ces algorithmes explorent ensuite cet échantillonnage, pour trouver les trajectoires sans collisions. Dans le cas des algorithmes présentés dans cette section, ces

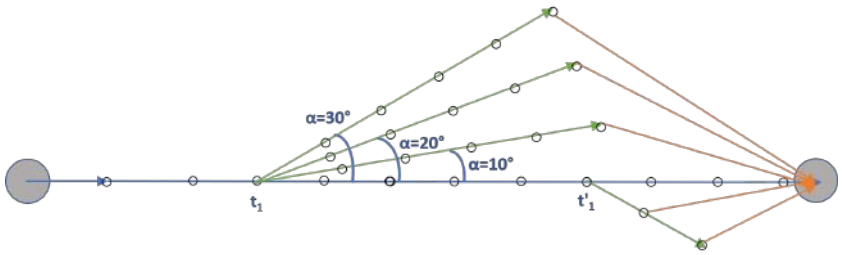


Figure 4.6 – Exemple de discrétisation d'une manœuvre de point tournant avec discrétisation de l'angle α (l'angle α peut prendre les valeurs -30, -20, -10, 0, 10, 20, ou 30 degrés), du temps de manière régulière (pour t_1 et t_2), avec un retour direct à la destination à la fin de la manœuvre (β et t_3 sont donc implicites). Le graphe ici n'est pas complet dans un souci de clarté, puisque le nombre de manoeuvres reste important

algorithmes cherchent à optimiser la trajectoire selon un ou plusieurs critères prédéfinis (comme la consommation en carburant), en trouvant un optimum local ou global parmi les trajectoires échantillonnées. Cette optimisation se fait selon deux approches, la première par des méthodes "exactes", l'autre par des méthodes "par approximation", c'est-à-dire des heuristiques ou des méta-heuristiques.

L'échantillonnage qui est effectué, ne se fait pas à partir d'une grille d'échantillonnage, mais par construction d'un graphe partant de la position de départ de mobiles, et arrivant à la position d'arrivée. Dans la grande majorité des cas, le graphe est construit en discrétisant les déplacements de l'entité mobile par des manœuvres, en discrétisant aussi ces manœuvres, et dans la plupart des cas en discrétisant aussi le temps (voir la figure 4.6). Dans ces algorithmes, en général une seule manœuvre est autorisée par entité mobile, ce qui, combiné avec la discrétisation des manœuvres, réduit de manière significative l'espace de recherche.

Nous présentons dans la suite de cette sous-section les différents algorithmes, en commençant par les méthodes exactes (section 4.3.1.1), pour ensuite présenter celles par approximation (section 4.3.1.2).

4.3.1.1 Méthodes exactes

Nous décrivons dans cette section des méthodes exactes utilisées pour la planification de trajectoires. Ces méthodes échantillonnent l'espace des configurations C par un ensemble de manoeuvres, qu'elles associent dans un graphe, pour ensuite explorer exhaustivement ce graphe afin de trouver la solution optimale. Elles diffèrent par la typologie de leur graphe, et par la méthode d'exploration de ce graphe. Nous présentons dans un premier temps des méthodes utilisant la programmation linéaire mixte en nombres entiers, pour ensuite présenter des méthodes utilisant la programmation par contraintes.

Programmation linéaire mixte en nombres entiers. L'auteur de [Lehouillier, 2015] propose de résoudre le problème de planification de trajectoires avec évitement en utilisant un algorithme pour un problème de clique maximale avec coût minimal. L'idée est de trou-

ver des trajectoires (le mieux étant bien sûr de trouver des trajectoires pour tous les avions, clique maximale), le tout en minimisant une fonction de coût (de coût minimal). L'algorithme discrétise un ensemble de manœuvres (changement de vitesse, changement de cap) en considérant que celles-ci commencent dès le début, et, dans un premier temps, sans calculer pendant combien de temps cette manœuvre est exécutée (sans calculer t_2, voir figure 4.5). Il associe un nœud à cette manœuvre sous spécifiée, et calcule pour chaque paire de nœud non voisin (pour des avions différents), si cette paire de manœuvre est sans conflit. Si oui, l'algorithme lie les deux nœuds par une arête et garde en mémoire le temps de fin de la manœuvre. Pour obtenir la clique de taille maximale, l'algorithme utilise la programmation linéaire mixte en nombre entier (Mixed-Integer Linear Programing (MILP)). Lorsque les cliques de tailles maximales sont obtenues, le coût de chaque manœuvre est calculé, et la clique de coût minimal est sélectionnée.

Les auteurs de [Lehouillier et al., 2017] améliorent l'algorithme précédant en introduisant des manœuvres verticales, en modifiant la fonction de coût et en utilisant un algorithme pour diminuer le nombre de nœuds. L'algorithme est testé pour un maximum de 60 avions dans le plan. Les mêmes auteurs introduisent l'incertitude sur les prévisions de vent, sur les mesures de vitesse et sur le temps d'exécution de la manœuvre.

Programmation par contraintes. Les auteurs de [Allignol et al., 2013] présentent un algorithme de résolution de conflits utilisant la programmation par contraintes. L'algorithme prend en compte des manœuvres de point tournant, mais en allant directement à la destination après le point tournant (voir figure 4.5). L'algorithme considère donc seulement le triplet (t_1, α, β, voir figure 4.5), plus exactement son équivalent en distance (d_1, α, d_2). Ces différents paramètres sont restreints à des valeurs discrètes, 5 valeurs pour d_1, 5 pour d_2 et 7 pour α (α peut prendre les sept valeurs -30, -20, -10, 0, 10, 20 ,30). L'algorithme explore un arbre de recherche par l'algorithme *backtracking* en utilisant l'heuristique *weigthed degree*. L'algorithme trouve la solution optimale au benchmark du rond-point (dans les conditions de manœuvres discrètes) pour 15 avions après avoir exploré l'ensemble de l'espace, et trouve des solutions pour 20 avions sans preuve d'optimalité en moins de 5 minutes (qui pour leur test est la durée maximale autorisée pour explorer exhaustivement l'espace de recherche).

4.3.1.2 Méthodes approximatives - Méta-heuristiques

Nous décrivons dans cette section des méthodes par approximation utilisées pour la planification de trajectoires. Ces méthodes optimisent une fonction de coût global, en explorant l'espace des configurations C grâce à différentes heuristiques. Nous détaillons dans un premier temps des méthodes qui réduisent l'espace de recherche par des manoeuvres d'*offset* et/ou de point tournant, en explorant alors cet espace par des méta-heuristiques (algorithme de colonie de fourmis et algorithme évolutionnaire, algorithmes génétiques). Puis nous décrivons des méthodes n'utilisant pas de manoeuvres d'*offset* ou de point tournant (algorithme génétique et recuit simulé).

Algorithme de colonie de fourmis. L'auteur de [Olive, 2006] utilise un algorithme de fourmis [Dorigo et al., 1996] généralisé pour planifier des trajectoires sans conflits. Pour ce faire, l'algorithme comporte autant de colonies de fourmis que d'avions. Dans l'algorithme, chaque départ d'avion est une colonie et chaque arrivée est une source de nourriture. Pour une colonie de fourmis, le dépôt de phéromones se fait en fonction du coût de la trajectoire, ou du nombre de conflits. Ils utilisent un seuil maximal de nombre de conflits par trajectoire qui est progressivement réduit à 0, si le seuil est dépassé aucune phéromone n'est déposée par la fourmi. Les avions peuvent effectuer des points tournants avec un angle α discret $(10, 20, 30, -10, 20, 30)$, en se dirigeant vers la destination après t_2. Les avions avancent avec des pas de temps, et peuvent décider d'effectuer une manœuvre à n'importe quel pas ; il revient ensuite vers sa destination. L'auteur utilise l'algorithme pour résoudre des ronds-points jusqu'à 29 avions. L'algorithme est repris dans [Durand and Alliot, 2009] et testé avec différents paramètres.

Hybridation : approche par intervalle et algorithme évolutionnaire. Dans sa thèse, Charlie Vanaret [Vanaret, 2015] propose un algorithme hybride utilisant deux algorithmes d'optimisation qui coopèrent entre eux. Le premier est un algorithme à évolution différentielle, qui est un algorithme à population évolutionnaire avec un opérateur de croisement probabiliste sur 4 individus. Le deuxième est un algorithme par *branch and bound* par intervalles. L'algorithme à évolution différentielle permet de donner un borne maximale de la fonction objective à l'algorithme de *branch and bound* par intervalle, et ce deuxième réduit l'espace de recherche du premier. Son application au trafic aérien utilise des manœuvres de point tournant (voir figure 4.5), où t_1, t_2 et α sont des valeurs continues, et en se dirigeant vers la destination après cette manœuvre unique. L'algorithme est testé pour résoudre des conflits à 3 avions avec une optimalité certifiée.

Algorithme génétique. L'auteur de [Durand, 1996] présente un algorithme génétique pour résoudre des conflits. Chaque avion peut uniquement effectuer des manœuvres d'*offset* et de point tournant (voir figure 4.5) avec $\alpha = \beta$. Chaque chromosome contient 4 gènes par avion, un pour chaque paramètre (α, t_1, t_2, t_3, voir figure 4.5), il contient donc $4n$ gènes en tout. Les valeurs possibles sont discrétisées, par exemple α peut prendre les valeurs -30, -20, -10, 10, 20 et 30 degrés. L'algorithme optimise des conflits allant jusqu'à 20 avions. L'algorithme est utilisé dans un simulateur pour détecter des conflits en temps réel : il détecte toutes les δ minutes (2 ou 3 minutes) les paires d'avions qui sont en conflit dans une fenêtre temporelle de Δ minutes (8 ou 12 minutes), pour ensuite les regrouper en cluster, et enfin résoudre les clusters par un algorithme génétique.

Les auteurs de [Delahaye et al., 2010][Peyronne, 2012] utilisent eux aussi les algorithmes génétiques pour résoudre des conflits. Cette fois, l'algorithme optimise des trajectoires représentées par des B-splines, qui sont une généralisation des courbes de Bézier dans le but de générer un trafic sans conflit. Les chromosomes représentent alors l'ensemble des points de contrôle des trajectoires. Les auteurs testent l'algorithme sur différents benchmarks, en particulier des ronds-points allant jusqu'à 8 avions.

Recuit simulé. Les auteurs de [Chaimatanan et al., 2013] proposent de résoudre des conflits en modifiant les temps de départ des avions, ou le profil horizontal, de manière discrète, par un algorithme de recuit simulé. La modification du profil horizontal se fait en discrétisant la trajectoire en N points, qui peuvent bouger horizontalement sous certaines contraintes. Les auteurs limitent les possibilités de l'algorithme en contraignant l'allongement de la trajectoire par une taille maximale, et les angles entre les points pour qu'ils soient suffisamment grands. Pour améliorer le trafic, l'algorithme tire aléatoirement une trajectoire parmi l'ensemble des trajectoires puis modifie celle-ci. La fonction d'évaluation est fonction du nombre de conflits.

L'auteur de [Girardet, 2014] présente un algorithme qui utilise la propagation d'une onde pour optimiser une trajectoire en fonction des vents. L'algorithme propage itérativement un front d'onde qui est influencé par le vent jusqu'à ce qu'il atteigne la destination. La trajectoire est ensuite déterminée en fonction des différents fronts d'ondes. L'auteur optimise aussi un ensemble de trajectoires pour diminuer la congestion par un algorithme de recuit simulé.

4.3.2 Évitement ans le domaine aérien par navigation itérative dans l'espace des configurations C

Les algorithmes décrits ci-après font évoluer les entités mobiles selon un pas de temps discret, et modifient lors de ces pas de temps discret la trajectoire du mobile en fonction d'une évaluation locale ou non de la situation de celui-ci. Cette modification peut être discrète (choix d'un incrément de cap par exemple) ou continue. Nous avons classé les différents algorithmes en 2 familles, selon qu'ils soient centralisés, ou décentralisés.

4.3.2.1 Méthodes centralisées

Nous décrivons dans cette section des méthodes qui font évoluer les avions selon un pas de temps discret, en modifiant la trajectoire des différents avions de manière centralisée, c'est-à-dire en regroupant la prise de décision dans une seule autorité. Nous décrivons différentes approches : une approche par programmation linéaire mixte en nombres entiers, une approche par analogie avec des ondes lumineuses, des approches par champs de potentiel et par fonction de navigation, et des approches géométriques.

Programmation linéaire mixte en nombres entiers. Les auteurs de [Pallottino et al., 2002] présentent un algorithme d'optimisation d'une trajectoire par des manœuvres latérales, par modifications de vitesses, ou par des manœuvres latérales et par modifications de vitesses. L'algorithme fait évoluer chaque avion selon un temps discret ($5min$) et résout à chaque pas la recherche de la modification minimale des trajectoires pour éviter les conflits. Pour cela, l'algorithme initialise à chaque pas l'avion comme étant dans la direction de sa destination, et calcule la déviation minimale pour chaque paire d'avions et en déduit la déviation .

Analogie. L'auteur de [Dougui, 2011] présente un algorithme d'optimisation d'une trajectoire par analogie avec la propagation d'ondes lumineuses. La lumière se déplace en suivant

des géodésiques (c'est-à-dire en suivant les trajectoires les plus courtes en temps), en passant par les milieux d'indice le plus faible (comme le vide). Par analogie, l'avion de l'algorithme évite itérativement les zones à haut niveau d'indice (les zones contenant des avions) en effectuant des manœuvres dans le plan horizontal. Pour cela, à chaque itération l'algorithme étudie un ensemble de manœuvres et décide celle qui le rapproche le plus de sa destination. L'algorithme est utilisé pour planifier plusieurs trajectoires en les planifiant séquentiellement.

Fonction Harmonique/Biharmonique. L'auteur de [Guys, 2014] propose deux algorithmes d'évitement de collisions, que nous notons dans la suite [Guys, 2014] (A) et [Guys, 2014] (B).

Le premier algorithme, [Guys, 2014] (A), utilise une fonction biharmonique pour créer un champ de potentiel sans minima locaux et planifier une trajectoire. La fonction biharmonique est ensuite utilisée pour planifier plusieurs trajectoires mais les temps de calcul ayant été jugés trop importants, l'auteur utilise une fonction de navigation harmonique pour planifier plusieurs trajectoires, [Guys, 2014] (B). L'auteur teste son algorithme pour planifier des trajectoires pour 4 et 2000 avions. L'algorithme permet de trouver des trajectoires reliant le départ et l'arrivée, mais sans garantie sur la faisabilité des trajectoires (en particulier pour les virages), ni sur l'absence de conflit (seulement sur l'absence de collisions).

Méthodes géométriques. Les auteurs de [Lin and Lee, 2015] proposent un algorithme de détection et de résolution de conflits par résolution géométrique. Les conflits détectés sont résolus à vitesse constante en calculant la modification d'angle minimale et en distribuant la modification sur les deux entités mobiles impliquées. L'algorithme est testé ensuite sur 2 drones.

Les auteurs de [Durand and Barnier, 2015] proposent une adaptation d'un algorithme d'évitement utilisé pour des robots nommé d'ORCA [Van Den Berg et al., 2011]. Le principe de [Van Den Berg et al., 2011] consiste à détecter dans un certain intervalle de temps τ les paires de conflits, et à calculer la modification minimale du vecteur vitesse pour éviter le conflit (en vitesse ou en angle donc). Du point de vue d'un mobile, cette modification minimale permet d'établir un demi-plan vers lequel son vecteur vitesse ne peut pas le diriger. Pour chaque entité mobile, un ensemble de demi-plans est calculé (par paire d'entités mobiles) et un espace dans lequel peut se trouver le vecteur vitesse d'une entité mobile est créé. Dans le cas où l'espace est vide, les demi-plans sont repoussés perpendiculairement jusqu'à obtenir une zone non vide. Si l'espace est non vide, la vitesse est choisie dans cette espace en fonction des objectifs de l'entité mobile.

Le vecteur vitesse n'est pas restreint dans [Van Den Berg et al., 2011]. Afin de prendre en compte les contraintes différentielles des avions, les auteurs de [Durand and Barnier, 2015] forcent chaque avion à choisir le vecteur vitesse dans un espace qui lui est propre, en plus des autres contraintes d'ORCA. Le vecteur vitesse choisi est celui le plus proche du vecteur vitesse optimale calculé pour la trajectoire. L'algorithme est testé avec de nombreux paramètres dans [Durand and Barnier, 2015] mais ne permet pas de planifier des trajectoires avec beaucoup d'avions, en particulier à vitesse constante. L'algorithme est utilisé dans

[Allignol et al., 2016] pour l'évitement de drones à vitesse constante. Il est ensuite modifié dans [Durand, 2018] pour que la modification de séparation soit plus forte et évite la mise en parallèle des entités mobiles, ce qui évite le conflit, mais empêche l'arrivée à destination. Prévu dans un premier temps pour être à vitesse constante, et nommé CSORCA, l'algorithme est ensuite testé en permettant des modifications de vitesses. Il permet de planifier des trajectoires sans conflit avec des modifications de vitesse jusqu'à 90 avions.

4.3.2.2 Méthodes décentralisées

Nous détaillons dans cette section des méthodes qui font évoluer les avions selon un pas de temps discret, en modifiant la trajectoire des différents avions de manière décentralisée, c'est-à-dire laissant les entités mobiles modifier leur trajectoire de manière autonome. Nous décrivons différentes approches : une approche par champs de potentiel et comportement d'essaim, une approche par fonction de navigation bipolaire, et des approches par système multi-agent.

Modified Voltage Potential. Les auteurs de [Maas et al., 2016] utilisent un algorithme décentralisé pour éviter et prévenir les conflits, utilisant une modification du champ de potentiel avec des comportements d'essaim. Dans leur algorithme, chaque drone avance en combinant le résultat de trois comportements indépendants :

— l'évitement de collisions (*Collision Avoidance*),

— l'homogénéisation de la vitesse (*Velocity Alignment*),

— la réduction de la taille de l'essaim (*Flock Centering*).

Chacun de ces comportements donne un vecteur vitesse que devrait effectuer l'avion pour satisfaire ce comportement, ces vecteurs vitesses sont ensuite sommés avec des poids pour donner le vecteur vitesse final. Chaque drone évite les collisions détectées en calculant le CPA (le point de la trajectoire où l'entité mobile est la plus proche de l'autre), puis en calculant la plus petite modification du vecteur vitesse pour éviter un conflit de manière géométrique, chaque agent (dans la paire en conflit) devant effectuer la moitié de cette modification. L'homogénéisation de la vitesse se fait par le calcul de la vitesse moyenne de l'entourage de l'avion. La réduction de la taille de l'essaim se fait de même en calculant le barycentre des avions proches. Les deux derniers comportements permettent de prévenir l'apparition de conflits en créant des flux d'avions. L'algorithme est testé pour réduire le nombre de conflits dans une zone circulaire synthétique contenant entre 50 et 350 avions sur 5 niveaux de vol.

Fonction de navigation bipolaire. Les auteurs de [Roussos et al., 2010] font le constat de l'échec des fonctions de navigation pour contrôler des véhicules avec des contraintes différentielles (comme les avions). Ils utilisent donc une fonction dipolaire, pour empêcher notamment l'apparition de rotation sur place du mobile. Chaque avion est un agent qui connaît l'ensemble des positions des autres avions, leurs orientations et leurs vecteurs vitesses, et calcule à chaque pas son déplacement avec la fonction de navigation bipolaire. L'algorithme est utilisé pour planifier des trajectoires sans conflits pour 4 avions.

Systèmes Multi-Agent. [Shandy and Valasek, 2001][Rong et al., 2002] proposent un système d'aide à la décision pour chaque avion basé sur une stratégie décentralisée à base d'agents. Les agents d'un même avion émettent des propositions de vecteurs vitesses en fonction de leurs contraintes (météorologie, collision, terrain) qui sont transmises à un agent central qui décide en fonction d'une base de manœuvres possibles la manœuvre de l'avion.

[Bicchi and Pallottino, 2000] proposent un système multi-agent ou chaque avion perçoit la position et l'objectif des avions dans une zone sphérique autour de lui. La résolution des conflits se fait en utilisant des trajectoires de Dubins (des arcs cercles). Chaque agent planifie sa trajectoire en minimisant le coût des trajectoires de tous les agents qu'il perçoit. Les auteurs testent leur algorithme avec 4 avions.

Dans sa thèse, [Breil et al., 2016] propose un système multi-agent pour résoudre les conflits par une modification de la vitesse des avions le long de leur trajectoire et la modification du réseau de route. Dans son système, chaque avion est un agent qui calcule, en fonction des messages reçus des autres agents, les conflits potentiels le long de sa trajectoire, et décide à chaque pas de temps de modifier sa vitesse ou non pour éviter les conflits détectés. Il introduit un autre agent qui détecte les zones congestionnées, et décide de modifier le réseau de routes dans le but de minimiser la congestion, et normalement de faciliter l'évitement de conflits.

4.3.3 Conclusion sur la planification de trajectoires dans le domaine aérien

Nous avons classé les différents algorithmes présentés précédemment selon sept critères :

- **Optimalité :** l'optimalité est recherchée sur une **trajectoire**, ou sur le **trafic** lui-même
  - **Trajectoire** : l'algorithme recherche soit l'optimalité de la trajectoire (*oui*), soit ne la recherche pas (*non*)
  - **Trafic :** l'optimalité sur le trafic est soit obtenue *sur un graphe*, soit *non garantie*, c'est-à-dire que l'algorithme recherche l'optimalité mais n'a pas de preuve d'optimalité, soit *sous-optimale*, c'est-à-dire que l'algorithme tel quel ne peut pas donner de solutions optimales mais tente d'optimiser les trajectoires (au moins individuellement), soit il n'y a pas de preuve d'optimalité ou de non optimalité (*non*), car l'algorithme ne recherche pas l'optimalité de manière explicite.
- **Modification de la trajectoire** : la modification de la trajectoire se fait soit par l'utilisation de *manœuvres* soit par des *modifications multiples*
  - **Manœuvres :** les manœuvres sont soit horizontales (H), soit verticales (V). Pour une illustration des manoeuvres horizontales, voir la figure 4.5. Les manoeuvres verticales correspondent uniquement à un changement de niveau de vol.
  - **Modifications multiples :** les modifications multiples sont soit discrètes (D), soit continues (Cn), c'est-à-dire que la trajectoire sera soit une suite de segments, soit une courbe. Ces modifications restent dans le plan, et peuvent être des modifications de vitesse, d'angle, ou des deux.

— **Moment de la planification** : la planification se fait soit dans une phase *amont*, c'est-à-dire avant que les avions ne partent, soit lors d'une *fenêtre temporelle*, c'est-à-dire pendant la phase de vol en planifiant entre le moment présent et un certain temps, soit de manière *dynamique*, c'est-à-dire au fur et à mesure qu'avance l'avion.

— **Incertitude** : les algorithmes gèrent l'incertitude *par la dynamique*, *par fenêtres* temporelles, et sur la *manœuvre* (début de la manœuvre, temps de la manœuvre, etc.), sur l'*heure de départ*, sur la *vitesse*, sur la *position*, ou ne gèrent pas l'incertitude

— **Retour à la trajectoire** : les algorithmes font revenir les avions sur leur trajectoire, ou les redirigent directement vers leur *destination*.

— **Passage à l'échelle** : le passage à l'échelle est soit *faible* (moins de 10 avions), soit *moyen* (entre 10 et 60), soit *fort* (plus de 60).

— **Généricité** : la généricité des systèmes est soit *faible*, soit *moyenne*, soit *forte*

Critères / Systèmes	Optimalité		Modification de trajectoire	Moment de la planification	Incertitude	Retour à la trajectoire	Passage à l'échelle	Généricité
	Trajectoire	Trafic						
[Dougui, 2011]	Oui	Sous-optimale	Modification multiples (D)	Fenêtre temporelle	Vitesse	Oui	Faible	Moyenne / Forte
[Durand, 1996]	Non	Non garantie	Manœuvre (H)	Fenêtre temporelle	Par fenêtre / Vitesse	Oui	Moyen	Faible
[Girardet, 2014]	Oui	Non garantie	Modification multiples (D)	Amont	Heure de départ	Destination	Faible	Faible
[Peyronne, 2012]	Non	Non garantie	Modification multiples (Cn)	Amont	Non	Destination	Faible	Faible
[Durand and Alliot, 2009]	Non	Non garantie	Manœuvre (H)	Amont	Non	Destination	Faible	Faible
[Chaimatanan et al., 2013]	Non	Non garantie	Modification multiples (D)	Amont	Non	Non garanti / Destination	Faible	Faible
[Lehouillier et al., 2017]	Non	Sur graphe	Manœuvre (H,V)	Amont	Manœuvre	Destination	Fort	Faible
[Allignol et al., 2013]	Non	Sur graphe	Manœuvre (H)	Amont	Manœuvre	Destination	Moyen	Faible / Moyenne
[Vanaret, 2015]	Non	Par précision	Manœuvre (H)	Amont	Non	Destination	Faible	Faible / Moyenne
[Guys, 2014] (A)	Non	Non	Modification multiples (D)	Amont	Non	Destination	Faible	Faible
[Guys, 2014] (B)	Non	Non	Modification multiples (D)	Dynamique	Par Dynamique / Non	Destination	Fort	Faible
[Pallottino et al., 2002]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Non	Destination	Moyen	Faible / Moyenne
[Lin and Lee, 2015]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Non	Destination	Faible	Faible / Moyenne
[Durand, 2018]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Non	Destination	Fort	Moyenne / Forte
[Maas et al., 2016]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Position	Destination	Non spécifié (Supposé Fort)	Faible
[Roussos et al., 2010]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Non	Destination	Non spécifié	Faible
[Shandy and Valasek, 2001]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Non	Destination	Non spécifié	Faible
[Bicchi and Pallottino, 2000]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Non	Destination	Fort	Moyenne / Forte
[Breil et al., 2016]	Non	Non	Modification multiples (D)	Dynamique	Par dynamique / Non	Oui	Fort	Moyenne / Forte

Table 4.1 – Comparaison des différents algorithmes de planifications de trajectoires dans le domaine aérien

Parmi les 19 systèmes présentés, 7 cherchent directement l'**optimalité** de la résolution pour le trafic [Lehouillier et al., 2017] [Allignol et al., 2013] [Durand, 1996] [Peyronne, 2012] [Durand and Alliot, 2009] [Chaimatanan et al., 2013] [Vanaret, 2015]. Deux autres cherchent à obtenir dans un premier temps l'optimalité des trajectoires, pour ensuite chercher à améliorer le trafic, en optimisant les trajectoires une à une [Dougui, 2011], ou en optimisant toutes les trajectoires séparément pour ensuite diminuer la congestion [Girardet, 2014], ce qui mène à des solutions sous-optimales. Les autres algorithmes cherchent uniquement à empêcher les conflits, sans rechercher l'optimalité [Pallottino et al., 2002] [Guys, 2014] (A) [Guys, 2014] (B) [Lin and Lee, 2015] [Durand, 2018] [Maas et al., 2016] [Roussos et al., 2010] [Shandy and Valasek, 2001] [Bicchi and Pallottino, 2000] [Breil et al., 2016]. Nous retrouvons globalement cette dichotomie dans les critères qui suivent. De manière générale, l'optimalité est atteinte par une centralisation des contraintes, et les méthodes décentralisées ne la recherchent pas.

La **modification de trajectoire**, lorsque le but est d'optimiser le trafic, se fait majoritairement par des manœuvres dites "opérationnelles", qui ressemblent à des ordres que les contrôleurs donneraient. C'est le cas pour 5 algorithmes [Allignol et al., 2013] [Durand, 1996] [Durand and Alliot, 2009] et [Vanaret, 2015], qui utilisent des manœuvres dans le plan, et [Lehouillier et al., 2017] qui utilise des manœuvres verticales. Ces méthodes présentent souvent des changements brusques dans les trajectoires qui sont peu réalistes. [Peyronne, 2012], [Chaimatanan et al., 2013], [Dougui, 2011] et [Girardet, 2014] sont les exceptions puisqu'ils optimisent en modifiant la trajectoire par des modifications multiples continues ou discrètes. Dans les autres cas, les modifications sont des modifications multiples discrètes, qui sont alors plus facilement adaptées aux capacités des avions. En résumé, les manoeuvres multiples semblent plus réalistes et génériques, surtout dans un contexte de simulation autonome et temps réel.

Le **moment de la planification** se fait en amont dans 7 des algorithmes, ou avec une fenêtre glissante dans 2 algorithmes. En suivant la dichotomie précédente, les autres méthodes planifient de manière réactive et dynamique [Pallottino et al., 2002] [Lin and Lee, 2015] [Durand, 2018] [Maas et al., 2016] [Roussos et al., 2010] [Shandy and Valasek, 2001] [Bicchi and Pallottino, 2000] [Breil et al., 2016]. Seule exception, [Guys, 2014] (B) planifie en amont sans avoir un but d'optimalité. De manière générale, la planification en amont semble peu appropriée pour l'évitement de collisions lorsque le système est ouvert et sujet à de nombreux imprévus.

Nous retrouvons un peu la dichotomie dans la prise en compte de l'**incertitude**, dans laquelle nous retrouvons le groupe des méthodes dites dynamiques et réactives (9 algorithmes), qui prennent en compte l'incertitude par la dynamique. Les mobiles peuvent réagir à des choses qui n'étaient pas prévues, comme une nouvelle météo, ou un recalibrage de la position qui était fausse, en s'adaptant par leur comportement [Pallottino et al., 2002] [Guys, 2014] (B) [Lin and Lee, 2015] [Durand, 2018] [Maas et al., 2016] [Roussos et al., 2010] [Shandy and Valasek, 2001] [Bicchi and Pallottino, 2000] [Breil et al., 2016]. Dans ces algorithmes dynamiques, [Maas et al., 2016] teste particulièrement l'incertitude sur la position. Dans l'autre moitié, la prise en compte de l'incertitude est plus partagée. Les deux méthodes qui utilisent une fenêtre temporelle pour optimiser apportent une prise en compte intermédiaire de l'incertitude entre les algorithmes dynamiques et ceux en amont [Dougui, 2011]

et [Durand, 1996] qui prennent en compte une partie de l'incertitude. Ces deux algorithmes introduisent aussi une incertitude sur la vitesse. Les 2 qui utilisent un graphe pour optimiser introduisent des incertitudes sur les manœuvres effectuées (début de la manœuvre, temps de la manœuvre, position de début de manœuvre...) [Lehouillier, 2015] [Allignol et al., 2013]. [Girardet, 2014] introduit une incertitude sur les départs des avions, et les 5 autres n'introduisent pas d'incertitude [Peyronne, 2012] [Durand and Alliot, 2009] [Chaimatanan et al., 2013] [Vanaret, 2015] [Guys, 2014] (A). Néanmoins, l'incertitude prise en compte traite majoritairement de petites modifications dans le trafic prévu, ce qui reste une hypothèse importante. Les systèmes dynamiques ou avec fenêtres temporelles semblent plus adaptés pour la gestion de l'incertitude, mais aussi par extension pour l'adaptation à des interactions lors de la simulation.

La quasi totalité des algorithmes ne fait pas en sorte que l'avion **retourne à sa trajectoire**, mais seulement à sa destination. Seuls [Dougui, 2011] et [Durand, 1996] obligent les avions à retrouver leur trajectoire à la fin de leur fenêtre temporelle. [Breil et al., 2016], ne modifie que la vitesse et oblige les avions à retourner à leur vitesse optimale. Cette composante est importante dans cette thèse, autant en terme de réalisme puisque le trafic aérien se construit dans la grande majorité sur un réseau de routes, mais aussi à la génération de *situations*, puisque générer des situations implique une adaptation des entités mobiles pour suivre des trajectoires données.

Le **passage à l'échelle** des différents algorithmes varie beaucoup. Les algorithmes qui passent le mieux à l'échelle sont globalement ceux qui utilisent des méthodes dynamiques. Parmi les autres, le seul algorithme qui montre un passage à l'échelle intéressant est [Lehouillier et al., 2017], grâce à des choix judicieux des paramètres d'échantillonage de l'espace des configurations C et l'utilisation de propriétés des graphes induites par l'application au trafic aérien. Les méthodes dynamiques semblent donc plus appropriées pour cette thèse où on vise la structuration d'un trafic, et donc de nombreuses entités.

La **généricité** est très majoritairement faible. Les méthodes qui s'appuient sur des algorithmes génétiques possèdent de nombreux paramètres à calibrer qui ralentissent celles-ci [Peyronne, 2012] [Durand, 1996], de même pour les algorithmes de recuit simulé [Girardet, 2014] [Chaimatanan et al., 2013], et celui utilisant des fourmis [Durand and Alliot, 2009]. Les fonctions de navigation induisent des oscillations dans la trajectoire jugées peu réalistes, des virages impossibles pour certaines entités mobiles, et généralement des allongements non nécessaires [Guys, 2014] (A) [Guys, 2014] (B) [Maas et al., 2016] [Roussos et al., 2010]. L'algorithme de [Pallottino et al., 2002] est jugé faiblement générique en raison d'une hypothèse non réaliste d'orientation vers la destination à chaque pas de résolution. Le système de [Shandy and Valasek, 2001] est très orienté avion et par conséquent faiblement générique. Les algorithmes utilisant des graphes ont une généricité faible voir moyenne, puisque la construction et la résolution des graphes peuvent être adaptées à d'autres entités mobiles, parfois difficilement [Allignol et al., 2013] [Vanaret, 2015]. En revanche [Lehouillier et al., 2017] a une généricité faible car il utilise des propriétés des graphes (comme la symétrie des manoeuvres) pour améliorer ses résultats qui ne sont pas communs à tous les domaines. De plus, ces trois différents algorithmes utilisent des changements de trajectoires brusques et peu réalistes. Les méthodes géométriques présentent une approche plus générique [Lin and Lee, 2015], en particulier [Durand, 2018].

L'approche de [Dougui, 2011] par exploration itérative d'un ensemble de modifications de trajectoires est une démarche qui peut être facilement modifiée pour d'autres entités mobiles. Enfin, les systèmes multi-agents de [Bicchi and Pallottino, 2000] et [Breil et al., 2016] apportent une décentralisation et une distribution propice à la généricité.

En **conclusion**, une majorité des algorithmes présentent des lacunes en termes de passage à l'échelle ce qui est préjudiciable pour la structuration d'un trafic aérien, et de généricité. Les méthodes décentralisées utilisant des systèmes multi-agents apportent une réponse intéressante en termes de généricité et de passage à l'échelle [Bicchi and Pallottino, 2000] [Breil et al., 2016], de même que les algorithmes géométriques [Durand, 2018]. Ces deux algorithmes sont des algorithmes réactifs, qui permettent aux différentes entités de réagir à l'état actuel de l'environnement, ce qui correspond à nos attentes liées à la structuration de simulations. L'algorithme de [Breil et al., 2016] gère plusieurs caractéristiques, comme le retour à la vitesse optimale, ce qui laisse prévoir une meilleure adéquation des systèmes multi-agents pour la généricité, le passage à l'échelle, et la structuration d'une simulation en fonction de plusieurs caractéristiques.

Dans un souci de généricité, l'évaluation d'un ensemble d'actions, comme le fait [Dougui, 2011] par l'évaluation de plusieurs déplacements, semble tout à fait appropriée et permet aussi une planification plus haut niveau.

L'optimalité est obtenue dans les différents algorithmes présentés par une centralisation. Nous émettons l'hypothèse que la relaxation des contraintes peut se faire de manière locale, par un partage d'information, non seulement sur l'état local des différentes entités mobiles, mais aussi sur leurs difficultés.

Après avoir présenté l'état de l'art sur la structuration de simulations de trafic aérien et celle de trafic routier (chapitre 3) nous avons conclu le besoin de donner des comportements d'adaptations locaux aux entités mobiles et de focaliser la génération de scénarios de trafic sur la génération de trajectoires. Nous avons donc effectué un deuxième état de l'art sur la planification de trajectoires (chapitre 4) et sur l'évitement de collisions, qui nous permet de finaliser le positionnement de cette thèse que nous présentons dans le chapitre suivant (chapitre 5).

5 Conclusion et positionnement de la thèse

Les deux état de l'art que nous avons effectués ont permis d'établir sur plusieurs axes plusieurs points faibles et points forts.

Le premier état de l'art sur le trafic aérien et le trafic routier a dégagé plusieurs points (section 3.3.2) que nous résumons ici :

— les simulateurs qui utilisent des comportements locaux, présentent une adaptation très intéressante lors de la simulation, très proche du paradigme des systèmes multi-agents (ou l'utilisent de manière explicite),

— certains systèmes ont été développés pour générer des *situations* lors de la simulation, et présentent une généricité intéressante [Wassink et al., 2006] en utilisant une métaphore du cinéma et une forte décentralisation,

— le couplage entre le scénario et la simulation n'existe pas, ce qui est un problème pour rejouer le scénario, mais aussi pour générer certaines situations particulières,

— les *situations* générées sont rarement des *situations* macroscopiques, et les mécanismes pour générer ces *situations* sont souvent limités.

Cet état de l'art a permis de motiver les principales fonctions à prendre en compte dans cette thèse :

— générer un scénario, peut être considéré comme la génération d'un ensemble de trajectoires respectant un ensemble de contraintes,

— apporter des comportements d'adaptation aux différentes entités mobiles, pour leur permettre de s'ajuster à des modifications de l'environnement, notamment l'évitement entre entités mobiles, mais aussi à des modifications du scénario,

— s'inspirer de la décentralisation observée dans [Bonakdarian et al., 1998] [Wassink et al., 2006] [Olstam et al., 2011], en prévoyant une adaptation lors de la simulation mais aussi lors de la génération du scénario,

— effectuer un couplage entre la structuration lors de la simulation et lors de la génération du scénario [Signor et al., 2004], notamment pour améliorer l'adaptation,

— utiliser des données pour générer des densités réalistes et des comportements contextuels, dans le but d'apporter du réalisme à la simulation, aussi bien en terme de trajectoires que de trafic.

A partir de ce premier positionnement, nous avons décidé de faire un état de l'art sur la planification de trajectoires et sur l'évitement de collisions dans le but de générer l'ensemble de trajectoires qu'est notre scénario, et de donner des comportements d'adaptation lors de la simulation aux différentes entités, afin qu'elles s'adaptent aux interactions, qu'elles s'évitent, et qu'elles soient capable de maintenir la génération du scénario voulu par le scénariste.

Cet état de l'art a permis de motiver les choix faits pour ce travail de thèse :

— afin de donner des comportements adaptatifs lors de la simulation aux différentes entités mobiles, nous utiliserons un système multi-agent,

— dans un souci de généricité, ces comportements s'appuieront sur un ensemble d'actions que peut effectuer l'entité mobile qui sont propres à celle-ci,

— le système devra relaxer les contraintes des différentes entités afin de chercher à satisfaire toutes les contraintes de la manière la plus optimale possible.

Les chapitres suivant décrivent les outils qu'utilisent cette thèse, qui sont la théorie Adaptive Multi-Agent System (AMAS), le pattern AMAS4Opt, et le système Environnement Virtuel Auto-Adaptatif (EVAA). Ces différents outils sont utilisé dans le système Autonomous GenerAtion of Traffic Simulations (AGATS), pour l'auto-structuration temps-réel multi-objective et multi-critère d'une simulation de trafic (chapitre 7).

Troisième partie

Une architecture pour résoudre l'auto-structuration d'une simulation de trafic : AGATS

6 Contexte et outils

Nous présentons dans ce chapitre les différents outils utilisés dans cette thèse. Nous décrivons dans un premier temps le paradigme des Systèmes Multi-Agents (section 6.1), ensuite la théorie des *Adaptive Multi-Agent Systems* et leur adéquation avec le contrôle de l'émergence et des systèmes complexes (section 6.2), et enfin deux modèles *AMAS* que nous utilisons dans la thèse, tout d'abord *AMAS4Opt* [Kaddoum, 2011] (section 6.3) pour la résolution collective de problème complexe sous-contraintes, et ensuite EVAA [Rantrua, 2017] (section 6.4) pour l'apprentissage et la simulation de trafic aérien.

6.1 Système Multi-Agent

Nous présentons dans cette section le concept de systèmes multi-agents. Un système multi-agent étant par définition composé d'agents, nous commencerons par présenter les agents (section 6.1.1), pour ensuite décrire les systèmes multi-agents (section 6.1.2), les caractéristiques de leurs environnements (section 6.1.3), les interactions du système multi-agent (section 6.1.4) et finir par la présentation des concepts de systèmes complexes et d'émergence (section 6.1.5).

6.1.1 Agent

Il existe dans la littérature de nombreuses définitions de ce qu'est un agent et des paradigmes l'utilisant. Une définition communément acceptée est celle de [Weiss, 1999] :

> "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives"

Cette définition est plus riche dans [Ferber, 1995], notamment sur la notion de localité. Nous associons généralement une liste de caractéristiques qui définissent un agent :

— un agent est autonome,

— il existe au sein d'un environnement qu'il est capable de percevoir et sur lequel il peut agir,

— il est mu par un ensemble de tendances (buts, objectifs individuels ou collectifs)

— il a une représentation partielle de cet environnement,

— il est capable de communiquer avec d'autres agents,
— il possède des ressources propres,
— il possède des compétences et peut offrir des services.

Le comportement d'un agent tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont il dispose, et en fonction de ses perceptions, de sa représentation de l'environnement et des communications qu'il reçoit. L'autonomie est souvent considérée comme l'aspect le plus important d'un agent [Di Marzo Serugendo et al., 2011]. Puisque l'agent est autonome, il est capable de prendre ses propres décisions en fonction de motivations qui lui sont propres, il est en particulier capable de refuser d'exécuter une tâche s'il la juge contradictoire avec ses objectifs, ou s'il ne peut pas l'effectuer (par manque de ressources ou de capacités).

Les décisions d'un agent sont régies par un **cycle de vie** en trois étapes distinctes, qu'il répète indéfiniment (figure 6.1) :

1. la phase de **perception** durant laquelle il observe et acquiert des informations sur son environnement,
2. la phase de **décision** durant laquelle il décide des prochaines actions qu'il va effectuer,
3. la phase d'**action** durant laquelle il effectue les actions qu'il a décidées précédemment.

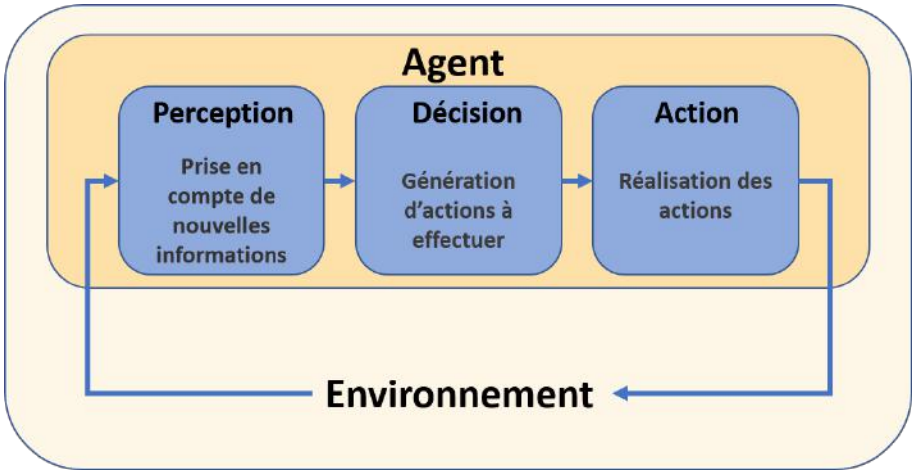


Figure 6.1 – Cycle de vie d'un agent

Un agent peut être considéré [Di Marzo Serugendo et al., 2011] :

— **Réactif / Cognitif** : un agent est dit *réactif* lorsqu'il réagit uniquement aux stimulis immédiats de son environnement, sans avoir de représentation interne de l'environnement. Au contraire, un agent est dit *cognitif* lorsqu'il dispose d'une base de connaissances importante pour réaliser ses tâches. Ces connaissances peuvent être construites au fur et à mesure par l'agent lui-même, et ne sont donc pas nécessairement données à l'initialisation de l'agent.

  Cette distinction est à prendre avec précaution, puisqu'il est rare de travailler avec des agents purement réactifs, ou purement cognitifs. Certains agents peuvent avoir une représentation interne de l'environnement, et être considérés comme des agents réactifs car leur représentation est assez simple.

— **Communiquant / Situé** : un agent est dit *communiquant* lorsque ses interactions ne dépendent pas de son positionnement dans l'environnement, ou qu'il n'a simplement pas de positionnement. Au contraire, un agent est qualifié de *situé* si ses perceptions et ses communications dépendent de son positionnement dans l'environnement.

Comme la distinction entre agent réactif et agent cognitif, la frontière entre agent communiquant et agent situé n'est pas absolue. Un agent peut, par exemple, avoir certaines interactions qui dépendent de son environnement sur un canal, et communiquer avec tous les agents sur un autre canal.

6.1.2 Système Multi-Agent

Un système multi-agent est défini comme étant un ensemble d'agents autonomes dans un même environnement, qui interagissent entre eux pour résoudre une tâche commune et cohérente telle que la résolution collective de problèmes [Di Marzo Serugendo et al., 2011], qui nous intéresse particulièrement dans le cadre de ce travail.

Dans un système multi-agent en général, les différents agents ne possèdent pas toutes les compétences, toutes les ressources, ni toutes les connaissances pour résoudre le problème individuellement. Ce sont les interactions des agents, et leur coopération qui permettent la réalisation de la tâche complète. De par la nature des agents, les systèmes multi-agents sont naturellement autonomes, et le contrôle d'un système multi-agent peut être décentralisé au niveau des agents.

Les systèmes multi-agents peuvent aussi être catégorisés en fonction de leur ouverture et de leur homogénéité :

— **Ouvert/Fermé :** dans un système multi-agent fermé, la population des agents présents dans le système n'évolue pas au cours du temps, c'est-à-dire qu'aucun agent n'est créé, ni supprimé. Au contraire, dans un système ouvert, il est possible d'en ajouter ou d'en supprimer au cours de l'exécution.

— **Homogène/Hétérogène :** dans un système multi-agent homogène, les agents sont du même type, c'est-à-dire qu'ils possèdent tous la même architecture. Dans un système hétérogène, plusieurs types d'agents existent.

6.1.3 Environnement d'un système multi-agent

Un système multi-agent est situé dans un environnement dans lequel chaque agent évolue en simultané. Cet environnement inclut en particulier ce qui est externe au système, que chaque agent peut interagir avec ou manipuler. Par exemple, si on considère une colonie de fourmis comme un système multi-agent, l'environnement correspond à la nature qui entoure cette colonie (maison, arbres, fleurs, animaux...).

Chaque agent possède son propre environnement qui est composé d'une partie de l'environnement du système multi-agent, et d'autres agents, qui sont alors dans son **voisinage**. Dans l'exemple de la colonie de fourmis, l'environnement d'une fourmi est composé de la nature et des fourmis environnantes. Le système multi-agent et l'environnement sont couplés par leurs interactions. Lorsque le système multi-agent effectue une action sur l'environ-

nement, celui-ci est modifié. La perception de cette modification agit comme un signal de retour sur le système multi-agent.

Comme les systèmes multi-agents et les agents, l'environnement d'un agent peut être caractérisé par différentes propriétés [Jonathan et al., 2010][Lind, 2001] :

— **Accessible/Inaccessible :** dans un environnement *accessible*, chaque agent peut accéder à une information complète, à jour, et juste de l'environnement. Dans le cas contraire, l'environnement est *inaccessible* et seulement une information partielle de l'environnement est disponible.

— **Déterministe/Non-déterministe :** un environnement est *déterministe* si le prochain état du monde est complètement déterminé par son état actuel, et par l'activité du système multi-agent. Dans le cas contraire, il est *Non-déterministe*. C'est en particulier le cas d'un environnement ouvert dans lequel des entités externes au système multi-agent peuvent apparaître ou disparaître.

— **Statique/Dynamique :** un environnement est *statique* s'il n'évolue pas au cours du processus de décision des agents. Dans le cas contraire, l'environnement est *dynamique*.

— **Discret/Continu :** un environnement est *discret* s'il existe un nombre fini d'actions et de perceptions, et est *continu* dans le cas contraire.

6.1.4 Interactions

Les interactions dans les systèmes multi-agents sont une part importante de leur définition. Dans [Camps et al., 1998][Gleizes et al., 1999] les auteurs distinguent trois types d'interactions :

— les interactions **antinomiques** : une entité a une interaction *antinomique* avec une autre entité si son action gêne cette entité dans l'accomplissement de sa tâche,

— les interactions **neutres** : si son action ne gêne pas mais ne favorise pas non plus l'accomplissement de la tâche de l'autre entité, l'interaction est *neutre*,

— les interactions **coopératives** : enfin, si l'action favorise l'accomplissement de la tâche de l'autre entité, l'interaction est *coopérative*.

Du point de vue de l'agent, la coopération correspond à la capacité des agents à travailler ensemble dans le but de réaliser leur objectif collectivement. Par conséquent, dans le but d'assurer l'état coopératif, les interactions entre les agents doivent satisfaire 4 propriétés [Glize, 2001] :

— **Sincérité** : un agent ne ment pas sur son état, ses croyances, ou sur ses intentions (mais il peut avoir des croyances fausses),

— **Volonté** : un agent satisfait toujours une requête si la requête est cohérente avec son état et qu'il en a la capacité,

— **Bienveillance** : quand c'est possible, l'agent le moins satisfait est favorisé pour être satisfait,

— **Réciprocité** : un agent connaît ces trois dernières propriétés, en tient compte dans ses raisonnements, et les respecte.

6.1.5 Émergence

Dans les méthodes traditionnelles de développement, les logiciels sont généralement mis au point pour répondre à des besoins explicités et pré-établis ainsi qu'aux exigences des différentes parties prenantes. De ces besoins, les concepteurs établissent une architecture logicielle, puis développent l'application, souvent en regardant les briques logicielles existantes et disponibles pour les réutiliser. Dans cette approche **top-down**, chaque brique logicielle, chaque partie, apporte quelque chose à la finalité du logiciel.

Cette approche est une bonne approche pour des systèmes, où le comportement global est bien défini, et où par conséquent le système peut être facilement découpé en parties. Néanmoins, elle peine à contrôler des systèmes **complexes**. De tels système sont généralement caractérisés par des comportements non-linéaires [Guespin-Michel, 2015], des rétroactions [Wiener, 1948], des causalités circulaires [Erdi, 2008], et sont généralement très sensibles aux conditions initiales. Cette dernière est souvent illustrée par le cas de la météorologie et *l'effet papillon* du mathématicien Lorenz [Lorenz, 2000]. De tels systèmes sont souvent composés par un grand nombre d'entités, et par un grand nombre d'interactions.

Une notion centrale de l'étude des systèmes complexes est la notion d'**émergence**. Cette dernière n'a pas de définition formellement acceptée de tous, mais on qualifie généralement celle-ci comme étant un phénomène perceptible au niveau global résultant des interactions locales des parties du système [De Wolf and Holvoet, 2004][Georgé et al., 2011]. L'apparition de la conscience humaine, qui est le résultat de l'interaction entre des milliards de neurones, peut être considérée comme un phénomène émergent.

Néanmoins, l'émergence ne peut pas être réduite uniquement au résultat des interactions locales des parties du système. En effet, on qualifie rarement la rotation des aiguilles d'une horloge comme émergent. D'autres aspects sont à prendre en compte.

Dans un premier temps, pour être émergent, un phénomène doit apparaître au cours de l'évolution du système (caractère dynamique), et se maintenir suffisamment longtemps pour être observé [Goldstein, 1999]. Nous retiendrons un deuxième critère qui est la *nouveauté* (*radical novelty*) [De Wolf and Holvoet, 2004]. Un phénomène est émergent s'il apporte quelque chose au niveau macroscopique de fondamentalement nouveau par rapport au niveau microscopique, qu'il est possible de paraphraser avec la citation attribuée généralement à Aristote, "le tout est supérieur à la somme de ses parties".

Afin d'aborder les systèmes complexes, il est nécessaire de changer l'approche traditionnelle de conception, et de ne plus étudier le tout, mais les parties, leur comportement, et leurs interactions.

Un système multi-agent est composé d'une multitude de parties en interaction, les agents, qui agissent localement entre eux et avec l'environnement de manière décentralisée, sans contrôle central du système. Dans le but de créer un système multi-agent qui produit la fonction voulue, les efforts de modélisation se concentrent alors sur le comportement local des différentes entités, en particulier le comportement des agents. La fonction du système émerge alors des interactions entre les différentes parties de ce système. La théorie AMAS [Georgé et al., 2011] propose un framework théorique pour créer de tels systèmes. Cette approche se base sur les interactions entre les agents, et la notion de coopération, dans le but

d'aborder les systèmes complexes, et de contrôler l'émergence.

6.2 L'approche AMAS

Nous avons défini dans la section précédente les concepts d'agent, de système multi-agent, d'interactions (pour les agents), et d'émergence. Dans cette section, nous présentons les concepts et les notions sur lesquels le travail de cette thèse porte : la théorie *Adaptive Multi-Agent System (AMAS)*.

Tel que nous l'avons défini dans la section 6.1.5, un système complexe n'a pas la capacité de s'observer lui-même, et par conséquent il est impossible du point de vue du système de juger si la tâche pour laquelle il a été conçu est satisfaite. Ainsi, le rôle du juge revient à un observateur externe qui peut observer l'ensemble et indiquer si le système est *fonctionnellement adéquat* (c'est-à-dire s'il réalise la fonction pour laquelle il a été conçu). L'*auto-organisation* est le processus permettant à un système logiciel de modifier dynamiquement son organisation interne (structure et fonctionnalité) pendant son temps d'exécution sans aucun mécanisme de direction externe explicite [Georgé et al., 2011]. Cependant, pour s'auto-organiser, le système doit s'évaluer lui-même. Dans le cas d'auto-organisation des systèmes multi-agents, les agents ne connaissent pas la fonction globale, mais n'en possèdent que des perceptions locales.

La clé est alors de trouver un moyen pour produire une bonne auto-organisation, afin de permettre au système de réaliser une fonction adéquate. La théorie AMAS propose la coopération comme moyen. En effet, la définition de l'adéquation fonctionnelle indique qu'un système fonctionnellement adéquat n'a pas d'interactions antinomiques avec son environnement. Ainsi, tous les systèmes à milieu intérieur coopératif sont fonctionnellement adéquats.

> "Pour tout système fonctionnellement adéquat, il existe au moins un système coopératif qui soit fonctionnellement adéquat dans le même environnement." [Glize, 2001]

Un système à milieu intérieur coopératif est un système où toutes les parties interagissent de façon coopérative. Ainsi, un système multi-agent où tous les agents sont dans un état coopératif (c'est-à-dire un système à milieu intérieur coopératif) existe pour chaque problème calculable.

Une démonstration informelle de ce théorème peut être trouvée dans [Glize, 2001]. Cette démonstration s'appuie sur l'inclusion de trois catégories différentes de systèmes : Systèmes Fonctionellement Adéquats (SFA), Systèmes à Etats Coopératifs (SEC) et Systèmes à Milieu Intérieur Coopératif (SMIC). La figure 6.2 illustre ces inclusions ($SFA \supset SEC \supset SMIC$)

Nous présentons donc dans ce qui suit la coopération comme définie dans la théorie AMAS.

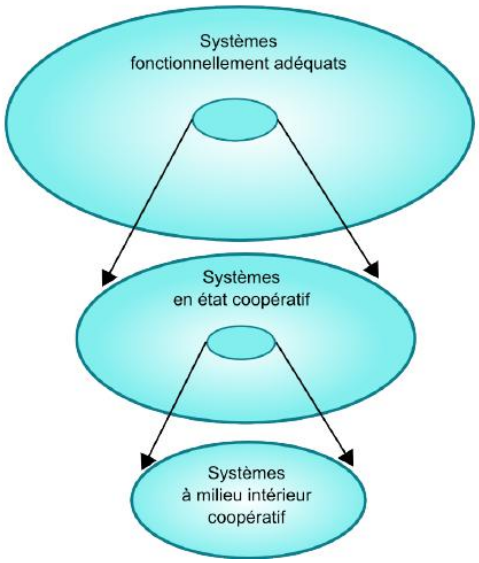


Figure 6.2 – Adéquation fonctionnelle des systèmes à milieu intérieur coopératif

6.2.1 Coopération

Dans la théorie AMAS, les agents ont un **comportement nominal**, c'est-à-dire un comportement normal qui assure l'adéquation fonctionnelle quand l'agent est dans un état coopératif. Les agents ont aussi un **comportement coopératif**, qu'ils adoptent lorsqu'ils sont en **situation de non coopération** (SNC) afin de revenir dans un état coopératif.

Une Situation de Non Coopération (SNC) est une situation dans laquelle un agent, de par son comportement, ou simplement l'état dans lequel il se trouve, ne peut atteindre son but, ou empêche d'autres agents d'atteindre leur but.

La théorie AMAS identifie 7 situations génériques de non coopération (SNC) pour un agent, réparties selon les trois étapes du cycle de vie d'un agent [Georgé et al., 2011]. Nous trouvons, associées à la perception, les situation de non coopération d'incompréhension et d'ambiguïté, puis, associées à la décision, les SNC d'incompétence et d'improductivité, et enfin, associés à l'action, les SNC de concurrence, de conflit et d'inutilité (voir figure 6.3) :

— **Incompréhension** : un agent est incapable d'extraire de l'information de ce qu'il perçoit.
— **Ambiguïté** : un agent peut interpréter de différentes manières ce qu'il perçoit.
— **Incompétence** : un agent est incapable de prendre une décision à partir de ses connaissances.
— **Improductivité** : l'agent est incapable de proposer une action à faire pendant l'action.
— **Concurrence** : l'agent perçoit qu'un autre agent est en train d'agir et que les conséquences de son action seront les mêmes que sa propre action.
— **Conflit** : les conséquences de l'action seront incompatibles avec le comportement d'un autre agent.
— **Inutilité** : l'action d'un agent n'a pas d'impact et n'est pas intéressante pour les autres agents.

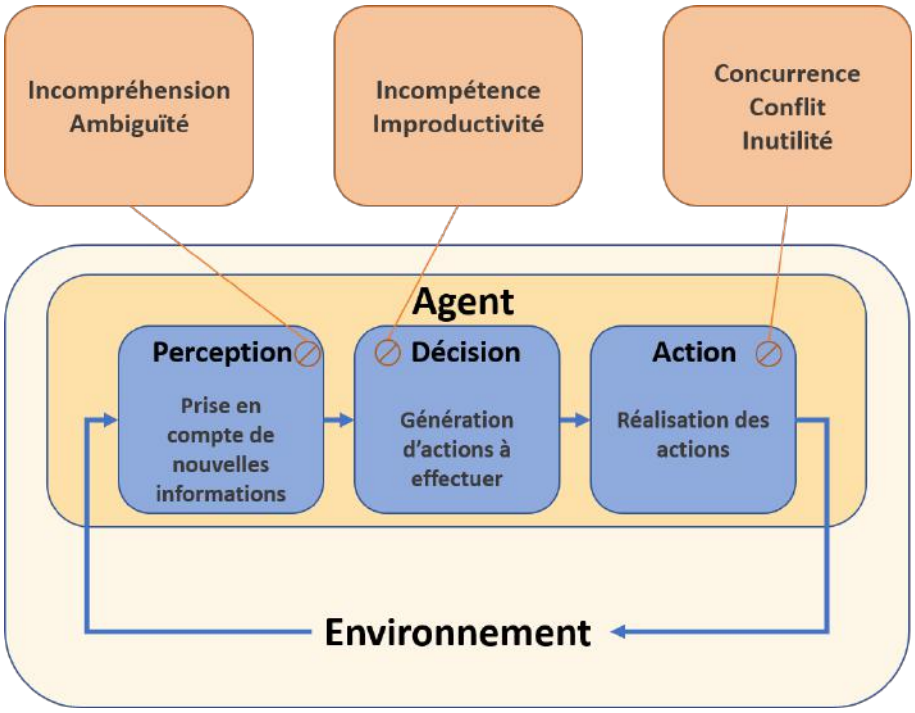


Figure 6.3 – Situations de non coopération (SNC) d'un agent

La résolution des situations de non coopération (SNC) se fait localement par les agents, en modifiant l'organisation du système multi-agent, par conséquent en s'auto-organisant. Un agent dispose de trois moyens génériques pour modifier cette organisation :

— **Réorganisation** : changer ses relations avec les autres agents, c'est-à-dire prendre contact avec d'autres agents, arrêter d'interagir avec un agent donné, ou bien modifier l'importance qu'il accorde aux relations existantes.

— **Ajustement** : modifier son comportement en ajustant ses paramètres internes.

— **Ouverture** : Décider de créer un nouvel agent, ou bien de se supprimer.

C'est au concepteur que revient la tâche de définir les règles de résolution des SNC. Elles se présentent sous la forme de règles comportementales qui décrivent la détection de la SNC et sa méthode de résolution. A ses règles s'ajoutent les règles d'interactions coopératives précédentes (section 6.1.4) : sincérité, bienveillance, réciprocité et volonté.

La règle de bienveillance implique un degré de satisfaction des différents agents. Dans la théorie AMAS, ce degré de non satisfaction d'un agent est appelé **criticité**. L'évaluation de la criticité et sa forme est spécifique à chaque type d'agent. La criticité peut être un réel, un entier, ou tout espace (discret ou continu) totalement ordonné. Dans la pratique, la criticité est souvent un réel, ou un n-uplet de réels.

La conception d'un AMAS, est différente de la conception d'approches traditionnelles. La méthode Atelier de Développement de Logiciels à Fonctionnalité Emergente (ADELFE) [Bonjean et al., 2014] guide le développement d'AMAS à travers 5 étapes que nous ne décrirons pas ici.

La théorie AMAS permet de définir des systèmes fonctionnellement adéquats capables

d'aborder efficacement les systèmes complexes. Nous présentons dans les sections suivantes deux modèles d'AMAS, le premier aborde la résolution collective de problème complexe sous-contrainte (section 6.3), et le deuxième l'apprentissage et la simulation de trafic aérien (section 6.4). Nous utilisons ces deux modèles dans la suite de la thèse.

6.3 Le pattern AMAS4Opt

Les problèmes d'optimisation complexes sont généralement formalisés sous la forme d'un ensemble d'entités qui doivent satisfaire un ensemble de contraintes. Dans les systèmes multi-agents, les entités sont généralement représentées en utilisant des agents. En fonction des interactions entre le système et son environnement, l'organisation des agents émerge et constitue la solution au problème.

Dans son travail, [Kaddoum, 2011] propose un modèle d'AMAS pour résoudre les problèmes d'optimisation, nommé AMAS4Opt. Ce modèle générique s'appuie sur la conception de 2 rôles d'agents :

— **Rôle contraint** : les agents dans le rôle contraint sont des agents qui ont besoin de l'aide d'autres agents,

— **Rôle service** : les agents dans le rôle service sont les agents qui aident coopérativement les agents dans le rôle contraint.

Les rôles dans le modèle ne sont pas figés, un agent peut passer dans le rôle contraint pour effectuer une tâche puis repasser dans le rôle service pour d'autres tâches. Les agents dans le rôle contraint sont considérés comme des activateurs de la résolution, puisque ce sont eux qui sont à l'initiative des interactions dans le but de satisfaire leurs contraintes. Les agents dans le rôle service étant coopératifs, ils aident en priorité les agents les plus critiques.

Pour assurer les négociations et les interactions entre les agents, AMAS4Opt propose 4 principaux types de messages :

— **demande de service** : ce message est envoyé par les agents dans le rôle contraint pour demander un service à des agents dans le rôle de service. Ce message dépend des connaissances de l'agent qui l'envoie. Le message contient aussi la criticité de l'expéditeur, pour que l'agent dans le rôle de service la connaisse,

— **demande d'information** : les deux rôles peuvent échanger ce type de message, lorsqu'ils ont besoin d'informations supplémentaires,

— **réponse à la demande** : après le traitement de la demande de service, l'agent dans le rôle de service répond en utilisant ce type de message,

— **message d'information** : les informations jugées utiles par les agents (dans les deux rôles) peuvent être échangées par le biais de ce genre de message.

Nous utilisons ce modèle dans la suite pour la gestion autonome de conflits, et la génération de scénarios, que nous présentons dans le chapitre suivant. La section suivante concerne le système EVAA, pour l'apprentissage et la simulation de trafic aérien.

6.4 EVAA : Environnement Virtuel Auto-Adaptatif

Habituellement, les simulateurs de trafic aérien utilisent un modèle mathématique plus ou moins complexe pour faire voler les avions, dans la majorité des cas en utilisant un plan de vol (voir section 3.1.4 et [Rantrua, 2017]).

Dans sa thèse, [Rantrua, 2017] propose un AMAS nommé EVAA pour apprendre à partir de données réelles de vols comment volent les avions en fonction du contexte dans lequel ils se trouvent. Il utilise ensuite cette base de données de comportements contextuels pour faire voler des avions virtuels dans un simulateur.

Dans son travail, il utilise 5 types d'agents :

— **avion de rejeu** : un avion qui rejoue une trajectoire enregistrée, c'est-à-dire un avion qui suit la suite de points qui constituent la trajectoire de l'avion,

— **situation d'intérêt** : un point d'une trajectoire jugé pertinent,

— **situation agrégée** : un groupe de situations d'intérêt qui présentent des similarités,

— **contextes** : une portion de comportements, c'est-à-dire les conditions dans lesquelles il faut faire une action,

— **avion simulé** : un avion qui interroge les contextes pour décider de sa trajectoire lors de la simulation.

Phase d'apprentissage. Dans la phase d'apprentissage, EVAA crée pour chaque avion dont il possède les enregistrements, un agent *avion de rejeu*, qui va suivre la suite de points qui constituent sa trajectoire enregistrée. Pendant qu'il suit cette suite de points, l'agent *avion de rejeu* détecte des points notables dans sa trajectoire. Ces points notables correspondent à des changements significatifs dans celle-ci, comme un virage ou une montée.

Pour chaque point notable qu'il détecte, il associe un agent *situation d'intérêt* avec les caractéristiques qui définissent ce point notable. Ces caractéristiques correspondent à la position (longitude et latitude) et l'altitude, mais aussi les caractéristiques de l'avion comme la compagnie, la vitesse horizontale et la vitesse verticale, le modèle avion, parmi d'autres.

Ces agents de *situations d'intérêt* connaissent l'agent *situations d'intérêt* précédant et suivant, et forment avec eux des *vecteurs*, que nous appelons dans la suite des **segments**.

Ces agents *situations d'intérêt* observent ensuite leur environnement pour se regrouper dans un agent *situation agrégée*.

La figure 6.4 montre le résultat de l'apprentissage d'EVAA sur une centaine de vols entre Bordeaux, Toulouse, Lyon et Paris.

Phase de simulation. Dans la phase de simulation, les agents *situations agrégées* n'évoluent plus et deviennent des agents *contextes*, et les agents situation d'intérêt liés entre eux deviennent des *segments*. L'ensemble des agents *contextes* correspond alors à une base de données distribuées et contextuelles de comportements, que les agents *avions simulés* utilisent pour voler durant la simulation.

Figure 6.4 – Apprentissage d'EVAA sur des vols entre Toulouse et Paris, Paris et Lyon, Lyon et Bordeaux, et Bordeaux et Paris. Les flèches rouges représentent les segments, les points oranges sont des situations d'intérêts, et les points verts sont des situations agrégées.

Dans la suite, nous utiliserons les résultats de l'apprentissage d'EVAA comme une *road-map* pour générer des scénarios réalistes, et le simulateur multi-agent d'EVAA pour expérimenter des comportements adaptatifs autonomes.

7 Le modèle de structuration de simulation de trafic AGATS ("Autonomous GenerAtion of Traffic Simulations")

Nous présentons dans ce chapitre le modèle *Autonomous GenerAtion of Traffic Simulations (AGATS)*, dont le but est de permettre une auto-structuration d'une simulation de trafic pour générer un trafic réaliste du point de vue microscopique et macroscopique (c'est-à-dire du point de vue des trajectoires et du trafic), et contenant des *situations* voulues par un scénariste. Le système s'appuie sur une modélisation à base d'agents, selon la théorie AMAS [Georgé et al., 2011], mais aussi sur le pattern AMAS4Opt [Kaddoum, 2011], pattern AMAS dédié à la résolution collective de problèmes d'optimisation, et sur EVAA [Rantrua, 2017], système multi-agent dédié à l'apprentissage et la simulation de trafic aérien.

Nous décrivons dans un premier temps l'ensemble du système *AGATS* après avoir rappelé brièvement ses objectifs (sections 7.1 et 7.2), pour ensuite détailler les différentes entités qui le composent (sections 7.5 à 7.9), et enfin présenter les différentes situations de non coopération et leur résolution (section 7.10).

7.1 Les objectifs et axes du système AGATS

Pour rappel, nous découpons l'objectif de structuration d'une simulation de trafic dans le but de générer un trafic réaliste, et contenant des *situations* voulues par un scénariste, en différents sous-objectifs :

1. Être capable de générer un scénario réaliste du point de vue microscopique et macroscopique, c'est-à-dire un scénario comportant des **trajectoires réalistes**, le tout dans un **trafic réaliste**,
2. Une composante importante du réalisme de trafic est l'**évitement de collisions**, le système doit donc permettre l'évitement de collisions entre les différentes entités de la simulation,
3. Lors de la simulation, l'utilisation du scénario doit prendre en compte l'apparition des différentes **situations** voulues par le scénariste,

4. Cette simulation doit être **interactive** pour que des acteurs humains agissent sur la simulation et que celle-ci réagisse à ces actions, en s'adaptant éventuellement pour continuer de générer les *situations* désirées.
5. Enfin, même si la thèse s'inscrit dans le trafic aérien, le modèle doit être **générique**, afin d'être applicable à d'autres trafics, tel que le trafic routier.

Nous voulons qu'à partir de :

— Une description de l'environnement. Par exemple dans la génération de trafic aérien, une description des espaces aériens dans lesquels les avions volent, des routes aériennes, et des aéroports (voir section 1).

— Une base de données de trajectoire. Dans notre système, nous utilisons le résultat de l'apprentissage d'EVAA (section 6.4) comme une *roadmap* (section 4.1). Chaque *situation d'intérêt* est alors un noeud, et chaque *segment* une arrête. Cette *roadmap* est utilisée comme une base de données de trajectoires, contenant un ensemble de parties de trajectoires (les *segments*).

— Un ensemble de *situations* requises par un scénariste, noté *Sit* ;

Les agents du système *AGATS* soient capables de générer un scénario de trafic réaliste, c'est-à-dire un ensemble de trajectoires réalistes, qui produira l'ensemble des situations *Sit* souhaitées lors de la simulation. Les agents du système *AGATS* doivent aussi être capables par leurs interactions d'adapter leur comportement afin de préserver le réalisme et de maintenir les *situations* requises.

7.2 Présentation des entités du système AGATS

Nous présentons dans cette section l'ensemble des entités de structuration de simulation de trafic du système *AGATS*. Cette structuration s'inspire de la métaphore du cinéma (voir section 3.2.3). Un scénario répond en général à 4 questions :

1. Quel est le lieu ?
2. Quel est le temps ?
3. Quelle est l'action ?
4. Quels sont les acteurs ?

Par hypothèse, les 2 premières questions sont entièrement spécifiées par le scénariste qui décrit l'environnement et le moment de la simulation. Il apporte également les grandes lignes de la réponse à la troisième, avec l'ensemble des *situations Sit* souhaitées. Le scénario, qui est une histoire, comporte des évènements majeurs, les *situations*. Dans un scénario de trafic, ces *situations* sont des évènements tels que : des densités de trafic, des collisions, des files d'attentes, etc.

Le but des différentes entités du système *AGATS* est de construire collectivement, grâce à leurs différentes interactions, l'ensemble des "actions" qui amène à la génération de ces *situations*, et l'ensemble des "acteurs" qui effectueront ces "actions" dans la simulation. Dans un scénario de trafic, ces "actions" sont des déplacements d'entités mobiles. Ainsi, l'ensemble des "actions" est un ensemble de déplacements, et correspond finalement à un ensemble de

trajectoires. Les "acteurs", quant à eux, sont uniquement des entités mobiles qui suivent ces trajectoires.

Puisque la simulation est interactive, les agents d'*AGATS* doivent aussi être capables de réagir à des modifications de l'environnement en adaptant leur organisation, et par conséquent les déplacements des entités mobiles, pour maintenir la cohérence du scénario. Comme notre travail concerne des simulations réalistes, cet ensemble de trajectoires devra être réaliste du point de vue macroscopique, c'est-à-dire du point de vue du trafic, mais aussi du point de vue microscopique, c'est-à-dire du point de vue de chaque trajectoire. Nous décrivons dans ce qui suit les différentes entités du système *AGATS*, et leur rôle dans cette structuration de trafic.

AGATS est composé de différents agents, entités passives et entités actives telles que définies dans la méthodologie ADELFE [Bonjean et al., 2014]. *AGATS* est composé de deux entités passives :

— La base de données de trajectoires, entité passive qui représente la base de données des parties de trajectoires. Dans l'implémentation des agents d'*AGATS*, cette base de données de trajectoires contient l'ensemble des agents contextes d'EVAA (section 6.4). Elle sert comme *roadmap* permettant aux agents (en particulier les *Agents Parties*) de générer des parties des trajectoires réalistes du point de vue physique et comportemental (5). Une partie de la trajectoire, peut représenter une courbe ou un segment, dans notre implémentation, elle représente uniquement des segments au sens d'EVAA. Ces parties de trajectoires contiennent d'autres caractéristiques qui décrivent l'entité mobile qui parcourt celle-ci, par exemple, les segments d'EVAA contiennent l'intervalle de temps (deux dates en temps unix) durant lequel l'avion a utilisé ce segment, le modèle avion (A320, A340...), un identifiant, la compagnie aérienne, en plus de la position de départ et d'arrivée du segment (en longitude, latitude) et leur altitude respective (en pied).

— L'*entité passive Caractéristique* est l'entité passive qui représente une caractéristique d'une situation.

Ces entités passives sont utilisées par 5 types d'agents coopératifs :

— L'*Agent Entité Mobile (Agent EM)*, représentant une entité mobile qui a pour but d'apporter des comportements d'adaptation aux différentes entités mobiles, pour éviter les collisions, suivre une trajectoire, et si l'entité mobile est impliquée dans une situation, de la maintenir.

— L'*Agent Situation*, représentant une situation requise par le scénariste qui mobilise les entités mobiles pour obtenir la *situation* qui lui est attribuée. Pour ce faire, il sollicite autant d'*Agents Trajectoires* que d'entités mobiles nécessaires à la création de sa *situation*.

— L'*Agent Trajectoire*, représentant une trajectoire qui a pour but de répondre au besoin de l'*Agent Situation* tout en assurant le réalisme de sa trajectoire. Sa trajectoire est générée par son propre ensemble d'*Agents Parties*.

— L'*Agent Partie*, représentant une partie de la trajectoire d'un *Agent Trajectoire* qui a pour but de générer une partie de trajectoire réaliste pour son *Agent Trajectoire*, en étant cohérent avec le reste de la trajectoire, c'est-à-dire les autres *Agents Parties* de l'*Agent Trajectoire*. La base de données de trajectoires alimente ces *Agents Parties* en parties de trajectoires potentielles.

— L'*Agent Extrémité*, représentant une zone d'apparition et de disparition d'entités mobiles qui génère les entités mobiles nécessaires à la simulation, que ce soit pour générer les *situations*, ou le trafic contextuel, c'est-à-dire le trafic qui ne participe pas à des situations mais participe au réalisme du trafic. Lors de cette génération d'entités mobiles, il doit respecter les données locales qui détaillent le trafic évoluant dans sa zone.

Les agents d'*AGATS* agissent selon deux phases, la première, la **phase de construction** permet aux agents de générer les premières trajectoires et les premières entités mobiles, et la deuxième, la **phase de simulation**, permet aux agents d'évaluer, d'auto-adapter, et d'améliorer les trajectoires ainsi générées. L'algorithme 7.1 décrit le comportement général du système et les interactions entre les différents agents pour une situation donnée selon ces deux phases. La figure 7.1 représente l'organisation des agents pour la génération d'une situation de collision (décrite ci-après).

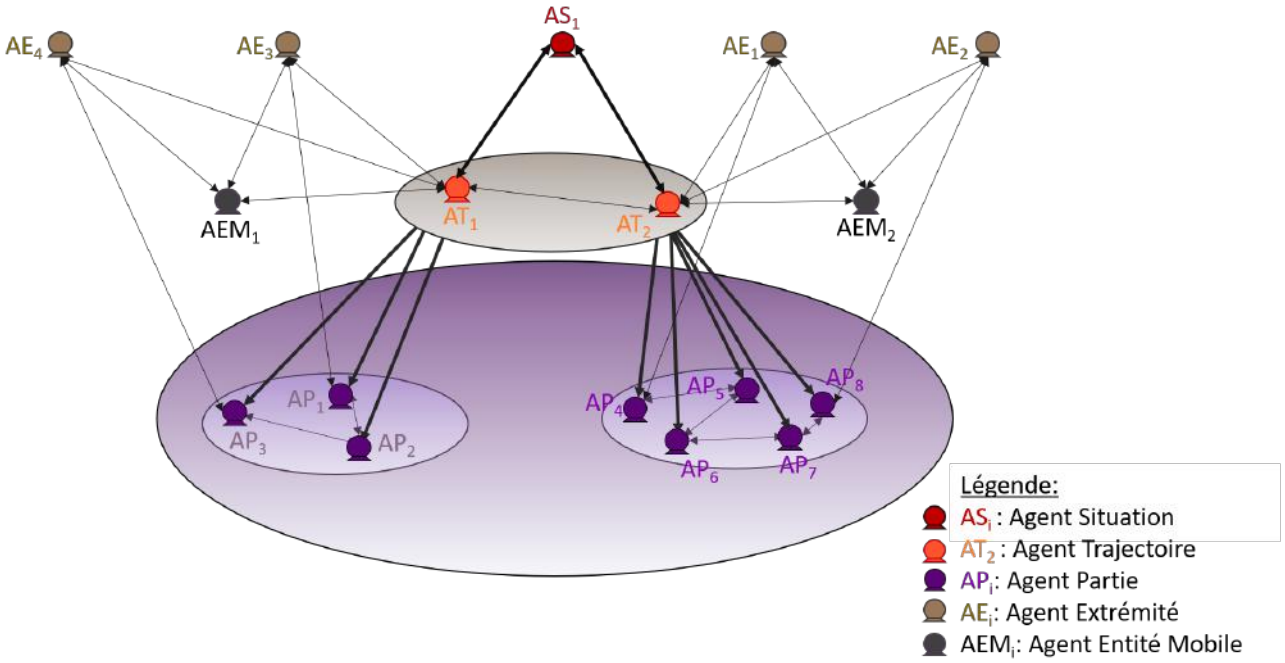


Figure 7.1 – Organisation des différents agents pour la génération de la *situation de collision* (décrite section 7.3)

Nous illustrons ce concept dans le cas de la génération d'une *situation de collision* dans la section suivante, qui servira de fil rouge dans le reste de ce chapitre.

7.3 Un exemple de situation : situation de collision

Afin d'illustrer le concept de *situation* et des caractéristiques de celle-ci, nous décrivons une *situation de collision* telle que définie pour le trafic aérien. Une *situation de collision* peut être définie par les caractéristiques suivantes :

1. le **nombre d'entités mobiles** impliquées dans la collision (N_{Sit}), par exemple 2 avions,

Algorithme 7.1 : Comportement général des agents d'*AGATS* pour générer une *situation*

Phase de construction

1. L'*Agent Situation* crée autant d'*Agents Trajectoires* que nécessaire pour sa *situation*
2. Chaque *Agent Trajectoire* choisit une partie de trajectoire satisfaisant au mieux ses caractéristiques dans la base de données de trajectoires et lui associe un *Agent Partie*.
3. L'*Agent Partie* ainsi créé déclenche la création d'autres *Agents Parties* jusqu'à ce qu'une trajectoire complète soit créée avec les deux *Agents Extrémités* adéquats. Les *Agents Parties* au fur et à mesure de leur création s'auto-organisent en se modifiant, en se substituant, en ajoutant d'autres *Agents Parties* et en se supprimant pour décrire coopérativement la trajectoire la plus réaliste possible.
4. L' *Agent Trajectoire* négocie avec les *Agents Extrémités* la création de l'*Agent Entité Mobile* qui suivra sa trajectoire.

Phase de simulation

repeter

1. L'*Agent Entité Mobile* suit sa trajectoire et envoie des feedbacks à son *Agent Trajectoire*.
2. L'*Agent Trajectoire* transmet les informations aux *Agents Parties* concernés.
3. Les *Agents Parties* auto-adaptent leur partie de trajectoire en se modifiant, en se substituant, en ajoutant d'autres *Agents Parties* et en se supprimant, afin d'aider l'*Agent Trajectoire* à prendre en compte les feedbacks de l'*Agent EM*

tant que Un trafic réaliste satisfaisant l'ensemble des *situations* est créé

2. le **temps** : la collision voulue doit se produire à un certain moment, par exemple à midi,
3. la **position** : la collision voulue doit se produire à un certain endroit, par exemple le point AS_1 dans la figure 7.2,
4. la **géométrie** : la collision voulue doit se produire selon une certaine géométrie, par exemple une entité vient du nord-ouest et une autre vient du sud-ouest,
5. la **proximité** : les 2 entités mobiles doivent se rencontrer.

La figure 7.2 illustre la génération d'une situation de collision. Afin de générer cette situation, l'*Agent Situation* AS_1 génère 2 *Agents Trajectoires*, AT_1 et AT_2, qui vont respectivement prendre dans la base de données de trajectoires les parties de trajectoires PT_2 et PT_6 pour générer chacun leur premier *Agents Parties*, respectivement AP_2 et AP_6. Ces deux *Agents Parties* et les *Agents Parties* qui sont créés après s'auto-organisent jusqu'à ce qu'une trajectoire complète soit créée pour chaque *Agent Trajectoire*. Les *Agents Parties* de AT_1 (AP_1, AP_2, AP_3) se modifient beaucoup pour générer la situation AS_1, tandis que les *Agents Parties* de AT_2 (AP_4, AP_5, AP_6, AP_7) se modifient peu. L'organisation des agents qui en résulte est représentée dans la figure 7.1, l'*Agent Situation* est relié à 2 *Agents Trajectoires*, qui sont chacun reliés à leurs *Agents Parties*. Chaque *Agent Partie* est relié aux 2 *Agents Parties* précédent et suivant dans la trajectoire de son *Agent Trajectoire*, ou à un *Agent Extrémité*.

Nous décrivons en détail les différents agents qui composent le système *AGATS* dans les sections qui suivent.

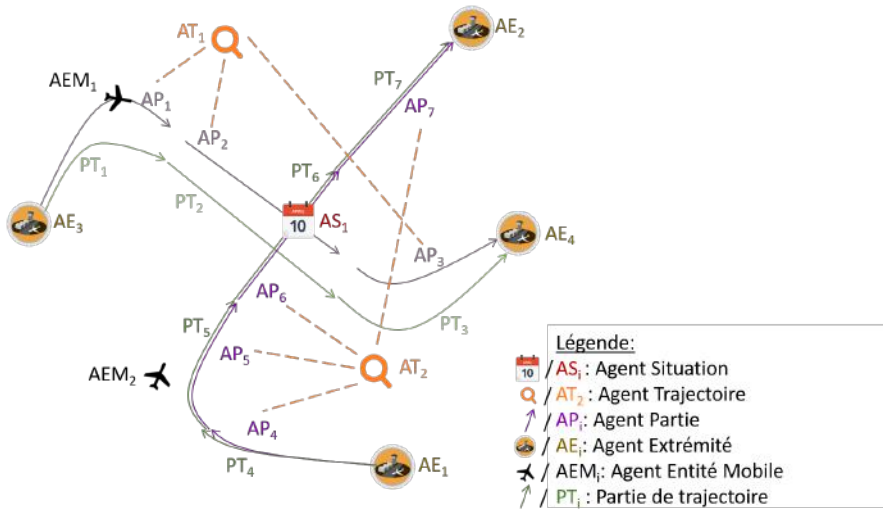


Figure 7.2 – Une représentation des agents d'*AGATS* pour la génération d'une *situation de collision*

7.4 Présentation de l'entité passive caractéristique

L'*entité passive Caractéristique* correspond à une caractéristique de l'*Agent Situation*. Par exemple, la *position* de la collision est une caractéristique d'une *situation de collision* représentée par une *entité passive Caractéristique*. Chaque fois que l'état de l'*Agent Situation* change, il demande à ses *entités passive Caractéristiques* de mettre à jour leur criticité.

Nous notons $Crit_{char,i}$ la criticité de la i^{eme} *entité passive Caractéristique*. Le calcul de la criticité de cette entité dépend de la caractéristique. Dans le cadre de ce travail, afin d'homogénéiser le calcul des criticités, nous proposons d'utiliser la fonction générique décrite par l'équation 7.1, où d_{max} est la valeur maximale de distance prise en compte pour la caractéristique, et d est une fonction de distance de la caractéristique, appliquée à 2 valeurs v_1 et v_2 : la valeur requise (v_1) pour la caractéristique, et la valeur actuelle (v_2).

$$C(x) = \begin{cases} \frac{1000}{d_{max}} d(v_1, v_2) & \text{if } d(v_1, v_2) < d_{max} \\ 1000 & \text{if } d(v_1, v_2) \geq d_{max} \end{cases} \tag{7.1}$$

7.5 Présentation de l'Agent Situation

Pour chaque *situation* que le scénariste souhaite observer dans la simulation, le système *AGATS* associe un *Agent Situation*. Par exemple, si le scénariste veut un conflit aéronautique dans son scénario, un *Agent Situation* s'occupera de générer cette situation. Ces agents sont, avec les *Agent Extrémité*, à l'initiation du processus de création du scénario et de la structuration de la simulation, puisque, ces agents, qui sont dans le rôle contraint dans le pattern AMAS4Opt (section 6.3), initient le processus de génération de trajectoires.

Une *situation* est composée d'un ensemble de caractéristiques qui la définissent. Dans le systèmes *AGATS* les caractéristiques d'un *Agent Situation* sont représentées par des *entités*

passive Caractéristiques (section 7.4). Le **but** de l'*Agent Situation* est alors d'avoir ses différentes caractéristiques satisfaites tout en préservant le réalisme. Sa criticité, $Crit_{sit}$, qui représente le degré de satisfaction de ce but, est alors un n-uplet composé des criticités de ses différentes caractéristiques, et par conséquent, $Crit_{sit} = \{Crit_{char,i}\}$.

La figure 7.3 décrit la structure de l'*Agent Situation* que nous utilisons pour décrire son comportement dans ce qui suit.

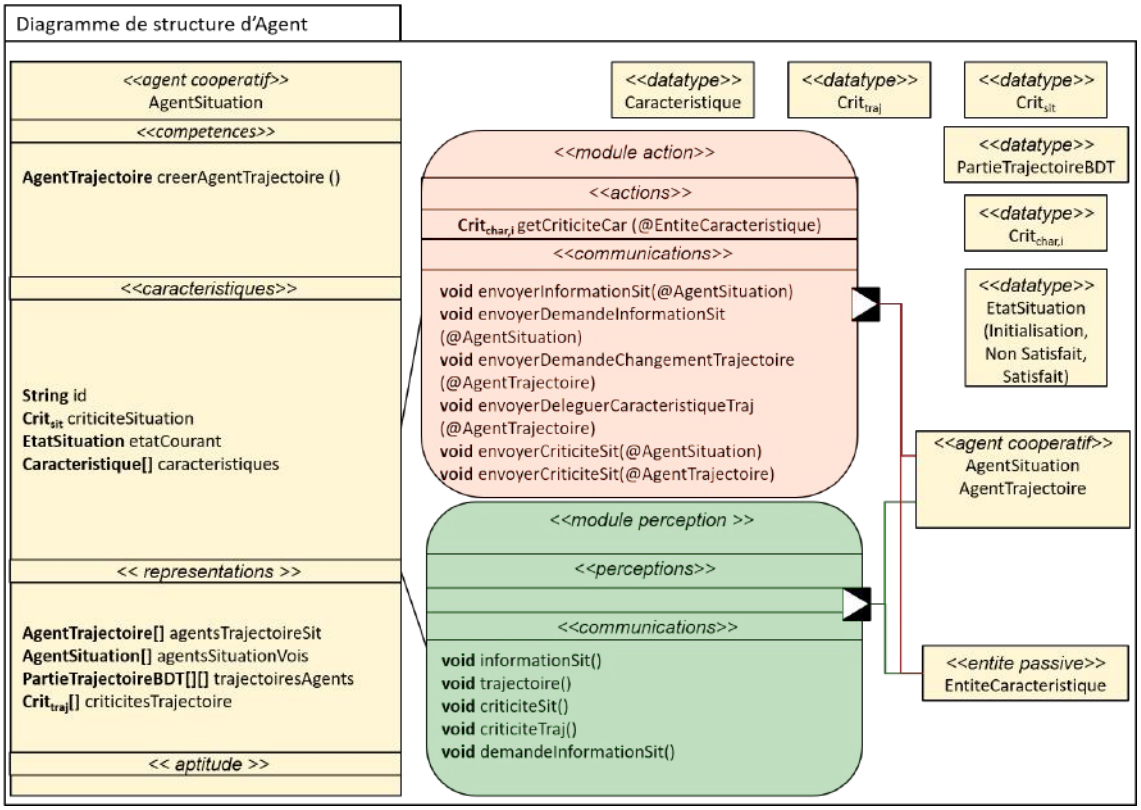


Figure 7.3 – Structure de l'*Agent Situation*

Dans l'état d'*Initialisation* (*EtatSituation etatCourant*, figure 7.3), l'*Agent Situation*, qui est dans le rôle contraint selon le pattern AMAS4Opt, cherche des agents pour satisfaire ses caractéristiques (*Caracteristique[] caracteristiques*), qui sont ses contraintes, et sortir de son état d'*Initialisation* (algorithme 7.2). Pour ce faire, l'*Agent Situation* crée autant d'*Agents Trajectoires* (section 7.6) qu'il faut d'entités mobiles pour satisfaire ses caractéristiques (*void creerAgentTrajectoire()*), puis délègue à chacun de ses *Agents Trajectoires* une partie de la satisfaction de ses caractéristiques (*void envoyerDeleguerCaracteristiqueTraj(@AgentTrajectoire)*), et passe alors dans l'état *Non Satisfait*.

Dans notre exemple de *situation de collision*, l'*Agent Situation* délègue en partie à un *Agent Trajectoire* l'obtention d'un premier avion en provenance du Nord-Ouest, arrivant à AS_1 et devant rencontrer un autre avion, et en partie à l'autre *Agent Trajectoire* l'obtention d'un deuxième avion en provenance du Sud-Ouest, arrivant à AS_1, et devant rencontrer le premier avion.

Afin de passer de l'état *non satisfait* à *satisfait*, l'*Agent Situation* doit avoir chacune

Algorithme 7.2 : Comportement d'un *Agent Situation* dans l'état d'initialisation

1: etatCourant ← Initialisation // L'*Agent Situation* n'a pas encore d'*Agents Trajectoires*
2: **pour** $k \leftarrow 1$ à N_{Sit} **faire**
3: agentsTrajectoireSit[k] ← creerAgentTrajectoire() // L'*Agent Situation* garde en mémoire l'adresse de ses *Agents Trajectoires* qu'il vient de créer
4: envoyerDeleguerCaracteristiqueTraj(agentsTrajectoireSit[k]) // L'*Agent Situation* délègue une partie de ses caractéristiques à chacun de ses *Agents Trajectoires*
5: **fin pour**
6: etatCourant ← NonSatisfait // L'*Agent Situation* passe dans l'état Non Satisfait

de ses caractéristiques satisfaites. Dans ce but, il envoie sa criticité (*void envoyerCritSit(@AgentTrajectoire)*) à ses *Agents Trajectoires* (*AgentTrajectoire[] agentsTrajectoireSit*), et demande à ces derniers de modifier leur trajectoire (*void envoyerDemandeChangementTrajectoire(@AgentTrajectoire)*), afin de faire diminuer sa criticité ($Crit_{Sit}$ *criticiteSituation*). Ses *Agents Trajectoires* lui renvoient leur trajectoire (*void trajectoire()*) et leur criticité (*void criticiteTraj()*) après avoir modifié leur trajectoire, que l'*Agent Situation* garde en mémoire ($Crit_{traj}$*[] criticitesTrajectoire* et *PartieTrajectoireBDT[][] trajectoiresAgents*) (algorithme 7.3).

Algorithme 7.3 : Comportement d'un *Agent Situation* dans l'état Non Satisfait

1: **tant que** $\exists i / Crit_{char,i} \neq 0$ **faire**
2: **pour tout** AgentTrajectoire a ∈ agentsTrajectoireSit **faire**
3: envoyerCriticiteSit(a) // L'*Agent Situation* envoie sa criticité à ses *Agents Trajectoires*
4: **fin pour**
5: **pour tout** AgentSituation a ∈ agentsSituationVois **faire**
6: envoyerCriticiteSit(a) // L'*Agent Situation* envoie sa criticité aux *Agents Situations* dans son voisinage (pour résoudre certaines SNC)
7: **fin pour**
8: **pour tout** AgentTrajectoire a ∈ agentsTrajectoireSit **faire**
9: envoyerDemandeChangementTrajectoire(a) // L'*Agent Situation* demande à ses *Agents Trajectoires* de modifier leur trajectoire afin de mieux satisfaire ses caractéristiques
10: **fin pour**
11: **pour tout** Caracteristique c ∈ caracteristiques **faire**
12: getCriticiteCar(c) // L'*Agent Situation* demande à ses *entités passive Caractéristiques* de mettre à jour leur criticité
13: **fin pour**
14: **fin tant que**
15: etatCourant ← Satistfait // L'*Agent Situation* a toutes ses caractéristiques satisfaites, et passe donc dans l'état Satisfait

Enfin, dans l'état *Satisfait*, l'*Agent Situation* reste à l'écoute des messages des autres agents, en particulier ses *Agents Situations* voisins (*AgentSituation[] agentsSituationVois*), et répond à leur demande (résolution de situation de non coopération, voir section 7.10, ou

demande d'information, *void envoyerDemandeInformationSit(@AgentSituation)*, *void envoyer-InformationSit(@Agent)*, *void envoyerCriticiteSit(@AgentSituation)*), et peut passer de nouveau dans l'état *Non Satisfait* si la trajectoire a été modifiée par l'*Agent Trajectoire*, par exemple pour s'adapter lors de la simulation.

7.6 Présentation de l'Agent Trajectoire

Dans le but d'avoir les *Agents EM* requis par ses différentes caractéristiques, chaque *Agent Situation* (figure 7.4) crée autant d'*Agents Trajectoires* que nécessaire. Il délègue par la suite à chacun de ses *Agents Trajectoires* la partie de ses caractéristiques le concernant notamment celles définissant le rôle de l'*Agent EM*. Dans notre exemple précédent (section 7.5), l'*Agent Situation* délègue à un premier *Agent Trajectoire* ses caractéristiques de position, temps, collision, et une partie de la caractéristique de géométrie, par exemple celle de venir du Nord-Ouest.

Le **but** de chaque *Agent Trajectoire* est alors de construire une trajectoire réaliste du point de vue **comportemental** et **physique** (section 3.2.3), noté τ, qui sera ensuite suivie par une entité mobile, dans le but de satisfaire les caractéristiques déléguées par l'*Agent Situation*. Ainsi, l'*Agent Trajectoire* est dans un rôle service du pattern AMAS4Opt vis-à-vis de son *Agent Situation*.

La figure 7.4 décrit la structure de l'*Agent Trajectoire* que nous utilisons pour décrire son comportement dans ce qui suit, qui dépend en majorité de son état (*EtatTraj etatCourrant*). Quelque soit son état, l'*Agent Trajectoire* envoie régulièrement sa trajectoire (*void envoyerTrajectoire(@AgentSituation)*) et sa criticité ($Crit_{traj}$ *criticiteTrajectoire*, *void envoyerCriticiteTraj(@AgentSituation)*) à son *Agent Situation* (*AgentSituation agentSituation*), et aux *Agents Trajectoires* de la même situation (*AgentTrajectoire[] agentsTrajectoire*, *void envoyerCriticite-Traj(@AgentTrajectoire)*, *void envoyerTrajectoire(@AgentTrajectoire)*)

Dans l'état *Initialisation*, l'*Agent Trajectoire* va chercher dans la base de données de trajectoires une première partie de trajectoire (*PartieTrajectoireBDT getDonneesBDT*()) pour satisfaire au mieux (*PartieTrajectoireBDT choisirPartieTrajectoire()*) les caractéristiques (*Caracteristique[] caracteristiquesSituation*) déléguées par son *Agent Situation*. Il crée ensuite un *Agent Partie* (*Agent Partie creerAgentPartie(PartieTrajectoireBDT)*) associé à cette partie de trajectoire (algorithme 7.4, et étape 1 de la figure 7.5).

Ce premier *Agent Partie* crée un *Agent Partie* à chacune de ses extrémités (étape 2 de la figure 7.5), ces deux nouveaux créent à leur tour un *Agent Partie* à leur extrémité qui n'est pas relié, et ainsi de suite jusqu'à ce qu'une trajectoire complète soit créée (étape 3). En parallèle et à la suite de cette première initialisation, la trajectoire peut évoluer en fonction des décisions des *Agents Parties*. La trajectoire de l'*Agent Trajectoire* est donc le résultat de la coopération de l'ensemble de ses *Agents Parties* (voir la figure 7.5).

Le **but** de l'*Agent Trajectoire* est de construire une trajectoire réaliste, nous divisons cet objectif en trois sous-objectifs :

1. La capacité de définir une trajectoire **complète** du départ jusqu'à la destination. La criticité $Crit_{traj,comp}$ est associée à ce sous-objectif. Elle est composée de l'ensemble

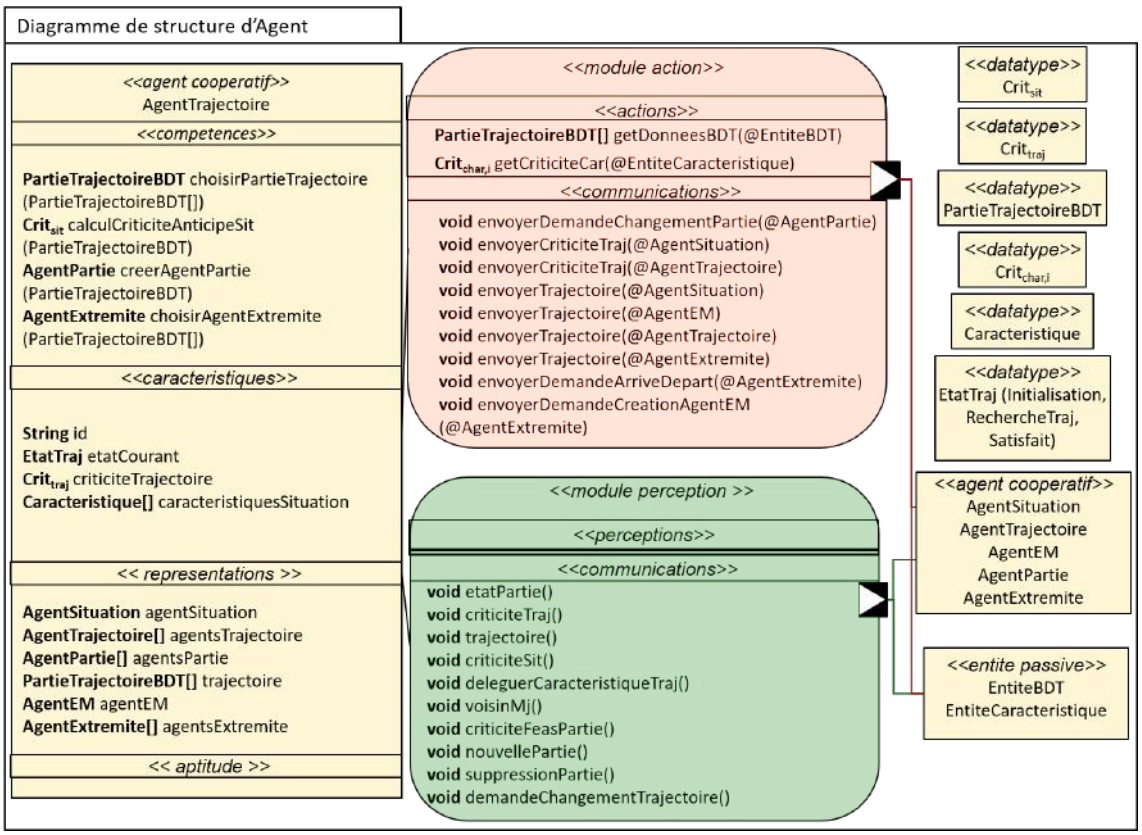


Figure 7.4 – Structure de l'*Agent Trajectoire*

Algorithme 7.4 : Comportement d'une *Agent Trajectoire* dans l'état Initialisation

1: PartieTrajectoireBDT[] ps // Variable temporaire pour contenir des parties de trajectoires venant de la base de données
2: $Crit_{sit}$[] cs // Variable temporaire pour contenir des criticités anticipées
3: etatCourant ← Initialisation // L'*Agent Trajectoire* est dans l'état d'Initialisation
4: caracteristiqueTraj() // L'*Agent Trajectoire* recupère de son *Agent Situation* les caractéristiques qui lui sont déléguées
5: ps ← getDonnesBDT() // Il cherche dans la base de données de trajectoires des parties de trajectoires qui pourraient satisfaire les caractéristiques de sa situation
6: **pour** k ← 1 à ps.taille() **faire**
7: cs[k] ← calculerCriticiteAnticipeSit(ps[k]) // L'*Agent Trajectoire* calcule la criticité anticipée de son *Agent Situation* s'il choisit ps comme première partie de trajectoire
8: **fin pour**
9: Déterminer l'indice l_{min} de la criticité minimale de cs // L'*Agent Trajectoire* choisit la partie de trajectoire qui diminue le plus la criticité de son *Agent Situation*
10: agentPartie[1] ← creerAgentPartie(p[l_{min}]) // Il crée l'*Agent Partie* associé
11: etatCourant ← RechercheTraj // L'*Agent Trajectoire* passe dans l'état RechercheTraj

des expressions locales de cet objectif au niveau de ses *Agents Parties*, c'est-à-dire que

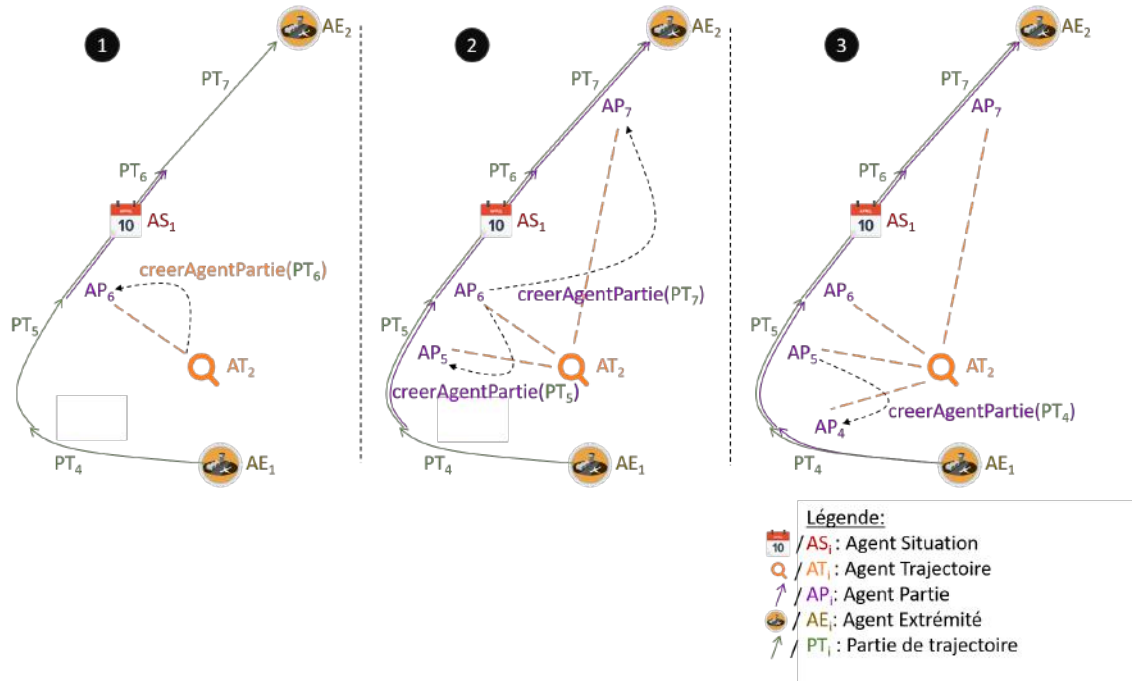


Figure 7.5 – Création de la trajectoire de l'*Agent Trajectoire* par ses *Agents Parties*. L'*Agent Partie*$_6$ (AP_6) crée les *Agents Parties* AP_5 et AP_7, et enfin AP_5 crée AP_4.

$Crit_{traj,comp} = \{Crit_{part,comp}\}$.

2. La définition d'une trajectoire réaliste du point de vue du trafic, c'est-à-dire qui montre des **comportements contextuels**. Ces comportements peuvent aller du respect de certaines règles du trafic, à des schémas plus avancés du trafic réel (par exemple, ne pas passer par un *waypoint* de la trajectoire, ou encore qu'une compagnie irlandaise, préfère passer le plus possible par l'espace aérien irlandais pour payer moins de taxes). La criticité $Crit_{traj,traff}$ est associée à ce sous-objectif. Elle s'exprime également par la criticité $Crit_{part,traff}$ des *Agents Parties* $Crit_{traj,traff} = \{Crit_{part,traff}\}$.
3. La **faisabilité** de la trajectoire, associée à $Crit_{traj,feas}$. Cette criticité est le retour reçu par son *Agent EM* qui la calcule pour chaque partie de trajectoire, noté $Crit_{part,feas}$.

Le calcul des criticités locales $Crit_{part,comp}$ et $Crit_{part,traff}$ est expliqué dans la section 7.7 présentant les *Agents Parties*. Ces trois criticités font partie de la criticité de l'*Agent Trajectoire*, $Crit_{traj} = \{Crit_{traj,comp}, Crit_{traj,traff}, Crit_{traj,feas}\}$.

Afin d'atteindre son but, dans la phase de construction, l'*Agent Trajectoire* demande à ses *Agents Parties* (*AgentPartie[] agentsPartie*) de s'auto-organiser (en modifiant les caractéristiques de leur partie de trajectoire, substituant leur partie de trajectorie par une autre, ajoutant une autre partie de trajectoire ou en se supprimant, voir section 7.7) dans le but de diminuer les criticités $Crit_{traj,comp}$ et $Crit_{traj,traff}$ (*void envoyerDemandeChangementPartie (@AgentPartie)*), c'est-à-dire améliorer la complétude de la trajectoire et le réalisme de la trajectoire du point de vue du trafic. Ses *Agents Parties* lui répondent après auto-organisation leur état (*void etatPartie()*), qu'il garde en mémoire (*PartieTrajectoireBDT[] agentsPartie*). Si sa trajectoire est suffisamment complète, l'*Agent Trajectoire* demande aux *Agents Extrémités* la

création de l'*Agent EM* qui va parcourir sa trajectoire pendant la phase de simulation (*void envoyerDemandeCreationAgentEM (@AgentExtremite)*).

Dans la phase de simulation, l'*Agent Trajectoire* donne sa trajectoire à son *Agent EM* (*AgentEM agentEM*) et lui demande de suivre celle-ci (*void envoyerTrajectoire(@AgentEM)*). Il reçoit ainsi les criticités $Crit_{part,feas}$ de son *Agent EM* qui sont relatives aux difficultés que celui-ci rencontre pour suivre cette trajectoire (*void criticiteFeasPartie()*), et demande aussi à ses *Agents Parties* de s'auto-organiser en conséquence (algorithme 7.5). Par conséquent, l'*Agent Trajectoire* et l'*Agent EM*, collaborent sous le rôle service du pattern AMAS4Opt dans le but de générer la trajectoire.

Enfin, dans l'état *Satisfait*, l'*Agent Trajectoire* reste à l'écoute des messages des autres agents. Si sa trajectoire ne correspond pas ou plus, il demande à ses *Agents Parties* de s'auto-organiser pour proposer une nouvelle trajectoire et retourne dans l'état *RechercheTraj*. Durant la phase de simulation, dans le cas où cette auto-organisation engendre une dégradation trop forte du réalisme par exemple, l'*Agent Trajectoire* est aussi en contact avec les *Agents EM* du voisinage de sa trajectoire (qu'il connaît grâce à son *Agent EM* qui lui envoie des messages *voisinMj*(@AgentTrajectoire)), et est capable de demander à l'une de ces entités mobiles de jouer le rôle de son *Agent EM*, c'est-à-dire de remplacer son *Agent EM* par un *Agent EM* plus proche.

7.7 Présentation de l'Agent Partie

La trajectoire d'un *Agent Trajectoire* est le résultat de l'auto-organisation d'un ensemble d'*Agents Parties* qui planifient localement leur trajectoire. Chaque *Agent Partie* correspond alors à une partie de trajectoire, lié à deux *Agents Parties*, un suivant et un précédent (sauf les agents de début et fin de trajectoire, qui sont chacun liés à un *Agent Extrémité*). L'*Agent Partie* assume un rôle service selon le pattern AMAS4Opt pour son *Agent Trajectoire*. Initialement, l'*Agent Partie* correspond exactement à une partie de trajectoire originale provenant de la base de données de trajectoires apprise par EVAA. Au cours de l'adaptation de l'*Agent Partie*, cette partie de trajectoire peut être modifiée afin de générer une meilleure trajectoire pour l'*Agent Trajectoire*, respectant toujours les contraintes de l'entité mobile. Sa structure est décrite par la figure 7.6.

La figure 7.6 décrit la structure de l'*Agent Partie* que nous utilisons pour décrire le reste de cet agent.

Le **but** de chaque *Agent Partie* est de définir localement une partie de trajectoire reliant les *Agents Parties* liés à lui et réaliste du point de vue physique et du point de vue du comportement. La **criticité** de cet agent, $Crit_{part}$, est décomposée en trois sous-objectifs :

1. La capacité de définir une partie de trajectoire **complète**, notée $Crit_{part,comp}$. Cette capacité s'exprime par la jonction avec les *Agents Parties* précédent et suivant.
2. La définition d'une partie de trajectoire réaliste du point de vue du comportement dans le **trafic**, notée $Crit_{part,traff}$.
3. La **faisabilité** de la partie de trajectoire, notée $Crit_{part,feas}$.

Les criticités $Crit_{part,comp}$ et $Crit_{part,traff}$ sont calculées selon au moins 4 axes, d'autres

Algorithme 7.5 : Comportement d'un *Agent Trajectoire* dans l'état Recherche Traj

1: **si** $Crit_{sit} \neq \overrightarrow{0}$ ou $Crit_{traj} \neq \overrightarrow{0}$ **alors** // Si la criticité de son *Agent Situation* et de sa trajectoire n'est pas nulle
2: **pour tout** Caracteristique c ∈ caracteristiquesSituation **faire** // L'*Agent Trajectoire* demande la criticité des caractéristiques ($Crit_{sit}$) qui lui sont déléguées aux entités passives Caractéristiques
3: getCriticiteCar(c)
4: **fin pour**
5: **pour tout** AgentPartie a ∈ agentsPartie **faire** // L'*Agent Trajectoire* demande à ses *Agents Parties* de modifier leur trajectoire pour améliorer la criticité de sa trajectoire $Crit_{traj}$
6: envoyerDemandeChangementPartie(a)
7: **fin pour**
8: **sinon**
9: etatCourant ← Satisfait // L'*Agent Trajectoire* passe dans l'état *Satisfait* car sa criticité et celle de son *Agent Situation* sont nulles
10: **fin si**
11: **pour tout** AgentTrajectoire a ∈ agentsTrajectoire **faire**
12: envoyerCriticiteTraj(a) // L'*Agent Trajectoire* envoie sa criticité aux autres *Agents Trajectoires* de son *Agent Situation*
13: envoyerTrajectoire(a) // L'*Agent Trajectoire* envoie sa trajectoire aux autres *Agents Trajectoires* de son *Agent Situation*
14: **fin pour**
15: **si** $agentExtremite[0] \neq null \wedge agentExtremite[1] \neq null$ **alors** // Si sa trajectoire est suffisamment complète, l'*Agent Trajectoire* cherche des *Agents Extrémités* d'où partira et arrivera l'*Agent EM* qui suivra sa trajectoire, puis leur demande de créer cette entité
16: **si** $agentEM = null$ **alors**
17: envoyerDemandeCreationAgentEM(agentExtremite[0]) // L'*Agent Trajectoire* demande à l'*Agent Extrémité* de départ de créer une *Agent EM*
18: **sinon**
19: **pour tout** AgentExtremite ae ∈ agentsExtremite **faire** // L'*Agent Trajectoire* envoie sa trajectoire à ses deux *Agents Extrémités*
20: envoyerTrajectoire(ae)
21: **fin pour**
22: **fin si**
23: **sinon**
24: choisirAgentExtremite(trajectoire) // S'il le peut, l'*Agent Trajectoire* décide de l'*Agent Extrémité* d'où part et où arrive sa trajectoire
25: **fin si**
26: envoyerCriticiteTraj(agentSituation) // L'*Agent Trajectoire* envoie sa criticité à son *Agent Situation*
27: envoyerTrajectoire(agentSituation) // L'*Agent Trajectoire* envoie sa trajectoire à son *Agent Situation*
28: envoyerTrajectoire(agentEM)// L'*Agent Trajectoire* envoie sa trajectoire à son *Agent EM*

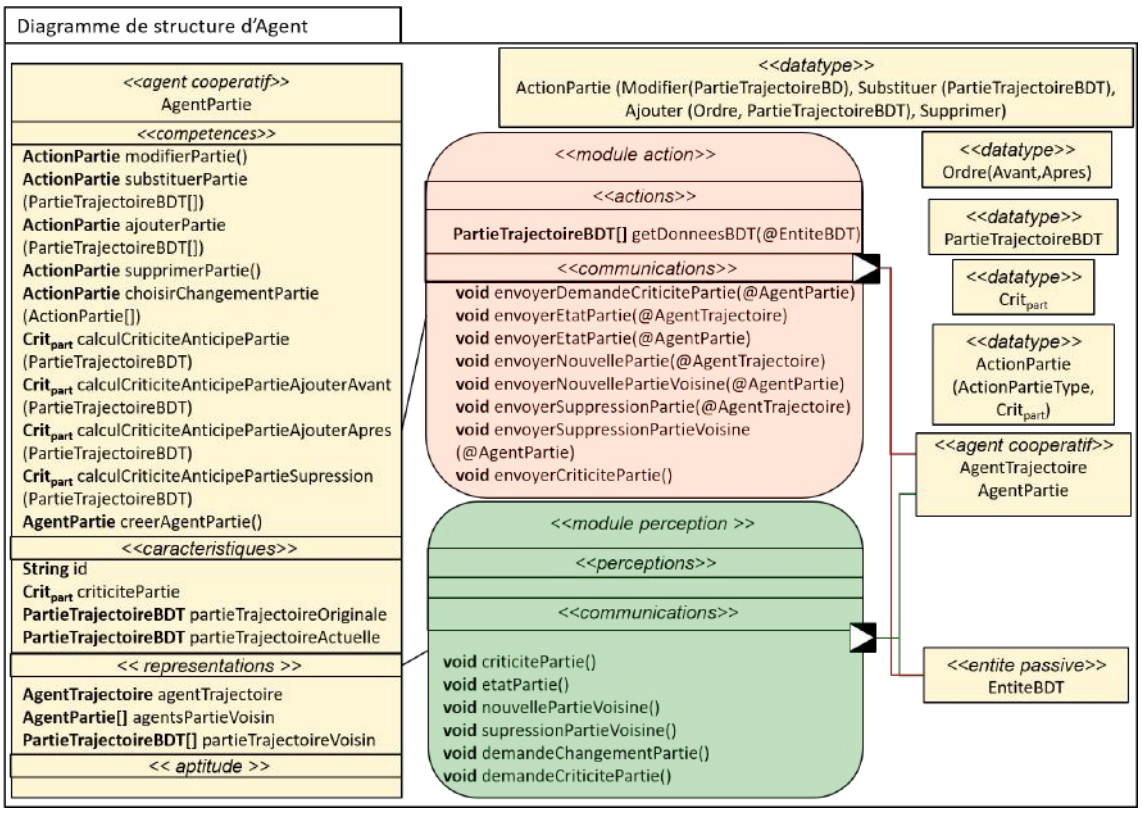


Figure 7.6 – Structure de l'*Agent Partie*

pouvant être ajoutés en fonction des applications, mais ces distances sont les bases nécessaire à la définition de ces criticités (l'altitude n'est pas toujours nécessaire, mais existe aussi dans le trafic routier, puisque celui-ci joue aussi avec l'altitude, par exemple les échangeurs, les ponts ou encore les parking à plusieurs étages) :

1. la distance *géographique*,
2. la distance *temporelle*,
3. la distance en *altitude*,
4. la distance de *vitesse*.

Dans *AGATS*, les parties de trajectoires sont des segments caractérisés, entre autres, par :

— une position géographique de départ et d'arrivée en longitude et latitude,
— une altitude de départ et d'arrivée en pied,
— un intervalle de temps durant lequel l'avion a utilisé ce segment,
— un modèle d'avion,
— un indicatif de vol, qui contient notamment la compagnie aérienne.

Ainsi, les criticités $Crit_{part,comp}$ et $Crit_{part,traff}$ sont composées d'au moins 4 sous-criticités que nous notons $Crit_{part,comp,l}$ et $Crit_{part,traf,l}$, avec $l \in [\![1,4]\!]$, selon l'énumération précédente.

Ces différentes criticités se calculent en utilisant la fonction générique de criticités décrite par l'équation 7.2, où $d_{max,l}$ est la valeur maximale de distance prise en compte pour l'axe l,

et d_l est une fonction de distance pour l'axe l, appliquée à 2 variables v_1, la valeur a obtenir, et v_2 la valeur actuelle.

$$C(x) = \begin{cases} \frac{1000}{d_{max,l}} d_l(v_1, v_2) & \text{if } d_l(v_1, v_2) < d_{max,l} \\ 1000 & \text{if } d_l(v_1, v_2) \geq d_{max,l} \end{cases} \tag{7.2}$$

Concrètement, les valeurs considérées pour chaque axe sont :

1. d_1 est la distance en kilomètres entre 2 points géographiques (v_1 et v_2) en longitude et latitude, et $d_{max,1} = 60km$, une distance utilisée pour déterminer le voisinage dans EVAA [Rantrua, 2017].
2. d_2 est la distance en secondes entre 2 dates (jour, heure, secondes) modulo le jour, et $d_{max,2} = 3600s$, le temps d'une simulation d'apprentissage pour un contrôleur,
3. d_3 est la distance entre 2 altitudes en m, et $d_{max,3} = 3048m = 10000ft$, la moitié de la hauteur de l'espace aérien inférieur, qui est suffisamment significative pour se traduire par un changement de typologie de trafic (voir section 1.2.1, en particulier figure 1.3),
4. d_4 est la distance entre 2 vitesses en kilomètres par heure, et $d_{max,4}$ est $450km.h^{-1}$, la moitié de la moyenne de vitesse de croisière des avions commerciaux.

Pour chaque *Agent Partie*, la criticité de complétude de sa partie selon chaque axe, $Crit_{part,comp,l}$, est composée de 2 criticités, dont le calcul se fait selon la fonction 7.2 :

— la distance d_l selon l'axe l, entre le point d'origine de la partie de trajectoire de l'*Agent Partie* (v_1), et la destination de l'*Agent Partie* précédant l'*Agent Partie* (v_2). Cette criticité est noté $Crit_{part,comp,l,prev}$,

— la distance d_l selon l'axe l, entre le point de destination de la partie de trajectoire de l'*Agent Partie* (v_1), et l'origine de l'*Agent Partie* suivant l'*Agent Partie* (v_2). Cette criticité est noté $Crit_{part,comp,l,foll}$.

Il est informé de l'état de la partie de ses deux *Agents Parties* voisins (*AgentPartie[] partieTrajectoireVoisin*) par des envois de messages (*void envoyerEtatPartie(@AgentPartie)*)

La figure 7.7 illustre ces criticités pour l'*Agent Partie* AP_6 en représentant les distances utilisées pour $Crit_{part,comp,l,foll}$ ($d_{1,foll}$), et pour $Crit_{part,comp,l,prev}$ ($d_{1,prev}$) pour l'axe des distances géographiques.

La criticité de l'*Agent Trajectoire* par rapport à la complétude de sa trajectoire ($Crit_{traj,comp}$) est donc formée par la criticité de chaque *Agent Partie* le composant (figure 7.8).

Le calcul de $Crit_{part,traff}$ se fait par comparaison de la partie de trajectoire actuelle de l'*Agent Partie* (*PartieTrajectoireBDT partieTrajectoireActuelle*) avec la partie de trajectoire originale (*PartieTrajectoireBDT partieTrajectoireOriginale*), c'est-à-dire celle qui vient de la base de données de trajectoires : plus la partie de trajectoire actuelle est proche de la partie de trajectoire originale, plus la criticité est basse et plus la partie de trajectoire est réaliste du point de vue du comportement. Ainsi, la partie de trajectoire contient le réalisme du comportement. Pour évaluer cette différence, nous utilisons les 4 même axes et les 4 même distances que précédemment, et nous avons donc une criticité $Crit_{part,traff,l}$ associée à chaque axe.

Chaque $Crit_{part,traff,l}$ est calculée en utilisant une fonction dérivée de la fonction générique 7.2. d_l ($l \in [\![1,4]\!]$), et appliquée à 2 couples puis sommée pour calculer la criticité,

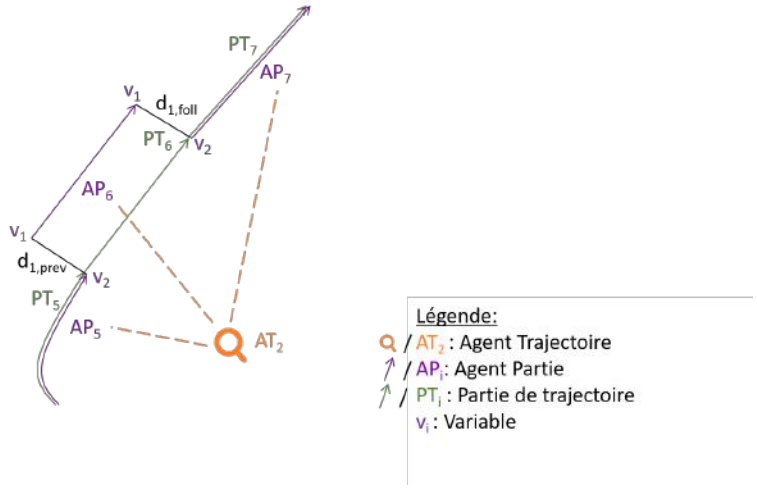


Figure 7.7 – Illustration des distances utilisées pour calculer la criticité de complétude selon l'axe géographique, $Crit_{part,comp,1}$.

$$Crit_{traj,comp}\left\{\begin{array}{l}\left.\begin{array}{l}Crit_{part,comp}\left\{\begin{array}{l}Crit_{part,comp,1}\left\{\begin{array}{l}\underbrace{\begin{array}{l}Crit_{part,comp,1,prev}\\Crit_{part,comp,1,foll}\end{array}}_{\text{Les 2 composantes de } Crit_{part,comp,1}}\end{array}\right.\\Crit_{part,comp,2}\\Crit_{part,comp,3}\\Crit_{part,comp,4}\end{array}\right.\end{array}\right\}\text{Expression de } Crit_{part,comp} \text{ selon les 4 axes}\\ \vdots\end{array}\right.}_{\text{Décomposition de } Crit_{traj,comp} \text{ selon les } \textit{Agents Parties} \text{ de l'} \textit{Agent Trajectoire}}$$

Figure 7.8 – Une vue hiérarchique de la criticité de l'*Agent Trajectoire* par rapport à la complétude de sa trajectoire, $Crit_{traj,comp}$

selon la fonction 7.3. Le premier couple, (v_1, v_2), correspond au départ de la partie de trajectoire actuelle de l'*Agent Partie* (v_1) et au départ de la partie de trajectoire originale (v_2). Le deuxième couple correspond à la destination de la partie de trajectoire de l'*Agent Partie* (v_3) et à la destination de la partie de trajectoire originale (v_4).

$$C(x) = \begin{cases} \frac{1000}{d_{max,l}}(d_l(v_1,v_2) + d_l(v_3,v_4)) & \text{if } (d_l(v_1,v_2) + d_l(v_3,v_4)) < d_{max,l} \\ 1000 & \text{if } (d_l(v_1,v_2) + d_l(v_3,v_4)) \geq d_{max,l} \end{cases} \tag{7.3}$$

La figure 7.9 illustre la valeur des deux distances utilisées pour le calcul de $Crit_{part,traff,1}$ pour l'axe des distances géographiques.

La criticité de l'*Agent Trajectoire* par rapport au réalisme de sa trajectoire du point de

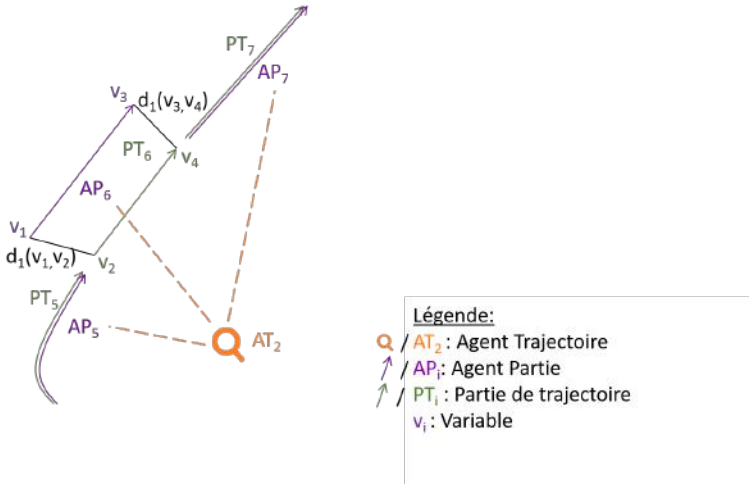


Figure 7.9 – Illustration des distances utilisées pour calculer la criticité de l'*Agent Partie* par rapport au réalisme de sa partie de trajectoire du point de vue du comportement, selon l'axe géographique, noté $Crit_{part,traff,1}$.

$$Crit_{traj,traff} \left\{ \begin{array}{l} \underbrace{Crit_{part,traff} \left\{ \begin{array}{l} Crit_{part,traff,1} \\ Crit_{part,traff,2} \\ Crit_{part,traff,3} \\ Crit_{part,traff,4} \end{array} \right.}_{\text{Expression de } Crit_{part,traff} \text{ selon les 4 axes}} . \\ . \\ . \end{array} \right.$$

Décomposition de $Crit_{traj,traff}$ selon les *Agents Parties* de l'*Agent Trajectoire*

Figure 7.10 – Une vue hiérarchique de la criticité de l'*Agent Trajectoire* par rapport au réalisme comportemental de sa trajectoire, $Crit_{traj,traff}$

vue du comportement ($Crit_{traj,traff}$) est donc formée par la criticité de chaque *Agent Partie* le composant (figure 7.10).

La criticité $Crit_{part,feas}$ est quant à elle calculée par l'*Agent EM* suivant la trajectoire de son *Agent Trajectoire*.

Afin d'atteindre ces différents buts, et de faire diminuer sa criticité ($Crit_{part}$ *criticitePartie*) l'*Agent Partie* est capable d'effectuer 4 **actions** différentes :

— Action 1 : Modifier les caractéristiques (*ActionPartie modifierPartie()*) de sa partie de trajectoire (*PartieTrajectoireBDT partieTrajectoireActuelle*), par exemple augmenter la vitesse de l'entité mobile qui suit sa partie de trajectoire.

— Action 2 : Substituer sa partie de trajectoire par une autre partie de trajectoire venant de la base de données de trajectoire (*ActionPartie substituerPartie(PartieTrajectoireBDT[])*).

— Action 3 : Ajouter un nouvel *Agent Partie* (*ActionPartie ajouterPartie(PartieTrajectoireBDT[])*), cette action peut être faite pour compléter la trajectoire, ou

pour l'étendre. Il peut ajouter un *Agent Partie* avec une partie de trajectoire spécifique avant ou après sa propre partie de trajectoire, en utilisant une partie de trajectoire venant de la base de données de trajectoires.

— Action 4 : Se supprimer (*ActionPartie supprimerPartie()*), cette action est exécutée lorsque l'*Agent Partie* considère qu'il est plus coopératif de laisser les *Agents Parties* précédant et suivant se rejoindre plutôt que d'effectuer l'une des trois actions précédentes.

Pour l'action 1, la **modification** d'une partie de trajectoire (*ActionPartie modifierPartie()*) se fait selon un seul des 4 axes utilisés suivants :

1. Une modification selon l'axe **géographique** se fait par translation de la partie de trajectoire, ou sa rotation, préservant ainsi la longueur et la forme de la partie de trajectoire, et maintient donc le réalisme physique. Le pseudo algorithme 7.6 décrit la translation d'un segment d'EVAA (illustrée par la figure 7.11), et le pseudo algorithme 7.7 décrit la rotation (illustré par la figure 7.12), puisque ces opérations sont moins triviales sur une sphère. Le point utilisé comme centre de la rotation est le milieu de la partie de trajectoire de l'*Agent Partie*
2. Une modification selon l'axe **temporel** se fait en modifiant le temps pour toute la partie de trajectoire.
3. Une modification selon l'axe de **l'altitude** se fait en modifiant la hauteur pour toute la partie de trajectoire.
4. Une modification selon l'axe de **la vitesse** se fait en augmentant ou en diminuant la vitesse sur toute la partie de trajectoire.

Algorithme 7.6 : Translation d'un segment d'EVAA

1: Choisir l'un des deux points extrêmes du segment, noté $\overrightarrow{p_1}$, disons le départ (respectivement l'arrivée) du segment, et translater ce point dans la direction voulue et la distance voulue. Nous notons le point translaté $\overrightarrow{p_3}$.
2: Translater $\overrightarrow{p_3}$ selon l'orientation (respectivement l'orientation contraire) et la longueur du segment. Nous notons le point translaté $\overrightarrow{p_4}$

Algorithme 7.7 : Rotation d'un segment d'EVAA par un point quelconque

1: Choisir le point de translation, généralement celui-ci appartient au segment, noté $\overrightarrow{p_{rot}}$.
2: Calculer la distance d_1 entre $\overrightarrow{p_{rot}}$ et le départ du segment $\overrightarrow{p_1}$
3: Calculer l'orientation α_1 de $\overrightarrow{p_{rot}p_1}$
4: Calculer l'orientation $\alpha'_1 = \alpha_1 + angleRotation$ de $\overrightarrow{p_{rot}p_1}$, avec $\overrightarrow{p_3}$ le futur point du segment translaté.
5: Translater $\overrightarrow{p_{rot}}$ selon l'orientation α'_1 et la distance d_1. Ce point translaté correspond à $\overrightarrow{p_3}$.
6: Translater $\overrightarrow{p_3}$ selon l'orientation $orientationSegment + angleRotation$ et la distance du segment. Ce point translaté correspond à $\overrightarrow{p_4}$.

L'*Agent Partie* modifie selon l'axe dont la criticité est maximale (parmi tous les axes) dans le but de diminuer celle-ci (algorithme 7.8).

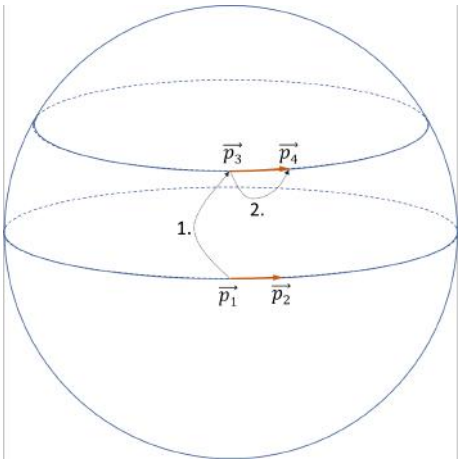


Figure 7.11 – Illustration de translation d'une partie de trajectoire (les étapes correspondent à celles de l'algorithme 7.6)

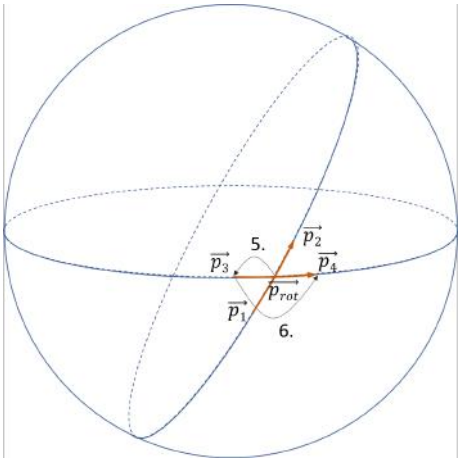


Figure 7.12 – Illustration de la rotation d'une partie de trajectoire (les étapes correspondent à celles de l'algorithme 7.7)

Pour l'action 2, la substitution d'une partie de trajectoire par une autre se fait en requérant des parties de trajectoires proches dans la base de données de trajectoire (*PartieTrajectoireBDT[] getDonneesBDT()*), en évaluant chacune de ces parties de trajectoire ($Crit_{part}$ *calculerCriticiteAnticipePartie(PartieTrajectoireBDT*)), et en choisissant celle qui fait diminuer le plus la criticité maximale actuelle (algorithme 7.9).

L'évaluation de l'action 3 d'ajout d'une nouvelle partie de trajectoire entre un *Agent Partie* et l'un de ses deux voisins, se fait par l'évaluation de la somme des criticités de ces 2 *Agents Parties* et sa comparaison avec la somme des deux criticités anticipées des *Agents Parties* avec celle du nouveau *Agent Partie* (algorithme 7.10). L'évaluation de l'ajout se fait en amont de l'*Agent Partie* dans la trajectoire ($Crit_{part}$ *calculerCriticiteAnticipePartieAjouterAvant (PartieTrajectoireBDT)*), et en aval ($Crit_{part}$ *calculerCriticiteAnticipePartieAjouterApres (PartieTrajectoireBDT)*), afin d'évaluer les différentes possibilités. La figure 7.13 illustre l'ajout en

Algorithme 7.8 : Modification d'une partie de trajectoire d'un *Agent Partie*

1: PartieTrajectoire pt
2: pt ← modifierPartie(partieTrajectoireActuelle) // L'*Agent Partie* calcule la modification de sa partie de trajectoire actuelle
3: Renvoyer new ActionPartie(Modifier(pt), calculeCriticiteAnticipePartie(pt)) // L'*Agent Partie* calcule sa criticité anticipée s'il effectue cette action

Algorithme 7.9 : Substitution d'une partie de trajectoire d'un *Agent Partie*

1: PartieTrajectoireBDT[] ps
2: $Crit_{part}$[] cs
3: ps ← getDonnesBDT() // L'*Agent Partie* cherche dans la base de données de trajectoires des parties de trajectoires qui pourraient convenir à la substitution
4: **pour** k ← 1 à ps.taille() **faire**
5: cs[k] ← calculerCriticiteAnticipePartie(ps[k]) // L'*Agent Partie* calcule la criticité anticipée de la substitution de la partie de trajectoire partieTrajectoireOriginale par la partie de trajectoire ps[k]
6: **fin pour**
7: Soit l_{min} / cs[l_{min}] = $\min_l cs[l]$ // L' *Agent Partie* détermine la partie de trajectoire qui fait le plus diminuer la criticité
8: Renvoyer new ActionPartie(Substitution(ps[l_{min}]), cs[l_{min}])

amont (AP'_5) et en aval (AP'_7) d'un *Agent Partie* pour l'*Agent Partie* AP_6 (à des fins de clarté, nous ajoutons une partie de trajectoire différente).

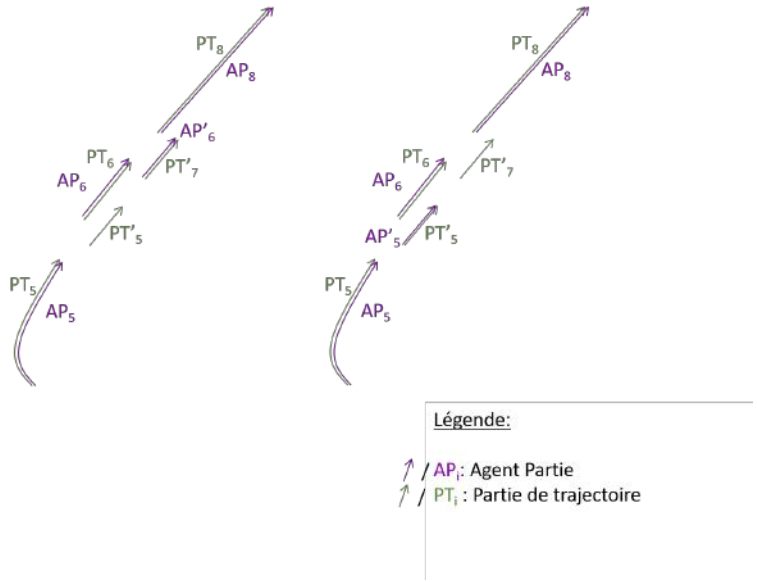


Figure 7.13 – Illustration de l'ajout d'une partie de trajectoire

L'évaluation de l'action 4 de suppression de l'*Agent Partie* se fait par évaluation de la cri-

Algorithme 7.10 : Ajout d'une partie de trajectoire d'un *Agent Partie*

1: PartieTrajectoire[] ps
2: $Crit_{part}$[] csApres
3: $Crit_{part}$[] csAvant
4: ps ← getDonnesBDT() // L'*Agent Partie* cherche dans la base de données de trajectoires des parties de trajectoire qui pourraient convenir à l'ajout
5: **pour** k ← 1 à ps.taille() **faire** // Pour chaque partie de de trajectoire ps[k], l'*Agent Partie* calcule la criticité anticipée de l'ajout de ps[s] avant ou après lui
6: csAvant[k] ← calculerCriticiteAnticipePartieAjoutAvant(ps[k])
7: csApres[k] ← calculerCriticiteAnticipePartieAjoutApres(ps[k])
8: **fin pour**
9: Soit $l_{min,avant}$ / csAvant[l_{min}] = $\min_{l} csAvant[l]$ // L'*Agent Partie* détermine la partie de trajectoire qui diminue le plus la criticité si elle est ajoutée avant lui
10: Soit $l_{min,apres}$ / csApres[l_{min}] = $\min_{l} csApres[l]$ // L'*Agent Partie* détermine la partie de trajectoire qui diminue le plus la criticité si elle est ajoutée après lui
11: **si** $csAvant[l_{min,avant}] > csApres[l_{min,apres}]$ **alors** // L'*Agent Partie* détermine lequel des deux ajouts (avant ou après) diminue le plus la criticité, et renvoi l'action appropriée
12: Renvoyer new ActionPartie(Ajouter(Apres,ps[$l_{min,apres}$]), csApres[$l_{min,apres}$])
13: **sinon**
14: Renvoyer new ActionPartie(Ajouter(Avant,ps[$l_{min,avant}$]), csAvant[$l_{min,avant}$])
15:
16: **fin si**

ticité maximale actuelle totale de l'*Agent Partie* avec celle de ses 2 voisins, et l'évaluation de la criticité totale de ses 2 voisins sans lui ($Crit_{part}$ *calculerCriticiteAnticipePartieSupprimer()*).

Tant que ses différentes sous-criticités ne sont pas nulles, l'*Agent Partie* étudie les différentes actions possibles (modifier, substituer, ajouter, se supprimer), et choisit l'action qui diminue le plus sa criticité (algorithme 7.11), applique l'action associée (modifier sa partie de trajectoire, substituer sa partie de trajectoire, ajouter une partie de trajectoire, se modifier), et indique à son *Agent Trajectoire* (*AgentTrajectoire agentTrajectoire*) et à ses deux *Agents Parties* voisins (*AgentPartie agentPartieVoisin*) son nouvel état (*void envoyerEtatPartie(@AgentPartie)*, *void envoyerEtatPartie(@AgentTrajectoire)*). Si l'action est un ajout, l'*Agent Partie* crée un nouvel agent (*AgentPartie creerAgentPartie(PartieTrajectoireBDT)*), et ce nouveau agent indiquera son existence à l'*Agent Trajectoire* (*void envoyerNouvellePartie(@AgentTrajectoire)*) et à ses *Agents Parties* voisins (*void envoyerNouvellePartieVoisine(@AgentPartie)*). Si l'action est une suppression, l'*Agents Parties* indique à son *Agent Trajectoire* et à ses deux *Agents Parties* voisins sa suppression (*void envoyerSupressionPartie(@AgentTrajectoire)* et *void envoyerSupressionPartieVoisine(@AgentPartie)*).

Algorithme 7.11 : Comportement de l'agent *Agent Partie*

```
1: PartieTrajectoireBDT[] ps
2: ActionPartie[] acs // Table contenant l'évaluation de l'ensemble des actions de l'Agent Partie
3: ActionPartie ac // Variable temporaire qui recevra l'action choisie
4: si Crit_part ≠ O→ alors // Si la criticité de l'Agent Partie est non nulle, il essaye de diminuer celle-ci
5:    ps ← getDonneesBDT()
6:    acs[1] ← modifierPartie() // L'Agent Partie calcule la meilleure modification de sa partie de trajectoire et sa criticité anticipée associée
7:    acs[2] ← substituerPartie(ps) // L'Agent Partie calcule la meilleure substitution de sa partie de trajectoire et sa criticité anticipée associée
8:    acs[3] ← supprimerPartie() // L'Agent Partie calcule la criticité anticipée de son voisinage s'il se supprime
9:    acs[4] ← ajouterPartie(ps) // L'Agent Partie calcule le meilleur ajout et la criticité anticipée de son voisinage associée
10:   ac ← choisirChangementPartie(acs) // L'Agent Partie détermine l'action qui fait le plus diminuer sa criticité anticipée
11:   ac.appliquer() // L'Agent Partie applique l'action
12:   etatPartie(agentTrajectoire) // L'Agent Partie envoi son nouvel état à son Agent Trajectoire
13:   pour tout AgentPartie a ∈ agentsPartieVoisin faire
14:      envoyerEtatPartie(a) // L'Agent Partie envoi son nouvel état à ses Agents Parties voisins
15:   fin pour
16: fin si
```

7.8 Présentation de l'Agent Extrémité

Un *Agent Extrémité* correspond à un point d'arrivée ou de départ d'une entité mobile. Une trajectoire commence forcément par un *Agent Extrémité* et se termine aussi par un *Agent Extrémité*. En fonction de la finalité de la simulation, ces points peuvent être des points réels de départ et d'arrivée c'est-à-dire un point d'où l'*Agent EM* s'arrête ou part (par exemple un aéroport, un parking), ou alors un point d'entrée ou de sortie de la simulation, c'est-à-dire les limites géographiques de l'espace simulé (par exemple les limites d'un secteur en aéronautique).

La figure 7.14 décrit la structure de l'*Agent Extrémité* utilisée dans la suite.

Un *Agent Extrémité* se caractérise par sa position dans l'espace (*Position position*), par exemple une position en longitude, latitude, altitude, et par un planning des arrivées et départs qui sont effectués depuis cet *Agent Extrémité* (*Planning planningArriveDepart*). D'autres caractéristiques peuvent compléter ces deux dernières, afin de représenter des contraintes sur l'entité que représente l'agent (*Caracteristiques[] caracteristiquesExtr*). Par exemple, il est possible d'ajouter des contraintes sur le nombre d'entités mobiles qui peuvent apparaître

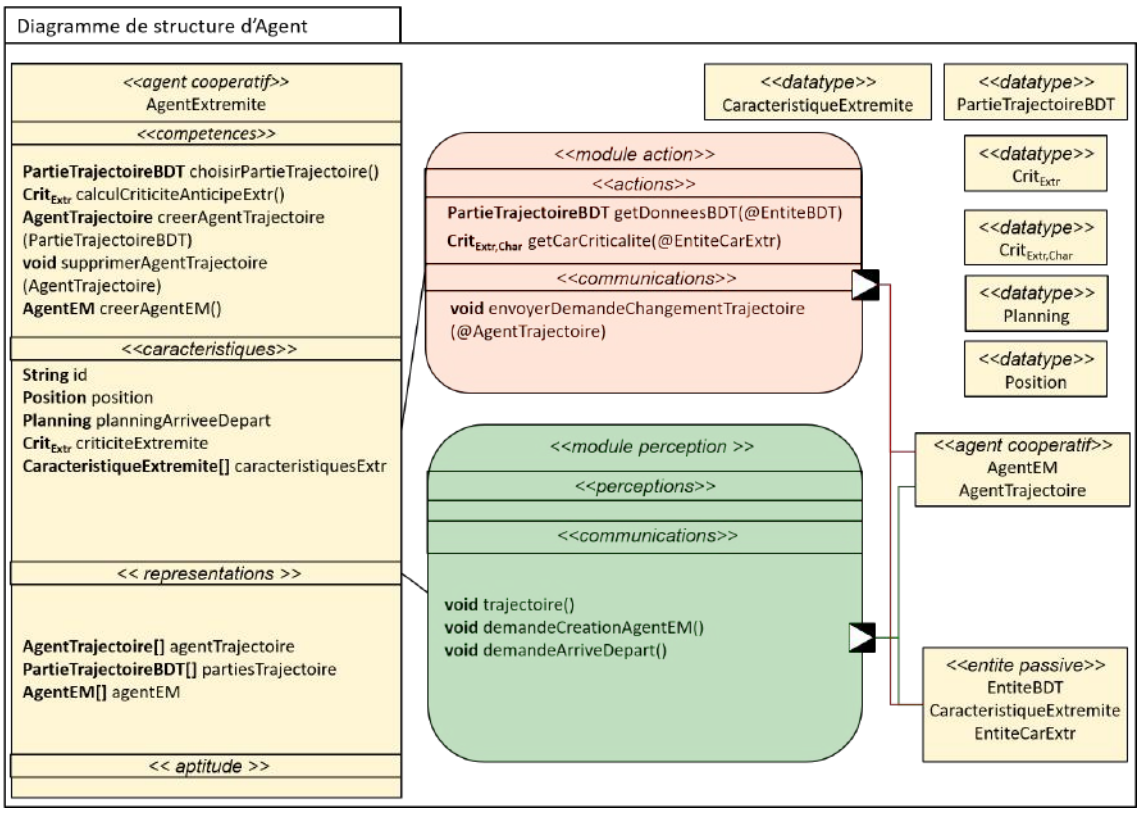


Figure 7.14 – Structure de l'*Agent Extrémité*

ou disparaître en même temps (par exemple, un aéroport avec une piste ne peut pas faire atterrir ou décoller plus d'un avion à la fois), sur leur nombre total (par exemple dans le trafic routier, un parking n'a pas un nombre illimité de places).

Son **but** est d'assurer le réalisme du trafic qui part et arrive à lui, c'est-à-dire de son planning des arrivées et des départs, mais aussi de respecter les différentes caractéristiques qui le définissent. Son but est donc à la fois d'assurer la création du trafic dit "contextuel", mais aussi que les départs et les arrivées soient réalistes pour lui, c'est-à-dire qu'ils correspondent à un trafic habituel et faisable.

Sa criticité $Crit_{extr}$ est donc composée de la criticité de son planning, $Crit_{extr,plan}$, et de la criticité de ses caractéristiques, $Crit_{extr,char}$.

$$Crit_{extr} = (Crit_{extr,plan}, Crit_{extr,char})$$

Afin de satisfaire ses buts, l'*Agent Extrémité* connaît les parties de trajectoires qui partent et arrivent à lui (*PartieTrajectoireBDT[] partiesTrajectoire*), et s'assure que chacune de ces parties de trajectoires soient utilisées dans une trajectoire, de la manière la plus réaliste possible. Pour cela, l'agent a la capacité de créer des *Agents Trajectoires* (*AgentTrajectoire creerAgentTrajectoire (PartieTrajectoireBDT)*) dont le but est de générer des trajectoires réalistes satisfaisant ses contraintes, et de générer des *Agents EM* pour parcourir ces trajectoires (*AgentEM creerAgentEM()*). Il peut également supprimer des *Agents Trajectoires* qui gêneraient la génération

du trafic dit "situationnel" (*void supprimerAgentTrajectoire(AgentTrajectoire)*), c'est-à-dire du trafic qui génère des situations requises par le scénariste, libérant ainsi des créneaux pour les *Agents EM* impliquées dans ces situations.

Dans notre système, le planning de l'*Agent Extrémité* est l'équivalent d'un ensemble de parties de trajectoires, qui correspondent à l'ensemble des arrivées et départs de cette extrémité. La criticité $Crit_{extr,plan}$ est alors un ensemble de criticités pour chacun de ces départs et arrivées, noté $Crit_{extr,char,i}$.

$$Crit_{extr,char,i} = \{Crit_{extr,char,i}\}$$

Chaque $Crit_{extr,char,i}$ est l'équivalent de la criticité de réalisme du point de vue du trafic de l'*Agent Partie* ($Crit_{part,traff}$), appliquée en un point. Ce point est le point d'arrivée, si la partie de trajectoire arrive à l'*Agent Extrémité*, ou est le point de départ, si la partie de trajectoire part à l'*Agent Extrémité*.

Ainsi, $Crit_{extr,char,i}$ est composée d'au moins 4 sous-criticités que nous notons $Crit_{extr,comp,i,l}$, $l \in [\![1,4]\!]$ selon les axes des *Agents Parties* (géographique, temporelle, altitude, vitesse). Le calcul de ces criticités se fait de la même manière que $Crit_{part,traffl}$, selon la fonction 7.4 appliquée au couple (v_1, v_2), qui correspond au point d'arrivée de la partie de trajectoire actuelle (respectivement de départ) (v_1), telle que l'utilise l'*Agent Trajectoire* et par extension l'*Agent Partie* (la trajectoire est transmise), et au point d'arrivée de la partie de trajectoire originale (respectivement de départ) (v_2). La figure 7.15 illustre la distance $d_1(v_1, v_2)$ pour l'axe géographique.

$$C(x) = \begin{cases} \frac{1000}{d_{max,l}}(d_l(v_1, v_2)) & \text{if } (d_l(v_1, v_2)) < d_{max,l} \\ 1000 & \text{if } (d_l(v_1, v_2)) \geq d_{max,l} \end{cases} \tag{7.4}$$

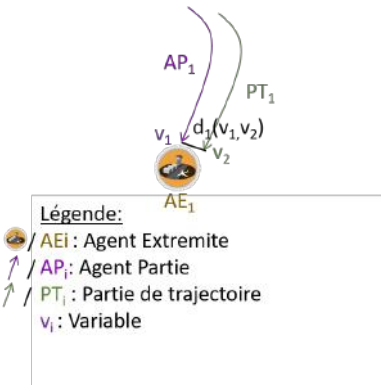


Figure 7.15 – Illustration des distances utilisées pour calculer la criticité de planification l'*Agent Extrémité* pour l'arrivée *i* selon l'axe géographique, $Crit_{extr,char,i}$

Le calcul de la criticité $Crit_{extr,char}$ dépend de la nature de l'extrémité, et donc de son implémentation.

7.9 Présentation de l'Agent Entité Mobile

L'*Agent EM* évolue uniquement durant la *phase de simulation*. Son **but** est soit d'étudier la faisabilité d'une trajectoire τ_i donnée par son *Agent Trajectoire*, soit de participer à la création d'un trafic contextuel, le tout en évitant d'entrer en collision avec les autres *Agents EM*, et en notifiant son *Agent Trajectoire* de sa difficulté à suivre sa trajectoire. Pour cela, l'*Agent EM* évolue dans son environnement en fonction de ce qu'il perçoit dans sa zone de perception.

La **zone de perception** de l'*Agent EM* est l'espace autour de sa position dans lequel l'*Agent Entité Mobile*$_i$ *(Agent EM*$_i$*)* percoit les *Agent Entité Mobile*$_j$ *(Agent EM*$_j$*)* voisins. Nous notons Zp_i la zone de perception de l'*Agent EM*$_i$. Ainsi cette zone doit être définie pour permette à l'*Agent EM* d'effectuer correctement sa tâche qui est d'atteindre sa destination tout en évitant les collisions. Dans notre système, elle est définie par la portée des échanges de messages.

Afin d'atteindre ses buts, l'*Agent EM* utilise ses différentes compétences, qui sont de 2 types : la communication et le déplacement. De par les différentes contraintes physiques qui régissent leurs mouvements, les *Agents EM* modifient leur trajectoire dans une certaine enveloppe. Par exemple, une voiture classique ne passe pas de 0 à 100 $km.h{-}1$ en moins d'une seconde. De même, la plupart des avions ne peuvent pas voler sans vitesse horizontale en vol soutenu.

Pour prendre en compte ses contraintes, l'*Agent EM* effectue des actions qui correspondent à ses capacités. Les actions qu'il peut effectuer correspondent à différentes sous-actions sur ses différents degrés de liberté. Par exemple, un avion peut ralentir et tourner en même temps.

Dans notre approche, nous ne considérons pas tout l'espace des actions possibles par l'entité mobile, qui est infini puisque la majorité (sinon la totalité) des dimensions sur lesquelles l'*Agent EM* peut agir sont des espaces continus (comme par exemple, la vitesse). Ainsi, nous considérons un échantillonage des actions possibles à l'instant t, qui est utilisé pour décider de la meilleure action à réaliser. Par exemple, dans le domaine aéronautique, les actions consistent en toutes les combinaisons possibles entre une modification de vitesse (accélérer, ne pas modifier la vitesse, ralentir), une modification d'orientation (tourner à gauche, ne pas tourner, tourner à droite), et une modification de l'altitude (monter, ne pas modifier l'altitude, descendre) (figure 7.16).

Dans ce qui suit, nous notons Ac_i l'ensemble des actions de l'*Agent EM*$_i$.

$$Ac_i = \left\{ Ac_{i,k} \right\}_{0<k\leq n} \text{ (n étant le nombre d'actions possibles)}$$

Les **communications** entre les *Agents EM* ont été volontairement limitées en contenu. Le contenu de ces messages est le suivant :

— La position de l'entité mobile M_i, notée $\overrightarrow{p_i}$.

— Le vecteur vitesse de l'entité mobile M_i, noté $\overrightarrow{v}_i$.

— La criticité de l'entité mobile (détaillée dans la section 7.9.1), notée $Crit_i$.

Les *Agents EM* communiquent entre eux dans leur zone de perception Zp_i, mais ne

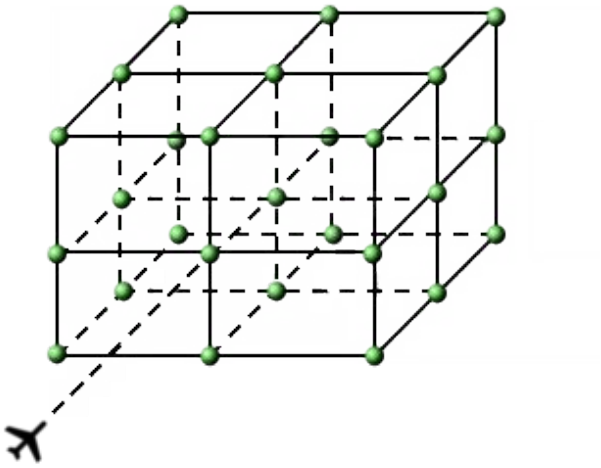


Figure 7.16 – Un avion et ses différentes actions

dialoguent pas, c'est-à-dire que les messages qu'ils s'envoient ne sont pas des réponses à d'autres messages.

Le comportement de l'*Agent EM* suit un cycle de vie présenté par l'algorithme 7.12.

Algorithme 7.12 : Cycle de vie d'un *Agent* EM_i le long de sa trajectoire

1: **repeter**
2: *Perception* : Récupérer la criticité et la situation dynamique actuelle (position+vecteur vitesse) des *Agents* EM_j voisins se trouvant dans sa zone de perception notée Zp_i;
3: *Décision* : Calculer la criticité de chaque action possible $Ac_{i,k}$ et décider de l'action qui minimise la criticité des voisins et la sienne;
4: *Action* : Effectuer l'action décidée $Ac_{i,k_{decided}}$ et informer le voisinage de sa nouvelle criticité et sa nouvelle situation;
5: **tant que** M_i n'est pas arrivé à destination

L'*Agent EM* effectue ce cycle de vie jusqu'à ce qu'il atteigne sa destination. Le calcul des différentes mesures de criticités utilisées et le fonctionnement détaillé de son raisonnement est décrit dans les sections suivantes.

7.9.1 Criticité d'un Agent EM $Crit_i$

Nous présentons dans cette section les différentes mesures permettant de calculer la criticité utilisée par les *Agents EM*, en partant de la criticité la plus générale pour aller vers la plus spécifique. Cette hiérarchie de criticité est résumée dans la figure 7.17.

L'*Agent* EM_i tel que nous l'avons défini, a 2 buts, qui sont de suivre sa trajectoire et d'éviter les collisions. Ainsi la criticité de l'Agent EM_i se définit comme un couple :

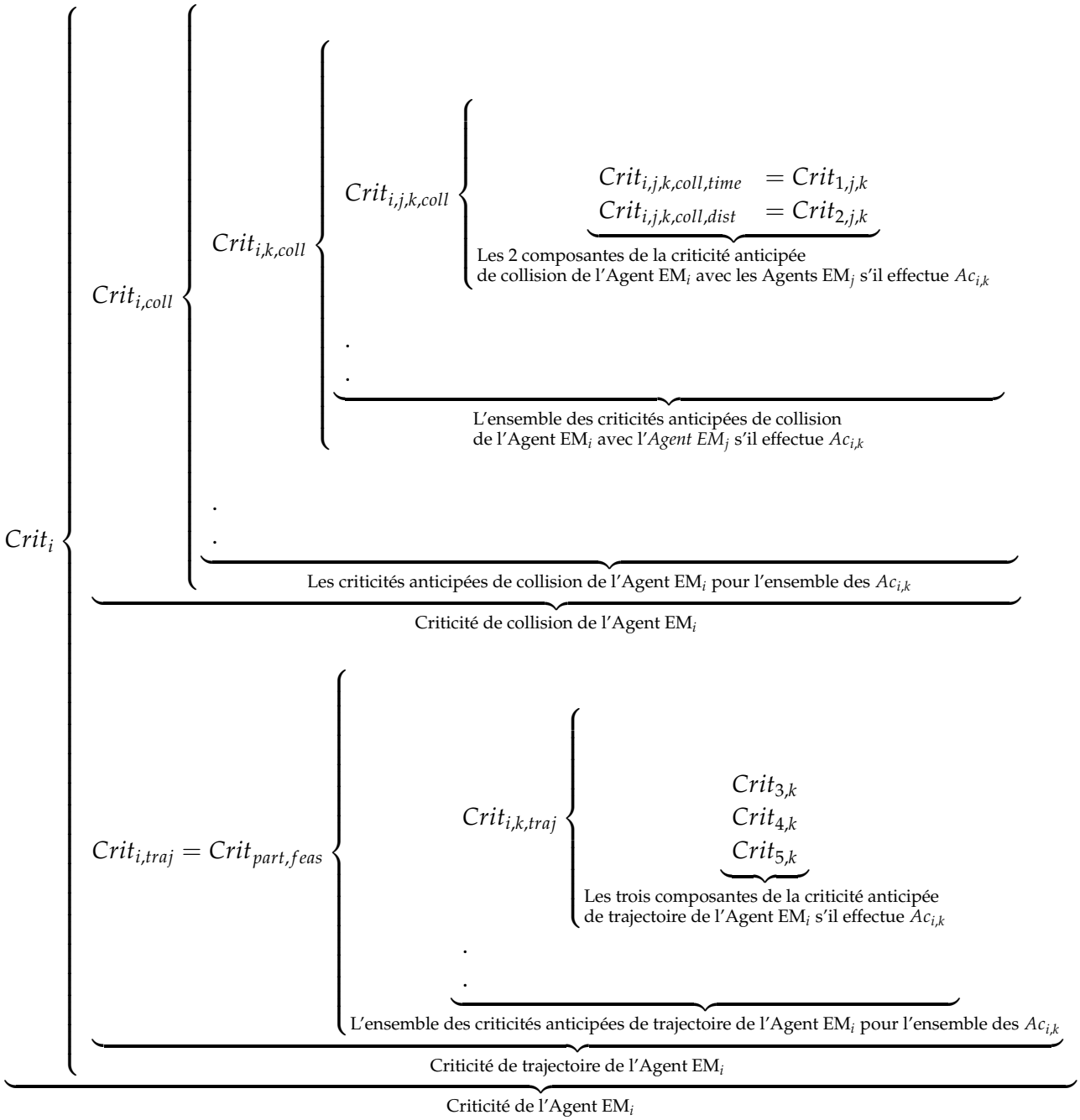


Figure 7.17 – Une vue hiérarchique des criticités de l'*Agent EM$_i$*

$$Crit_i = (Crit_{i,traj}, Crit_{i,coll})$$

Avec :

— $Crit_{i,traj}$, la criticité de l'*Agent EM*$_i$ par rapport à la trajectoire,

— $Crit_{i,coll}$, la criticité de l'*Agent EM*$_i$ par rapport à la collision.

La criticité à un instant t de l'*Agent EM*$_i$ correspond à la criticité de l'action qu'il effectue, ainsi, nous avons :

$$Crit_{i,traj} = Crit_{i,k_{decided},traj}$$

Et,

$$Crit_{i,coll} = Crit_{i,k_{decided},coll}$$

Avec $k_{decided}$, le numéro de l'action $Ac_{i,k_{decided}}$ que l'*Agent EM*$_i$ effectue. La criticité $Crit_{i,traj}$ qui correspond à la criticité $Crit_{part,feas}$ est envoyée à son l'*Agent Trajectoire*, qui la renvoie à l'*Agent Partie* correspondant.

L'*Agent EM*$_i$ décide à chaque cycle de vie d'une action parmi l'ensemble d'actions $Ac_{i,k}$. Pour cela, il évalue la criticité des différentes actions $Ac_{i,k}$ selon les 2 critères précédents, à savoir la collision et le suivi de trajectoire. Nous notons :

— $Crit_{i,k,traj}$, la criticité de l'*Agent EM*$_i$ par rapport à la trajectoire s'il effectue l'action $Ac_{i,k}$.

— $Crit_{i,k,coll}$, la criticité de l'*Agent EM*$_i$ par rapport à la collision s'il effectue l'action $Ac_{i,k}$.

Enfin, la criticité anticipée de l'*Agent EM*$_i$ s'il effectue l'action $Ac_{i,k}$ est notée

$$Crit_{i,k} = (Crit_{i,k,traj}, Crit_{i,k,coll})$$

La criticité anticipée de trajectoire de l'*Agent EM*$_i$ par rapport à la trajectoire s'il effectue l'action $Ac_{i,k}$, $Crit_{i,k,traj}$, correspond à l'éloignement entre l'*Agent EM*$_i$ et sa trajectoire voulue τ_i donnée par son *Agent Trajectoire*. Cette criticité est communiquée ensuite à l'*Agent Trajectoire*, qui la communique à l'*Agent Partie* concerné. S'il le peut, c'est-à-dire si cela n'implique pas de rapprochement trop important entre son entité mobile et un obstacle, l'*Agent EM*$_i$ suit sa trajectoire.

Suivre la trajectoire τ_i revient à suivre les différentes parties de la trajectoire qui la composent. Ainsi, la criticité de la trajectoire est évaluée par rapport à la partie de la trajectoire qui est suivie actuellement par l'*Agent EM*$_i$. Cette criticité, $Crit_{i,k,traj}$, est composée de 3 criticités, $Crit_{3,k}$, $Crit_{4,k}$, $Crit_{5,k}$:

$$Crit_{i,k,traj} = (Crit_{3,k}, Crit_{4,k}, Crit_{5,k})$$

Le calcul de ces 3 criticités anticipées dépend de la notion de partie de trajectoires utilisée (voir chapitre 8 pour un exemple). Dans *AGATS*, nous considérons que ce sont des segments, néanmoins, l'utilisation et le sens de ces criticités restent les mêmes si la partie de trajectoire change de nature, et nous considérons que toute partie de trajectoire peut se découper en segments :

— $Crit_{3,k}$ est fonction de la distance entre la partie de la trajectoire et l'*Agent EM*$_i$ au prochain cycle de vie $(t+1)$ si l'*Agent EM*$_i$ effectue l'action $Ac_{i,k}$. $Crit_{3,k}$ sert à rapprocher l'*Agent EM*$_i$ de sa trajectoire.

$$Crit_{3,k} = \min_{\overrightarrow{x} \in [\overrightarrow{p_{start}}, \overrightarrow{p_{end}}]} \{||\overrightarrow{p_{i,k}}(t+1) - \overrightarrow{x}||\}$$

L'algorithme 7.13 permet de calculer le point $\overrightarrow{x}$ pour lequel la distance $||\overrightarrow{p_{i,k}}(t+1) - \overrightarrow{x}||$ est minimale.

— $Crit_{4,k}$ est fonction de la distance entre l'*Agent EM*$_i$ et la destination de la partie de la trajectoire $\overrightarrow{p_{end}}$ au prochain cycle de vie $(t+1)$ si l'action $Ac_{i,k}$ est effectuée. $Crit_{4,k}$ sert à rapprocher l'*Agent EM*$_i$ de la fin de la partie de la trajectoire.

$$Crit_{4,k} = ||\overrightarrow{p_{i,k}}(t+1) - \overrightarrow{p_{end}}||$$

— $Crit_{5,k}$ est fonction de l'angle α_k entre la partie de trajectoire et le vecteur vitesse de *Agent EM*$_i$ au prochain cycle de vie $(t+1)$ si l'action $Ac_{i,k}$ est effectuée. $Crit_{5,k}$ sert à orienter l'*Agent EM*$_i$ dans le bon sens de la trajectoire.

$$Crit_{5,k} = \begin{cases} 0 & \text{if } \alpha_k < \frac{\pi}{2} \\ \frac{100}{\frac{\pi}{2}} \times (\alpha_k - \frac{\pi}{2}) & \text{if } \alpha_k \geq \frac{\pi}{2} \end{cases}$$

Algorithme 7.13 : Calcul de la projection généralisée d'un point sur un segment

1: Calculer $\overrightarrow{q_{i,k}}(t+1)$ le projeté de $\overrightarrow{p_{i,k}}(t+1)$ sur la droite D contenant le segment ;
2: **si** $\overrightarrow{q_{i,k}}(t+1) \in [\overrightarrow{p_{start}}, \overrightarrow{p_{end}}]$ **alors**
3: Retourner $\overrightarrow{q_{i,k}}(t+1)$;
// La projection de $\overrightarrow{p_{i,k}}(t+1)$ sur la droite appartient au segment, comme la projection de A dans la figure 7.18
4: **sinon si** $||\overrightarrow{q_{i,k}}(t+1) - \overrightarrow{p_{start}}|| > ||\overrightarrow{q_{i,k}}(t+1) - \overrightarrow{p_{end}}||$ **alors** // Sinon, l'algorithme renvoie l'extrémité la plus proche
5: Retourner $\overrightarrow{p_{end}}$;
// L'extrémité la plus proche est $\overrightarrow{p_{end}}$, comme pour A" dans la figure 7.18
6: **sinon**
7: Retourner $\overrightarrow{p_{start}}$;
// L'extrémité la plus proche est $\overrightarrow{p_{start}}$, comme pour A' dans la figure 7.18
8: **fin si**

Dans la suite, nous considérons qu'une trajectoire est une suite de segments. Suivre une trajectoire revient à suivre chaque partie de trajectoire, et à effectuer la transition d'une partie de trajectoire à l'autre. Nous décrivons dans un premier temps le suivi d'une partie de trajectoire, puis les transitions entres les parties de trajectoires.

Lors du suivi de sa partie de trajectoire actuelle, l'*Agent EM*$_i$ utilise les criticités $Crit_{3,k}$, $Crit_{4,k}$ et $Crit_{5,k}$, dans 2 ordres leximin possibles :

— $(Crit_{5,k}, Crit_{3,k}, Crit_{4,k})$, cet ordre est utilisé pour que l'*Agent EM*$_i$ rejoigne au plus vite sa trajectoire

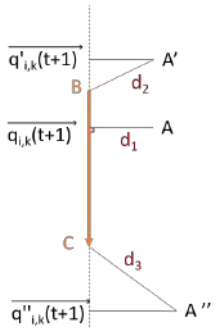


Figure 7.18 – Illustration de la distance minimale entre un point et un segment.

— $(Crit_{5,k}, Crit_{4,k}, Crit_{3,k})$, cet ordre est utilisé pour que l'*Agent* EM_i rejoigne au plus vite sa destination

Coopérativement, l'*Agent* EM_i rejoint sa partie de trajectoire au plus vite afin de la suivre au maximum, et de satisfaire cette partie de trajectoire. Pour ce faire, il passe d'un ordre à l'autre. Le choix de l'ordre est résumé par l'algorithme 7.14. Nous décrivons dans la suite cet algorithme et le calcul des seuils pour décider de l'ordre, qui passe par l'approximation d'un cercle par un polygone.

De manière mathématique, plus on augmente le nombre m de faces d'un polygone régulier, plus celui-ci se rapproche d'un cercle, de telle manière que (travaux d'Archimède pour déterminer le nombre π) :

$$\lim_{m \to \infty} Polygone(m) = Cercle$$

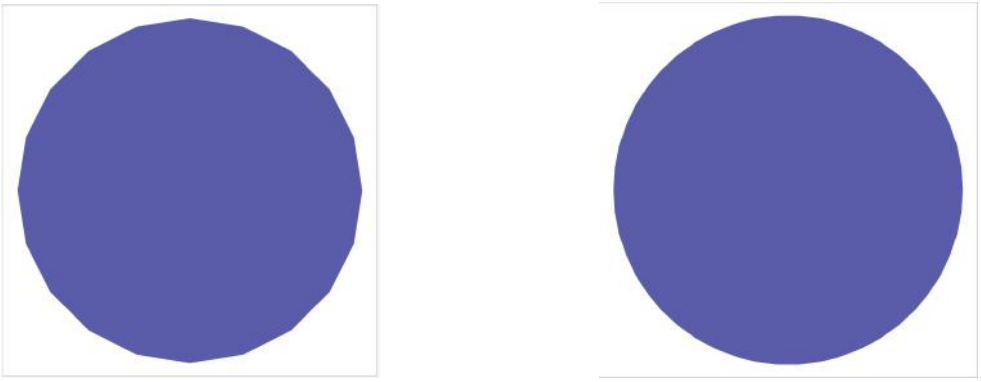
Ce résultat peut se voir graphiquement dans les images a et b de la figure 7.19. Le périmètre d'un polygone régulier à m côtés inscrit dans un cercle de rayon r est de $2msin(\frac{\pi}{m})r$. Dans notre implémentation, le virage maximum est de $3\,\mathrm{deg}.s^{-1}$, et nous allons jusqu'à $5\,\mathrm{deg}.s^{-1}$ dans l'étude de sensibilité, avec ces virages, nous faisons respectivement une erreur de 0.046% et 0,127% sur le périmètre du polygone en approximant les polygones de la suite comme étant des cercles.

Pour décider de l'ordre à appliquer, l'*Agent* EM_i utilise deux seuils adaptatifs, R et β, illustrés dans la figure 7.20.

Le premier seuil, R, est un seuil "gros grain" qui permet de savoir si l'*Agent* EM_i est loin de son segment, et dans ce cas doit le rejoindre. Pour R, si l'*Agent* EM_i effectue un virage à vitesse constante $||\overrightarrow{v_{i,k}}||$ et vitesse de rotation constante $\dot{\alpha}$, celui-ci effectue un cercle (ou une approximation satisfaisante), en un nombre de cycles de vie égal à $\frac{2\pi}{\dot{\alpha}}$, ce qui veut dire que le périmètre R du cercle est :

$$R = ||\overrightarrow{v_{i,k}}||\frac{2\pi}{\dot{\alpha}}$$

Si la distance $r = \min_{\overrightarrow{x} \in segment} \{||\overrightarrow{p_{i,k}}(t+1) - \overrightarrow{x}||\}$ entre l'*Agent* EM_i et la partie de trajectoire est supérieure à R, l'*Agent* EM_i se dirige directement vers la partie de trajectoire pour



(a) Polygone régulier à 20 côtés (b) Polygone régulier à 50 côtés

Figure 7.19 – Approximation d'un cercle par un polygone régulier à n côtés

la rejoindre en utilisant l'ordre leximin $(C_{5,i,k}, C_{4,i,k}, C_{3,i,k})$.

Dans le cas où $r < R$, l'*Agent* EM_i utilise le deuxième seuil, β. Le seuil β correspond à l'angle optimal que devrait avoir l'*Agent* EM_i pour rejoindre son segment, et permet de rejoindre la trajectoire le plus souplement possible. Cette angle est déterminé de la manière suivante :

$$\beta = acos(\frac{R-r}{R});$$

L'*Agent* EM_i compare alors l'angle $\gamma = (\overrightarrow{v_{i,k}}, \overrightarrow{p_{start}p_{end}})$ (l'angle que fait le vecteur vitesse de l'*Agent* EM_i avec la partie de trajectoire) et l'angle optimal β, dans le but de réduire à 0 la différence entre ces deux angles. Si l'on considère que l'*Agent* EM_i est à gauche de sa partie de trajectoire actuelle (c'est-à-dire si $\theta > 0$, sinon inverser le signe de β et considérer les inégalités dans l'autre sens), alors :

— si $\gamma \geq \beta$ alors l'*Agent* EM_i utilise l'ordre $(C_{5,i,k}, C_{3,i,k}, C_{4,i,k})$, pour se rapprocher de la partie de trajectoire actuelle et faire décroître l'angle γ jusqu'à β,

— sinon l'*Agent* EM_i utilise l'ordre $(C_{5,i,k}, C_{4,i,k}, C_{3,i,k})$, pour s'orienter vers la destination de sa partie de trajectoire actuelle, et faire décroître l'angle γ jusqu'à β.

Coopérativement, l'*Agent* EM_i **change d'une partie de trajectoire** à l'autre, en équilibrant la criticité de suivi des deux parties de trajectoires, c'est-à-dire en suivant autant l'une que l'autre.

Si les parties de trajectoires sont jointes, l'*Agent* EM_i suit coopérativement la première partie de trajectoire jusqu'à ce que la limite de sa vitesse de rotation l'oblige à passer à l'autre partie de trajectoire. Dans cette optique, l'*Agent* EM_i suit la partie de trajectoire jusqu'à être à une distance L de la destination de celle-ci qui correspond au début de l'autre partie de trajectoire (figure 7.21). Lorsqu'il est à une distance inférieure à L de la destination, l'*Agent* EM_i change de partie de trajectoire.

Pour calculer L, remarquons que le centre du cercle (figure 7.21) est sur la bissectrice de l'angle β, par conséquent, nous avons bien L de chaque côté et ainsi :

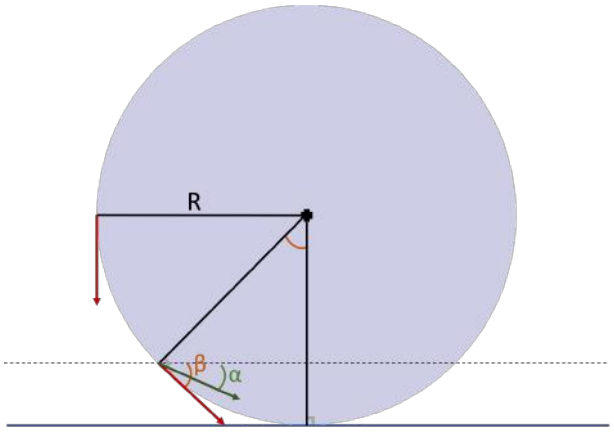


Figure 7.20 – Illustration de l'algorithme de décision pour le classement des criticités

Algorithme 7.14 : Choix de l'ordre des criticités pour le suivi de segment

1: Calculer $R = ||\overrightarrow{v_{i,k}}||\frac{2\pi}{\hat{\alpha}}$
2: Calculer $r = \min_{\overrightarrow{x} \in segment} \{||\overrightarrow{p_{i,k}}(t+1) - \overrightarrow{x}||\}$
3: **si** $r > R$ **alors**
4: Calculer l'angle $\theta = (\overrightarrow{p_{start}p_{end}}, \overrightarrow{p_{start}p_{i,k}(t)})$;
5: Calculer l'angle $\beta = acos(\frac{R-r}{R})$;
6: Calculer l'angle $\gamma = (\overrightarrow{v_{i,k}}, \overrightarrow{p_{start}p_{end}})$
7: **si** $(\theta \geq 0 \wedge \gamma \geq \beta) \vee (\theta \leq 0 \wedge -\beta \geq \gamma)$ **alors**
8: Renvoyer $(C_{5,i,k}, C_{3,i,k}, C_{4,i,k})$
9: **sinon**
10: Renvoyer $(C_{5,i,k}, C_{4,i,k}, C_{3,i,k})$
11: **fin si**
12: **sinon**
13: Renvoyer $(C_{5,i,k}, C_{4,i,k}, C_{3,i,k})$
14: **fin si**

$$\frac{R}{L} = \tan\beta$$

Si les parties de trajectoire sont non jointes, le point B qui était commun aux deux parties est remplacé par le point B_1 qui est la fin de la partie de trajectoire actuelle et B_2 qui est le début de la partie de trajectoire suivante (figure 7.22). Ainsi, si l'*Agent* EM_i est à une distance $\leq L$ de B_2, il décide de changer de partie de trajectoire et de passer au suivant, de même, s'il est à une distance $\leq d_\epsilon$ de B_1. Le seuil d_ϵ a été configuré comme étant la distance parcourue par l'*Agent* EM_i en un cycle de vie.

L'algorithme 7.15 résume ce raisonnement pour les parties de trajectoires non jointes, et jointes.

La criticité anticipée de collision de l'*Agent* EM_i par rapport à la collision s'il effectue l'action $Ac_{i,k}$, $Crit_{i,k,coll}$, correspond au maximum des criticités anticipées $Crit_{i,j,k,coll}$ pour

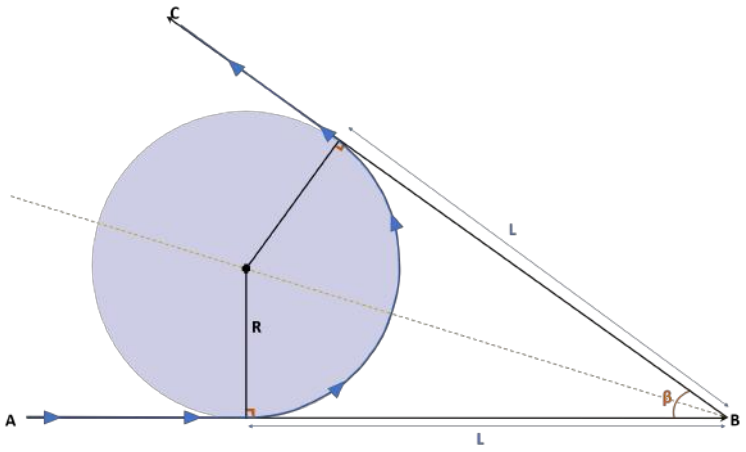


Figure 7.21 – Illustration de l'algorithme de décision pour le changement de parties de trajectoires lorsque les parties sont jointes

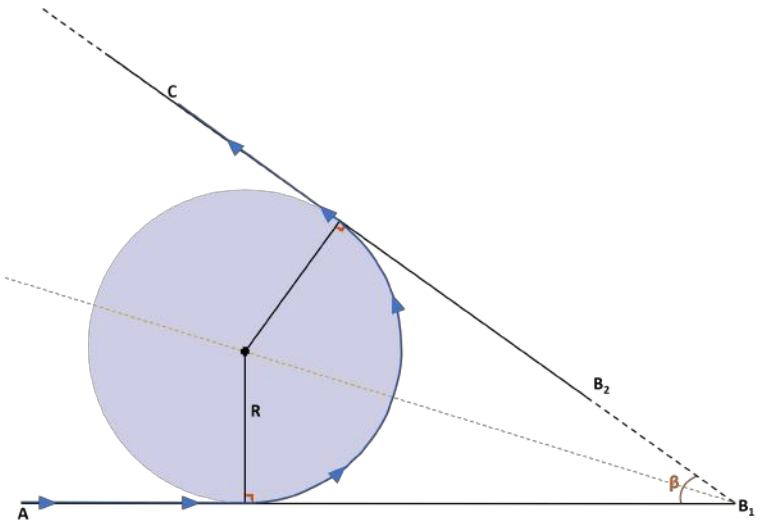


Figure 7.22 – Illustration de l'algorithme de décision pour le changement de parties de trajectoires lorsque les parties sont disjointes

l'ensemble des *Agents* EM_j dans sa zone de perception.

Algorithme 7.15 : Changement de segment

1: Calculer $R = ||\overrightarrow{v_{i,k}}||\frac{2\pi}{\dot{\alpha}}$
2: Calculer $d1 = ||\overrightarrow{p_i(t)} - \overrightarrow{B1}||$
3: Calculer $d2 = ||\overrightarrow{p_i(t)} - \overrightarrow{B2}||$
4: Calculer $L = \frac{R}{\tan\beta}$
5: Retourner $L > d_2 || d_\epsilon > d_1$

$$Crit_{i,k,coll} = \max_{j} Crit_{i,j,k,coll}$$

Chaque criticité $Crit_{i,j,k,coll}$ est la criticité anticipée de collision de l'*Agent EM*$_i$ avec l'*Agent EM*$_j$ s'il effectue l'action $Ac_{i,k}$. Une collision correspond à un rapprochement dans le temps. Ainsi la criticité $Crit_{i,j,k,coll}$ est une fonction de la distance entre les 2 trajectoires des *Agents EM*. Afin de déterminer cette criticité nous avons décidé :

— de baser son calcul sur une extrapolation appelée *nominal projection* dans la littérature [Kuchar and Yang, 2000]. Cette extrapolation consiste à considérer que M_j évolue de la même manière qu'il évolue actuellement, c'est-à-dire en suivant les vecteurs vitesses communiqués et perçus par l'*Agent EM*$_i$, ce qui permet de calculer une potentielle intersection de manière simple,

— de nous concentrer sur la distance entre les mobiles M_i et M_j à un instant t si M_i effectue l'action $Ac_{i,k}$, distance que nous notons $d_{i,j,k}(t)$,

— de considérer que les *Agents EM* peuvent être réduits à des points, ou englobés dans des sphères.

La première hypothèse permet de prendre des décisions sur des éléments simples, et rapides à calculer, et se justifie par le fait que cette prise de décision est très régulière. Ainsi, si l'*Agent EM*$_j$ venait à modifier sa trajectoire, il la modifierait peu, et surtout, le prochain cycle de décision de l'*Agent EM*$_i$ viendrait rapidement corriger l'erreur induite par l'hypothèse. Les 2 autres hypothèses sont justifiées par le contexte du système : nous voulons éviter tout type de collisions (une collision de côté est aussi grave qu'une collision de face).

Afin de calculer cette distance $d_{i,j,k}(t)$, nous utilisions la norme euclidienne notée $||.||$ sur n'importe quel espace $\mathbb{R}^n$ pour donner la distance $d_{i,j,k}(t) = ||\overrightarrow{p_{i,k}}(t) - \overrightarrow{p_j}(t)||$.

Nous utilisons la valeur minimale de la fonction $d_{i,j,k}(t)$, noté $d_{min,i,j,k}$, et le temps $t_{min,i,j,k}$ où cette distance minimale est atteinte pour déterminer cette criticité anticipée, et décider des actions à effectuer. Les valeurs de distance et de temps sont utilisées dans deux criticités distinctes qui composent $Crit_{i,j,k,coll}$:

$$Crit_{i,j,k,coll} = (Crit_{1,j,k}, Crit_{2,j,k})$$

avec :

— $Crit_{1,j,k}$, la criticité associée à $d_{min,i,j,k}$

— $Crit_{2,j,k}$, la criticité associée à $t_{min,i,j,k}$

Avec ces hypothèses, nous avons la valeur minimale de la distance entre l'*Agent EM*$_i$ avec l'*Agent EM*$_j$ si l'*Agent EM*$_i$ effectue l'action $Ac_{i,k}$:

$$d_{min,i,j,k} = \min_{t \geq 0}(d_{i,j,k}(t)) = \min_{t \geq 0} ||\overrightarrow{p_{i,k}}(t) - \overrightarrow{p_j}(t)|| \tag{7.5}$$

Avec $\overrightarrow{p_{i,k}}(t)$ la position de l'*Agent EM*$_i$ s'il effectue l'action $Ac_{i,k}$ en fonction du temps, et $\overrightarrow{p_j}(t)$ la position de l'*Agent EM*$_j$.

Puisque les vecteurs vitesses $\overrightarrow{v_{i,k}}$ et $\overrightarrow{v_j}$ sont considérés comme constant, $d_{min,i,j,k}$ peut s'écrire :

$$d_{min,i,j,k} = \min_{0 \leq t} ||(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}}) - (\overrightarrow{p_j}(0) + t.\overrightarrow{v_j})||$$

L'instant où cette distance minimale est atteinte, $t_{min,i,j,k}$, est la valeur à laquelle la dérivée de $||(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}} - (\overrightarrow{p_j}(0) + t.\overrightarrow{v_j})||$ est nulle. Intuitivement, cette équation a au plus une solution puisque les 2 mobiles avancent en mouvement rectiligne uniforme, ce qui veut dire que soit ils avancent en parallèle, soit ils se rapprochent l'un de l'autre puis s'éloignent. Mathématiquement, cette équation admet exactement un minimum puisque c'est la racine carrée d'une équation vectorielle du second degré, sous réserve que $\overrightarrow{v_{i,k}}$ et $\overrightarrow{v_j}$ ne soient pas égaux. On peut donc se permettre d'étudier son carré :

$$\begin{aligned} d_{i,j,k}(t)^2 &= ||(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}} - (\overrightarrow{p_j}(0) + t.\overrightarrow{v_j})||^2 \\ &= \Big((\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)) + t(\overrightarrow{v_{i,k}} - \overrightarrow{v_j})\Big).\Big((\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)) + t(\overrightarrow{v_{i,k}} - \overrightarrow{v_j})\Big) \\ &= ||\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)||^2 + 2t(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)).(\overrightarrow{v_{i,k}} - \overrightarrow{v_j}) + t^2||\overrightarrow{v_{i,k}} - \overrightarrow{v_j}||^2 \end{aligned}$$

dont la dérivée est,

$$\frac{d}{dt}\Big(d_{i,j,k}(t)^2\Big) = 2(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)).(\overrightarrow{v_{i,k}} - \overrightarrow{v_j}) + 2t||\overrightarrow{v_{i,k}} - \overrightarrow{v_j}||^2$$

qui s'annule pour,

$$t_{min,i,j,k} = \frac{-(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)).(\overrightarrow{v_{i,k}} - \overrightarrow{v_j})}{||\overrightarrow{v_{i,k}} - \overrightarrow{v_j}||^2} \tag{7.6}$$

et de manière évidente,

$$d_{min,i,j,k} = d_{i,j,k}(t_{min,i,j,k}) \tag{7.7}$$

Nous avons constaté qu'en utilisant uniquement $t_{min,i,j,k}$ dans l'algorithme de décision, celui-ci avait tendance à effectuer 2 comportements minoritaires mais non voulus :

— Repousser les collisions à plus tard, en faisant augmenter le temps de rapprochement $t_{min,i,j,k}$

— Créer des collisions à un instant t parce que dans un temps T ($T > t$) la situation sera bien pire ($d_{i,j,k}(T) > d_{i,j,k}(t)$)

Afin de pallier à ces comportements, nous considérons l'intervalle de temps $\left[t_{start,i,j,k}, t_{end,i,j,k}\right]$ durant lequel les agents *Agent* EM_i et *Agent* EM_j sont en collision, et nous repérons les interblocages, qui sont des *conflits* d'agents, traités dans la section 7.10.3.4.

Nous développons dans ce qui suit le calcul de cet intervalle. Une collision est détectée si la distance minimale est inférieure à une certaine distance qu'on note d_{coll}.

$$d_{min,i,j,k} < d_{coll} \Rightarrow Collision$$

Dans le cas où une collision est détectée, nous déterminons l'intervalle $[t_{start,i,j,k}, t_{end,i,j,k}]$ de temps durant lequel la collision se produit. Nous utilisons alors, le temps $t_{start,i,j,k}$ au lieu de $t_{min,i,j,k}$.

Pour calculer cet intervalle, nous résolvons l'équation suivante :

$$\begin{aligned} & 0 \leq ||(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}}) - (\overrightarrow{p_j}(0) + t.\overrightarrow{v_j})|| \leq d_{coll} \\ \Leftrightarrow \quad & 0 \leq ||(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)) + t.(\overrightarrow{v_{i,k}} - \overrightarrow{v_j})||^2 \leq d_{coll}^2 \end{aligned}$$

Ce qui revient à calculer les 2 racines de :

$$\begin{aligned} & ||(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)) + t.(\overrightarrow{v_{i,k}} - \overrightarrow{v_j})||^2 - d_{coll}^2 = 0 \\ \Leftrightarrow \quad & ||\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)||^2 + 2t(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)).(\overrightarrow{v_{i,k}} - \overrightarrow{v_j}) + t^2||\overrightarrow{v_{i,k}} - \overrightarrow{v_j}||^2 - d_{coll}^2 = 0 \quad (7.8) \end{aligned}$$

Les 2 solutions étant :

$$t_{start,i,j,k} = \frac{-b - \sqrt{\delta}}{2a} \tag{7.9}$$

$$t_{end,i,j,k} = \frac{-b + \sqrt{\delta}}{2a} \tag{7.10}$$

avec,

$$\begin{aligned} \delta = b^2 - 4ac \qquad & a = ||\overrightarrow{v_{i,k}} - \overrightarrow{v_j}||^2 \\ b = 2(\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)).(\overrightarrow{v_{i,k}} - \overrightarrow{v_j}) \quad & c = ||\overrightarrow{p_{i,k}}(0) - \overrightarrow{p_j}(0)||^2 - d_{coll}^2 \end{aligned}$$

Enfin, si le temps retenu ($t_{min,i,j,k}$ ou $t_{start,i,j,k}$) est inférieur à 0, nous considérons qu'il est égal à 0.

Ainsi, **la criticité anticipée** $Crit_{1,j,k}$ est calculée selon l'équation 7.11 et suit la forme décrite dans la figure 7.23

$$Crit_{1,j,k} = \begin{cases} 100 - \frac{100}{2d_{coll}} d_{min,i,j,k} & d_{min,i,j,k} < 2d_{coll} \\ 0 & d_{min,i,j,k} \geq 2d_{coll} \end{cases} \tag{7.11}$$

Cette criticité permet de trier les rapprochements entre M_i et les $M_j \in Zp_i$ (dans la zone de perception) en fonction de leur gravité en terme de distance. Lorsque la distance est nulle, la criticité est maximale, et lorsque les 2 mobiles sont au moins à une distance suffisante, la criticité est nulle. Cette notion de distance "suffisante" est définie comme étant 2 fois la distance de collision d_{coll}. La pertinence de ce choix est discutée dans les expérimentations (section 8.2).

La criticité anticipée $Crit_{2,j,k}$ est calculée selon l'équation 7.12 et suit la forme décrite dans la figure 7.24.

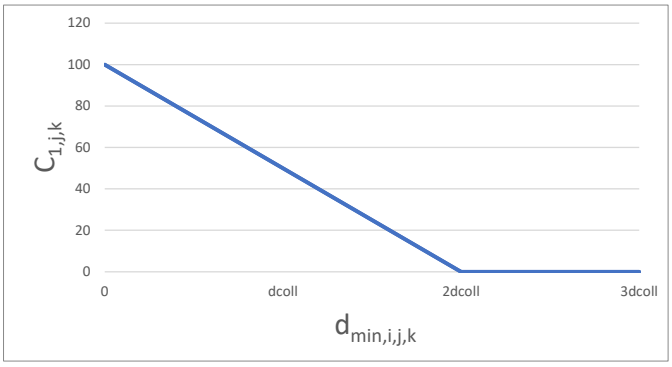


Figure 7.23 – La fonction de criticité $Crit_{1,j,k}$

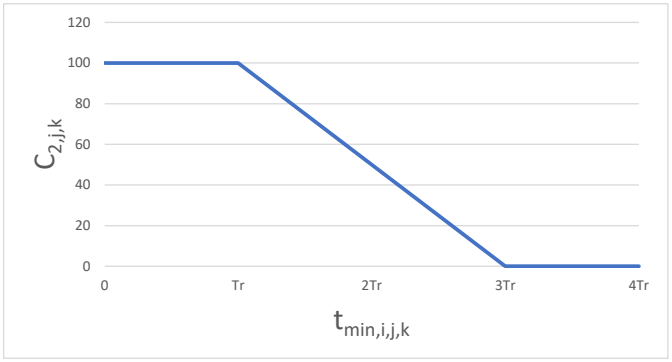


Figure 7.24 – La fonction de criticité $Crit_{2,j,k}$

$$Crit_{2,j,k} = \begin{cases} 100 & t_{min,i,j,k} < t_r \\ 100 - \frac{100}{2t_r}(t_{min,i,j,k} - t_r) & t_{min,i,j,k} \in [t_r, 3t_r] \\ 0 & t_{min,i,j,k} \geq 3t_r \end{cases} \tag{7.12}$$

Cette criticité permet de trier les rapprochements entre M_i et les $M_j \in Zp_i$ en fonction de leur gravité en terme de temps. Lorsque la collision ou le rapprochement est "trop tôt", la criticité est maximale. Ensuite, celle-ci doit être nulle lorsque la collision est "suffisamment loin" dans le temps, puisqu'elle est très incertaine et n'arrivera sûrement pas. Nous considérons le temps que met l'*Agent* EM_i pour traverser sa zone de perception Zp_i, noté t_r, comme premier jalon, et $3t_r$ comme deuxième jalon. De même, la pertinence de ces choix est

discutée dans les expérimentations (section 8.2).

7.9.2 Fonctionnement interne de l'Agent EM

Nous avons présenté précédemment le fonctionnement de l'*Agent EM*$_i$ de manière générale. Nous introduisons ici son fonctionnement interne, qui se découpe en 4 types de modules différents décrits dans les sections suivantes :

— Le *Module Perception de l'Agent EM (Module Perception)* : c'est le module qui permet à l'agent de collecter des informations de son environnement. Il y a un *Module Perception* par *Agent EM*$_i$. Son comportement est présenté dans la section 7.9.3.

— Les *Modules Voisins de l'Agent EM (Module Voisins)* : ils correspondent à la mémoire à court terme de l'*Agent EM*$_i$. Il existe un *Module Voisin* par *Agent EM*$_j$ dans le voisinage, tel que $M_j \in Zp_i$. Leur comportement est présenté dans la section 7.9.4.

— Les *Modules Actions de l'Agent EM (Module Actions)* : ils correspondent aux actions possibles de l'*Agent EM*$_i$. Il existe autant de *Module Action* que d'actions dans l'ensemble Ac_i. Leur comportement est présenté dans la section 7.9.5.

— Le *Module Décision de l'Agent EM (Module Décision)* : il correspond à la prise de décision de l'*Agent EM*$_i$. Il y a un *Module Décision* par *Agent EM*$_i$. Son comportement est décrit dans la section 7.9.6.

Le fonctionnement interne de l'*Agent EM*$_i$ est représenté par la figure 7.25. Cette distribution répond au besoin de généricité, et permet aussi de distribuer les calculs. Le *Module Perception* peut recevoir des informations de nombreux capteurs. Les différentes entités mobiles qui sont dans le voisinage de l'*Agent EM*$_i$ peuvent être de types différents (par exemple, dans le trafic aérien nous pouvons avoir des avions de types différents, des drones, des planeurs, parmi d'autres). L'analyse de leur comportement différera en fonction de ce type, justifiant l'utilisation de plusieurs *Modules Voisins*. Les actions sont différentes les unes des autres, et si dans notre système elles correspondent à des modifications du vecteur vitesse, elles peuvent correspondre à une suite d'actions simples (par exemple tourner à droite puis à gauche), justifiant l'utilisation de plusieurs *Modules Actions*. Enfin, le fonctionnement du *Module Décision* est indépendant du type des actions.

La figure 7.26 décrit la structure de l'*Agent EM*$_i$ dans sa globalité.

7.9.3 Présentation du Module Perception

Le rôle du *Module Perception* est d'observer l'environnement de l'*Agent EM*$_i$ et de notifier ses observations auprès des *Modules Voisins*.

Le *Module Perception* peut représenter un système de reconnaissance d'images, le transpondeur d'un avion, ou l'ensemble des capteurs de l'*Agent EM*$_i$. Dans un système où les données d'observations sont multiples et multi-modales, ce module est en charge de l'observation de ces données et de leurs différentes interprétations.

Le *Module Perception* commence par percevoir son environnement, c'est-à-dire les différents messages des Agent EM$_j \in Zp_i$ et les différents obstacles $O_i \in Zp_i$. Il envoie ses per-

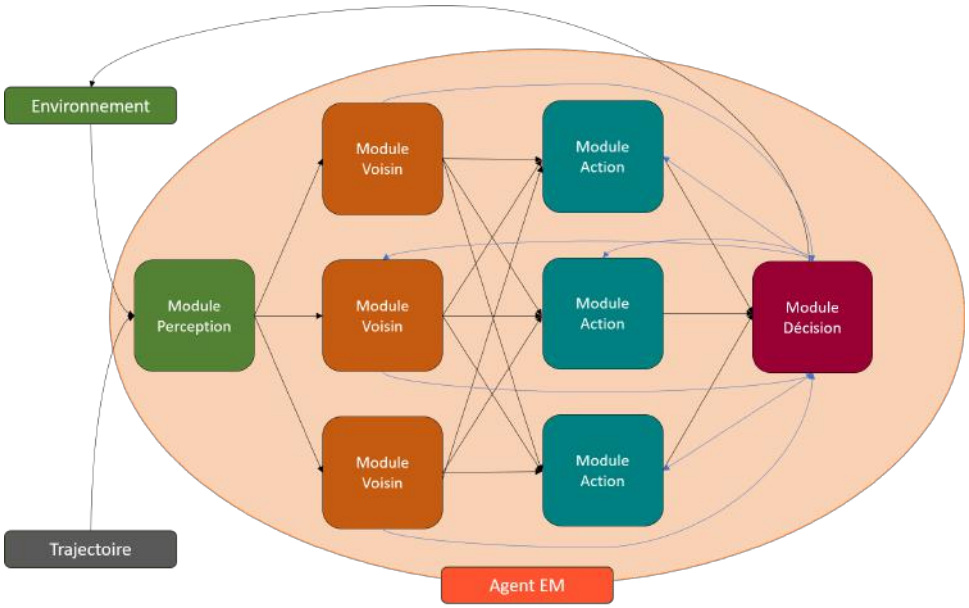


Figure 7.25 – Vue globale du fonctionnement interne de l'*Agent EM*$_i$

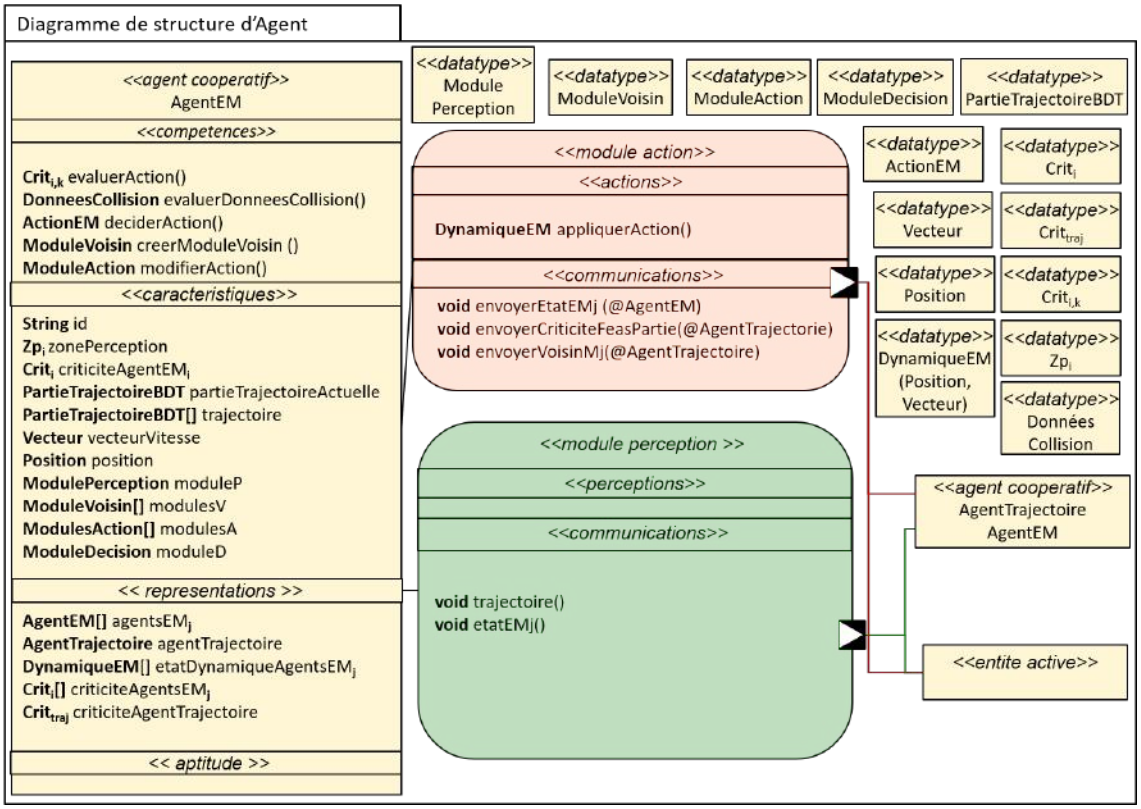


Figure 7.26 – Structure de l'*Agent EM*$_i$

ceptions aux différents *Modules Voisins*. Ce comportement est représenté dans l'algorithme 7.16.

Algorithme 7.16 : Comportement du *Module Perception*

1: Percevoir l'ensemble des *Agents* EM_j dans la zone de perception Zp_i de l'*Agent* EM_i.
2: **pour tout** Les *Agents* $EM_j \in Zp_i$ **faire**
3: **si** Le *Module Voisin* associé à l'*Agents* EM_j n'existe pas **alors**
4: creerModuleVoisin()
5: **fin si**
6: Envoyer l'information perçue au *Module Voisin* approprié
7: **fin pour**

7.9.4 Présentation du Module Voisin

Pour accomplir leurs fonctions, les *Modules Actions* et le *Module Décision* doivent avoir une vison de leur environnement à jour et suffisamment détaillée. Cette tâche incombe aux *Modules Voisins*, qui sont la mémoire à court terme de l'*Agent* EM_i.

Les *Modules Voisins* effectuent 2 tâches essentielles pour l'*Agent* EM_i. La première est d'informer les *Modules Actions* qui dépendent d'eux de l'état des *Agents* EM_j qu'ils observent. La deuxième est d'inférer des informations sur l'environnement à partir de ses perceptions, afin de prévoir, de comprendre, voire d'anticiper les déplacements de l'*Agent* EM_j dans la zone de perception de leur agent, et ainsi améliorer le fonctionnement des autres modules. Chaque *Module Voisin* effectue cette tâche pour un *Agent* EM_j qui lui est attribué.

Inférer des informations sur l'environnement à partir des perceptions est utile dans 2 cas. Dans un premier temps, il est important de déterminer l'état des voisins, même quand les informations reçues par le *Module Voisin* (venant du *Module Perception*) sont incomplètes. Ce cas est décrit par la situation de non coopération de la section 7.10.3.2. Dans un deuxième cas, il est intéressant d'inférer des informations qui ne sont pas communiquées. En effet, les *Agents EM* ne se communiquent que leur position, leur vecteur vitesse, et leur criticité. Par conséquent, il est intéressant que l'*Agent* EM_i détermine si des *Agents* EM_j sont en difficulté (en particulier en situation de non coopération), et comment les aider. Par exemple, un *Agents* EM_j peut être dans l'incapacité de revenir sur sa trajectoire (situation que nous décrivons dans la section 7.10.3.5), sans pouvoir communiquer cette information, et il est alors intéressant que le *Module Voisin* associé à cet *Agent* EM_j détecte cette situation.

Le *Module Voisin* effectue cette tâche tant qu'il perçoit l'*Agent* EM_j auquel il est associé dans son environnement, et se supprime sinon. L'algorithme 7.17 décrit le comportement du *Module Voisin*.

7.9.5 Présentation du Module Action

Un *Module Action* représente une action que peut effectuer l'*Agent* EM_i. Une action correspond à une modification du vecteur vitesse, que ce soit dans son orientation (sa vitesse de rotation), sa norme (et donc sa vitesse) sur ses différents degrés de liberté. Il y a autant de *Module Action* que d'actions $Ac_{i,k}$ réalisables par l'*Agent* EM_i.

Chaque *Module Action* reçoit des informations des *Modules Voisins*, à partir desquelles

Algorithme 7.17 : Comportement du *Module Voisin*

```
1:  DeathCounter ← 0;
2:  si a reçu un message de Module Perception alors
3:      DeathCounter ← 0;
4:      Ajouter cet état à sa mémoire;
5:      Inférer informations;
6:      Envoyer l'état et les information inférées aux Modules Actions et au Module Décision
        (en cas de SNC, voir section 7.10);
7:  sinon
8:      DeathCounter ← DeathCounter + 1;
9:      si DeathCounter > 2 alors
10:         Détruire Module Voisin
11:     fin si
12: fin si
```

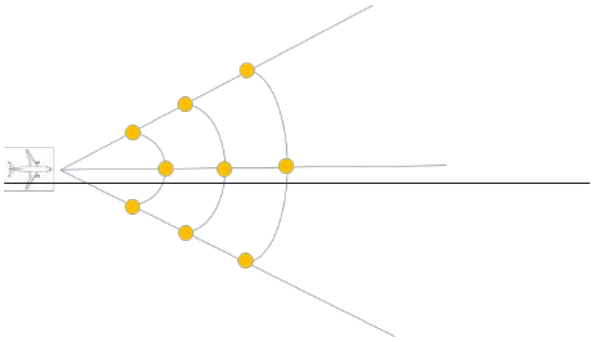


Figure 7.27 – Illustration de l'intérêt de la modification d'une action

il calcule la criticité de l'*Agent* EM_i si celui-ci effectue son action $Ac_{i,k}$, à savoir $Crit_{i,k,coll}$ et $Crit_{i,k,traj}$. Il envoie ensuite son action et les criticités associées au *Module Décision*.

Ce *Module Action* peut décider de modifier légèrement son action s'il considère que modifier celle-ci peut améliorer la criticité du voisinage ou de l'*Agent* EM_i. Par exemple, sans modifier les actions, un *Agent* EM_i peut être dans l'incapacité de rejoindre exactement sa trajectoire, en raison de la discrétisation qui a été faite, et être à une distance faible, mais non nulle, de celle-ci (figure 7.27) ; en modifiant très légèrement une action, l'*Agent* EM_i peut alors rejoindre exactement sa trajectoire. Le comportement du *Module Action* est décrit dans l'algorithme 7.18.

7.9.6 Présentation du Module Décision

Le *Module Décision* est la partie de l'*Agent* EM_i qui décide quelle action $Ac_{i,k_{decided}}$ est effectuée par l'agent.

Pour ce faire, le *Module Décision* prend en compte à chaque cycle les propositions des différents *Modules Actions*, et choisit d'appliquer l'action proposée qui diminue le plus la criticité locale de l'agent, c'est-à-dire des autres *Agents* EM_j et de lui même. Pour cela, il

Algorithme 7.18 : Comportement du *Module Action*

1: Calculer $Crit_{i,k,traj}$;
2: Recevoir les différentes informations associées à l'*Agent* EM_j ($Crit_j$, $\overrightarrow{p_j}$, $\overrightarrow{v_j}$) ;
3: **pour tout** *Agent* EM_j **faire**
4: Calculer $Crit_{i,j,k,coll}$;
5: Mettre $Crit_{i,j,k,coll}$ dans la liste $Crit_{i,k,coll}$;
6: **fin pour**
7: **tant que** Le *Module Action* modifie son action **faire**
8: Recalculer $Crit_{i,k,traj}$;
9: **pour tout** *Agent* EM_j **faire**
10: Recalculer $Crit_{i,j,k,coll}$;
11: Mettre à jour $Crit_{i,j,k,coll}$ dans la liste $Crit_{i,k,coll}$;
12: **fin pour**
13: **fin tant que**
14: Envoyer $Crit_{i,k} = (Crit_{i,k,traj}, Crit_{i,k,coll})$ et son action au *Module Décision*

choisit le sous-ensemble d'actions qui diminue le plus la criticité de l'*Agent EM* le plus critique dans sa zone de perception (potentiellement lui même), puis réduit ce sous-ensemble avec les actions qui diminuent le plus la criticité du deuxième *Agent EM* le plus critique dans sa zone de perception, et ainsi de suite, récursivement, jusqu'à ce qu'il n'y ait plus qu'une action, ou qu'il n'y ait plus d'autres *Agents EM* (algorithme 7.19).

7.10 Situations de non coopération des agents d'AGATS

Une Situation de Non Coopération (SNC) est une situation dans laquelle un agent, de par son comportement, ou simplement l'état dans lequel il se trouve, ne peut atteindre son but, ou empêche d'autres agents d'atteindre leurs buts. Nous décrivons dans cette section les SNC prévues dans lesquelles peuvent se trouver les différents agents d'*AGATS*, leur détection, et leur résolution.

7.10.1 SNC de l'Agent Situation

Cette section va décrire les 2 SNC de l'*Agent Situation*.

7.10.1.1 SNC 1 : Conflit d'Agent Situation

Deux *Agents Situations* se gênent dans la génération de leurs situations respectives.

Détection La situation est détectée par deux *Agents EM* appartenant à deux situations différentes qui entrent en conflit lors de la phase de simulation.

Algorithme 7.19 : Comportement du *Module Décision*

Soit L, la liste contenant l'ensemble des couples ($Ac_{i,k}$, $Crit_{i,k}$)
Soit V, la liste contenant les couples (*Agent* EM_j,$Crit_j$) et le couple (*Agent* EM_i,$Crit_i$)
Trier V en fonction de $Crit_j$, du plus critique au moins critique
$L_{temp} \leftarrow L$
$j_{actuel} \leftarrow 1$ //Cet indice est utilisé dans la suite pour parcourir les *Agents* EM_j voisins du plus critique au moins critique
tant que L_{temp}.taille()>1 et j_{actuel}<V.taille() **faire**
  si V[j_{actuel}]$\neq$*Agent* EM_i **alors**
    Trouver l'action $Ac_{i,k_{j_{actuel}}}$ telle que $Crit_{i,j_{actuel},k_{j_{actuel}}}$ soit minimale pour tout couple ($Ac_{i,k}$, $Crit_{i,k}$) dans L_{temp} // On cherche l'action qui minimise la criticité avec l'agent $EM_{j_{actuel}}$, qui est le j_{actuel}ème agent le plus critique. Pour rappel, $Criti, j, k$ est contenue dans $Crit_{i,k}$
    Enlever toutes les actions de L_{temp} dont la criticité $Crit_{i,j,k}$ dans $Crit_{j,k}$ est telle que $Crit_{i,j_{actuel},k} > Crit_{i,j_{actuel},k_{j_{actuel}}} + \epsilon$ // On enlève toute les actions dont la criticité est trop élevée par rapport à cette action de criticité minimale
  sinon
    Trouver l'action $Ac_{i,k_{j_{actuel}}}$ telle que $Crit_{i,k_{j_{actuel}}}$ soit minimale pour tout couple ($Ac_{i,k}$, $Crit_{i,k}$) dans L_{temp} // On cherche l'action qui minimise la criticité de l'*Agent* EM_i.
    Enlever toutes les actions de L_{temp} dont la criticité $Crit_{i,k}$ est telle que $Crit_{i,k} > Crit_{i,k_{j_{actuel}}} + \epsilon$ // On enlève toute les actions dont la criticité est trop élevée par rapport à cette action de criticité minimale
  fin si
  $j_{actuel} \leftarrow j_{actuel} + 1$
fin tant que

Résolution Les deux *Agents EM* transmettent cette information à leur *Agents Trajectoires* respectifs, qui eux-même transmettent cette information à leurs *Agents Situations* respectifs. L'*Agent Situation* le moins critique demande alors à son *Agent Trajectoire* de modifier sa trajectoire, qui demande alors à ses *Agents Parties* de modifier leurs parties de trajectoire.

7.10.1.2 SNC 2 : Incompétence d'un *Agent Situation*, non respect du réalisme

L'*Agent Situation* ne peut pas améliorer la satisfaction de ses contraintes sans dégrader le réalisme, ou inversement. La *situation* requise par le scénariste n'est alors pas compatible avec le réalisme demandé.

Détection La SNC est détectée par l' *Agent Situation*, par observation de sa criticité. Si celle-ci ne baisse pas, ou suit un cycle, l'*Agent Situation* est alors dans cette SNC.

Résolution L'*Agent Situation* ne peut pas construire la *situation* telle que voulue par le scénariste, il doit donc signaler son incompétence et demander au scénariste de modifier une ou plusieurs caractéristiques de la situation, en proposant en premier celles ayant la criticité

la plus élevée.

7.10.2 SNC de l'Agent Trajectoire

L'*Agent Trajectoire* traite la SNC suivante.

7.10.2.1 SNC 3 : Conflit d'Agents Trajectoires, génération de collision non désirée

Les trajectoires de deux *Agents Trajectoires* génèrent une ou plusieurs collisions entre les *Agents EM* qui parcourent leur trajectoire.

Détection La détection de cette SNC se fait lors de la phase de simulation, les *Agents EM* qui suivent ces trajectoires ont dû s'éviter l'un l'autre, et par conséquent ils ont envoyé cette information à leur *Agent Trajectoire* respectif.

Résolution La résolution dépend du fait que les *Agents Trajectoires* appartiennent à la même situation ou non.

— Si les deux *Agents Trajectoires* appartiennent à la même situation, l'*Agent Trajectoire* le moins critique demande à ses *Agents Parties* de se modifier, en commençant par l'*Agent Partie* qui était suivi par l'*Agent EM* lors de l'interaction.

— Sinon, l'information est remontée aux *Agents Situations* respectifs, et ces derniers négocient pour savoir quel *Agent Trajectoire* doit se modifier. L'*Agent Situation* le moins critique demande alors à son *Agent Trajectoire* de modifier sa trajectoire, qui demande alors à ses *Agents Parties* de modifier leurs parties de trajectoire.

Cette SNC est résolue par les *Agents Trajectoires* qui demandent la modification de la trajectoire à leurs *Agents Parties*.

7.10.3 SNC de l'Agent EM

L'*Agent EM* traite les 6 SNC suivantes.

7.10.3.1 SNC 4 : Incompréhension Agent EM

Détection. L'*Agent EM* reçoit des observations qui ne concernent aucun de ses *Module Voisin*, son *Module Perception* se retrouve donc dans l'incapacité de le transmettre au bon *Module Voisin*.

Résolution. L'*Agent EM* crée le *Module Voisin* approprié et transmet *via* son *Module Perception* le message qu'il n'arrivait pas à transmettre.

7.10.3.2 SNC 5 : Ambiguïté d'un Agent EM, message incomplet

Détection. Quand un *Agent EM*$_i$ reçoit une information incomplète venant d'un *Agent EM*$_j$ voisin, en particulier pour des informations comme la criticité de l'agent, ou son vecteur vitesse, il est dans une situation d'ambiguïté. Cette information importante peut potentiellement prendre n'importe quelle valeur. En raison de l'absence de cette information, le *Module Voisin* associé à l'*Agent EM*$_j$ est incapable d'effectuer les tâches qu'il doit effectuer, comme inférer les SNC de l'*Agent EM*$_j$ qu'il observe, ou sa tâche principale qui est de transmettre une information complète au *Module Action*.

Résolution. L'*Agent EM*$_i$ tente de résoudre cette SNC en utilisant la mémoire du *Module Voisin*. Le module estime les informations manquantes en utilisant les informations qui sont à sa disposition, c'est-à-dire les informations qu'il perçoit à l'instant, ou celles qu'il a mémorisées. L'information estimée par le *Module Voisin* est transmise aux *Modules Actions* pour qu'ils réussissent à estimer les informations ou non. L'échec est dû à un manque d'information, notamment lorsque l'*Agent EM*$_j$ vient d'entrer dans la zone de perception de l'*Agent EM*$_i$, et que ce dernier n'a donc pas de mémoire le concernant.

7.10.3.3 SNC 6 : Incompréhension d'un Agent EM

Détection. Cette SNC est détectée par le *Module Action* qui reçoit une information venant d'un *Module Voisin* et que celle-ci n'est pas complète, en particulier pour des informations comme la criticité d'*Agent EM*$_j$, ou son vecteur vitesse. Cette situation peut arriver quand le *Module Voisin* n'arrive pas à résoudre la SNC5 (présentée dans 7.10.3.2).

Résolution. La résolution dépend de la nature de l'information manquante :

— Si l'information manquante est le vecteur vitesse de l'*Agent EM*$_j$, le *Module Action* considère que la vitesse est nulle.

— Si l'information manquante est la criticité de l'*Agent EM*$_j$, le *Module Action* considère que la criticité de l'*Agent EM*$_j$ est maximale.

7.10.3.4 SNC 7 : Inutilité de l'action d'un Agent EM

Détection. Cette SNC apparaît lorsque l'action d'un *Module Action* est appliquée plusieurs fois par le *Module Décision* sans que la criticité de l'*Agent EM*$_i$ ne s'améliore.

Résolution. L'*Agent EM*$_i$ décide de désactiver ce module jusqu'à ce qu'il observe que sa criticité baisse suffisamment.

7.10.3.5 SNC 8 : Incompétence d'un Agent EM, suivi de trajectoire

Détection. Cette situation apparaît en raison de la priorité que l'évitement de collision a sur le suivi de la trajectoire. Voulant éviter les conflits, mais aussi en même temps se rappro-

cher de sa trajectoire un *Agent EM*$_i$ peut se retrouver en situation d'interblocage, avec un ou plusieurs autres *Agent EM*$_j$. Cette situation est détectée de deux manières :

— Si la situation concerne l'*Agent EM*$_i$ lui-même, elle est détectée par le *Module Action* lors de la détection de la SNC 7 (section 7.10.3.4).
— Si la situation concerne un autre *Agent EM*$_j$, elle est détectée par le *Module Voisin* qui observe cet *Agent EM*$_j$

Résolution.
— Si la situation concerne l'*Agent EM*$_i$, il choisit une action différente de l'action précédemment effectuée, et effectue cette action tant que la criticité par rapport à la collision n'est pas nulle.
— Si la situation concerne un autre *Agent EM*$_j$ qui est plus critique que lui, le *Module Décision* demande au *Module Voisin* qui observe l'*Agent EM*$_j$ d'estimer sa trajectoire. Tant qu'aider l'*Agent EM*$_j$ ne le pénalise pas, le *Module Décision* de l'*Agent EM*$_i$ choisit l'action qui aide le plus l'*Agent EM*$_j$ à rejoindre sa trajectoire

7.10.3.6 SNC 9 : Improductivité d'un Agent EM, pas d'action

Détection. Le *Module Décision* ne peut pas choisir d'action car il n'a reçu aucune proposition d'action de la part de ses *Modules Actions*. Cette SNC peut arriver notamment parce que l'*Agent EM* est en SNC 5 (7.10.3.4) pour toutes ses actions.

Résolution. Le *Module Décision* indique la SNC, et l'*Agent EM* réactive les différents *Modules Actions* qui étaient désactivés provisoirement.

7.10.4 SNC de l'Agent Partie

L'*Agent Partie* traite 2 SNC.

7.10.4.1 SNC 10 : Conflit d'Agents Parties

Deux *Agents Parties* veulent modifier leur partie de trajectoire en même temps, de manière antinomique. Par exemple, l'un veut décaler vers l'est sa partie de trajectoire, et l'autre vers l'ouest, alors qu'ils étaient parfaitement liés entre eux.

Détection. En effectuant leurs actions respectives, les deux agents remarquent la SNC.

Résolution. L'*Agent Partie* qui était le plus critique reste dans son état, et l'autre agent retourne dans l'état antérieur.

7.10.4.2 SNC 11 : Inutilité d'un Agent Partie

Une partie de trajectoire est inutile dans la trajectoire de son *Agent Trajectoire*.

Détection. L'*Agent Partie* évalue que la criticité de ses deux *Agents Parties* voisins sera meilleure s'il ne participe pas à la création de la trajectoire de leur *Agent Trajectoire*.

Résolution. L'*Agent Partie* décide alors de se supprimer.

7.10.5 SNC de l'Agent Extrémité

L'*Agent Extrémité* traite une SNC.

7.10.5.1 SNC 12 : Conflit d'Agents Extrémité

Deux *Agents Extrémités* veulent qu'un même *Agent Trajectoire* arrive (ou parte) de leur extrémité afin de satisfaire leur planning et/ou leur contrainte.

Détection. L'*Agent Trajectoire* reçoit deux propositions d'*Agents Extrémités* pour faire arriver (ou partir) un *Agent EM* depuis leur extrémité.

Résolution. L'*Agent Trajectoire* choisit l'*Agent Extrémité* le plus critique.

7.11 Conclusion sur le système AGATS

Nous avons décrit dans ce chapitre le système *Autonomous GenerAtion of Traffic Simulations (AGATS)*, pour l'auto-structuration d'une simulation de trafic. Dans cette thèse, nous considérons la structuration de trafic comme étant l'organisation d'un trafic qui permet l'émergence de *situations* requises par un scénariste, et le réalisme de ce trafic. Le réalisme doit être réalisé du point de vue macroscopique, c'est-à-dire que le trafic soit réaliste, et du point de vue microscopique (par exemple, sans collision), c'est-à-dire que les trajectoires soient réalistes que ce soit par rapport à la physique (par exemple, aucun avion commercial actuel n'a la capacité de voler à plus de Mach 2, c'est-à-dire 2 fois la vitesse du son à son altitude [1]), ou le comportement (par exemple, beaucoup d'avions commerciaux volent environ à Mach 0.8 en croisière). Une simulation étant souvent interactive, et sujette à des modifications, cette structuration doit être adaptable de manière autonome, et résiliente, notamment en apportant des comportements d'adaptation aux différentes entités mobiles.

Afin d'atteindre ces différents objectifs, nous nous sommes appuyés sur la théorie AMAS (section 6.2), le pattern AMAS4Opt (section 6.3) et le système EVAA (section 6.4). Le système EVAA est utilisé pour l'apprentissage de comportement contextuel, afin de générer une base de données de trajectoires distribuée et contextuelle. En suivant le pattern AMAS4Opt, nous avons proposé différents agents pour l'auto-structuration d'une simulation de trafic, qui peuvent se distinguer en deux ensembles. Le premier ensemble d'agents est dédié à l'adaptation des entités mobiles lors de la simulation, en s'inspirant des comportements d'adaptation observés dans la structuration de trafic routier (section 3), et d'approches utilisées dans

1. La vitesse du son varie avec la température, qui varie notamment en fonction de l'altitude

le trafic aérien (section 3.1). Ces agents (*Agent EM*) permettent aux entités mobile de s'éviter, de suivre leur trajectoire, et de manière générale de s'adapter en fonction de l'évolution de la simulation. Le deuxième ensemble d'agents est dédié à la génération d'un scénario et à son adaptation en temps réel en fonction de l'évolution de la simulation. Ces agents (*Agent Situation*, *Agent Trajectoire*, *Agent Partie*, *Agent Extrémité*) ont pour but de générer des trajectoires réalistes qui permettent l'émergence des *situations* et d'un trafic réaliste. Ces deux ensembles interagissent afin d'auto-structurer dynamiquement la simulation.

Nous présentons dans les chapitres suivants l'implémentation d'*AGATS*, et les expérimentations qui ont été effectuées. Les agents du système *AGATS* peuvent être séparés en deux ensembles suffisamment autonomes. C'est pourquoi nous testons ces deux ensembles séparément. Ainsi, nous présentons dans un premier temps l'implémentation et l'expérimentation du premier ensemble qui apporte des comportements d'adaptation aux différentes entités mobiles, *Collision Avoidance Adaptive Multi-Agent System (CAAMAS)*. Et nous présentons dans un deuxième temps l'implémentation et l'expérimentation du deuxième ensemble qui permet de générer dynamiquement des scénarios, *Autonomous GEneration of trAffic Scenarios (AGEAS)*.

Quatrième partie

Expérimentation et validation

8 CAAMAS : Implémentation, expérimentations et analyse

Les comportements adaptatifs des entités mobiles constituent une partie non négligeable du modèle *AGATS*, globalement indépendante de la génération de scénarios et de trajectoires, et pleine de spécificités et de comportements intéressants à tester. Ainsi, nous avons mis en place le système *CAAMAS* permettant de tester intensivement cette partie avant de la replacer dans le modèle complet.

Le reste de ce chapitre présente l'implémentation de l'*Agent EM*, son adaptation aux spécificités de l'aviation, et enfin les expérimentations qui ont été effectuées.

8.1 Implémentation du système CAAMAS

Nous décrivons dans cette section l'implémentation des *Agents EM* qui constituent le système CAAMAS. Cette section présente les adaptations de ces agents aux spécificités de l'aviation, ainsi que les points d'implémentation que nous avons traités. Dans ce qui suit, les *Agents EM* sont désormais uniquement des avions, et la notation M_i désigne donc l'avion i. Dans cette implémentation, comme la plupart des implémentations sur la planification de trajectoires d'avions, nous nous abstrayons du problème des coordonnées géographiques (longitude, latitude, altitude) pour nous concentrer sur l'évitement de collisions et l'adaptation des entités mobiles à leur environnement. Nous décrivons dans ce qui suit l'implémentation de l'*Agent* EM_i et de ses situations de non coopération.

8.1.1 Présentation générale de CAAMAS

Le système *CAAMAS* est une partie du système *AGATS* composé uniquement des *Agents EM*. Les agents de *CAAMAS* requièrent en entrée un ensemble d'entités mobiles $M = \{M_i\}$, et leur trajectoires prévues, noté τ_i, et doivent fournir en sortie des trajectoires sans collision, respectant au maximum les trajectoires τ_i prévues.

CAAMAS est composé des *Agents EM*, qui doivent s'éviter entre eux et suivre leur trajectoire τ_i. Chaque trajectoire est l'équivalent de la trajectoire envoyée par l'*Agent Trajectoire* du système *AGATS* à son *Agent* EM_i, elle est donc composée de parties de trajectoires, que l'*Agent* EM_i suit l'une après l'autre.

L'implémentation suit globalement la description de l'*Agent EM*$_i$ d'*AGATS* (section 7.9). Le système *CAAMAS* suit une exécution quasi-synchronisée. A chaque pas de temps, tous les *Agents EM* exécutent un cycle de vie (perception, décision, action). L'ordre dans lequel ces agents effectuent ce cycle de vie est totalement aléatoire. Ce type d'exécution influence certains résultats, comme nous le verrons dans les expérimentations (section 8.2).

Nous décrivons dans les sections suivantes l'implémentation de la trajectoire, puis celle de l'*Agent EM*$_i$.

8.1.2 Implémentation des trajectoires de l'Agent EM

Dans le monde aérien, les trajectoires planifiées pour les avions sont des suites de balises aéronautiques (chapitre 1.2.2), appelées "waypoints", qui peuvent se résumer à un nom et une position géographique. Une trajectoire d'avion est donc une suite de segments joints, néanmoins, le but étant d'utiliser le résultat de l'apprentissage d'EVAA (section 6.4) comme une base de données de trajectoires, nous considérons que ces segments sont potentiellement non joints. Suivre une trajectoire revient alors à suivre chaque partie de la trajectoire, c'est-à-dire suivre chaque segment.

Une **partie de trajectoire** est déterminée par un point de départ et un point de destination dans l'espace cartésien, noté respectivement $\overrightarrow{p_{start}} = (x_{start}, y_{start}, z_{start})$ et $\overrightarrow{p_{end}} = (x_{end}, y_{end}, z_{end})$.

Une **trajectoire** est alors une suite ordonnée de segments τ_i, supposés non joints, que doit suivre l'*Agent EM*$_i$ associé.

8.1.3 Implémentation de l'Agent EM

Afin d'être utilisé dans le trafic aérien, l'*Agent EM*$_i$ nécessite quelques ajustements, que nous décrivons dans ce qui suit.

Étant donné les spécificités de l'aviation, l'évitement d'*Agents EM* ne correspond pas uniquement à éviter les collisions, mais aussi les conflits entre les avions (section 1.3.1). Ainsi, nous considérons dans la suite deux avions en perte de séparation comme étant en "collision" pour notre système. Nous préservons alors le but de l'*Agent EM*$_i$ qui est de suivre sa trajectoire τ_i, et d'éviter les collisions (conflits aéronautiques).

La **zone de perception** de l'*Agent EM*$_i$ a été déterminée par trois facteurs :

— La portée de l'*Automatic Dependent Surveillance – Broadcast (ADS-B)*.

— L'espace aérien et son utilisation.

— La zone de conflit aéronautique des avions.

Dans un premier temps, la zone de perception Zp_i était calée sur la distance de perception des signaux de l'*ADS-B*[1] environ égale à $200NM$, puisque les données dynamiques utilisées par *CAAMAS* (et *AGATS*) sont déjà contenues dans les messages échangés par les

1. système de surveillance dans lequel les avions déterminent leur position gps, et la diffusent par radio avec d'autres données qui le caractérisent (identifiant, vitesse, entre autres)

avions et qu'il suffit d'enrichir ceux-ci avec les mesures de criticités. Ainsi, la zone de perception était une sphère (figure 8.1) de rayon égal à la moitié de la distance de perception des signaux *ADS-B* (à savoir $100NM$), limitant ainsi la perte d'information.

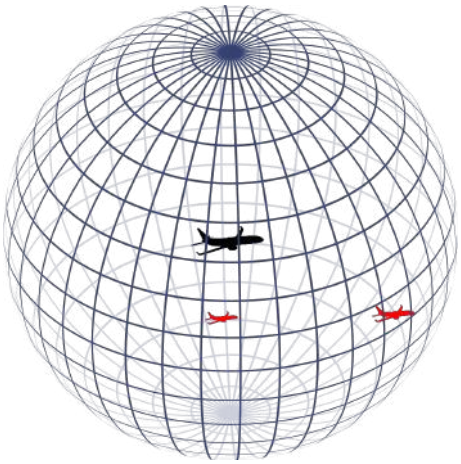


Figure 8.1 – Sphère de rayon $100NM$ représentant la première version de la zone de perception de l'*Agent EM*$_i$ noir au centre qui perçoit ici deux avions rouges

Cependant, nous avons constaté que les avions volent à une altitude de quelques kilomètres de la surface de la terre (par exemple, les avions commerciaux volent dans la troposphère, dont la limite haute est de $20km$), alors que le rayon de cette sphère de perception est de $100NM = 185,2km$. Nous avons donc décidé de caler la zone de perception sur la forme de la zone de conflit aéronautique (qui pour rappel est un cylindre orienté selon l'axe de l'altitude), en ayant un cylindre de rayon r_{Z_p} de $100NM$ ($100NM = 182,5km$) et de hauteur h_{Z_p} $4000ft$ ($4000ft = 1219,2m$), représenté dans la figure 8.2. Le rayon r_{Z_p} correspond toujours à la moitié de la portée horizontale de l'*ADS-B*, afin de prendre en compte de possibles congestions et pertes de signaux. La hauteur permet à l'*Agent EM*$_i$ de percevoir deux voies aériennes au dessus de lui et en dessous, selon la structure la plus répandue dans la gestion du trafic aérien décrite dans la section 1.2.2.

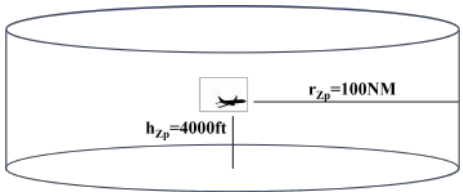


Figure 8.2 – La zone de perception de l'*Agent EM*$_i$

Les **compétences** de l'*Agent EM*$_i$ sont celles qui se spécifient le plus pour s'adapter aux spécificités de l'aviation.

De par les différentes contraintes physiques qui régissent leurs mouvements, les avions ne peuvent modifier leur trajectoire que dans une certaine enveloppe. Pour correspondre à ces contraintes, les **actions** des *Agents EM* sont de petites modifications de leur trajectoire

sur trois aspects :

— La vitesse : l'*Agent* EM_i peut décider d'accélérer, de ralentir, ou de ne pas modifier sa vitesse. La vitesse de l'avion est comprise dans un intervalle déterminé, $\left\| \overrightarrow{v}_{A_i} \right\| \in [v_{min}, v_{max}]$. Par la suite, ses bornes seront comprises entre [vopt - 6% ; vopt + 3%] comme dans le projet ERASMUS [Bonini et al., 2009]. De même, son accélération et sa décélération seront au maximum de $1,11NM.s^{-2}$.

— La direction horizontale : l'*Agent* EM_i peut décider de tourner à droite, à gauche, ou de ne pas modifier sa direction avec une vitesse de virage de $3°.s^{-1}$.

— L'altitude : l'*Agent* EM_i peut décider de monter, de descendre, ou de ne pas modifier son altitude. Dans l'expérimentation, cette compétence n'est pas utilisée, mais sa valeur par défaut est de $10m.s^{-1}$ en descente ou en montée.

L'*Agent* EM_i peut effectuer ses modifications en même temps. Il peut par exemple accélérer, tourner à droite, et garder son altitude actuelle. L'*Agent* EM_i peut donc effectuer à chaque instant t un total de 27 modifications de sa trajectoire qui correspondent aux différentes combinaisons d'actions sur la vitesse, la direction et l'altitude. Ces différentes actions peuvent être représentées comme des points dans l'espace vers lesquels peut se diriger l'avion (figure 8.3).

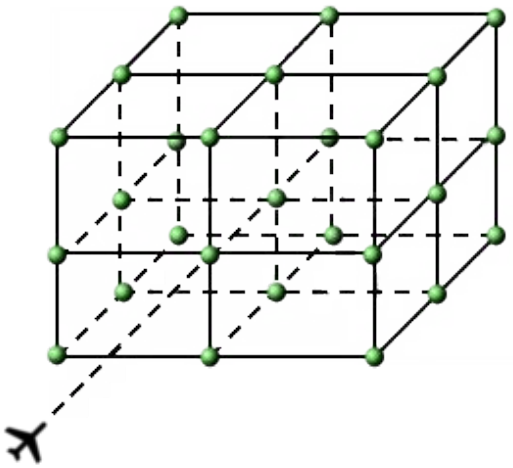


Figure 8.3 – Un avion et l'ensemble de ses actions (Ac_i)

L'*Agent* EM_i **communique** uniquement par des envois de messages contenant les mêmes informations décrites dans *AGATS* (position, vecteur vitesse et criticité).

Le calcul de la **criticité de collision** de l'*Agent* EM_i évolue légèrement, puisque nous considérons désormais les conflits aéronautiques au lieu des collisions. Un avion M_i est en conflit aéronautique avec un autre avion M_j s'il est dans son espace de sécurité (et par conséquent, puisque les zones de sécurité sont les mêmes, M_j est aussi en conflit aéronautique avec M_i).

Logiquement, $Conflit =$ Perte de séparation horizontale $\wedge$ Perte de séparation verticale. Partant de cette définition, nous avons adapté les criticités $Crit_{1,j,k}$ et $Crit_{2,j,k}$ pour présenter cette séparation. Dans ce qui suit, les conflits d'avions sont nommés des collisions.

Pour $d_{min,i,j,k}$ et $t_{min,i,j,k}$, les calculs considèrent les deux dimensions, c'est-à-dire $d_{min,i,j,k,ver}$, $d_{min,i,j,k,hor}$, $t_{min,i,j,k,ver}$ et $t_{min,i,j,k,hor}$,ainsi que les intervalles $\left[t_{start,i,j,k,ver}, t_{end,i,j,k,ver}\right]$, $\left[t_{start,i,j,k,hor}, t_{end,i,j,k,hor}\right]$ en cas de collision.

Pour calculer ces différentes valeurs, il suffit d'utiliser les formules 7.6, 7.7, 7.9, 7.10 en projetant les vecteurs de la manière suivante :

— Pour obtenir les valeurs horizontales, les vecteurs sont projetés sur le plan horizontal, ils sont donc remplacés comme suit : $\overrightarrow{p_{i,k}}(0) = (x_{i,k,0}, y_{i,k,0}, z_{i,k,0})$ par $(x_{i,k,0}, y_{i,k,0}, 0)$, $\overrightarrow{p_j}(0) = (x_{j,0}, y_{j,0}, z_{j,0})$ par $(x_{j,0}, y_{j,0}, 0)$, $\overrightarrow{v_{i,k}} = (v_{x,i,k}, v_{y,i,k}, v_{z,i,k})$ par $(v_{x,i,k}, v_{y,i,k}, 0)$ et $\overrightarrow{v_j} = (v_{x,j}, v_{y,j}, v_{z,j})$ par $(v_{x,j}, v_{y,j}, 0)$.

— Pour obtenir les valeurs verticales, les vecteurs sont projetés sur l'axe vertical, ils sont donc remplacés comme suit : $\overrightarrow{p_{i,k}}(0) = (x_{i,k,0}, y_{i,k,0}, z_{i,k,0})$ par $(0, 0, z_{i,k,0})$, $\overrightarrow{p_j}(0) = (x_{j,0}, y_{j,0}, z_{j,0})$ par $(0, 0, z_{j,0})$, $\overrightarrow{v_{i,k}} = (v_{x,i,k}, v_{y,i,k}, v_{z,i,k})$ par $(0, 0, v_{z,i,k})$ et $\overrightarrow{v_j} = (v_{x,j}, v_{y,j}, v_{z,j})$ par $(0, 0, v_{z,j})$.

La criticité $Crit_{1,j,k}$, est donc représentée par deux criticités, une pour chacune des deux dimensions, $Crit_{1,j,k,hor}$ et $Crit_{1,j,k,ver}$. Les deux criticités ont la même forme que $Crit_{1,j,k}$, en remplaçant d_{coll} par $d_{coll,hor}$ pour $Crit_{1,j,k,hor}$ et $d_{coll,ver}$ pour $Crit_{1,j,k,ver}$.

$$Crit_{1,j,k,ver} = \begin{cases} 100 - \frac{100}{2d_{coll,ver}} d_{min,i,j,k,ver} & d_{min,i,j,k,ver} < 2d_{coll,ver} \\ 0 & d_{min,i,j,k,ver} \geq 2d_{coll,ver} \end{cases} \tag{8.1}$$

$$Crit_{1,j,k,hor} = \begin{cases} 100 - \frac{100}{2d_{coll,hor}} d_{min,i,j,k,hor} & d_{min,i,j,k,hor} < 2d_{coll,hor} \\ 0 & d_{min,i,j,k,hor} \geq 2d_{coll,hor} \end{cases} \tag{8.2}$$

Avec,

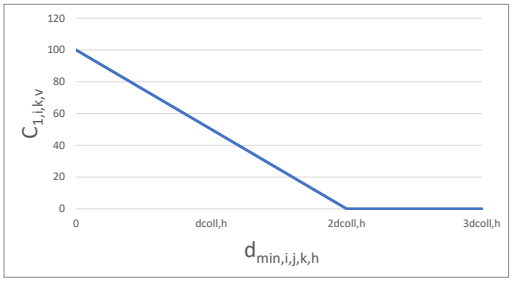
$$d_{coll,hor} = r_{conflit}$$
$$d_{coll,ver} = h_{conflit}$$

Ces deux criticités sont représentées par les figures 8.4 a et b.

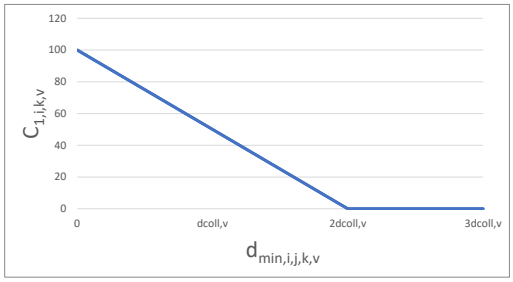
Une collision étant une rencontre dans la dimension verticale et dans le plan, la criticité $Crit_{1,j,k}$ est une combinaison de ses deux sous-criticités, et afin de maintenir l'esprit de la mesure, nous utilisons le minimum des deux criticités (ainsi, deux avions l'un au dessus de l'autre ne sont pas considérés en collision selon $Crit_{1,j,k}$ tant qu'ils ne sont pas véritablement en collision) :

$$Crit_{1,j,k} = \min(C_{1,j,k,ver}, C_{1,j,k,hor}) \tag{8.3}$$

Afin de calculer **la criticité** $Crit_{2,j,k}$, nous utilisons les différentes valeurs $t_{min,i,j,k,ver}$ et $t_{min,i,j,k,hor}$, $\left[t_{start,i,j,k,ver}, t_{end,i,j,k,ver}\right]$, $\left[t_{start,i,j,k,hor}, t_{end,i,j,k,hor}\right]$.



(a) $C_{1,j,k,hor}$



(b) $C_{1,j,k,ver}$

Figure 8.4 – Adaptation de $C_{1,j,k}$ et prise en compte des deux dimensions

S'il y a une collision, et que les deux intervalles sont calculés, nous calculons leur intersection $\left[t_{start,i,j,k}, t_{end,i,j,k}\right]$ et nous utilisons $t_{start,i,j,k}$.

$$\left[t_{start,i,j,k}, t_{end,i,j,k}\right] = \left[t_{start,i,j,k,hor}, t_{end,i,j,k,hor}\right] \cap \left[t_{start,i,j,k,ver}, t_{end,i,j,k,ver}\right]$$

Sinon, nous utilisons la valeur maximale entre $t_{min,i,j,k,ver}$ et $t_{min,i,j,k,hor}$.

$$t_{min,i,j,k} = \begin{cases} \max(t_{min,i,j,k,ver}, t_{min,i,j,k,hor}) & \text{si pas de collision} \\ t_{start,i,j,k} & \text{si collision} \end{cases}$$

Et ensuite nous utilisons la valeur $t_{min,i,j,k}$ (ou $t_{start,i,j,k}$ s'il y a une collision), dans la même fonction de criticité :

$$Crit_{2,j,k} = \begin{cases} 100 & t_{min,i,j,k} < t_r \\ 100 - \frac{100}{2t_r}(t_{min,i,j,k} - t_r) & t_{min,i,j,k} \in [t_r, 3t_r] \\ 0 & t_{min,i,j,k} \geq 3t_r \end{cases}$$

Avec,

$$t_r = \frac{d_{t_r}}{||\overrightarrow{v_i}||}, d_{t_r} = r_{Z_p}$$

La figure 8.5 montre l'adaptation des criticités des *Agents EM* du système *AGATS* pour le système *CAAMAS*.

Concernant les différents modules de l'*Agent* EM_i, ceux-ci sont implémentés et sont les mêmes que dans le système *AGATS* à quelques adaptations près. Les *Modules Voisins* n'estiment pas les SNCs des *Agents* EM_j. Les *Modules Actions* ajoutent une préférence dans les actions des *Agents EM* afin de prendre en compte les préférences du contrôle aérien (section 1.3.1), mais ne peuvent pas modifier leur action (8.1).

N'ayant pas implémenté d'algorithme capable d'estimer la trajectoire voulue par ses voisins, l'algorithme de décision de notre ***Module Décision*** a été adapté pour faire diminuer

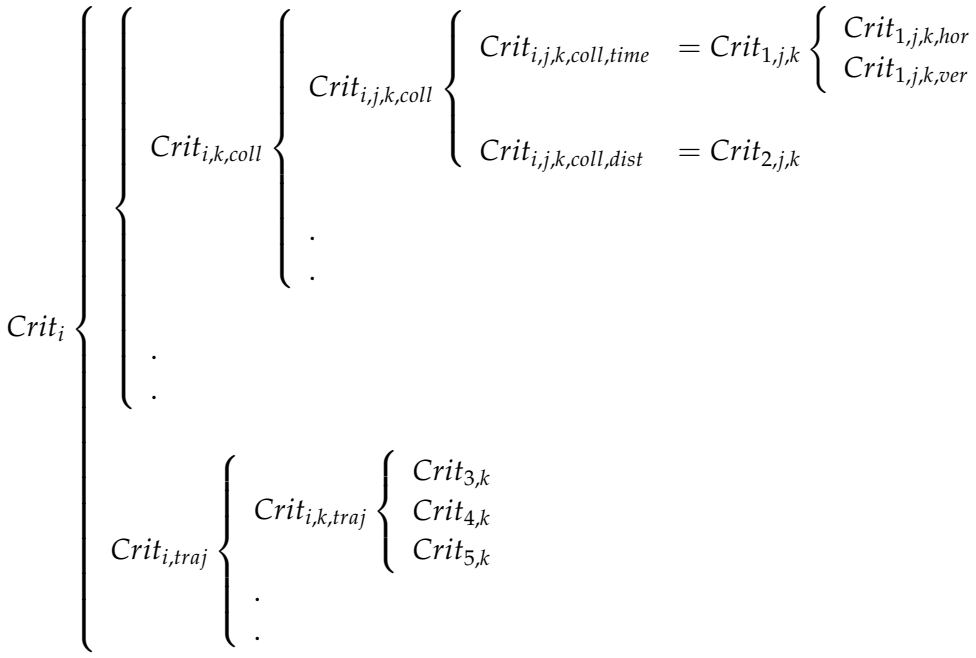


Figure 8.5 – Adaptation des criticités de l'*Agent EM*$_i$ pour l'aviation

Algorithme 8.1 : Implémentation du cycle de vie du *Module Action*

1: Calculer $Crit_{i,k,traj}$;
2: Recevoir les différentes informations des *Agent EM*$_j$ ($Crit_j$, $\overrightarrow{p_j}$, $\overrightarrow{v_j}$) ;
3: **pour tout** *Agent EM*$_j$ **faire**
4: Calculer $Crit_{i,j,k,coll}$;
5: Mettre $Crit_{i,j,k,coll}$ dans la liste $Crit_{i,k,coll}$;
6: **fin pour**
7: Envoyer $Crit_{i,k} = (Crit_{i,k,traj}, Crit_{i,k,coll})$ et son action au *Module Décision*

la criticité de collision pour les autres et les criticités de trajectoire et de collision pour M_i (algorithme 8.2).

En revanche, nous avons implémenté la préférence des manoeuvres décrite dans la section 1.3.1, ainsi, lors de la phase (7) de l'algorithme 8.2, si l'ensemble des actions $\{Ac_{i,k}\}$ contient différents types d'actions, seront extraites les actions, si elles existent, qui modifient uniquement la vitesse, sinon, si elles existent, les actions qui modifient uniquement le cap de l'avion, puis, toujours si elles existent, celles qui modifient uniquement l'altitude, puis uniquement les actions avec les deux types de modifications (avec la même préférence).

Nous avons implémenté les SNC suivantes :

— SNC 4 : Incompréhension Agent EM

— SNC 5 : Ambiguïté d'un Agent EM, message incomplet. L'estimation de la criticité pour cette SNC n'a pas été implémenté, en revanche, l'estimation de la vitesse, de l'accélération ou de la position l'ont été. Afin d'estimer ces dernières, ce travail utilise leurs

Algorithme 8.2 : Comportement du *Module Décision* dans l'implémentation

Soit L, la liste contenant l'ensemble des couples ($Ac_{i,k}$, $Crit_{i,k}$)
Soit V, la liste contenant les couples (*Agent* EM_j,$Crit_j$) et le couple (*Agent* EM_i,$Crit_i$)
Trier V en fonction des criticités, du plus critique au moins critique
$L_{temp} \leftarrow L$
$j_{actuel} \leftarrow 1$ //Indice utilisé dans la suite pour parcourir les agents du plus critique au moins critique
tant que L_{temp}.taille()>1 et j_{actuel}<V.taille() **faire**
  si V[j_{actuel}]$\neq$*Agent* EM_i **alors**
    Trouver l'action $Ac_{i,k}$ telle que $Crit_{i,j_{actuel},k,coll}$ soit minimale pour tout couple ($Ac_{i,k}$, $Crit_{i,k}$) dans L_{temp} // On cherche l'action qui minimise la criticité de collision avec l'agent $EM_{j_{actuel}}$, qui est le j_{actuel}ème agent le plus critique. Pour rappel, $Crit_{i,j,k,coll}$ est contenue dans $Crit_{i,j,k}$, elle même contenue dans $Crit_{i,k}$.
    Enlever toutes les actions de L_{temp} dont la criticité $Crit_{i,j,k,coll}$ dans $Crit_{j,k}$ est telle que $Crit_{i,j_{actuel},k,coll} > Crit_{i,j_{actuel},k_{j_{actuel}},coll} + \epsilon$ // On enlève toutes les actions dont la criticité est trop élevée par rapport à cette action de criticité minimale
  sinon
    Trouver l'action $Ac_{i,k}$ telle que $Crit_{i,k}$ soit minimale pour tout couple ($Ac_{i,k}$, $Crit_{i,k}$) dans L_{temp} // On cherche l'action qui minimise la criticité globale (trajectoire et collision) de l'*Agent* EM_i.
    Enlever toutes les actions de L_{temp} dont la criticité $Crit_{i,k}$ est telle que $Crit_{i,k} > Crit_{i,k_{j_{actuel}}} + \epsilon$ // On enlève toutes les actions dont la criticité est trop élevée par rapport à cette action de criticité minimale
  fin si
  $j_{actuel} \leftarrow j_{actuel} + 1$
fin tant que

définition respectives, et les approximations habituelles :

$$\overrightarrow{v(t)} = \frac{d\overrightarrow{p(t)}}{dt} \approx \frac{\overrightarrow{p_1} - \overrightarrow{p_0}}{t_1 - t_0}$$
$$\overrightarrow{a(t)} = \frac{d\overrightarrow{v(t)}}{dt} \approx \frac{\overrightarrow{v_1} - \overrightarrow{v_0}}{t_1 - t_0}$$
$$\overrightarrow{p(t)} = \int \overrightarrow{v(t)}dt \approx \overrightarrow{p_0} + (t_1 - t_0)\overrightarrow{v_0}$$

— SNC 6 : Incompréhension d'un Agent EM

— SNC 7 : Inutilité de l'action d'un Agent EM

— SNC 8 : Incompétence d'un Agent EM, suivi de trajectoire. Cette SNC a été implémentée selon sa description si la SNC concerne l'*Agent* EM_i, elle n'a en revanche pas été implémentée si elle concerne l'*Agent* EM_j.

— SNC 9 : Improductivité d'un *Agent EM*, pas d'action.

8.2 CAAMAS : Expérimentations et analyse

Nous présentons ici les différentes expérimentations effectuées sur le système CAAMAS adapté au contexte aéronautique présenté dans la section précédente. Nous présentons dans un premier temps des expérimentations sur le benchmark dit du rond-point afin de comparer les solutions données par *CAAMAS* avec des solutions optimales (section 8.2.2), puis des expérimentations sur un benchmark dit aléatoire, pour tester le système dans un contexte réaliste (section 8.2.3), afin d'étudier la sensibilité sur certains des paramètres de *CAAMAS* (section 8.2.4) et aussi de comparer le système *CAAMAS* avec un autre système (section 8.2.5).

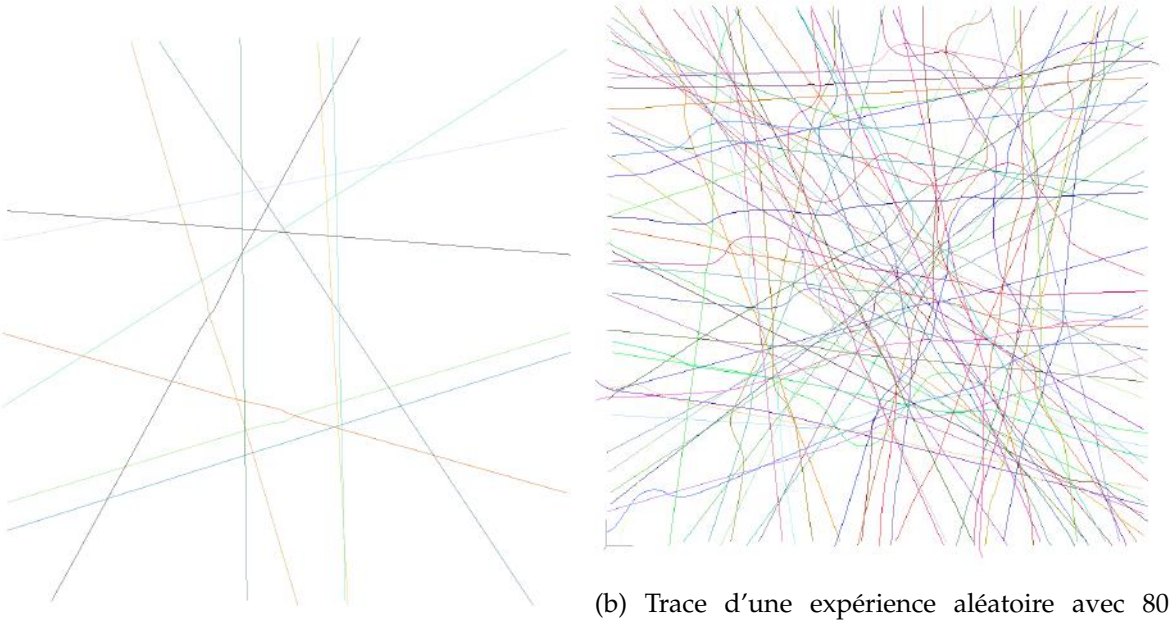
8.2.1 Présentation des benchmarks

Nous utilisons dans ces expérimentation deux benchmarks présentés dans [Durand and Barnier, 2015] et [Durand, 2018], le benchmark du rond-point et le benchmark aléatoire.

Le benchmark du **rond-point** est un benchmark classique du monde aéronautique qui consiste à placer tous les avions dans un même plan, sur le bord d'un disque de rayon R. Les entités mobiles sont placées à intervalle régulier sur le bord, et convergent toutes vers le centre du disque avec un angle de $\frac{2\Pi}{AgNb}$ entre elles (figure 8.8a). Des variantes existent, en plaçant les avions uniquement sur un côté du disque, comme dans [Durand and Barnier, 2015]. Le nom du benchmark vient de la solution optimale pour le collectif, qui est de créer un rond-point autour du centre du disque empêchant les avions d'entrer en collision.

Le benchmark **aléatoire** quant à lui consiste à placer les avions dans le même plan, sur le bord d'un carré de côté l. Les avions sont divisés en quatre groupes égaux, et chaque avion d'un même groupe part d'un point choisi aléatoirement sur l'une des faces du carré et doit rejoindre un point choisi aléatoirement sur la face opposée du carré (figure 8.6a et 8.6b). Deux points de départ ou d'arrivée, doivent être distants au minimum d'une certaine distance d_e. La distance d_e est choisie de telle sorte que les avions ne soient pas en difficulté dès le début de la simulation, en particulier qu'ils ne soient pas à une distance $d_e < d_{coll}$.

Ces deux benchmarks se déroulant dans le plan, nous avons réduit l'ensemble des actions des *Agents EM* aux actions dans le plan horizontal (pas de montée ni de descente, donc 9 actions possibles), comme le montre la figure 8.7.



(a) Trace d'une expérience aléatoire avec 12 avions, tous arrivent à destination

(b) Trace d'une expérience aléatoire avec 80 avions : 76 sont arrivés, 4 sont sortis excessivement, et 0 collision

Figure 8.6 – Deux expérimentations de *CAAMAS* pour le benchmark aléatoire

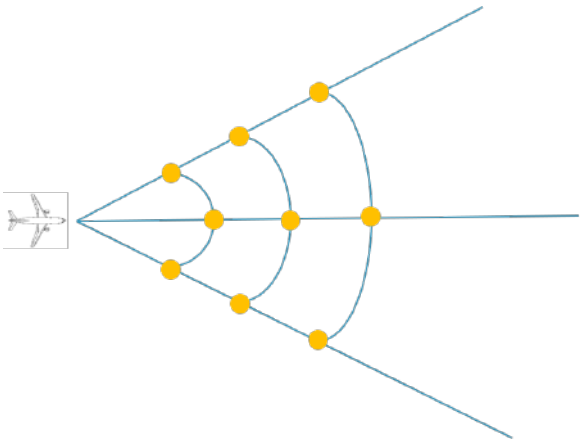


Figure 8.7 – Actions possibles pour un avion dans le plan (n'est pas à l'échelle)

Dans nos différentes expérimentations, nous utilisons les métriques suivantes :

— temps de calcul,

— nombre d'avions arrivant à destination, pour cela, l'avion doit arriver suffisamment proche de sa destination (cette distance est fixée à la distance parcourue par un *Agent EM* en un cycle de vie) sans sortir excessivement de la zone carrée (d'une distance $2r_{conflit}$),

— collisions restantes, le nombre de paires d'avions qui sont entrées en collision,

$AgNb$	Délais Relatif Moyen	Déviation Standard	Q1	Q3
6	1.00735	0.01210	1.0	1.01353
8	1.01310	0.02028	1.00000	1.02029
10	1.02291	0.03447	1.00338	1.03721

Table 8.1 – Délais relatif dus à l'évitement de collisions

— allongement du temps de trajectoire moyen, défini par :

$$\frac{|l_{i,wanted} - l_{i,final}|}{l_{i,wanted}}$$

Avec $l_{i,wanted}$ la longueur de la trajectoire prévue τ_i, et $l_{i,final}$ la longueur de la trajectoire effectuée par l'agent EM.

Nous utilisons ces benchmarks et ces métriques dans les sections suivantes.

8.2.2 Benchmark du rond-point

Dans ce benchmark, [Durand and Barnier, 2015] utilise un disque de rayon $R = 125NM = 231,5km$. Nous avons donc adapté le comportement des *Agents EM* et généré des scénarios avec les mêmes paramètres pour permettre la comparaison des résultats obtenus. Ainsi, seuls les changements de cap sont autorisés, les vitesses seront donc fixes et normalisées, ainsi le nombre d'actions possibles est réduit au nombre de trois (tourner à gauche, tourner à droite, ne pas modifier l'orientation).

Les paramètres de ce benchmark sont :

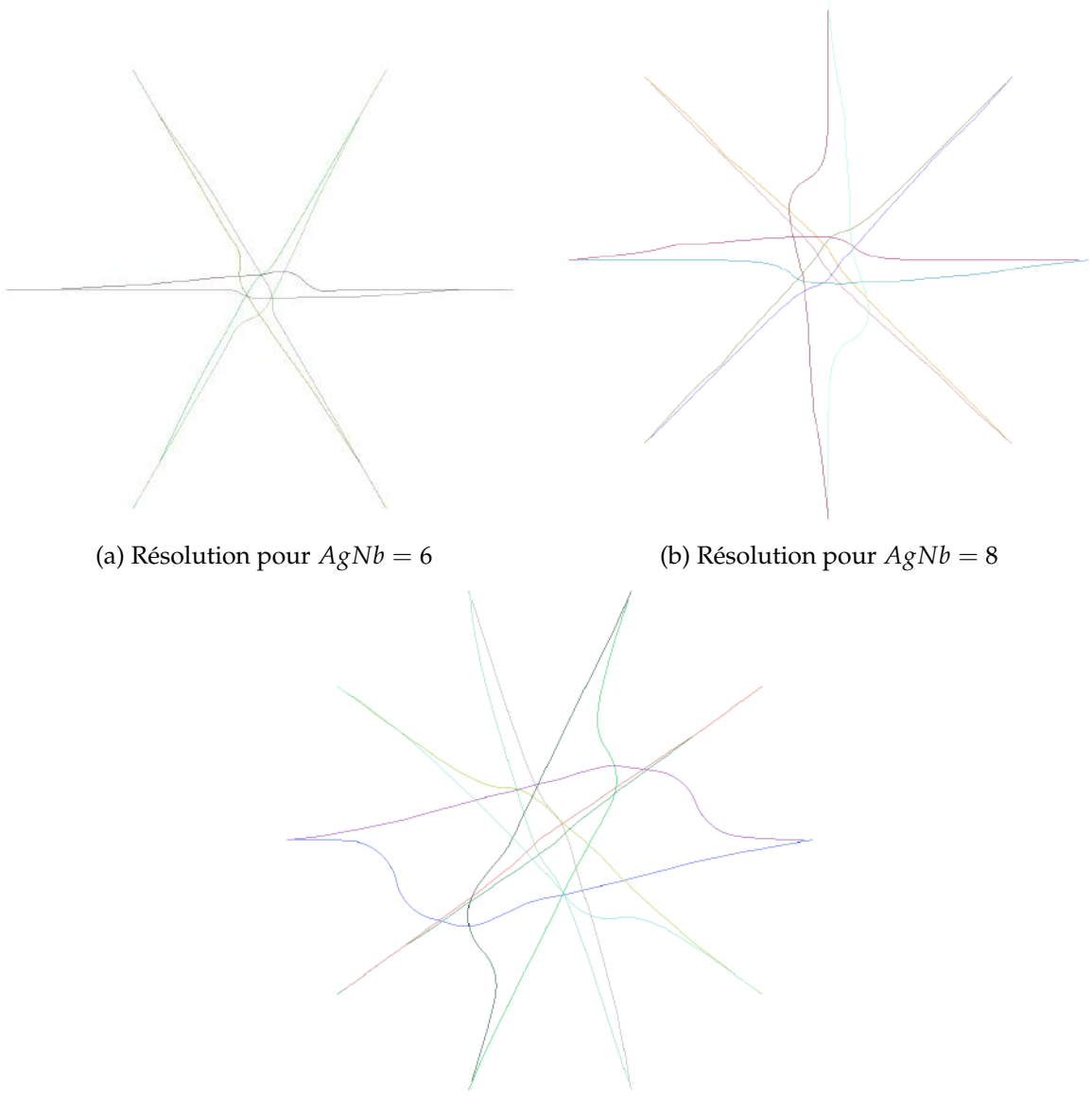
— le nombre d'avions, $AgNb$, qui varie dans l'ensemble $\{6, 8, 10\}$,

— la vitesse de rotation maximale de l'avion, qui est de $3\,\mathrm{deg}\,.s^{-1}$,

— la vitesse est constante, et donc l'accélération est nulle.

Les figures 8.8a, 8.8b et 8.8c présentent la résolution du benchmark par *CAAMAS* pour $AgNb = 6$, $AgNb = 8$ et $AgNb = 10$ respectivement, et le tableau 8.1 regroupe les résultats sur l'allongement de trajectoire.

Le pattern du rond-point émerge des différentes interactions entre les *Agents EM*, en particulier pour le cas $AgNb = 6$. Néanmoins, l'écart-type souligne le fait que le retard relatif n'est pas toujours réparti de manière égale entre les avions (en particulier pour le cas $AgNb = 10$), et le rond-point se dégénère légèrement. Ceci est dû à l'implémentation faite pour cette expérimentation dans laquelle les *Agents EM* décident de manière itérative, ainsi les premiers agents qui agissent sont plus pénalisés que les autres (voir figure 8.8c). Enfin, dans notre approche, les agents retournent sur leur trajectoire une fois la collision évitée avant de se rendre à leur destination, alors que dans la plupart des algorithmes, les avions se rendent directement à la destination, ce qui augmente le retard relatif causé aux avions. Néanmoins, les *Agents EM* sont capables de trouver une solution quasi optimale, malgré l'implémentation, pour des cas avec peu d'avions.



(a) Résolution pour $AgNb = 6$

(b) Résolution pour $AgNb = 8$

(c) Résolution pour $AgNb = 10$

Figure 8.8 – Résolution du benchmark du rond-point par CAAMAS

8.2.3 Benchmark aléatoire

Nous testons les comportements des *Agents EM* sur le benchmark aléatoire pour les paramètres prédéterminés afin de tester les comportements d'adaptation de ces agents dans un contexte réaliste.

Pour rappel, les paramètres prédéterminés de l'*Agent EM* sont :

— La vitesse, qui peut varier dans un intervalle $[v_{nom} - 5\%, v_{nom} + 5\%]$, avec $v_{nom} = 450NM.h^{-1}$,

— L'accélération, qui est fixée à $1.11NM.h^{-2}$,

— La vitesse de rotation maximale, qui est de $3°.s^{-1}$

— Le temps de chaque cycle de décision, qui est de $\Delta t = 1s$

Pour ce benchmark, la longueur de côté du carré est fixée à $l = 500NM$, et $d_e = 2d_{coll}$. Le nombre d'avions $AgNb$ varie dans l'ensemble $\{12, 20, 32, 40, 52, 60, 72, 80, 92, 100, 112, 120\}$. Pour chaque $AgNb$, 100 tests aléatoires sont réalisés, avec des positions générées aléatoirement pour chaque test en respectant la contrainte d'espacement pour chaque test.

Enfin, nous observons 4 métriques dans cette étude :

— le temps de calcul,

— le nombre d'avions arrivant à destination,

— le nombre de collisions restantes,

— l'allongement du temps de trajectoire moyen.

Les figures 8.9, 8.10, 8.11 et 8.12 présentent les résultats de cette étude sur ces 4 métriques. Chaque figure représente pour chaque valeur de $AgNb$ la valeur minimale, le premier quartile, la valeur médiane, le troisième quartile, et la valeur maximale de la métrique représentée.

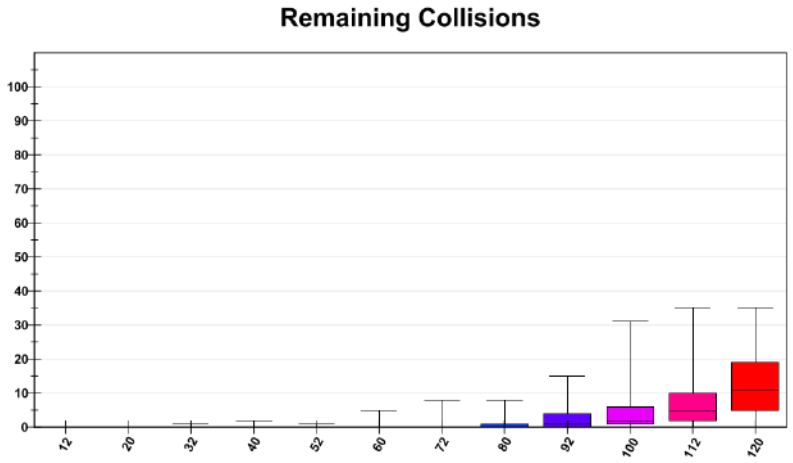


Figure 8.9 – Nombre de collisions (perte de séparation) entre les avions pour différentes valeurs d'$AgNb$

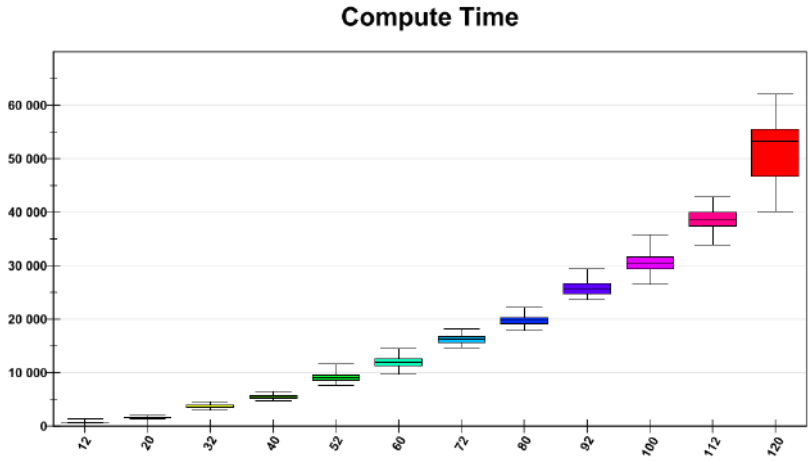


Figure 8.10 – Temps de calcul des agents (ms) de *CAAMAS* pour différentes valeurs d'$AgNb$

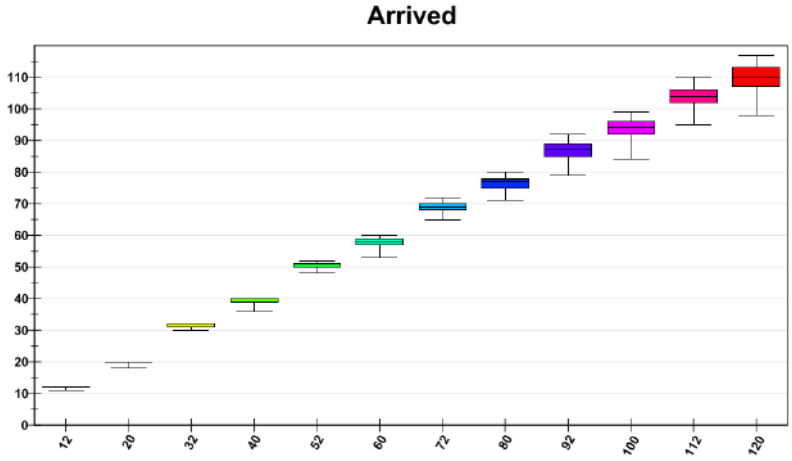


Figure 8.11 – Nombre d'avions arrivant à destination pour différentes valeurs d'$AgNb$

Nous avons effectué les tests dans un espace carré de côté $l = 500NM = 926km$. La taille de cet espace correspond globalement à un carré englobant la France en entier (la distance entre Brest et Strasbourg est de 900km, et celle entre Lille et Perpignan est de 882km), qui correspond à la taille de l'espace aérien français sur un niveau de vol (voir section 1.2, notamment la figure 1.3). Le trafic en juin 2018, mois le plus chargé sur l'année 2018, était de 175 milliers de vols IFR (avion respectant les règles IFR, section 1.2) volant dans l'espace aérien français [DGAC, 2018]. Cela correspond à 42 avions en même temps sur un seul niveau de vol dans tout l'espace aérien français, si :

— ce trafic est réparti uniformément sur tous les jours du mois de juin ;
— les avions volent seulement entre 6h et 24h dans l'espace aérien ;
— les avions volent deux heures dans l'espace aérien ;
— le trafic est réparti uniformément sur la journée ;
— les avions volent uniquement entre le FL240 et FL370.

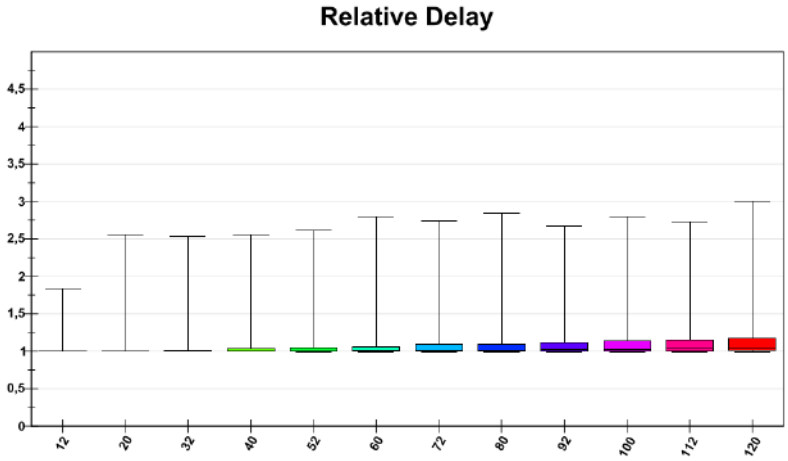


Figure 8.12 – Allongement relatif des trajectoires pour différentes valeurs d'$AgNb$

Nos expérimentations montrent que les agents de *CAAMAS* sont capables dans la grande majorité des cas d'éviter des collisions entre les différents avions, malgré une densité forte d'avions et plus élevée que dans la réalité (figure 8.9), et dans une configuration très différente et plus compliquée, dans laquelle, en probabilité, les avions convergent tous en même temps vers le centre du carré. En effet, pour des cas avec $AgNb < 80$ avions, les *Agents EM* sont capables de s'adapter pour éviter totalement toute collision entre eux dans la grande majorité des cas (100% pour $AgNb < 32$ et entre 96% et 100% pour $AgNb = 32, 40, 52 et 60$ et au moins 75% pour $AgNb = 72$). Les cas $AgNb = 92$ et $AgNb = 100$ semblent être des cas pivots pour le nombre de collisions puisque les tests avec au moins une collision deviennent majoritaires. A noter que dans cette étude, une perte de séparation est considérée comme conflit. Une perspective intéressante est d'étudier l'importance des pertes de séparation détectées et notamment voir si dans le cadre d'une automatisation complète, il n'est pas possible de réduire la taille de cette zone de séparation.

Du point de vue du temps de calcul, les résultats montrent globalement une non linéarité du temps de calcul avec le nombre d'agents. Pour autant nous notons une linéarité quasi-exacte de temps de calculs médians entre les valeurs $AgNb = 40$ et $AgNb = 80$ (coefficient de corrélation de $0,9979$) et un peu moins exacte entre les valeurs $AgNb = 12$ et $AgNb = 40$ (coefficient de corrélation de $0,9931$). A partir de $AgNb \geq 80$, la courbe semble plutôt suivre une fonction carré [2], nous notons tout de même une forte variabilité des résultats pour le cas $AgNb = 120$ en comparaison avec les autres cas (une différence de $15,9\%$ entre le premier et troisième quartile). Ceci dit, les résultats montrent un passage à l'échelle intéressant, avec une capacité du système multi-agent à adapter les trajectoires afin de prendre en compte l'environnement de l'entité mobile en quelques secondes (ou quelques dizaines de secondes pour les cas les plus denses). Les décisions et la plupart des calculs étant naturellement décentralisés, une perspective intéressante est de distribuer les calculs des agents, et les

2. Notons qu'ici nous ne parlons pas de la complexité algorithmique théorique du système multi-agent ; mais seulement d'un interpolation pour un nombre d'agents satisfaisant (plus d'une centaine ce qui correspond à 3 fois le trafic dit réaliste) pour l'application

calculs des modules de l'*Agents EM* en particulier ceux des *Modules Actions*.

La limitation de l'espace n'étant pas spécifiée dans les agents, un nombre non négligeable d'agents peuvent sortir de la zone lorsque la densité augmente, ou à cause du fait que les départs et les destinations se font depuis le bord du carré et sont comptés comme n'arrivant pas à destination. Ainsi, pour $AgNb < 30$ les avions arrivent tous à destination dans plus de 75% des tests, pour $AgNb = 52$ au plus deux avions sortent dans 75% des cas, et pour $AgNb \geq 100$, tous les tests ont au moins un avion qui n'arrive pas à destination.

Pour les cas avec $AgNb < 80$, l'allongement général des trajectoires est relativement faible. En effet, l'allongement des trajectoires est au maximum de $8,3\%$ pour le troisième quartile et de $1,4\%$ pour la médiane (pour $AgNb = 80$). En revanche, pour les cas avec $AgNb > 80$, l'allongement des trajectoires est plus important, avec au maximum de $17,4\%$ pour le troisième quartile et de $1,4\%$ pour la valeur médiane (pour $AgNb = 120$). Certains pics très important dans l'allongement de trajectoire sont à étudier, malgré leur non-persistance qui suggère que ce sont des cas très isolés.

8.2.4 Étude de sensibilité

Nous étudions dans cette section la sensibilité de *CAAMAS* à différents paramètres utilisés. Pour cela, nous faisons évoluer un paramètre de *CAAMAS* et le nombre d'avions, et nous fixons les autres valeurs de paramètres.

Les paramètres de base sont :

— la vitesse qui peut varier dans un intervalle $[v_{nom} - 6\%, v_{nom} + 3\%]$, avec $v_{nom} = 450NM.h^{-1}$,

— l'accélération qui est fixée à $1.11NM.h^{-2}$,

— la vitesse de rotation maximale qui est de $2°.s^{-1}$.

Les paramètres de *CAAMAS* qui nous faisons évoluer sont :

1. $d_{coll,hor}$, avec $d_{coll,hor} = r_{conflit} = 5NM$ comme valeur par défaut, utilisé dans $Crit_{1,j,k,hor}$,
2. $t_r = \frac{d_{tr}}{||\overrightarrow{v_i}||}$, en faisant varier d_{tr}, avec $d_{tr} = r_{Z_p}$ comme valeur par défaut, utilisé dans $Crit_{2,j,k}$,
3. r_{Z_p}, avec $r_{Z_p} = 100NM$ comme valeur par défaut pour le rayon de la zone de perception,
4. $\dot{\alpha}$, avec $\dot{\alpha} = 3°.s^{-1}$ comme valeur par défaut pour la vitesse de rotation,
5. Δt, avec $\Delta t = 1s$ comme valeur par défaut pour le temps entre chaque cycle de vie des agents.

Nous étudions la sensibilité de *CAAMAS* à ces paramètres en utilisant le benchmark aléatoire. Pour cette étude, nous gardons une longueur $l = 500NM$ pour le carré, et nous utilisons une distance $d_e = 2d_{coll,hor}$.

Afin de tester la sensibilité de ces 5 paramètres, nous considérons 5 cas d'études. Pour chaque cas, au moins 5 valeurs sont considérées pour le paramètre étudié, les autres paramètres sont fixés à leur valeur par défaut. Pour chaque valeur considérée, le nombre d'avions $AgNb$ évolue dans $\{12, 20, 32, 40, 52, 60, 72, 80, 92, 100, 112, 120\}$. Pour chaque valeur de $AgNb$, 100 exécutions du système (avec des positions aléatoires différentes pour

chaque exécution) sont effectuées. Ainsi, pour chaque cas d'étude, un minimum de $5 \times 12 \times 100 = 6000$ exécutions sont réalisées.

Enfin, nous observons 4 métriques dans cette étude de sensibilité :

— temps de calcul,

— nombre d'avions arrivant à destination,

— collisions restantes,

— allongement du temps de trajectoire moyen.

Chaque courbe affiche l'une de ces métriques. Elles représentent les valeurs médianes sur les 100 tests.

8.2.4.1 Sensibilité au paramètre $d = d_{coll,hor}$ de la fonction de criticité $Crit_{1,j,k,hor}$

Pour cette étude nous faisons varier $d_{coll,hor}$, utilisé dans la fonction de criticité $Crit_{1,j,k,hor}$ (section 8.1.3), en préservant $r_{conflit}$:

— $d_{coll,hor} = r_{conflit} = 5NM$, la valeur qui avait été prédéterminée pour le système AGATS,

— $d_{coll,hor} = 10NM (= 18,52,3km)$, soit $2r_{conflit}$,

— $d_{coll,hor} = 15NM (= 27,78km)$, soit $3r_{conflit}$,

— $d_{coll,hor} = 20NM (= 37,04km)$, soit $4r_{conflit}$,

— $d_{coll,hor} = 25NM (= 46,30km)$, soit $5r_{conflit}$,

— $d_{coll,hor} = 30NM (= 55,560km)$, soit $6r_{conflit}$.

Les figures 8.13, 8.14, 8.15 et 8.16 présentent les valeurs médianes des différentes métriques en fonction de $AgNb$ pour chaque valeur de $d_{coll,hor}$.

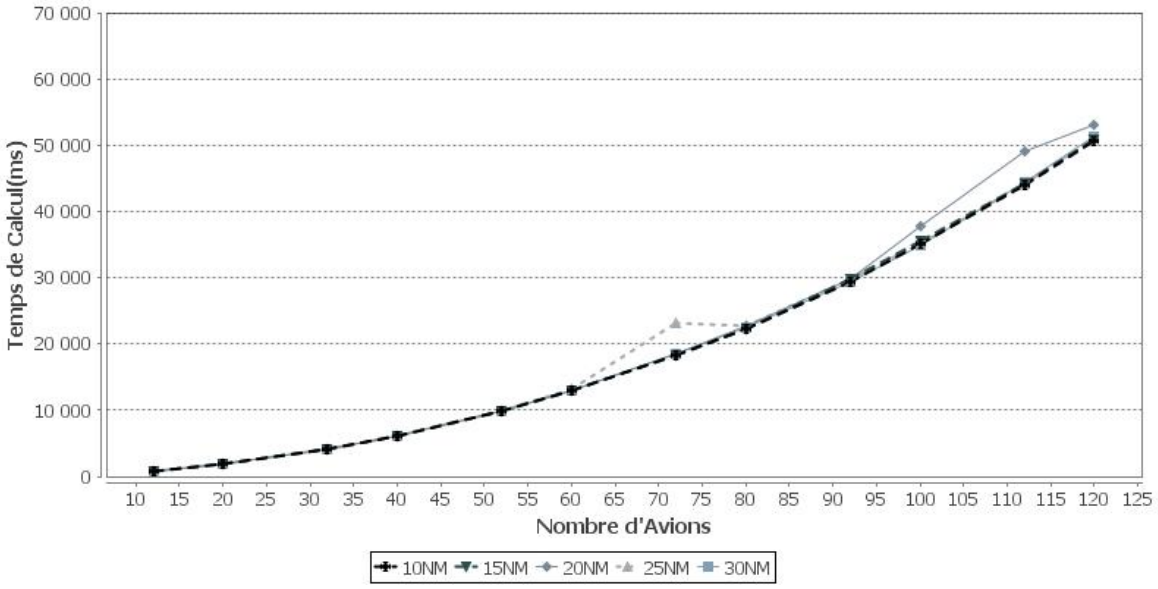


Figure 8.13 – Temps de calcul en fonction d'$AgNb$ pour différentes valeurs de $d_{coll,hor}$

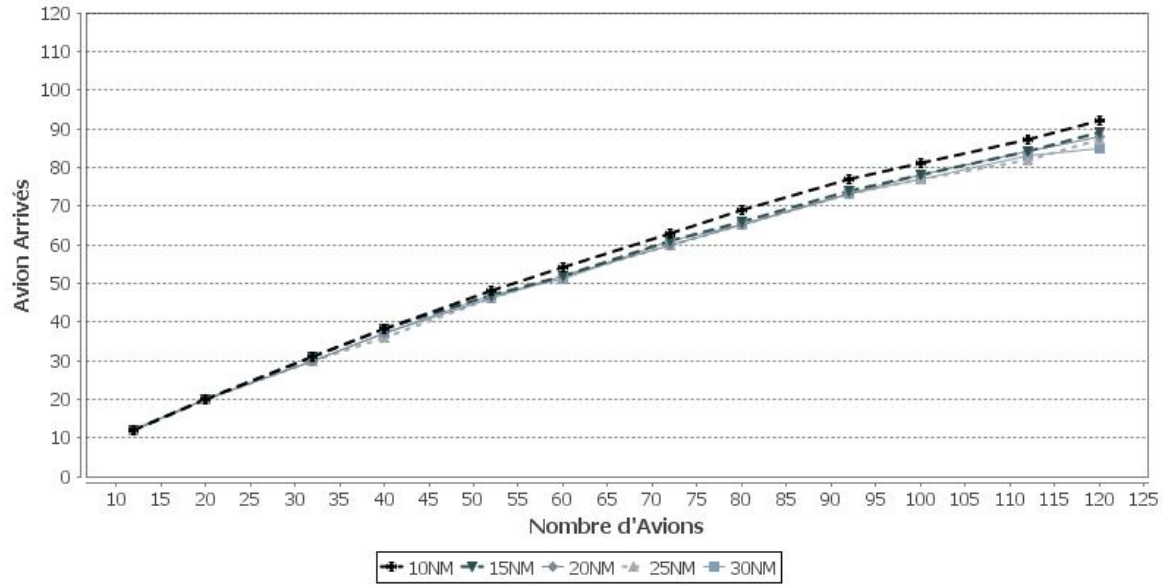


Figure 8.14 – Nombre d'avions arrivants en fonction d'$AgNb$ pour différentes valeurs de $d_{coll,hor}$

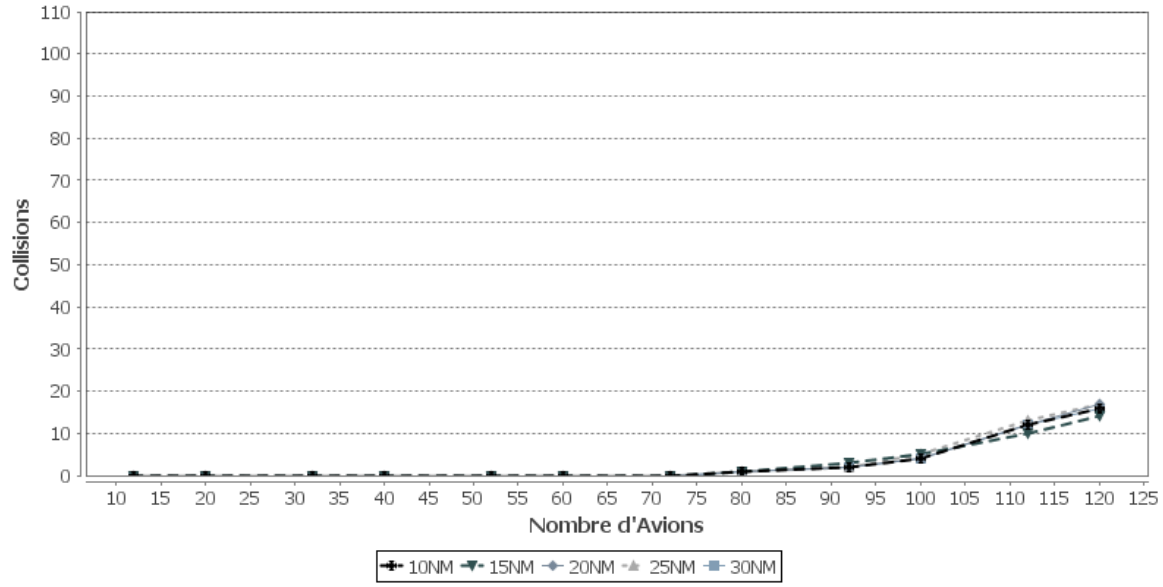


Figure 8.15 – Collisions restantes en fonction d'$AgNb$ pour différentes valeurs de $d_{coll,hor}$

Discussion de la sensibilité du système aux modifications de $d_{coll,hor}$. Les 4 figures (figures 8.13, 8.14, 8.15 et 8.16) montrent la robustesse du système quant aux variations de ce paramètre, de l'adaptabilité du comportement agent et valide le rôle d'ordonnancement de la fonction de criticité $Crit_{1,j,k,hor}$. Deux points sont à noter :

1. La courbe représentant le temps de calcul pour $AgNb$ (figure 8.13). Étant donné que les autres valeurs sont assez concentrées, nous pensons que cet écart est dû à l'utilisation de la machine de calcul pour d'autres tâches pendant une partie de cette expérimentation.
2. Nous remarquons une petite influence du paramètre $d_{coll,hor}$ sur le nombre d'avions qui

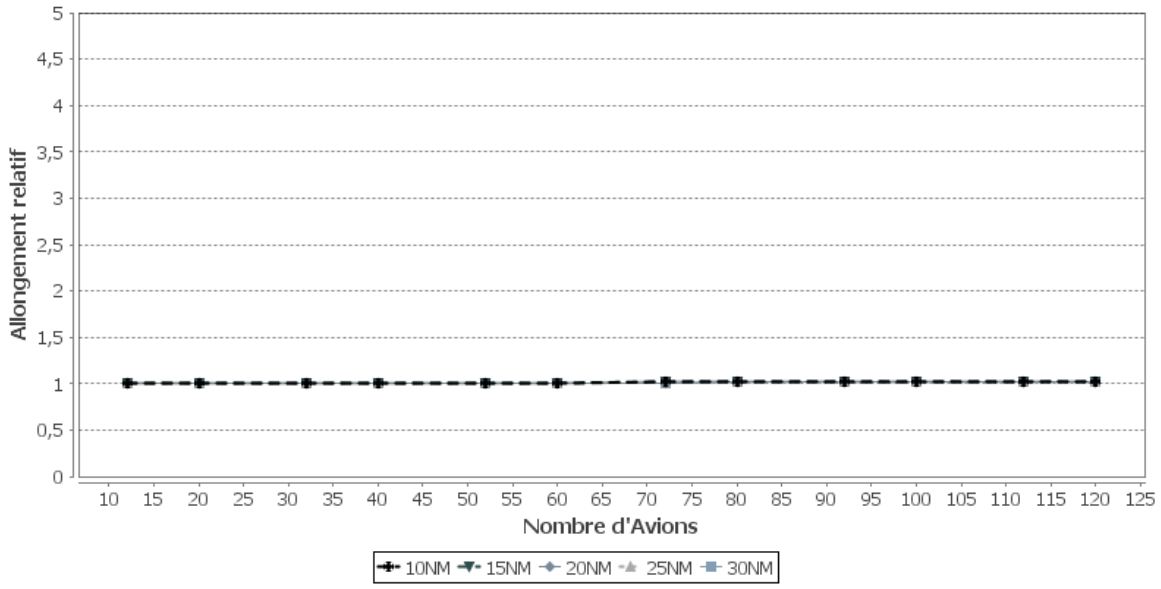


Figure 8.16 – Allongement du temps de trajectoire en fonction d'$AgNb$ pour différentes valeurs de $d_{coll,hor}$

arrivent à destination (figure 8.14). Nous supposons que ce résultat est dû à la résolution de la SNC 8 (section 7.10.3.5), en effet, lorsque $d_{coll,hor}$ augmente, la criticité de l'*Agent EM* décroît moins rapidement, imposant à l'*Agent EM* de s'éloigner plus de l'autre *Agent EM*.

8.2.4.2 Sensibilité au paramètre t_r de la fonction de criticité $Crit_{2,j,k}$

Nous faisons évoluer $t_r = \frac{d_{t_r}}{||\overrightarrow{v_i}||}$, utilisé dans la fonction de criticité $Crit_{2,j,k}$ (section 7.9.1), en faisant varier d_{t_r}. Pour cela, nous utilisons les valeurs :

— $d_{t_r} = 25NM(= 46,3km)$, soit $\frac{1}{4}d_{t_r}$,

— $d_{t_r} = 50NM(= 92,6km)$, soit $\frac{2}{4}d_{t_r}$,

— $d_{t_r} = 75NM(= 138,9km)$, soit $\frac{3}{4}d_{t_r}$,

— $d_{t_r} = d_{t_r} = 100NM(= 185,2km)$,

— $d_{t_r} = 125NM(= 231,5km)$, soit $\frac{5}{4}d_{t_r}$.

Les figures 8.17, 8.18, 8.19 et 8.20 présentent les valeurs médianes des différentes métriques en fonction de $AgNb$ pour chaque valeur de d_{t_r}.

Discussion de la sensibilité du système aux modifications de d_{t_r}. Comme le montrent les 4 figures, t_r ne semble pas avoir d'influence sur les différentes métriques, ceci valide le rôle d'ordonnancement de la fonction de criticité $Crit_{1,j,k,hor}$, mais aussi la robustesse du système quant à ces variations, et la capacité du comportement agent à intégrer ces différentes valeurs. Ces résultats étayent notre hypothèse de la section précédente par rapport à la résolution de la SNC 8.

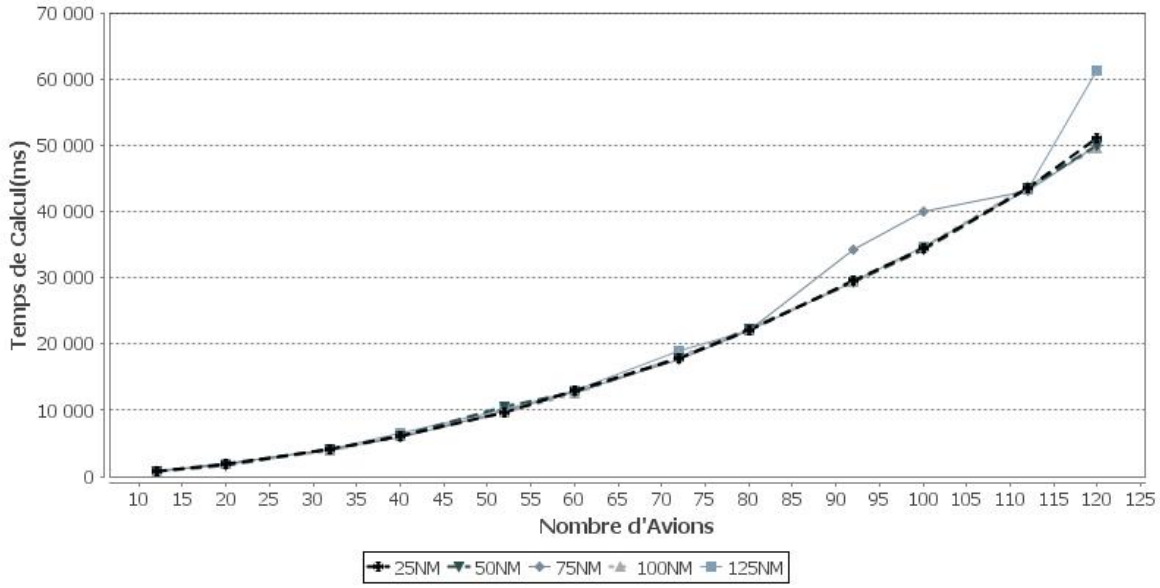


Figure 8.17 – Temps de calcul en fonction d'$AgNb$ pour différentes valeurs de d_{t_r}

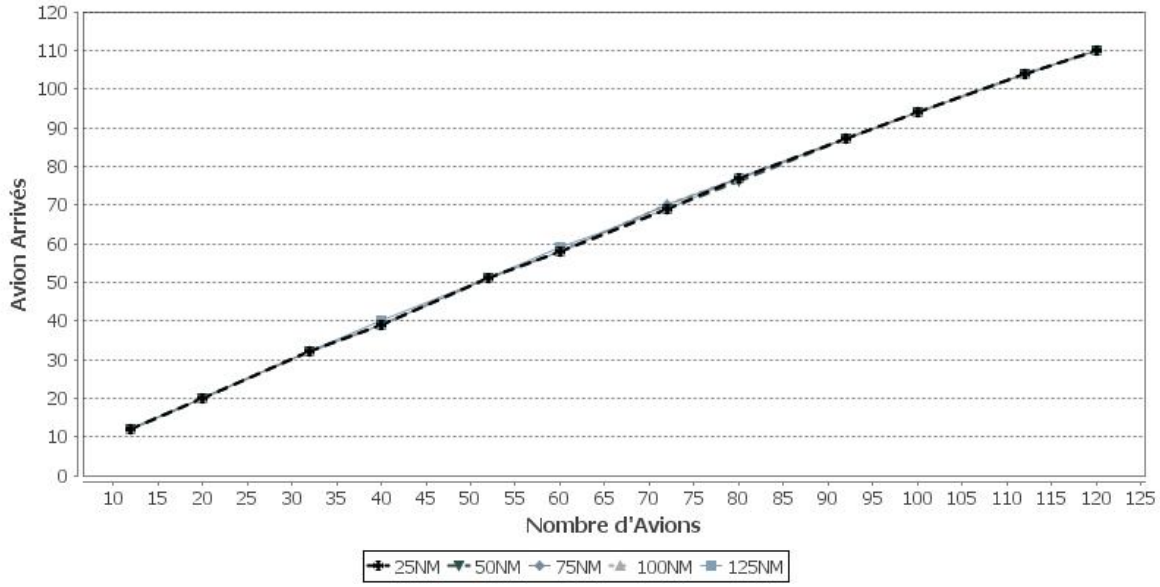


Figure 8.18 – Nombre d'avions arrivants en fonction d'$AgNb$ pour différentes valeurs de d_{t_r}

8.2.4.3 Sensibilité au rayon r_{Z_p} de la zone de perception

Nous faisons évoluer la distance de vision horizontale de l'*Agent EM*, r_{Z_p}, en prenant comme valeurs :

— $r_{Z_p} = 25NM$, soit $\frac{1}{4}$ de la valeur par défaut,
— $r_{Z_p} = 50NM$, soit $\frac{2}{4}$ de la valeur par défaut,
— $r_{Z_p} = 75NM$, soit $\frac{3}{4}$ de la valeur par défaut,
— $r_{Z_p} = 100NM$, la valeur de départ
— $r_{Z_p} = 125NM$, soit $\frac{5}{4}$ de la valeur par défaut.

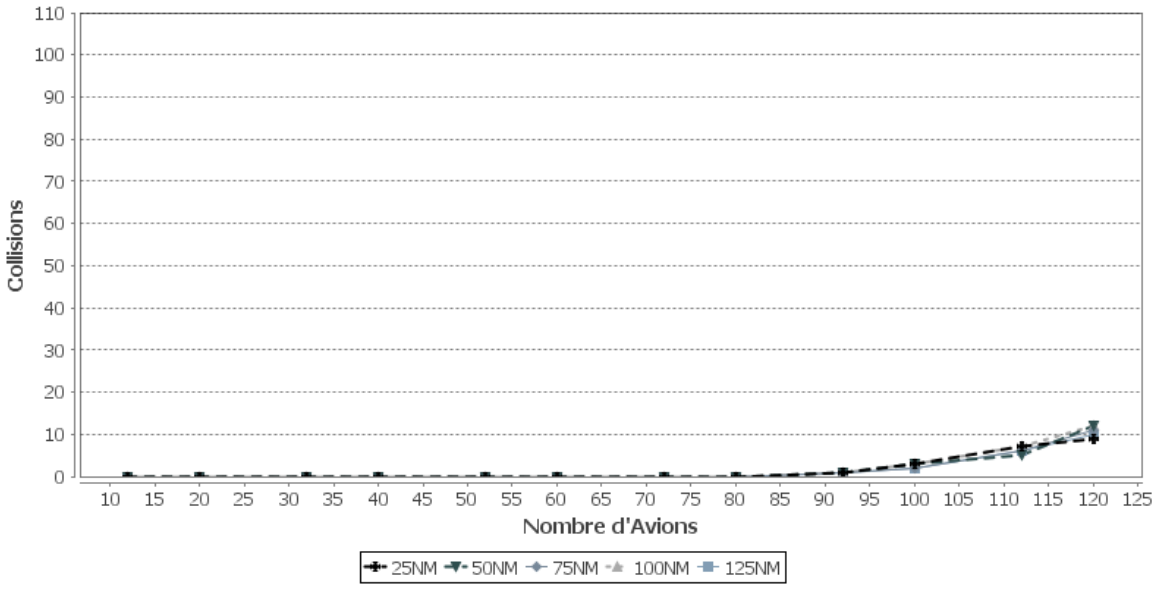


Figure 8.19 – Conflits aéronautiques restants en fonction d'$AgNb$ pour différentes valeurs de d_{t_r}

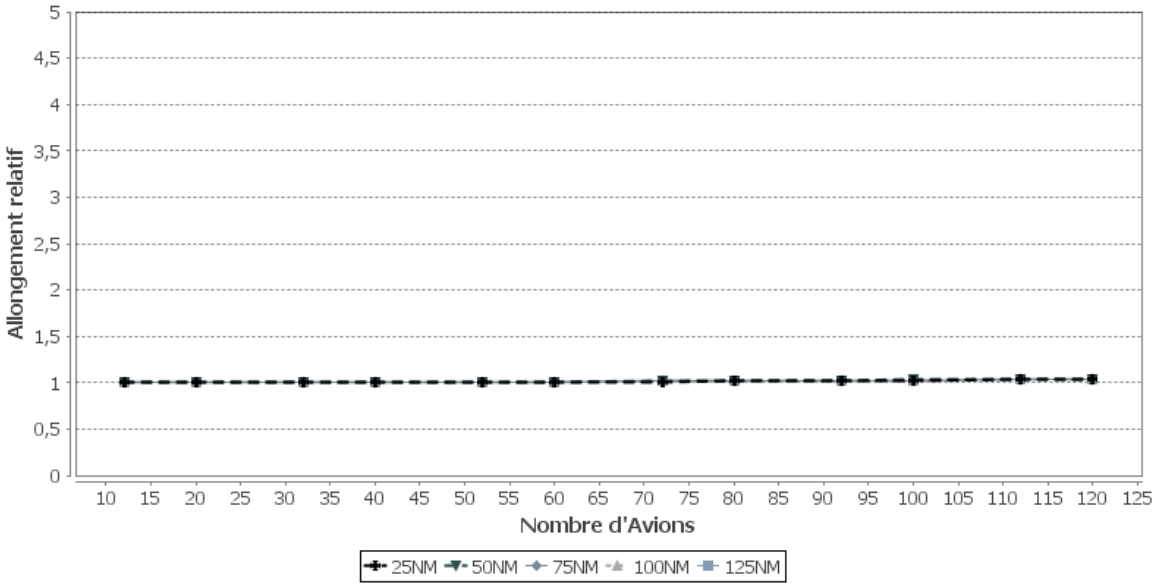


Figure 8.20 – Allongement du temps de trajectoire en fonction d'$AgNb$ pour différentes valeurs de d_{t_r}

Les figures 8.21, 8.22, 8.23 et 8.24 présentent les valeurs médianes de différentes métriques en fonction de $AgNb$ pour chaque valeur de r_{Z_p}.

Discussion de la sensibilité du système aux modifications de r_{Z_p}. Les 4 figures montrent que les variations de r_{Z_p} n'ont pas d'influence sur le système. Deux points sont à noter :

1. Le paramètre ne semble pas avoir d'influence sur le temps de calcul (figure 8.21), or, en diminuant r_{Z_p}, seule la taille de la zone de perception est influencée, et par consé-

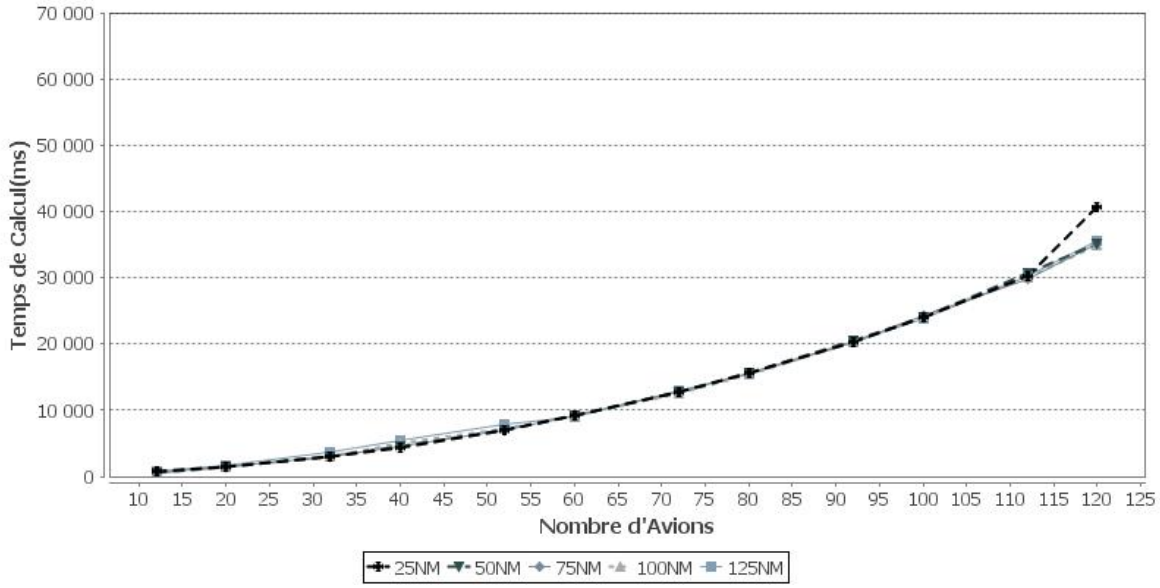


Figure 8.21 – Temps de calcul en fonction d'$AgNb$ pour différentes valeurs de r_{Z_p}

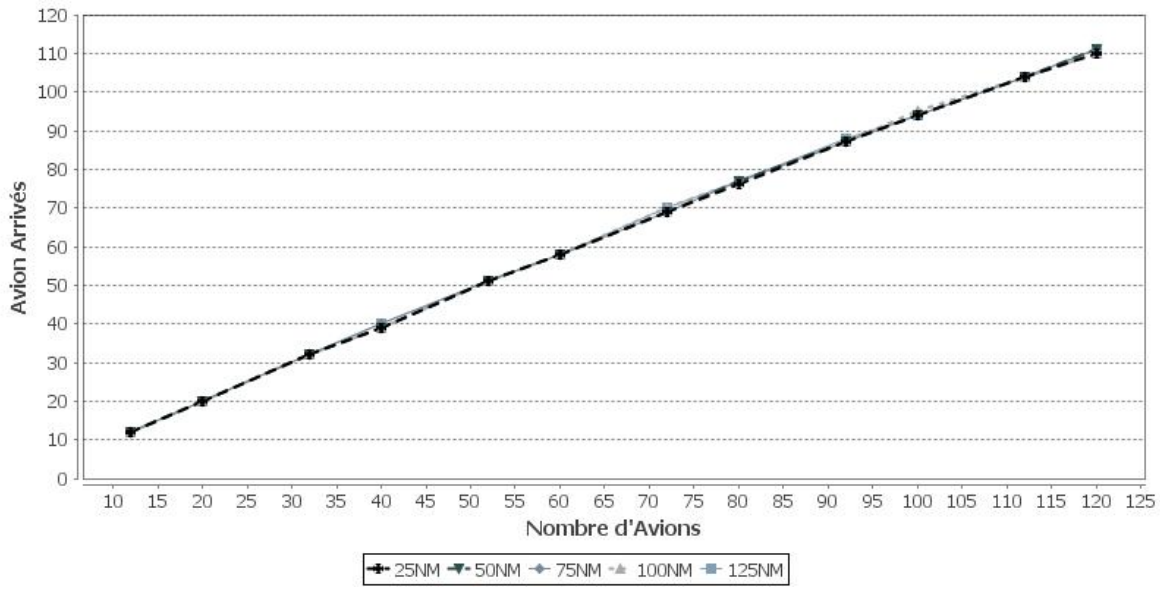


Figure 8.22 – Nombre d'avion arrivants en fonction d'$AgNb$ pour différentes valeurs de r_{Z_p}

quent l'*Agent EM* devrait percevoir moins d'*Agents EM*, ce qui devrait *a priori* réduire le nombre d'opérations de calcul, et par conséquent le temps de calcul. Puisque le temps de calcul évolue de manière non-linéaire lorsque $AgNb$ varie seul, nous pensons que la diminution de r_{Z_p} ne fait pas diminuer le nombre d'*Agents EM* perçus. En effet, en probabilité, la très grande majorité des *Agents EM* convergent vers le centre du carré en même temps, et il est possible que les valeurs de r_{Z_p} testées soient trop grandes. Sinon, la diminution du nombre d'*Agents EM* perçus est compensé par d'autre opérations, par exemple le choix d'une action devient plus long.

2. Le nombre de collisions n'augmente pas lorsque r_{Z_p} diminue (figure 8.23), alors que

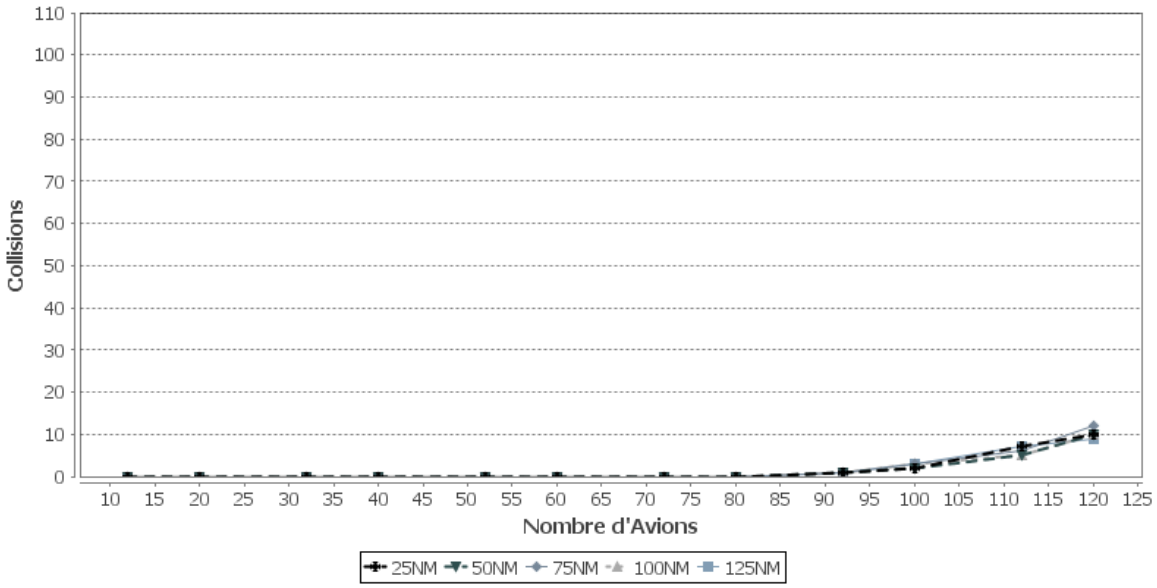


Figure 8.23 – Collisions restantes en fonction d'$AgNb$ pour différentes valeurs de r_{Z_p}

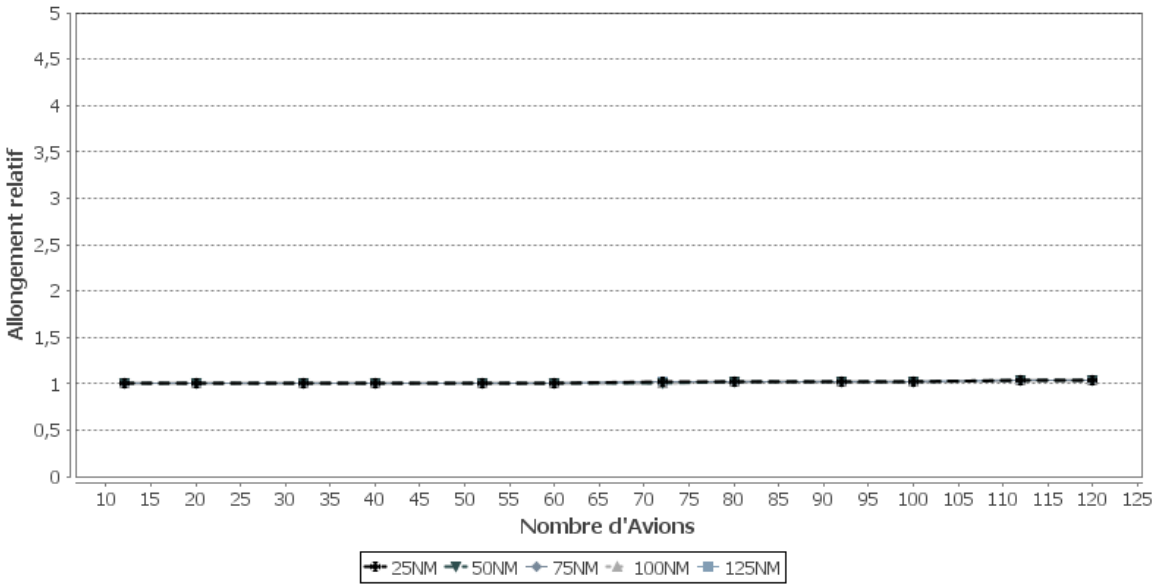


Figure 8.24 – Allongement du temps de trajectoire en fonction d'$AgNb$ pour différentes valeurs de r_{Z_p}

l'*Agent EM* a moins de temps pour modifier sa trajectoire et éviter les collisions.

Ce résultat montre qu'en fonction de son environnement, l'*Agent EM* pourrait écouter plus d'*Agents EM* autour de lui en augmentant r_{Z_p} (par exemple lorsque la densité est faible), ou se concentrer sur des *Agents EM* plus proche (par exemple lorsque la densité est forte) afin d'améliorer son temps de calcul, sans pour autant modifier son efficacité.

8.2.4.4 Sensibilité à la vitesse de rotation

Nous faisons évoluer le paramètre $\dot{\alpha}$, la vitesse de rotation de l'avion, en prenant comme valeurs :

— $\dot{\alpha} = 1\,\text{deg}\,.s^{-1}$,
— $\dot{\alpha} = 2\,\text{deg}\,.s^{-1}$,
— $\dot{\alpha} = 3\,\text{deg}\,.s^{-1}$,
— $\dot{\alpha} = 4\,\text{deg}\,.s^{-1}$,
— $\dot{\alpha} = 5\,\text{deg}\,.s^{-1}$.

Les figures 8.25, 8.26, 8.27 et 8.28 présentent les valeurs médianes de différentes métriques en fonction de $AgNb$ pour chaque valeur de $\dot{\alpha}$.

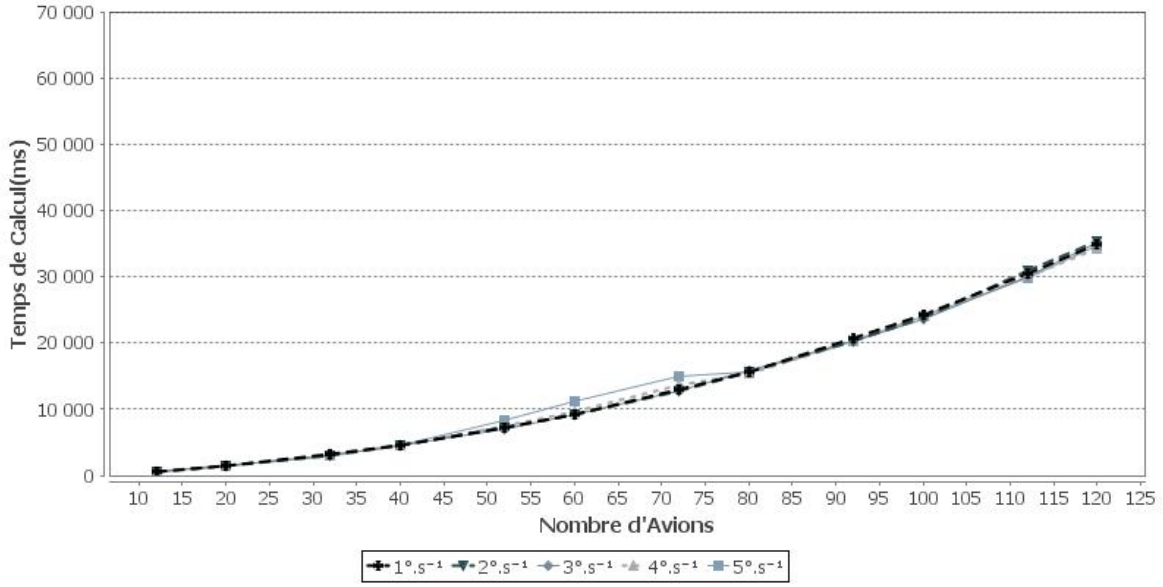


Figure 8.25 – Temps de calcul en fonction d'$AgNb$ pour différentes valeurs de $\dot{\alpha}$

Discussion de la sensibilité du système aux modifications de $\dot{\alpha}$. Comme nous pouvions nous y attendre, le paramètre $\dot{\alpha}$, qui influe sur les actions de l'*Agent EM*, influe sur certains des résultats du système *CAAMAS* :

— Le temps de calcul n'est pas influencé (figure 8.25).
— Le nombre de collisions restantes est légèrement influencé par $\dot{\alpha}$ lorsque le nombre d'$AgNb$ est supérieur à 70 (figure 8.27), le plus fort gain étant pour 112 avions (226% entre $1\text{deg}\,.s^{-1}$ et $5\text{deg}\,.s^{-1}$). Cette amélioration se maintient généralement (avec une augmentation moins forte), sauf pour 120 où la valeur médiane pour $1\text{deg}\,.s^{-1}$ et $5\text{deg}\,.s^{-1}$ sont les mêmes.
— Le nombre d'avions arrivant à destination sans sortir excessivement de la zone (figure 8.26) est très légèrement influencé par $\dot{\alpha}$ pour $AgNb > 30$, et légèrement influencé pour $AgNb > 80$ (environ 5% entre $\dot{\alpha}$). Ce résultat est certainement dû à la capacité de l'*Agent EM* à modifier plus vite sa trajectoire une fois les collisions passées, et donc sa capacité à rester dans la zone carrée.

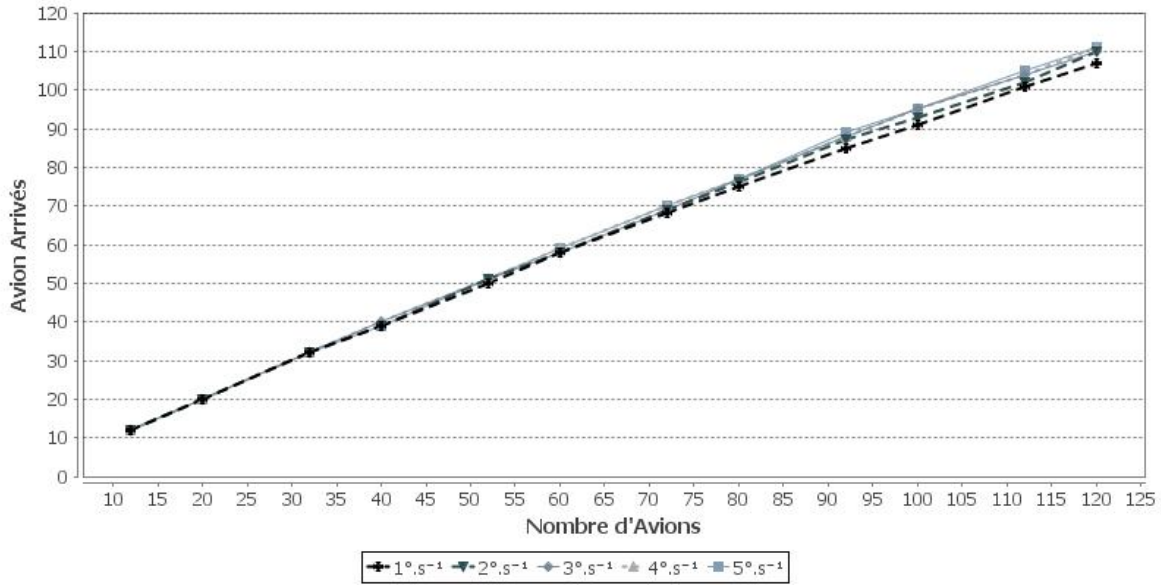


Figure 8.26 – Nombre d'avions arrivants en fonction d'$AgNb$ pour différentes valeurs de $\dot{\alpha}$

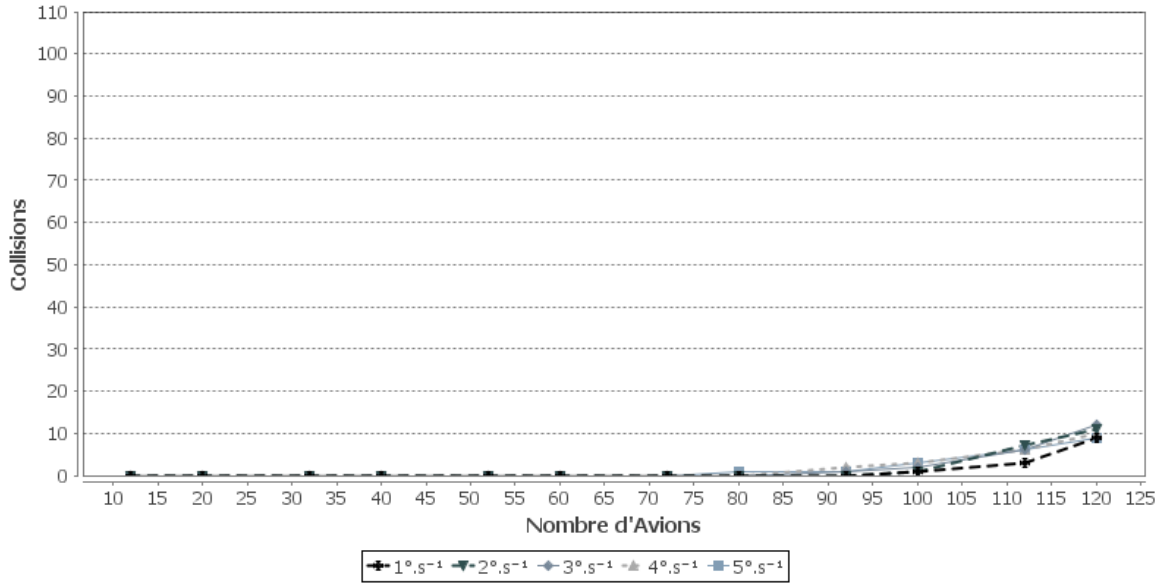


Figure 8.27 – Collisions restantes en fonction d'$AgNb$ pour différentes valeurs de $\dot{\alpha}$

8.2.4.5 Sensibilité au temps entre chaque cycle de vie d'agent Δt

Nous faisons évoluer le paramètre Δt entre chaque cycle de vie d'agent, en prenant comme valeurs :

— $\Delta t = 1s$,

— $\Delta t = 2s$,

— $\Delta t = 3s$,

— $\Delta t = 4s$,

— $\Delta t = 5s$,

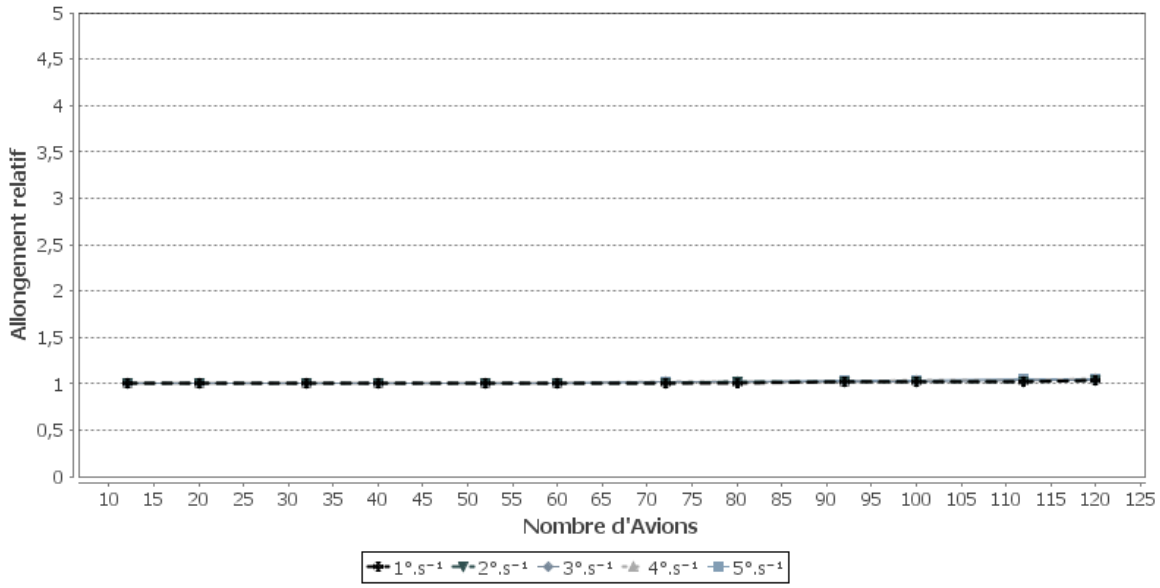


Figure 8.28 – Allongement du temps de trajectoire en fonction d'$AgNb$ pour différentes valeurs de $\dot{\alpha}$

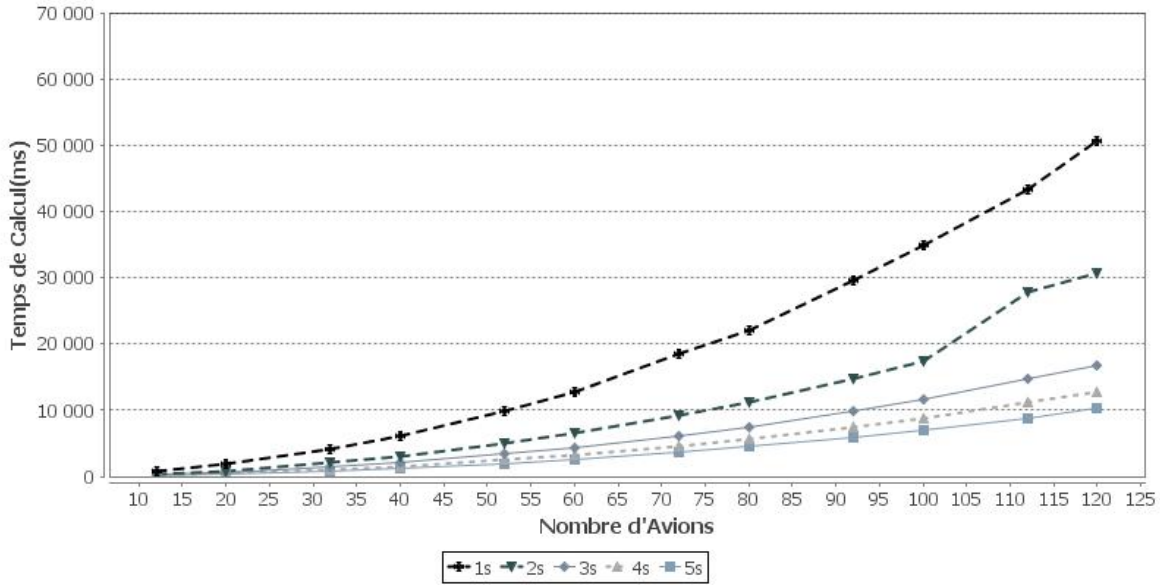


Figure 8.29 – Temps de calcul en fonction d'$AgNb$ pour différentes valeurs de Δt

Les figures 8.29, 8.30, 8.31 et 8.32 présentent les valeurs médianes de différentes métriques en fonction de $AgNb$ pour chaque valeur de Δt.

Discussion de la sensibilité du système aux modifications de Δt. De manière assez évidente, le système est sensible au modification de Δt :

1. Le temps de calcul est proportionnel à Δt, et décroît lorsque Δt augmente (figure 8.29). Par conséquent, le temps de calcul est directement proportionnel au nombre de cycles.
2. Le nombre de collisions est sensiblement le même pour les différentes valeurs de Δt,

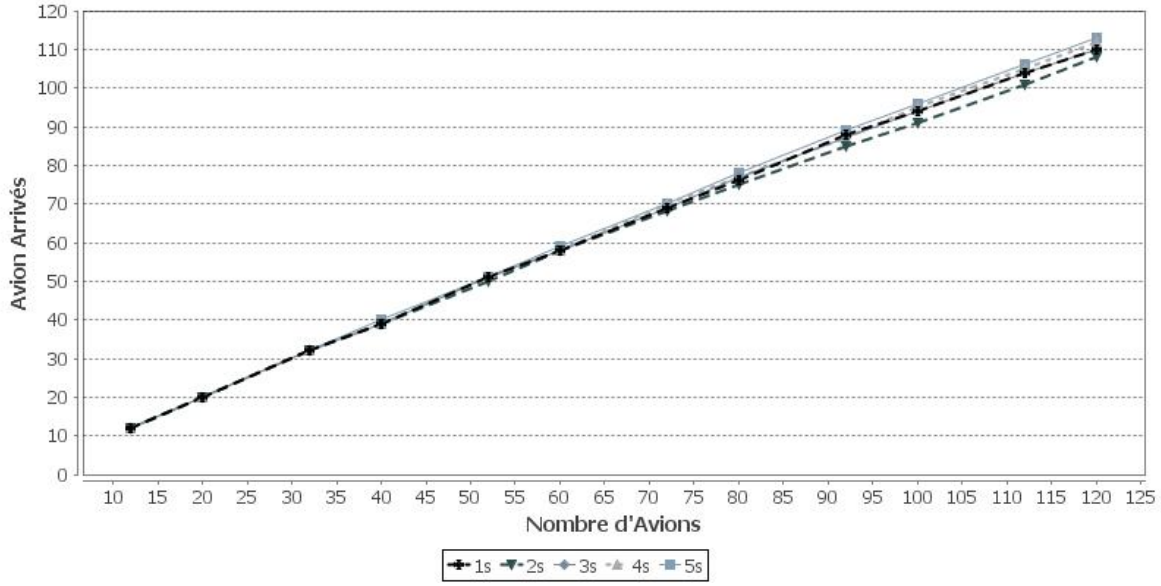


Figure 8.30 – Nombre d'avions arrivants en fonction d'$AgNb$ pour différentes valeurs de Δt

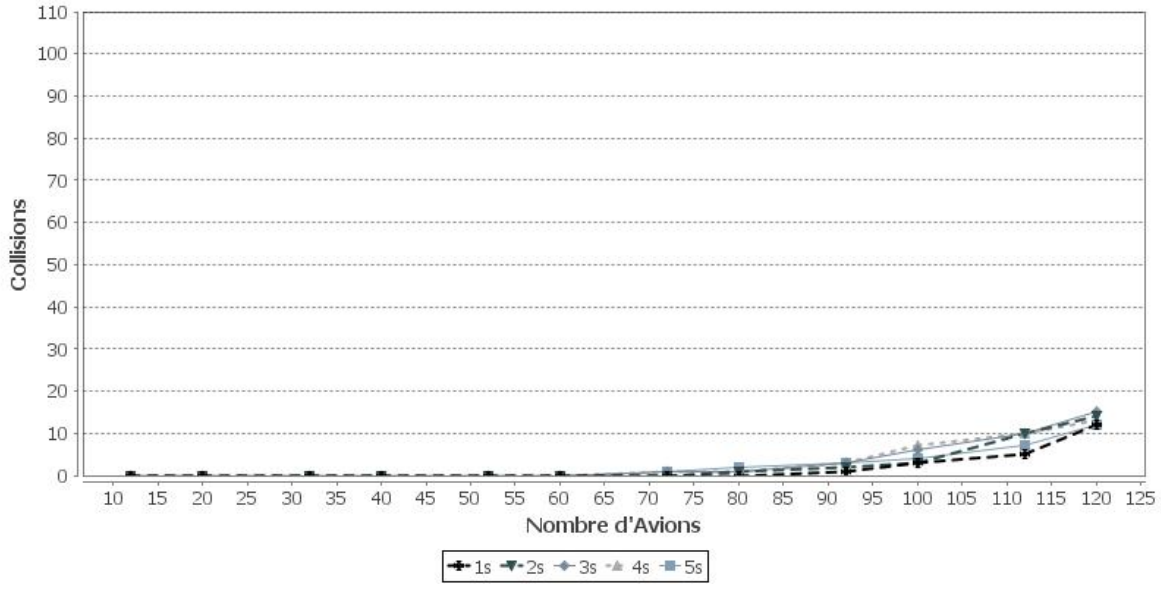


Figure 8.31 – Collisions restantes en fonction d'$AgNb$ pour différentes valeurs de Δt

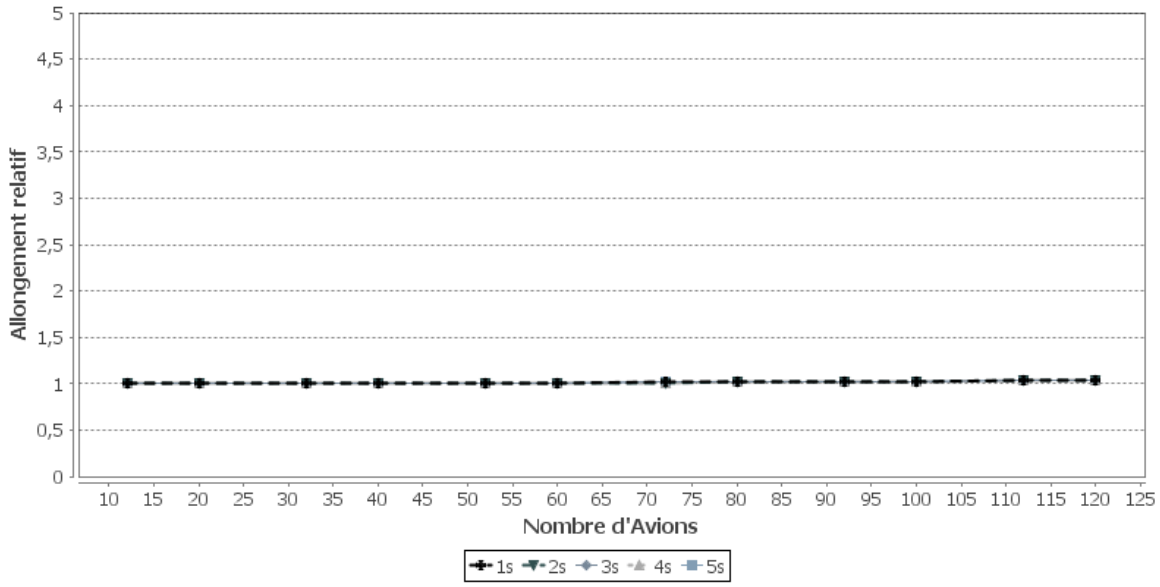


Figure 8.32 – Allongement du temps de trajectoire en fonction d'$AgNb$ pour différentes valeurs de Δt

et il n'y a pas de tendance fixe, sauf pour $\Delta t = 3s$ et $\Delta t = 4s$ qui semblent avoir plus régulièrement de meilleurs résultats (figure 8.31). Il est possible que ces valeurs de Δt permettent d'anticiper un peu plus que les autres valeurs, sans pour autant provoquer des changements de trajectoire brusques. Deux biais sont à noter ; 1) Les virages sont effectués automatiquement, donc plus Δt est important, moins le virage est réaliste. 2) Puisque le simulateur fait agir un avion après l'autre, ce qui introduit un biais, plus Δt est important plus le biais est fort. Dans sa thèse, [Dougui, 2011] présente ce biais à l'extrême lorsqu'un avion planifie toute sa trajectoire avant les autres.

3. Le nombre d'avions arrivant sans sortir excessivement (figure 8.30) est très légèrement plus important lorsque Δt augmente (environ 2% entre $1s$ et $5s$). Il est possible que l'agent a moins de cycles de vie pour décider de modification et arriver à destination

Le fait que le temps de calcul soit directement lié au nombre de cycles de vie montre qu'il est possible de diminuer le temps des calcul en distribuant ceux-ci, et nous supposons aussi en distribuant le calcul des différents modules (notamment les *Modules Actions*). L'*Agent EM* pourrait aussi adapter son temps Δt en fonction de son environnement, par exemple en augmentant Δt lorsqu'il y a peu d'*Agents EM* autour de lui.

8.2.4.6 Conclusion sur l'étude de sensibilité

Cette étude de sensibilité nous a permis de montrer l'adéquation du système *CAAMAS* pour donner des comportements adaptatifs aux entités mobiles.

Cette étude a montré la robustesse du système quant aux variations de $d_{coll,hor}$ et t_r, ainsi que l'adaptabilité du comportement des *Agents EM* et valide le rôle d'ordonnancement de $Crit_{1,j,k}$ et $Crit_{2,j,k}$.

Elle a permis de montrer le potentiel de la distribution du calcul des cycles de vie des

agents et de leurs *Modules Actions*. Les expérimentations ont validé la modification adaptative du temps entre chaque cycle de vie Δt, de la taille de la zone de perception, mais aussi des actions.

8.2.5 Comparaison

Nous comparons dans cette section l'implémentation de CAAMAS pour le trafic aérien avec les résultats de CSORCA [Durand, 2018] (présenté dans la section 4.3) sur le benchmark aléatoire. La longueur de côté du carré est fixée à $l = 500NM$, et $d_e = 3d_{coll}$.

Nous effectuons deux comparaisons avec CSORCA [Durand, 2018], la première à vitesse constante (table 1 de [Durand, 2018]), et la seconde à vitesse variable (table 5 de [Durand, 2018]). Dans ces comparaisons, le nombre d'avions $AgNb$ varie dans l'ensemble $\{10, 20, 30, 40, 50, 60, 70, 80, 90\}$.

Pour chaque $AgNb$, 100 tests aléatoires (avec des configurations différentes) sont réalisés, avec des positions générées aléatoirement en respectant la contrainte d'espacement pour chaque test. Si le test contient une perte de séparation, il est considéré comme ayant échoué. Pour tous les tests qui n'ont pas échoué, des statistiques sont calculées sur l'allongement relatif du temps de la trajectoire.

Pour la **comparaison à vitesse constante**, nous utilisons donc les paramètres suivants (qui sont ceux de la table 1 de [Durand, 2018]) :

— La vitesse de rotation maximale est de $3°.s^{-1}$,

— La vitesse est constante, fixée à $v_{nom} = 450NM.h^{-1}$,

— L'accélération est nulle.

Pour la **comparaison avec une vitesse variable**, les paramètres sont (*cf.* table 5 de [Durand, 2018]) :

— La vitesse peut varier dans un intervalle $[v_{nom} - 5\%, v_{nom} + 5\%]$, avec $v_{nom} = 450NM.h^{-1}$,

— L'accélération, qui est fixée à $1.11NM.h^{-2}$,

— La vitesse de rotation maximale, qui est de $3°.s^{-1}$

Les tableaux 8.2 et 8.3 contiennent les résultats de la comparaison à vitesse constante et à vitesse variable. Nous comparons les deux systèmes sur l'allongement relatif de la trajectoire (voir section 8.2.1), et sur le nombre de tests qui ont échoué à cause de collisions (c'est-à-dire le nombre de tests où il existe au moins une collision ou une perte de séparation).

Avec $l_{i,wanted}$ étant la longueur de la trajectoire prévue τ_i, et $l_{i,final}$ la longueur de la trajectoire effectuée par l'*Agent EM*.

Échecs en raison de perte de séparation. A vitesse constante, *CAAMAS* propose des résultats équivalents à ceux de CSORCA pour les tests avec $AgNb < 70$, avec des résultats bien meilleurs pour les cas $AgNb \in \{30, 40\}$ en terme de planification sans collisions. En revanche, nous observons un palier important pour $AgNb \in \{70, 80, 90\}$ et des résultats moins bon. Nous retrouvons aussi des bonds brusques dans les résultats de CSORCA entre $AgNb = 20$ et $AgNb = 30$ et entre $AgNb = 80$ et $AgNb = 90$.

	CAAMAS			CSORCA [Durand, 2018]		
Nb d'Avions($AgNb$)	Nb de tests en échecs dû à des collisions	Allongement		Nb de tests en échecs dû à des collisions	Allongement	
		Moyen	Max		Moyen	Max
10	0	1.01311	2.34988	3	1.00172	1.00763
20	1	1.02474	2.18661	5	1.00477	1.01088
30	4	1.03558	2.01421	23	1.01159	1.02739
40	16	1.04872	2.22742	26	1.01979	1.03626
50	34	1.06590	2.66802	39	1.02708	1.04028
60	52	1.05945	1.69313	55	1.04270	1.06213
70	82	1.07694	2.78372	53	1.05514	1.08631
80	93	1.08510	2.92088	57	1.06571	1.10463
90	99	1.09948	2.89704	79	1.07024	1.07024

Table 8.2 – Comparaison avec la table 1 de [Durand, 2018]

	CAAMAS			CSORCA [Durand, 2018]		
Nb d'Avions($AgNb$)	Nb de tests en échecs dû à des collisions	Allongement		Nb de tests en échecs dû à des collisions	Allongement	
		Moyen	Max		Moyen	Max
10	0	1.01576	2.02537	0	1.00061	1.00331
20	0	1.02840	2.08766	0	1.00168	1.00730
30	0	1.03911	2.06485	0	1.00382	1.00969
40	1	1.04374	2.46300	1	1.00664	1.01412
50	4	1.05093	1.90796	2	1.01179	1.02745
60	4	1.05345	2.09812	12	1.01634	1.03171
70	30	1.07244	2.08912	8	1.01634	1.03884
80	41	1.06389	2.60254	15	1.03050	1.05314
90	60	1.07065	2.68834	25	1.03485	1.04844

Table 8.3 – Comparaison avec la table 5 de [Durand, 2018]

Nous retrouvons des résultats équivalents à vitesse variable, puisque les résultats de CAAMAS sont comparables à ceux de CSORCA pour $AgNb < 70$ voire même meilleurs pour $AgNb = 60$. En revanche, on observe une brusque augmentation du nombre d'échecs à partir de $AgNb = 70$ et des résultat moins bons pour les nombres supérieurs.

Les deux comparaisons montrent des paliers dans le nombre d'échecs, en particulier entre 50 et 60 avions pour *CAAMAS* et entre 80 et 90 pour CSORCA à vitesse variable. La persistance de ces paliers dans les deux tableaux suggère que le niveau de vol commence à être trop saturé pour l'une puis pour l'autre des méthodes. Le fait que ce palier intervienne avant pour *CAAMAS* pourrait venir de la discrétisation des actions : l'espace étant plus saturé, il requiert d'utiliser plus finement les possibilités de l'entité mobile en terme de déplacement, et donc de faire évoluer le nombre d'actions possibles. Globalement, les résultats des deux systèmes sont meilleurs lorsque la vitesse peut varier, néanmoins, pour *CAAMAS* les résultats sont quasiment identiques pour $AgNb < 40$, ce qui suggère que donner à l'*Agent EM* la possibilité de créer de nouvelles actions, ou de modifier légèrement celles qu'il utilise déjà, pourrait améliorer les résultats (par exemple en temps de calcul). A noter que dans cette étude, une perte de séparation est considérée comme conflit, suite à quoi le test est interrompu. Une perspective intéressante est d'étudier l'importance des pertes de séparation détectées. En effet, la zone de séparation est définie pour prendre en compte de nombreux

paramètres (voir section 1.3.1), notamment le temps de réaction d'acteurs humains. Dans le cadre d'une automatisation de la gestion de conflit, il est peut-être intéressant de réduire la taille de cette zone de séparation afin d'augmenter la densité.

Allongement de la trajectoire. Les allongements des trajectoires induits par *CAAMAS* sont globalement moins bon pour la moyenne, et toujours largement moins bon pour le maximum. La non constance dans les maximums suggère que ce sont des cas très isolés, mais dont la valeur tire la valeur moyenne vers le haut. A noter que l'allongement du temps de trajectoire ne prend pas correctement en compte les modifications de la vitesse qui induisent des modifications de trajectoire qui peuvent passer inaperçues, et d'autres métriques devraient être introduites.

Conclusion. Nos expérimentations montrent que les deux systèmes sont capables dans la grande majorité des cas d'éviter des collisions entre les différents avions, avec plus d'avions que dans la réalité (en particulier le tableau 8.3), et dans une configuration très différente et plus compliquée, dans laquelle, en probabilité, les avions convergent tous en même temps vers le centre du carré.

En l'absence de tests sur le non-déterminisme des *Agents EM* et la sensibilité du système à la façon avec laquelle décident les agents, il est difficile d'affirmer que nous pourrions obtenir de meilleurs résultats en rejouant un test (même positions de départ et d'arrivée). Néanmoins, il est évident que l'ordre dans lequel les agents agissent peut influencer les trajectoires générées. Nous observons cet effet dans les systèmes qui génèrent entièrement une trajectoire après l'autre (par exemple [Dougui, 2011] et [Girardet, 2014] présenté dans la section 4.3), et nous suspectons encore cet effet dans un système comme *CAAMAS* dans les résultats du benchmark du rond-point (section 8.2.2). Dans le cas ou le système se révélerait non déterministe, puisque les trajectoires sont générées en quelques secondes pour des vols qui prennent des heures, il serait possible de générer des trajectoires plusieurs fois dans le but d'obtenir de meilleurs résultats.

Enfin, nous observons dans cette comparaison que le système *CAAMAS* peut gérer un trafic et éviter les collisions dans l'espace aérien actuel, et globalement aussi bien que *CSORCA* [Durand, 2018] pour des cas avec moins de 70 avions, voire mieux dans la plupart des cas. En revanche, *CAAMAS* donne de moins bons résultats pour un nombre d'avions plus élevé. Nous pensons que la discrétisation des actions commence à pénaliser les résultats de *CAAMAS* dans cet environnement très dense en avion, et que donner de nouveaux comportements d'adaptation des *Agents EM*, comme la capacité de modifier ses actions, ou d'en créer de nouvelles, permettrait d'améliorer ces résultats.

8.3 Conclusion sur le système CAAMAS

Nous avons présenté dans ce chapitre le système multi-agent *Collision Avoidance Adaptive Multi-Agent System (CAAMAS)*, composé des *Agents EM* du système multi-agent *AGATS*.

Nous avons présenté dans un premier temps l'adaptation du modèle pour le trafic aérien

Critères		*CAAMAS*
Optimalité	Trajectoire	Non
	Trafic	Non prouvé
Modification de trajectoire		Modification multiple (D)
Moment de la planification		Dynamique
Incertitude		Par Dynamique / Position, Vitesse, Orientation
Retour à la trajectoire		Oui
Passage à l'échelle		Fort
Généricité		Forte

Table 8.4 – Le système multi-agent adaptatif *CAAMAS* positionné selon les critère de l'état de l'art sur l'évitement de collisions (chapitre 4, en particulier la table 4.1)

(section 8.1), qui concerne majoritairement la définition non homogène d'une collision dans le trafic aérien, et la définition des actions des avions.

Ce système a été ensuite utilisé dans 4 expérimentations différentes :

— le benchmark du rond-point,

— le benchmark aléatoire,

— une étude de sensibilité sur le benchmark aléatoire,

— une comparaion avec CSORCA [Durand, 2018] sur le benchmark aléatoire.

Ces expérimentations permettent de situer le système multi-agent *CAAMAS* par rapport à l'état de l'art sur la planification de trajectoires et l'évitement de collisions. De manière générale, *CAAMAS* se place à la frontière entre les méthodes qui cherchent à éviter les collisions et les méthodes qui cherchent l'optimalité, avec une forte composante dynamique (Table 8.4).

Le calcul, la diffusion, et l'utilisation de différentes mesures de criticités dans le processus de décision permet la relaxation locale de la criticité des *Agents EM*, et permet de trouver des solutions quasi-optimales pour le trafic (section 8.2.2), bien que non prouvée.

Par l'utilisation d'actions discrètes, les *Agents EM* de *CAAMAS* construisent une trajectoire avec des modifications multiples discrètes, et une planification dynamique. En plus de prendre en compte l'incertitude par cette dynamique, grâce à ses comportements d'adaptation (en particulier la résolution de ses situations de non coopération, voir section 7.10.3), le système *CAAMAS* est capable de prendre en compte l'incertitude de position, de vitesse et d'orientation, qui sont peu prises en compte dans les différents systèmes dynamiques (voir table 4.1).

De manière générale, les différentes expérimentations montrent la capacité d'adaptation des *Agents EM* pour leur permettre de s'ajuster à des modifications de l'environnement, en particulier l'évitement entre entités mobiles, et de retrouver leur trajectoire une fois ces perturbations prises en compte. La décentralisation de la prise de décision au niveau des entités mobiles, et l'utilisation d'actions spécifiques aux entités mobiles rend le système fortement générique et facilite son adaptation pour des trafics différents. De plus, les comportements des *Agents EM* ont montré leur robustesse quant aux variations de leurs paramètres com-

portementaux, ainsi que leur adaptabilité à ceux-ci et aux variations de l'environnement (section 8.2.4).

Enfin, les comportements adaptatifs des *Agents EM* leur permettent d'éviter les collisions entre entités mobiles dans un contexte proche du réel, et dans des scénarios plus compliqués (section 8.2.5 et section 8.2.3). Les expérimentations (en particulier section 8.2.3) montrent un fort passage à l'échelle du système multi-agent *CAAMAS*, avec une augmentation du temps de calcul presque linéraire avec le nombre d'avions (figure 8.10) jusqu'à 80 avions sur un même niveau de vol.

Ces expérimentations ont aussi permis de montrer le potentiel de certaines adaptations et de nouveaux comportements des *Agents EM* de *CAAMAS*, notamment sur la distribution des calculs, mais aussi sur la modification de la taille de la zone de perception, sur l'intervalle de temps entre chaque cycle de vie, ou encore sur la modification des actions par les *Agents EM*.

Nous présentons dans le chapitre suivant la deuxième partie de ces expérimentations, qui concernent les agents d'*AGATS* dédiés à la génération de scénarios et à leur adaptation.

9 AGEAS : Implémentation, expérimentations et analyse

Nous avons expérimenté les comportements adaptatifs des entités mobiles dans le chapitre précédent, qui agissent exclusivement pendant la phase de simulation. Dans ce chapitre, nous expérimentons les comportements des autres agents composant le système *AGATS* lors de la phase de création. Dans cette phase, ces agents sont dédiés à la génération de scénarios, et nous nommons le sous-système qu'ils constituent *AGEAS*.

Le reste de ce chapitre présente la conception des différents agents agissant durant la phase de création, leurs éventuelles adaptations aux spécificités de l'aviation (section 9.1), et enfin les expérimentations qui ont été effectuées sur ce système (section 9.2).

9.1 Implémentation d'AGEAS

Nous présentons dans cette section l'implémentation d'*AGEAS* pour la génération de scénario de trafic aérien pour les simulations d'entraînement des contrôleurs, en présentant dans un premier temps les différentes caractéristiques de ces simulations, puis l'implémentation d'*AGEAS* pour ce contexte.

9.1.1 Simulation de contrôle de trafic aérien

Différentes caractéristiques des simulations ont été définies grâce à des discussions avec des experts de génération de scénarios de trafic aérien. Les situations se placent selon les trois niveaux du trafic (micro, méso, macro) et l'environnement.

Au niveau **microscopique**, nous retrouvons les caractéristiques qui définissent un avion et son comportement :

— la compagnie, par exemple Air France,

— le type de vol, par exemple un vol low-cost, militaire ou encore charter,

— le type d'avion, par exemple un A320,

— la trajectoire prévue de l'avion,

— le respect du plan de vol,

— le respect des procédures,

— la demande de modification de "directe", c'est-à-dire une modification du plan de vol,
— la mise en situation d'urgence (panne moteur, réserves basses, situation sanitaire ou tout autre situation).

Au niveau **mésoscopique**, nous retrouvons les caractéristiques suivantes :

— des collisions, ou des conflits,
— des flux : saisonniers, descendants, montants,
— des volumes horaire d'entrée/sortie.

Enfin, au niveau **macroscopique** nous retrouvons les caractéristiques suivantes :

— la densité de trafic,
— la complexité du trafic,
— un ensemble de flux, par exemple le trafic l'été qui est caractérisé par plusieurs flux d'été (par exemple en Europe, ce sont majoritairement des flux allant vers le sud).

L'environnement quant à lui se caractérise par :

— le réseau de routes,
— l'espace aérien simulé de manière générale, par exemple le secteur,
— les bornes temporelles pendant lesquelles la simulation se déroule,
— la météorologie, par exemple la présence de cellules orageuses, ou de turbulences,
— les zones militaires, qui peuvent s'activer ou se désactiver,
— les aéroports, qui peuvent se fermer, dont des pistes peuvent devenir impraticables, ou dont le sens d'utilisation des pistes peut varier.

Dans la suite, nous nous intéressons en particulier aux *situations de collision* du niveau mésoscopique. Nous définissons une *situation de collision* par les caractéristiques suivantes :

1. le **nombre d'entités mobiles** impliquées dans la collision (N_{Sit}), par exemple 2 avions,
2. la **géométrie** : la collision requise doit se faire selon une certaine géométrie, par exemple un avion vient du Nord-Ouest et un autre vient du Sud-Ouest,
3. la **position** : la collision requise doit arriver à un certain endroit, déterminée par sa longitude et latitude, par exemple (44.516698, 1.722166),
4. l'*altitude* : la collision requise doit arriver à une certaine altitude, par exemple $30000ft$,
5. le **temps** : la collision requise doit arriver à un certain moment, par exemple à midi,
6. la **proximité** : les 2 entités mobiles doivent se rencontrer.

Les sections suivantes présentent l'implémentation d'*AGEAS* pour la génération de scénarios d'entraînement de contrôleurs aériens.

9.1.2 Autonomous Generation of trAfic Scenario

Les agents qui composent le système *AGEAS* sont les agents d'*AGATS* qui agissent durant la phase de création du scénario. Ces agents sont :

— l'*Agent Situation*, représentant une situation requise par le scénariste. Dans cette implémentation, les situations sont uniquement des situations de collision,

— l'*Agent Trajectoire*, représentant une trajectoire,

— l'*Agent Partie*, représentant une partie de la trajectoire d'un *Agent Trajectoire*,

— l'*Agent Extrémité*, représentant un aéroport.

Ces agents utilisent deux entités passives :

— base de données de trajectoires, entité passive représentant la base de données des parties de trajectoires, qui est un ensemble d'agents contextes d'EVAA (voir section 6.4),

— *entité passive Caractéristique*, entité passive représentant une caractéristique d'une situation.

L'**implémentation de l'*Agent Situation*** ne diffère pas de celle dans *AGATS*, à la différence de ses caractéristiques, dont la criticité qui dépend de la nature de chaque caractéristique. Ces criticités sont toutes calculées selon la fonction générique 9.1. Le calcul de ces criticités se fait par évaluation de la partie de trajectoire de l'*Agent Trajectoire* utilisée par l'*Agent EM* lors de la génération de la situation.

$$C(x) = \begin{cases} \frac{1000}{d_{max}} d(v_1, v_2) & \text{if } d(v_1, v_2) < d_{max} \\ 1000 & \text{if } d(v_1, v_2) \geq d_{max} \end{cases} \tag{9.1}$$

Pour la situation de collision décrite précédemment, les caractéristiques 2 à 5 influencent chaque avion indépendamment pour qu'il ait le cap souhaité, qu'il passe par la position souhaitée avec l'altitude souhaitée, au moment souhaité. La caractéristique numéro 6, par contre, influence les différents avions pour qu'ils soient au même endroit au même moment.

Pour les caractéristiques 2 à 5, la distance d est définie comme suit :

2. $d_{geometry}$ est la distance en degrés entre le cap souhaité spécifié par l'*Agent Situation* et le cap réel de la partie de trajectoire d'EVAA. x_{max} est ici 180°. Dans la figure 9.1, cette distance est de 0 pour la partie de trajectoire $[AB]$ et γ_2 pour la partie de trajectoire $[CD]$,
3. $d_{position}$ est la distance en kilomètres entre la position souhaitée spécifiée par l'*Agent Situation* et la partie de trajectoire d'EVAA. x_{max} est ici $60km$, une distance utilisée pour déterminer le voisinage dans la simulation EVAA [Rantrua, 2017]. Dans la figure 9.1, cette distance est d_1 (respectivement d_2) pour la partie de trajectoire $[AB]$ (respectivement $[CD]$),
4. $d_{altitude}$ est la distance en pieds entre l'altitude souhaitée spécifiée par l'*Agent Situation* et l'intervalle d'altitude décrit par la partie de trajectoire d'EVAA (altitude au début et à la fin). x_{max} est ici $10000ft$,
5. d_{time} est la distance en secondes entre le temps souhaité spécifié par l'*Agent Situation* et l'intervalle de temps décrit par la partie de trajectoire d'EVAA (temps au début et à la fin). $x_{max} = 3600s$.

La caractéristique de proximité crée une rencontre dans le temps et l'espace entre les différentes trajectoires des *Agents Trajectoires* de l'*Agent Situation*. Pour ce faire, elle crée pour chaque *Agent Trajectoire* une rencontre dans le temps et l'espace entre la partie de trajectoire impliquée dans la génération de la situation avec la partie de trajectoire impliquée dans la

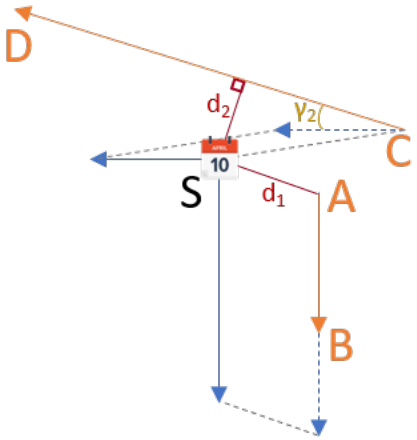


Figure 9.1 – Illustration des caractéristiques de géométrie et de position d'une situation de collision

génération de la situation du premier *Agent Trajectoire* créé par l'*Agent Situation*. Par simplification, cette dernière partie de trajectoire est numérotée 1, et nous numérotons j la partie de trajectoire du j^{eme} *Agent Trajectoire* créée par l'*Agent Situation* impliquée dans la génération de la situation. Cette caractéristique est donc composée de $N_{Sit} - 1$ criticités. Chacune de ces criticités est notée $Crit_{char,prox,j}$. Pour calculer $Crit_{char,prox,j}$, quatre étapes sont utilisées (voir figure 9.2 pour une représentation des notions utilisées ci-dessous) :

1. calculer les deux points de croisement (P et Q dans la figure) entre le cercle contenant la partie de trajectoire 1 (utilisée comme référence) et le cercle contenant la partie de trajectoire j,
2. prendre le point le plus proche de la première partie de trajectoire (Q dans la figure),
3. calculer le temps $t_{prox,j,1}$ auquel un avion j suivant la partie de trajectoire j atteindrait ce point s'il volait avec le même cap et la même vitesse que ceux décrits par la partie de trajectoire j et le temps $t_{prox,1,j}$ auquel l'avion 1 suivant la partie de trajectoire 1 atteindrait ce point s'il volait avec le même cap et la même vitesse que ceux décrits par la partie de trajectoire 1,
4. calculer $Crit_{char,prox,j}$ où d_{prox} est la fonction de distance en secondes entre $t_{prox,j,1}$ et $t_{prox,1,j}$ (pas de modulo). Ici, $x_{max} = 3600s$.

L'**implémentation de l'*Agent Trajectoire*** suit celle décrite dans *AGATS*.

L'**implémentation de l'Agent Partie** diffère légèrement de son implémentation dans *AGATS* : lors de l'initialisation, l'*Agent Partie* initialise l'ensemble de la trajectoire en prenant les parties de trajectoires qui sont associées à sa partie de trajectoire, et en créant autant d'*Agents Parties* que de parties de trajectoires. Le reste de son fonctionnement reste le même.

Dans cette première **implémentation, l'*Agent Extrémité*** est simplement une entité passive qui génère les *Agents EM* demandés par les *Agents Trajectoires*.

Dans cette implémentation, nous avons implémenté les SNC suivantes :

— SNC 10 : conflit d'Agents Partie,

— SNC 11 : inutilité d'un Agent Partie.

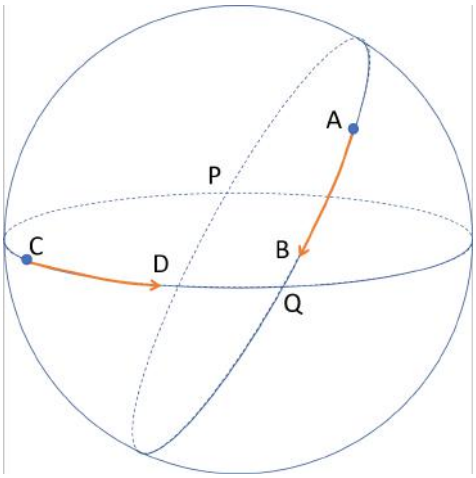


Figure 9.2 – Deux parties de trajectoires, $[AB]$ et $[CD]$ et leurs cercles. Les points géographiques P et Q sont les points de croisement de ces cercles utilisés pour déterminer la criticité de la proximité.

Les SNC de l'*Agent Situation* et de l'*Agent Trajectoire* n'ont pas été implémentées puisqu'elles sont détectées dans la phase de simulation, et les SNC de l'*Agent Extrémité* ne sont pas prises en compte.

9.2 Expérimentation d'AGEAS

Dans cette expérience, nous testons séparément la génération de trois *situations de collision* (AS_1, AS_2 et AS_3) représentées dans 9.3, et représentées séparément dans la suite dans les figures 9.4, 9.7 et 9.9 respectivement.

Nous utilisons pour cette expérimentation, une base de données de trajectoires composée de parties de trajectoires apprises à partir du trafic enregistré entre trois paires de villes françaises : (Paris, Toulouse), (Bordeaux, Paris) et (Bordeaux, Lyon). Cette base de données de trajectoires est représentée dans la figure 9.3 (traits rouges). Les données d'apprentissage utilisées contenant certaines données erronées ou incomplètes, certaines trajectoires apprises n'arrivent pas aux aéroports cités précédemment (c'est le cas notamment de la trajectoire apprise qui se dirige vers le Nord-Est de la France, mais aussi de trajectoires apprises qui s'arrêtent avant destination).

Dans ce qui suit, les *Agents Parties* peuvent uniquement modifier leur partie de trajectoire, afin de tester cette action qui est la plus complète. Les *Agents Situations* n'interagissant pas dans cette implémentation, nous générons chaque *situation de collision* séparément pour plus de clarté.

Génération de AS_1 La situation AS_1 représentée dans la figure 9.4 est définie par :

— deux avions A320,

— deux caps souhaités, 180° (du nord au sud) et 90° (de l'ouest à l'est),

— une position géographique, près de $(46,01°, 1,62°)$,

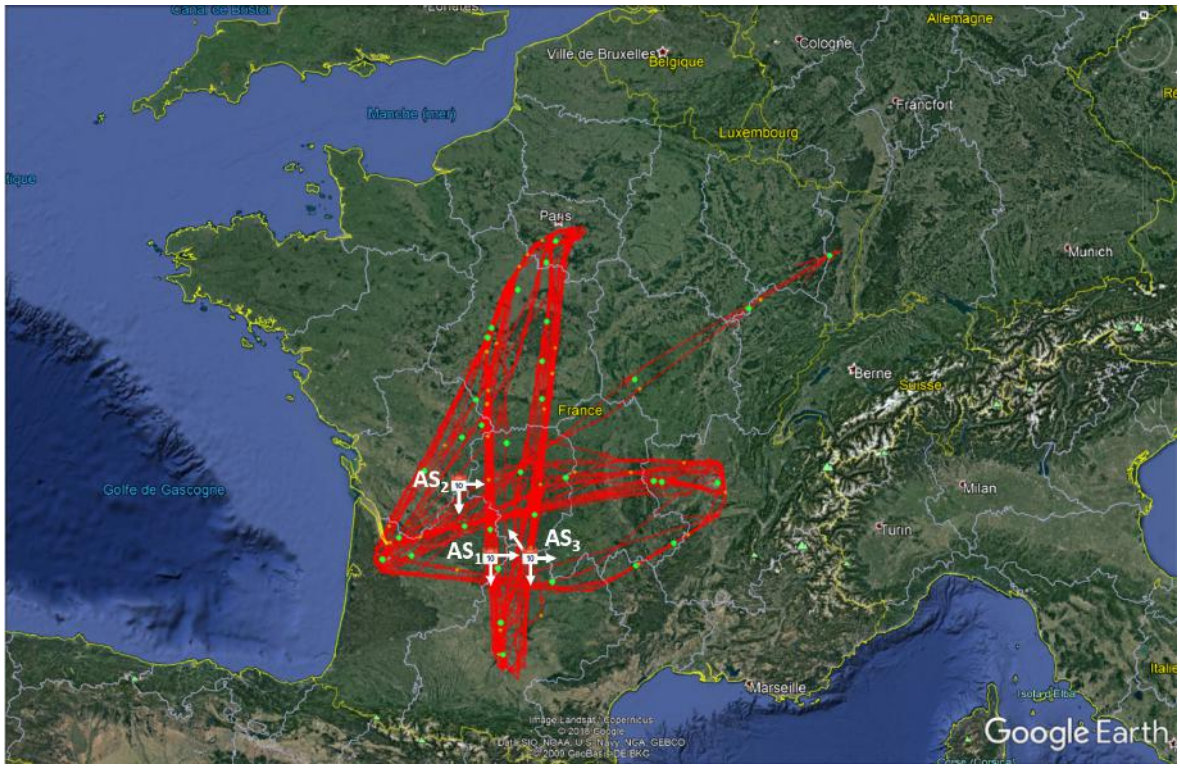


Figure 9.3 – Base de données de trajectoires utilisée pour la génération de 3 situations de collisions avec AGEAS (AS_1, AS_2, AS_3)

— une altitude de 30000ft.

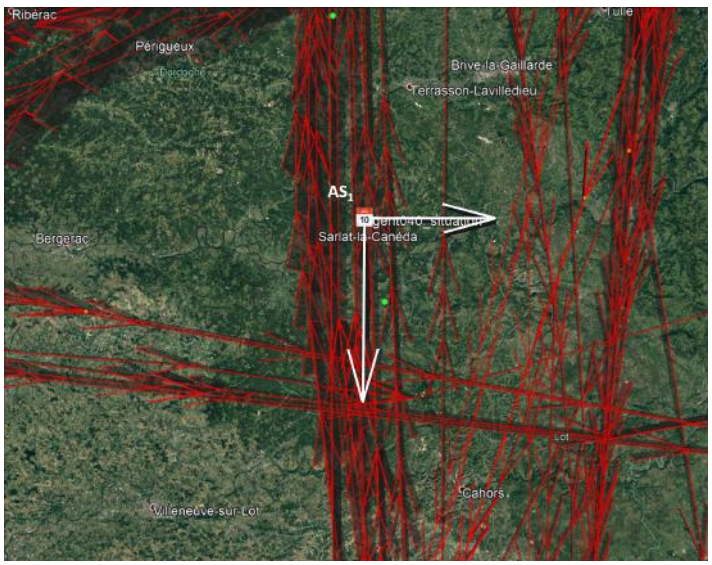


Figure 9.4 – Description de la *situation de collision* AS_1

Nous avons positionné la situation AS_1 dans une zone géographique où aucune partie de trajectoire ne correspond parfaitement pour la situation, mais des parties de trajectoires ayant quasiment le bon cap existent dans le voisinage (figure 9.4).

Comme l'*Agent Situation* nécessite deux avions pour satisfaire ses caractéristiques, il crée deux *Agents Trajectoires* pour créer des trajectoires d'avions qui satisferont ses caractéristiques. Puisque l'*Agent Situation* exige un avion allant de l'ouest vers l'est et un avion allant du nord vers le sud, les deux trajectoires ont respectivement choisi une partie de la trajectoire apprise par EVAA sur une trajectoire de Bordeaux à Lyon et de Paris à Toulouse. Ensuite, ils créent des *Agents Parties* correspondant à ces parties de trajectoire, et les laissent créer les *Agents Parties* pour compléter la trajectoire. Comme les trajectoires ne satisfont pas à la situation, les *Agents Parties* créés commencent à adapter leurs parties de trajectoire (ici, ils décident de modifier les caractéristiques géographiques et temporelles de leur partie de trajectoire d'origine) pour mieux satisfaire l'*Agent Situation*. Le figure 9.5 montre les deux trajectoires générées qui permettent aux avions de créer la *situation de collision*.

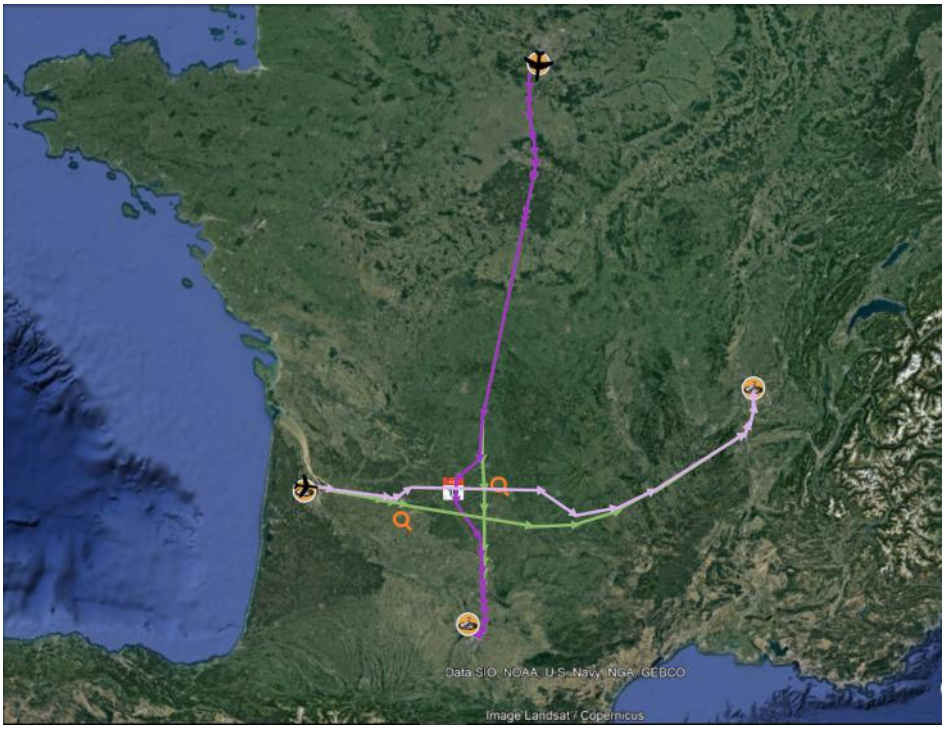


Figure 9.5 – Résultats d'une génération de scénario avec la *situation de collision* AS_1, en violet les *Agents Parties*, et en vert les parties de trajectoires de la base de données de trajectoire utilisées par ces agents

La figure 9.6 représente toutes les criticités de la *situation de collision*. Dans un premier temps, les *Agents Parties* tentent de réduire la criticité des caractéristiques de géométrie, de position, d'altitude et de temps, puis de réduire la criticité de la caractéristique de proximité lorsque toutes les autres criticités sont inférieures à un seuil de 10. Aux alentours du 90^{eme} cycle, toutes les criticités sont inférieures à 20, ce qui signifie que dans la *situation de proximité* générée, les deux avions passent par la même position (à $1,2km$ de la position souhaitée) avec une différence d'une minute.

Les premiers résultats montrent que les agents d'AGEAS sont capables de générer la situation en moins de 100 cycles de vie des agents (figure 9.6), ce qui représente environ $5,45s$.

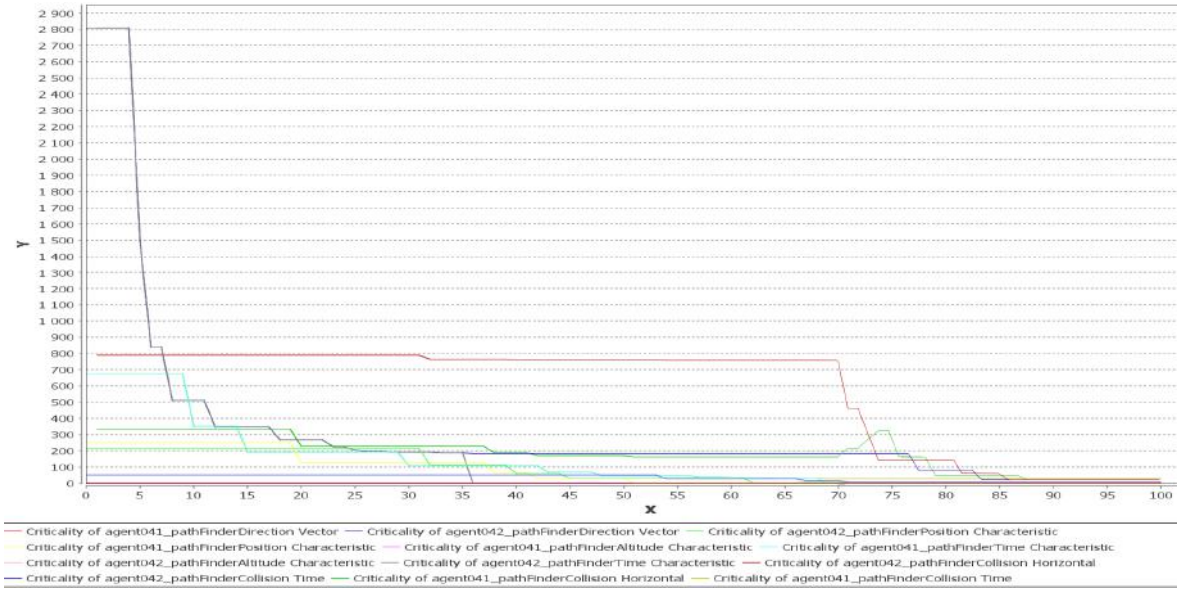


Figure 9.6 – Évolution des différentes criticités de la situation en fonction du nombre d'étapes des *Agents Parties* lors de la génération de la *situation de collision* AS_1

Génération de AS_2 La situation AS_2 représentée dans la figure 9.7 est définie par :

— deux avions quelconques,

— deux caps souhaités, 180° (du nord au sud) et 90° (de l'ouest à l'est),

— une position géographique, près de $(45,51°, 2,52°)$,

— une altitude de $30000 ft$.

Figure 9.7 – Description de la *situation de collision* AS_2

Nous avons positionné la situation AS_2 dans une zone géographique où aucune partie de trajectoire ne correspond parfaitement pour la situation, mais des parties de trajectoires passent à proximité.

Se situant trop loin d'autres parties de trajectoires, les *Agents Trajectoires* de l'*Agent Situation* AS_2 sont obligés d'utiliser des parties de trajectoires apprises par EVAA entre Bordeaux

et Paris. Dans cette situation, les *Agents Parties* impliqués dans la génération de la *situation* sont obligés d'effectuer des rotations de leur partie de trajectoire afin de passer exactement par la position requise, et d'avoir l'angle requis. Les autres *Agents Parties* s'adaptent à leur modification afin de préserver au maximum la trajectoire (figure 9.8). Les *Agents Parties* de l'*Agent Trajectoire* venant de Bordeaux ont dans notre cas utilisé des parties de trajectoires apprises sur des données erronées dans lesquelles l'avion qui volait indiquait voler vers Lyon alors qu'il ne s'y rendait pas (voir [Rantrua, 2017]). Dans cette expérimentation, les *Agents Parties* ne peuvent pas utiliser d'autres actions que la modification (et l'*Agent Extrémité* n'est pas non plus implémenté complètement), ainsi la trajectoire venant de Paris (cet effet se remarque aussi dans l'autre sens), qui détourne sa trajectoire pour générer la *situation de collision* AS_2, ne peut arriver à Bordeaux. L'étude de l'ensemble des actions (notamment l'ajout) en perspective permettra de vérifier l'adéquation du comportement de l'*Agent Partie* pour résoudre ses cas.

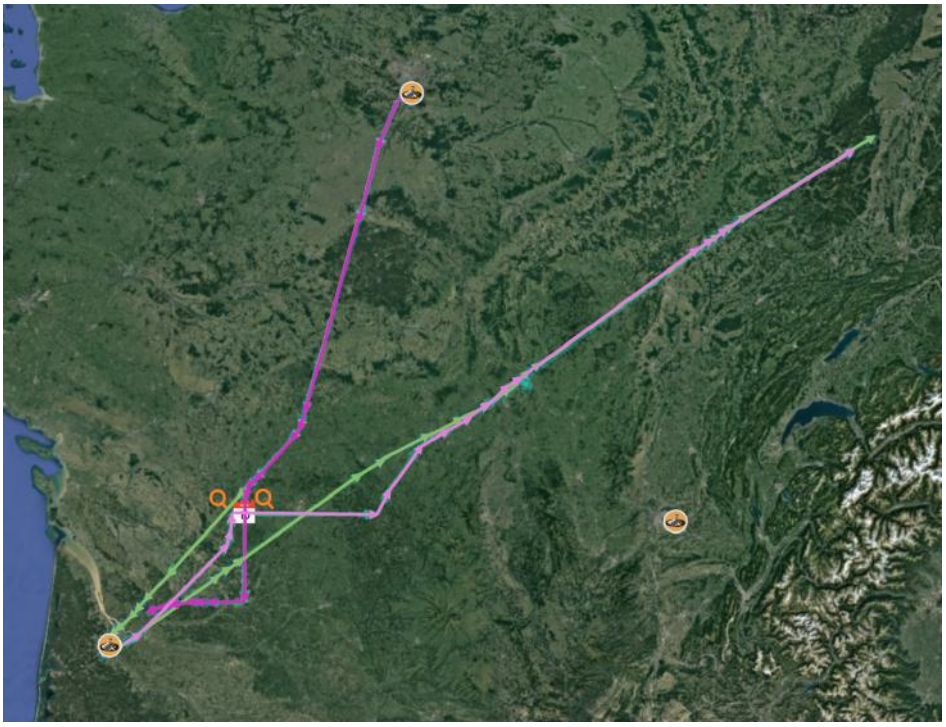


Figure 9.8 – Résultats d'une génération de scénario avec la *situation de collision* AS_2, en violet les *Agents Parties*, et en vert les parties de trajectoires de la base de données de trajectoires utilisées par ces agents

Génération de AS_3 La situation AS_3 représentée dans la figure 9.9 est définie par :

— trois avions A320,

— trois caps souhaités, 180° (du nord au sud), 90° (de l'ouest à l'est) et 270° (du sud-est vers le nord-ouest),

— une position géographique, près de $(46,61°, 1,62°)$,

— une altitude de $30000 ft$.

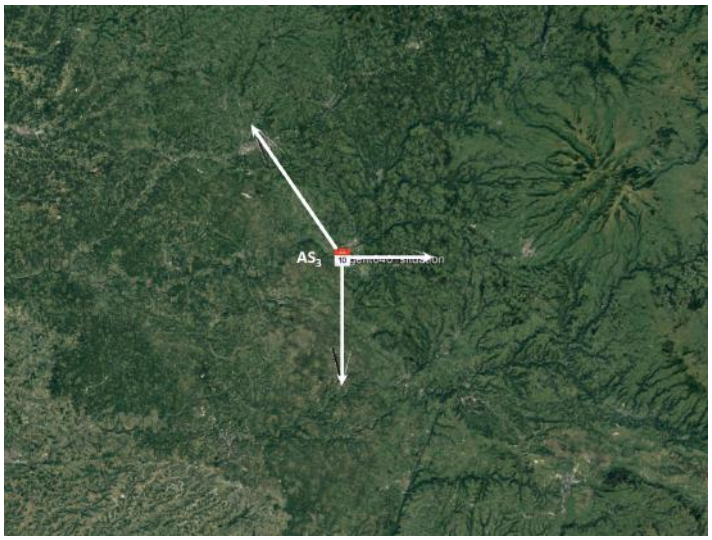


Figure 9.9 – Description de la *situation de collision* AS_3

La *situation de collision* AS_3 permet de montrer la généricité de la définition de la collision et la capacité des agents d'AGEAS à générer des collisions avec plus de deux avions, mais aussi sa capacité à générer une trajectoire lorsqu'aucune partie de trajectoire ne correspond pour la position et le cap (cap 270°).

Trois points sont à noter lors de cette génération (figure 9.10) :

— les *Agents Parties* montrent leur capacité d'adaptation afin de générer une trajectoire selon le cap 270 en passant par la position requise, malgré le fait qu'il n'y ait pas de partie de trajectoire proche ayant un cap proche. Néanmoins, l'angle de virage qui découle de cette génération est peu probable pour un avion. Dans AGATS, le problème sera détecté lors de la phase de simulation par l'*Agent EM* qui suit la trajectoire. Dans AGEAS, ce problème pourrait être détecté en ajoutant d'autres mesures de criticité afin de vérifier l'angle entre deux parties de trajectoires,

— les *Agents Parties* ne pouvant pas ajouter de parties de trajectoires, les trajectoires venant de Toulouse et de Bordeaux n'arrivent pas à destination. Pour le cas de la trajectoire venant de Toulouse, le problème est le même que pour la situation AS_2 (les *Agents Parties* n'ont pas le droit d'ajouter les parties de trajectoires nécessaires),

— pour le cas de la trajectoire venant de Bordeaux, le problème est légèrement différent : lors de la génération de la première trajectoire par l'*Agent Trajectoire*, les parties de trajectoire utilisées viennent d'une trajectoire incomplète (par exemple parce que le transpondeur de l'avion a été arrêté). Par conséquent, la première trajectoire (verte) n'arrive pas à destination, et pour les raisons précédentes, les *Agents Trajectoires* ne complètent pas la trajectoire.

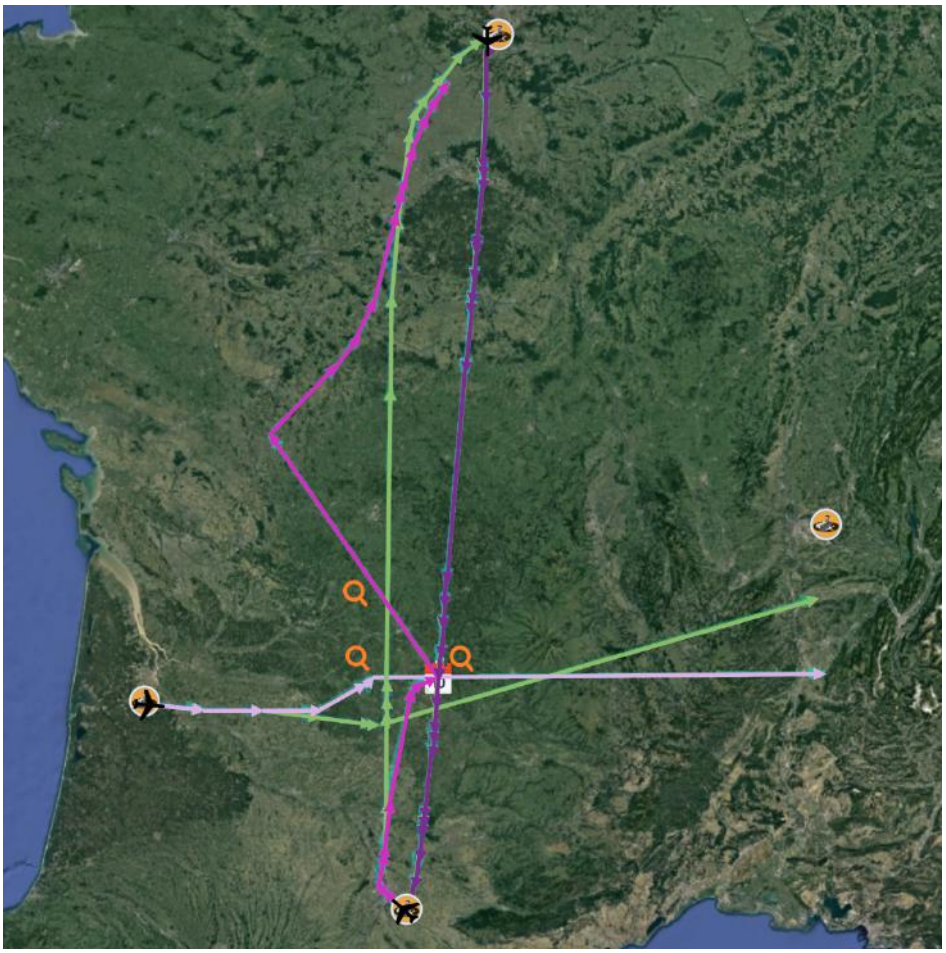


Figure 9.10 – Résultats d'une génération de scénario avec la *situation de collision* AS_3, en violet les *Agents Parties*, et en vert les parties de trajectoires de la base de données de trajectoires utilisées par ces agents

9.3 Conclusion sur le système AGEAS

Ces expérimentations ont permis de montrer la capacité des agents d'*AGEAS* à générer des *situations de collision*, et la généricité de la définition de ses *situations*. Ces expérimentations montrent la capacité d'adaptation des agents d'*AGEAS*, en particulier des *Agents Parties*, qui sont capables de générer pour leur *Agent Trajectoire* une trajectoire la plus réaliste possible, et respectant les caractéristiques de l'*Agent Situation*, malgré les caractéristiques de la *situation* qui n'autorisent qu'une génération simple, proche des parties de trajectoires provenant de la base de données de trajectoires.

Ces expérimentations préliminaires sont encourageantes, et laissent envisager des résultats similaires pour l'utilisation des autres actions, permettant de générer des trajectoires complètes une fois que les *Agents Extrémités* seront implémentés complètement (table 9.1).

Les perspectives d'*AGEAS* sont très liées avec l'implémentation complète d'*AGATS*, qui permettra de lier totalement la simulation et la génération de scénarios. En effet, le système *AGEAS* se concentre sur la phase de création de scénario d'*AGATS*, et ne traite pas de l'adap-

Système			AGEAS
Structuration simulation			Scénario (et Simulation, Perspective couplage *AGEAS* et *CAAMAS*)
Catégorie de méthodologie			Enregistrement et De zéro
Automatisation			Forte
Utilisation de l'environnement			Indirectement (Perspective couplage *AGEAS* et *CAAMAS*)
Situations	Micro		Collision
	Macro		Non testé
Réalisme	Trajectoire	Physique	Apprentissage sur Enregistrements
		Comportement	Apprentissage sur Enregistrements
	Trafic		Apprentissage sur Enregistrements
Adaptation du trafic		Scénario	Aucune (Perspective couplage *AGEAS* et *CAAMAS*)
		Simulation	Aucune (Perspective couplage *AGEAS* et *CAAMAS*)
Nécessite plan de vol			Non

Table 9.1 – Le système multi-agent adaptatif *AGEAS* positionné selon les critère de l'état de l'art sur l'évitement de collisions (chapitre 3, en particulier les tables 3.1 et 3.2)

tation du trafic, que se soit à la simulation (travail effectué dans *CAAMAS*, voir chapitre 8), ou au scénario (travail perspectif de *CAAMAS*), ni de sa structuration lors de la simulation (qui est prévue à travers l'adaptation des *Agents Trajectoires* et *Agents Parties* grâce aux retours des *Agents EM*). De même, l'utilisation de l'environnement est pour l'instant indirecte avec l'utilisation de l'apprentissage, mais elle sera plus complète avec le couplage entre les *Agents EM* et les *Agents Trajectoires*.

En l'état actuel, *AGEAS* permet d'obtenir un réalisme intéressant au niveau des trajectoires grâce à l'utilisation de l'apprentissage d'EVAA (voir 6.4). Au niveau du trafic, le réalisme devrait être fort lorsque l'*Agent Extrémité* sera complètement implémenté et qu'il générera le trafic contextuel.

Enfin, *AGEAS* est fortement automatisé, contrairement aux systèmes et méthodes étudié dans l'aviation (chapitre 3 en particulier la table 3.1), et permet de lier les méthodologies partant de zéro (*via* les comportements des agents) et d'enregistrement (*via* l'apprentissage).

Conclusion et Perspectives

Conclusion

Le trafic aérien (chapitre 1) est complexe. En effet, c'est un domaine vaste, composé de plusieurs milliers d'entités en interaction avec des buts qui leur sont propres et parfois conflictuels (liés au pilote, au type d'avion, à la ligne de vol, à la compagnie, entre autres), sujets à des aléas structurels (ouverture ou fermeture d'une portion de l'espace aérien, ouverture ou fermeture d'une piste) ou météorologiques (vent, orages, éruptions volcaniques, *ect.*). Son organisation en temps réel est critique, aussi bien pour la sécurité (par exemple l'évitement de collisions) que pour son bon déroulement en terme d'optimisation (par exemple éviter que les avions soient mis en attente, accélérer le trafic).

Ainsi, la simulation, qui est un outil puissant d'apprentissage, de visualisation, et de compréhension de l'impact d'une décision à un moment donné sur tout un système, est fortement utilisée afin d'éprouver les outils de demain (chapitre 2). Néanmoins, la structuration d'une simulation de trafic est un processus long qui nécessite la mobilisation d'une équipe conséquente pour contrôler son déroulement dans le monde aérien.

La structuration de simulation de trafic est l'organisation d'une simulation de trafic pour qu'elle soit réaliste du point de vue du trafic et des trajectoires des entités mobiles qui la composent, et pour qu'elle contienne des *situations* requises par un scénariste. Les techniques de structuration de simulation de trafic présentent certaines limites (chapitre 3). Les techniques de structuration de trafic aérien (section 3.1) se focalisent majoritairement sur la structuration à partir d'un scénario, sont peu automatisées, et sont souvent peu génériques et ne s'adaptent pas lors de la simulation. Les techniques de structuration de trafic routier (section 3.2) apportent plusieurs points positifs à ces problèmes, notamment par l'utilisation de comportements locaux des entités mobiles. Certaines en particulier présentent une généricité et une flexibilité intéressantes lors de la simulation par l'utilisation de systèmes multi-agents.

Partant du constat qu'un scénario de trafic correspond à un ensemble de trajectoires, nous nous sommes intéressés à la génération de trajectoires, et à l'évitement de collisions, ce qui nous a permis d'étudier les comportements locaux des entités mobiles dans le trafic aérien (chapitre 4).

Les solutions les plus prometteuses dans ces deux états de l'art laissent à penser que la solution se trouve, en partie, dans la décentralisation. Ainsi, les systèmes multi-agents, naturellement décentralisés, et en particulier les systèmes multi-agents adaptatifs, repré-

sentent une alternative séduisante. Dans de tels systèmes, l'auto-organisation coopérative des agents leur permet de trouver collectivement, une solution au problème qui leur est présenté (chapitre 6).

Nous appuyant sur ces systèmes multi-agents adaptatifs, nous proposons le système *Autonomous GenerAtion of Traffic Simulations (AGATS)* pour la structuration en temps réel d'une simulation de trafic (chapitre 7), combinant la structuration durant la phase de génération de scénario, et pendant la phase de simulation.

Nous expérimentons séparément les comportements des agents qui composent *AGATS* dans les deux phases. Notre première expérimentation porte sur les comportements locaux d'adaptation des entités mobiles, en se concentrant sur les *Agents EM*, système *Collision Avoidance Adaptive Multi-Agent System (CAAMAS)* (chapitre 8). Notre deuxième expérimentation se concentre sur le système *Autonomous GEneration of trAffic Scenarios (AGEAS)* (chapitre 9) et notamment, sur la phase de génération de scénario, et sur les agents qui agissent majoritairement dans cette phase (*Agents Situations*, *Agents Trajectoires*, *Agents Parties*, *Agents Extrémités*).

Contributions et Perspectives

Dans ce qui suit, nous présentons les contributions et les perspectives selon 3 axes :

— Le cadre scientifique des systèmes multi-agents et le modèle *AGATS*.

— Le cadre applicatif.

— Le cadre des expérimentations et de la validation

Cadre scientifique des Systèmes Multi-Agents et le modèle *AGATS*

Le système multi-agent *AGATS* est la principale contribution de cette thèse. Il offre une solution originale et innovante pour l'adaptation des entités mobiles lors de la simulation à des modifications de l'environnement, pour la construction de scénarios, leur adaptation lors de la simulation, et plus généralement pour la théorie *AMAS*.

Concernant l'**adaptation des entités mobiles lors de la simulation**, le système multi-agent *AGATS* apporte des comportements d'adaptation locaux, robustes, capables de passer à l'échelle, génériques, et capables de s'adapter à l'évolution même des paramètres internes du système. Ces comportements permettent de gérer des trafics dans des conditions réalistes mais aussi pouvant contenir plus d'avions que dans la réalité. Ils donnent des solutions en quelques secondes (quelques dizaines de secondes dans les cas les plus denses), pour des trajets de plus d'une heure, améliorant ainsi de manière continue les solutions proposées, et la prise en compte de l'incertitude au cours du vol.

Plusieurs perspectives sont possibles pour enrichir cette adaptation des entités mobiles lors de la simulation. Les *Agents EM* étant au centre de cette adaptation, ces perspectives concernent de nouveaux comportements pour ces agents :

— Les *Agents EM* pourraient décider de faire des cycles de vie à des intervalles plus ou moins long, en fonction de leur environnement, ce qui pourrait améliorer le passage à

l'échelle, et le temps de calcul (suggéré par l'étude de sensibilité section 8.2.4.5).

— Les *Agents EM* pourraient décider d'augmenter ou de diminuer leur zone de perception, ce qui est équivalent à ignorer des entités mobiles en fonction d'une distance variable, en fonction de son état et de celui de son environnement (suggéré par l'étude de sensibilité section 8.2.4.5).

— En plus de la modification des actions des *Modules Actions*, permettre à l'*Agent EM* de créer ou de supprimer des *Modules Actions* pour qu'il puisse adapter son comportement en fonction de l'état de son environnement, afin d'explorer plus l'espace des actions possibles si nécessaire, comme nous le soulignons dans l'étude de sensibilité (particulièrement la section 8.2.4.4) et dans la comparaison (section 8.2.5).

— Enfin, les expérimentations, en particulier celles du rond-point (section 8.2.2), suggèrent l'utilisation d'une nouvelle criticité pour les *Agents EM*, qui prenne en compte le temps, afin de pallier certaines situations ou un agent est beaucoup plus pénalisé que les autres agents.

Ces quatre perspectives posent finalement la question de la flexibilité du cycle de vie de l'agent, de la notion de perception locale, et plus généralement de toutes les notions qui dans le modèle *AMAS4Opt* et la théorie *AMAS* sont décidés par l'expert du domaine. L'expert pourrait finalement apporter les premiers calibrages, et le système devrait alors se calibrer lui-même, en apprenant tout au long de sa vie, ou en amont de son utilisation.

Concernant la **construction adaptative de scénarios de trafic**, le système multi-agent *AGATS* apporte une automatisation qui est peu présente dans l'état de l'art (section 3), une généricité dans la construction des situations, dans les comportements d'adaptation et de modification locale des parties de trajectoires qui conduisent à une génération décentralisée et adaptative des trajectoires.

Afin d'enrichir cette construction adaptative de scénarios de trafic, il serait intéressant d'implémenter complètement les *Agents Extrémités* et certaines de leurs caractéristiques qui ont été laissées de côté dans les expérimentations. En effet, ces agents apportent des contraintes supplémentaires sur le réalisme pour tempérer les modifications parfois nombreuses des *Agents Trajectoires*.

Enfin, dans notre cas d'application, chaque *Agent EM* intervient au maximum dans une *situation*, ce qui est peut-être le cas dans le trafic aérien sur un secteur (notre cas d'application), néanmoins, cette hypothèse est restrictive dans le cadre général, d'une part parce qu'une entité mobile pourrait intervenir plus tard dans une autre situation (par exemple, une collision dans un secteur et dans le secteur suivant), d'autre part parce que la construction de situations plus complexes le nécessitent (dans le trafic routier, un conducteur qui dépasse dangereusement plusieurs fois). Cette modification nécessite l'ajout de mécanismes pour que les situations communiquent entre elles, par exemple pour qu'elles se partagent un *Agent Trajectoire*, et permettre aussi aux *Agents Situations* de se déléguer des caractéristiques.

Enfin, *AGATS* est avant tout un **modèle de structuration de trafic temps réel**, liant phase de simulation et phase de création de scénario, pour une adaptation du scénario aux évolutions des entités mobiles dans la simulation, mais aussi de l'environnement, afin de maintenir la cohérence et le réalisme du scénario requis, dont les deux principales parties, *CAAMAS* et *AGEAS*, sont déjà fonctionnelles.

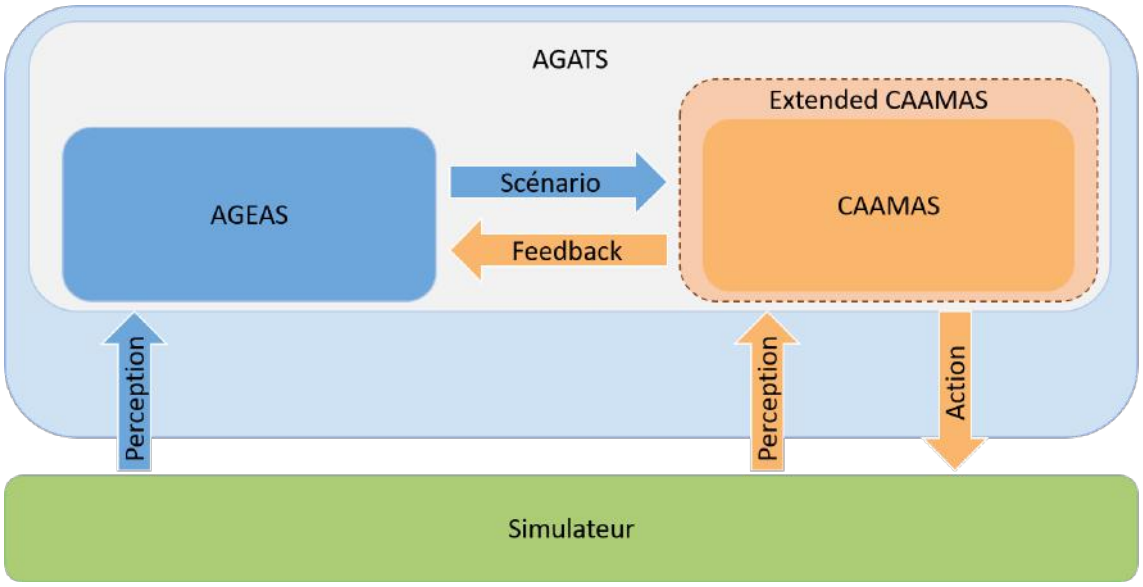


Figure 9.11 – Le modèle AGATS complet et ses deux sous-modules

L'une des perspectives les plus pressantes du système *AGATS* reste alors le couplage entre ces deux parties c'est-à-dire les interactions entre les *Agents EM* et les *Agents Trajectoires*, afin de lier complètement la phase de génération de scénarios et la phase de simulation et d'adaptation du scénario lors de la simulation. Ce couplage est une partie importante du modèle qu'il faut tester pour évaluer entièrement l'apport d'*AGATS*.

Dans nos expérimentations, nous avons pris à part les deux groupes d'agents, c'est-à-dire les *Agents EM* dans *CAAMAS*, et les *Agents Situations*, les *Agents Trajectoires*, les *Agents Parties* et les *Agents Extrémités* dans *AGEAS*. La deuxième étape est d'étendre les comportements des *Agents EM* de *CAAMAS* afin qu'ils puissent intervenir dans une *situation*. Pour réaliser cette étape, il faut ajouter une nouvelle valeur de criticité plus importante que celle de collision et de trajectoire, pour que les *Agents EM* intervenant dans des *situations* soient prioritaires. Il est également nécessaire de compléter le comportement des *Agents EM* pour ne pas gérer une collision lorsque cette dernière est requise par un agent situation en permettant l'interaction entre *Agents EM* impliqués dans la situation. La deuxième étape consiste à enrichir le système en prenant en compte ces nouvelles interactions, terminant le couplage entre *AGEAS* et *CAAMAS* représenté dans la figure 9.11.

Cette thèse contribue à enrichir la théorie des **AMAS** sur les points suivants :

— La criticité, en particulier sur les fonctions de criticités et sur la gestion de l'hétérogénéité des criticités des agents.

— L'auto-organisation.

La notion de **criticité** est fondamentale dans les *AMAS*, et une grande partie de la définition des agents réside dans la définition de ses criticités. La criticité, qui est une mesure de la non satisfaction des buts de l'agent, permet aux agents de s'évaluer eux-mêmes et de se comparer aux autres agents dans leur environnement. C'est donc un mécanisme essentiel pour mettre en oeuvre la coopération entre les agents.

La **fonction de criticité** varie beaucoup en fonction du cadre théorique et applicatif.

[Perles, 2017] utilise une fonction basée sur les moindres carrées entre deux valeurs (celle du voisinage et celle calculée par l'agent) dans sa thèse sur l'estimation du voltage des noeuds dans un réseau électrique. [Rantrua, 2017] utilise, entre autre, une distance de Manhattan généralisée et pondérée pour déterminer le contexte le plus proche de l'état de l'avion simulé. [Boes, 2014] utilise différentes fonctions (fonction "seuil", "consigne", "optimisation") proches du cadre applicatif, en l'occurrence la calibration de moteur à combustion, et construit, si possible (cadre bornée), ces fonctions avec des fonctions "barrières" plus simples.

Dans cette thèse, les fonctions de criticités sont composées de trois parties linéaires, qui représentent respectivement la satisfaction (criticité nulle), un degré de non satisfaction, et un degré de non satisfaction maximal (1000 ou 100). Pour le contrôle d'un système, ce travail souligne que la fonction de criticité n'a pas besoin de représenter parfaitement l'évolution de l'état d'un agent, et ainsi n'a pas besoin d'être une fonction compliquée. Au contraire, une fonction trop compliquée peut avoir tendance à générer d'autres problèmes. Par exemple, une fonction dont la dérivée est très faible aux alentours de la valeur visée donne une criticité très faible, alors que la valeur obtenue reste loin de la valeur requise (annexe A de [Boes, 2014]). La fonction peut alors être une fonction linéaire par morceaux (et continue), et réduite à son rôle premier : donner un ordre d'idée de la satisfaction des agents.

La **gestion de l'hétérogénéité** des criticités, et par extension l'hétérogénéité des agents prend différentes formes dans les travaux sur la théorie *AMAS*. La criticité d'un véhicule a explicitement ou, la plupart du temps, implicitement, une sémantique parfois extrêmement différente. Dans la thèse [Esteoul, 2019] sur la prévision de production de parcs éoliens (et d'éoliennes), la criticité des agents qui représentent des parcs éoliens est normalisée par leur taille avant d'être comparée à la criticité d'autres agents du même type. Dans la thèse [Bouziat, 2017] portant sur la coopération entre systèmes de systèmes, les agents qui représentent un système peuvent normaliser les criticités qu'ils perçoivent par un facteur qu'ils apprennent, sous réserve que les autres agents respectent un standard.

Dans cette thèse, nous retrouvons une problématique similaire, dans des agents de même type, mais avec plusieurs criticités hétérogènes (par exemple $Crit_{traj,comp}$ qui se partage selon les axes géographique, temporel, de vitesse, et d'altitude, voir section 7.7). Nous avons utilisé une fonction de criticité générique basée sur les distances entre l'état requis et l'état actuel, avec en paramètre une valeur maximale pour standardiser les valeurs dans une plage (entre 0 et 1000), ce qui introduit un taux de conversion implicite entre les différents axes. De plus, nous ajoutons des mécanismes d'ordonnancement adaptatifs préétablis des criticités des *Agents EM* pour les criticités qui composent $Crit_{i,k,traj}$ (section 7.9).

Ces deux aspects de la criticité (fonction et hétérogénéité) présentent différentes perspectives pour les *AMAS* en général, et pour *AMAS4Opt*. Il serait intéressant de **standardiser un ensemble de criticités élémentaires**, dont les propriétés sont connues pour qu'il soit utilisable dans différents travaux. Les fonctions de criticités de cette thèse, linéraires par morceaux et continues, semblent suffisantes dans différents cas. Toujours concernant la criticité, la **généralisation des mécanismes de taux de conversion**, et des **mécanismes d'ordonnancement adaptatif** des criticités devrait être étudiée. Pour résoudre certaines SNC de manière générique, il faudrait que les agents soient capables d'adapter les criticités perçues

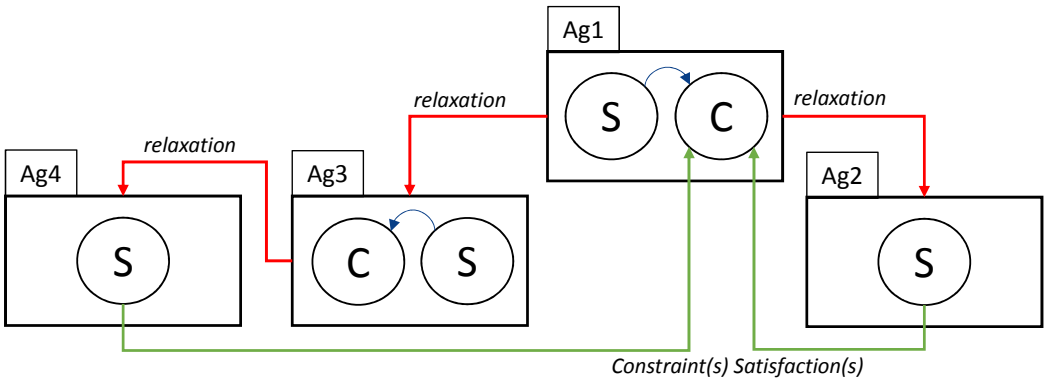


Figure 9.12 – Relaxation de la criticité dans [Bonnet et al., 2015]

ou envoyées, et de modifier l'importance de certains critères. De tels mécanismes requièrent d'avoir un ordre dans les criticités qui peut évoluer et des paramètres de calcul qui peuvent être modifiés (par exemple, la valeur maximales des criticités dans le cadre de cette thèse).

L'**auto-organisation** reste le coeur des *AMAS* et cette thèse y contribue selon différents axes notamment sur le pattern *AMAS4Opt* :

— La relaxation de criticité.

— La délégation de service.

— La coopération entre agents ayant le rôle service.

La **relaxation de la criticité** est un mécanisme qui permet à un agent service de gérer une partie d'une contrainte d'un agent service et de propager dans son voisinage ce qu'il n'est pas capable de gérer. Ce mécanisme est présenté dans le cadre d'AMAS4Opt dans [Bonnet et al., 2015], afin de "faire de la place" dans un planning pour une nouvelle tâche (figure 9.12), en l'occurrence des prises de vues de satellites. Dans cette thèse, cette relaxation est effectuée implicitement (sans dialogue), par chaque *Agent EM* qui recherche l'action qui diminue le plus les criticités ***des*** *Agents EM* les plus critiques parmi son ensemble d'actions, montrant aussi la résilience du pattern AMAS4Opt pour de tels cas.

La **délégation de service** est un comportement ajouté pour la conception du système ATLAS [Bonnet et al., 2015], qui trouve dans cette thèse un nouveau cas d'application, et se trouve généralisé par une délégation multi-niveau et l'hétérogénéité des agents qui y participent (figure 9.13). L'*Agent Situation* délègue à ses *Agents Trajectoires* la tâche de générer des trajectoires respectant certaines de ses caractéristiques, qui délèguent eux-mêmes cette tâche à leurs *Agents Parties* respectifs. Dans le cas où les *Agent Situation* peuvent se déléguer des contraintes (cas envisagé précédemment qu'un *Agent EM* intervienne dans plusieurs situations), d'autres niveaux seraient ajoutés à cette délégation. Dans le cadre plus général des *AMAS*, cette dernière délégation serait l'équivalent de mécanismes pour que les agents reconnaissent qu'ils ont un ou plusieurs buts communs, et qu'ils peuvent agir ensemble, au moins momentanément, pour réaliser ce but.

Cette structure multi-niveau n'est autre qu'une représentation des liens entres les différents niveaux micro, meso, et macro d'un système. L'auto-organisation qui émerge des inter-

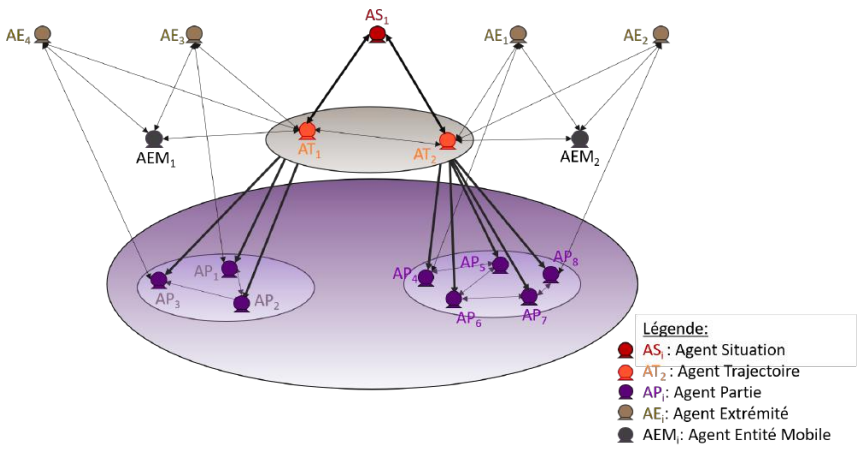


Figure 9.13 – Généralisation de la délégation de service dans *AGATS*

actions entre les agents fait apparaître ces différents niveaux. Le comportement coopératif qui anime les agents permet l'émergence d'évènements macroscopique ou mesoscopique (ici les situations) du point de vue de l'agent (ici l'entité mobile).

Enfin, cette thèse apporte des mécanismes de **coopération entre agents services** au travers de la coopération des *Agents Trajectoires* et *Agents Extrémités* avec leurs *Agents EM* respectifs. En effet, l'*Agent Trajectoire* demande à son *Agent EM* de suivre une trajectoire, ce que fait l'*Agent EM*, lui rendant donc un service, mais ce dernier demande aussi à l'*Agent Trajectoire* de modifier la trajectoire s'il ne peut pas la suivre, parce qu'il n'en a pas la capacité ou qu'il s'en est éloigné pour s'adapter aux évolutions de l'environnement.

Cadre applicatif

Dans le cadre applicatif, cette thèse contribue à la structuration de trafic aérien, mais aussi à la structuration d'une simulation de trafic, voir à la structuration de trafic (outil d'aide à la décision) ou de simulation.

Du côté du **trafic aérien**, cette thèse contribue à l'enrichissement du logiciel EVAA développé par la thèse de [Rantrua, 2017] en complétant ce premier travail par une générateur de trafic aérien réaliste. Les expérimentations valident le modèle *AGATS* et son adaptation au domaine aérien, et cette adaptation pourrait être explorée plus en profondeur sur différents plans :

1. Dans les trafics réels, les perceptions sont souvent multiples, et multi-modales. Par exemple, la plupart des avions commerciaux se transmettent des données *via* leur transpondeur [1], mais ils possèdent aussi d'autres capteurs, comme le radar météo pour détecter des précipitations ou des orages, des sondes Pitot pour détecter la pression atmosphériques, des accéléromètre, des GPS... Nous nous sommes abstraits de la plupart de ces sources en considérant que la position des *Agents EM* était absolue et sans erreur, mais le *Module Perception* de l'*Agent EM* pourrait être en charge de concorder des

1. émetteur/récepteur radio permettant d'envoyer des données entre des avions

sources hétérogènes d'informations, et de détecter des anomalies. Une telle modification demanderait peut être de donner au *Module Perception* des comportements et des capacités de décisions qui en ferait un ou plusieurs agents, comme le suggèrent certains travaux sur la détection d'anomalies [Verstaevel et al., 2018].

2. Les calculs des *Agents EM* dans les expérimentations de *CAAMAS* ne sont pas distribués physiquement, ainsi, expérimenter la distribution de ces calculs permettrait d'appréhender pleinement le potentiel en terme de passage à l'échelle de ces agents. Cette distribution physique devrait se faire facilement grâce à la décentralisation naturelle des agents et l'indépendance dans le calcul des *Modules Actions* des *Agents EM*.
3. Concernant *AGEAS*, les expérimentations ont validé la capacité du système à générer des *situations de collision*. Si cette *situation* reste primordiale, de nouvelles situations sont à ajouter pour permettre la génération d'un scénario complet. L'implémentation de ces *situations* nécessite l'étude de leurs caractéristiques afin d'en déduire les criticités correspondantes et l'implication des différents agents dans leur construction. Une piste intéressante consiste à rendre plus générique la génération de ces différentes situations. Certaines réponses se trouvent sans doute du côté des travaux sur la composition de services Web ou de composants informatiques [Noël, 2012][Degas et al., 2016], et des pistes communes avec la délégation de caractéristiques entre situations semblent envisageables.

Ensuite, le système multi-agent *AGATS* a été conçu pour être générique. Il serait donc opportun de l'utiliser dans le contexte d'**autres simulations de trafics** pour juger de son degré de généricité, par exemple sur un trafic de drones ou sur le trafic routier. Les *Agents EM* de part leur architecture sont assez génériques, et demandent uniquement de définir ce que sont les actions possibles. Les autres agents nécessitent des adaptations variables pour être utilisés dans d'autres trafics. Les *Agents Situations* ne devraient pas présenter plus de travail que pour la génération d'une nouvelle situation dans le domaine aérien. Les *Agents Trajectoires* et les *Agents Parties* devraient peut être nécessiter d'autres mesures de criticités propres au domaine pour les parties de trajectoires. Enfin, les *Agents Extrémités* nécessitent un travail de généralisation. Le travail d'adaptation nécessitera aussi d'avoir une base de données de trajectoires que les agents pourront utiliser. Un travail actuellement en cours dans un projet de première année de Master pour utiliser les comportements des *Agents EM* sur des Drones quadri-moteurs dans le simulateur Paparazzi [2], a pour but de tester le comportement des *Agents EM* dans un simulateur réaliste et valider la généricité de l'approche.

Une étape suivante serait alors d'utiliser *AGATS* pour un trafic hétérogène, mais aussi d'aborder l'inter-modalité par l'angle pris par cette thèse sur la structuration de trafic. Cette adaptation pourrait utiliser par exemple l'esprit de l'approche générique de [Wassink et al., 2006] et spécialiser les *Agents Trajectoires* et *Agents Parties* en fonction du type d'entité mobile avec lesquels ils travaillent (par exemple, une voiture et un piéton), ou en ajoutant d'autres mesures de criticités qui traduiront la différence entre ces entités mobiles.

D'un autre côté, cette thèse pourrait évoluer comme **outil d'aide à la décision**. En effet, cette thèse est dédiée à la structuration de trafic virtuel, mais ce trafic pourrait être réel, et les *situations* seraient alors d'une nature légèrement différente. Par exemple, cette thèse

2. simulateur drone développé par l'ENAC

pourrait être utilisée comme support d'aide à la décision d'un contrôleur aérien, pour demander à un flux d'avions de se diviser en deux sous-flux, empêcher les avions de passer par une zone, ou au contraire les faire passer dans une zone en priorité. Cette perspective reposerait majoritairement sur les aspects dynamiques d'*AGATS* et s'éloigne finalement peu de la problématique. La différence majeure est que les entités mobiles ne sont pas créées à la demande, mais existent déjà. De même, cette thèse pourrait s'appliquer à la thématique des flottes de véhicules autonomes accessibles à la demande. Fonctionnant comme nos taxis actuels (un véhicule cherche un utilisateur, le dépose à sa destination, et va chercher un nouvel utilisateur), ou comme le taxi collectif (un véhicule peut prendre plusieurs utilisateurs avec la même destination, ou des destinations complémentaires). Ce dernier est particulièrement intéressant pour la gestion des ressources, la génération de trajectoires répondant au maximum de critères (les départs et destinations des usagers), et retrouve de nombreux points avec cette thèse, avec une contrainte supplémentaire : le nombre de véhicules est limité, et on cherche à diminuer au maximum la durée des trajectoires.

Enfin, dans une thématique plus éloignée, cette thèse parle de génération de scénarios et de leur adaptation lors de la **simulation**, et il serait très intéressant de se rapprocher d'autres domaines nécessitant la génération de scénarios (ici au sens large), comme la structuration de jeux vidéos, par exemple pour préserver une trame d'histoire, ou adapter dynamiquement le scénario aux actions d'un joueur, ou la génération de textes. L'*Agent Trajectoire* (Agent Rôle) devraient alors générer un ensemble d'actions que son *Agent EM* (Agent Acteur) devrait effectuer, et les *Agents Parties* (Agent Action) s'occuperaient de ces actions élémentaires.

Cadre des expérimentations et de la validation

Dans cette thèse, plusieurs expérimentations valident de nombreux aspects et hypothèses d'*AGATS*, mais certains aspects n'ont pas été testés.

A court terme, différentes expérimentations peuvent être effectuées afin de valider certains comportements prévus dans le modèle *AGATS* :

— Expérimenter les pertes ou la détérioration des communications des *Agents EM* (non envoi de la position, non envoi du vecteur vitesse, non envoi de la criticité, ou toute combinaison de ces détériorations), pour valider les comportements prévus pour cette situation de non coopération (SNC 5 et SNC 6).

— Expérimenter les *Agents EM* en tenant compte de l'altitude pour valider toutes les actions prévues permettrait aussi de montrer les avantages de la décentralisation, notamment sur le passage à l'échelle.

— Expérimenter les *Agents EM* dans un trafic mixte, composé d'entités mobiles non coopératives, et d'entités coopératives permettrait de montrer la possibilité d'une intégration progressive dans un trafic réel, mais aussi l'adaptation des *Agents EM* face à des obstacles mobiles non coopératifs.

— Expérimenter le comportement des *Agents EM* dans un environnement contenant du vent, afin de tester la robustesse de leur comportement pour cet évènement particulier, mais aussi plus généralement face à des incertitudes dans le résultat des actions.

— Expérimenter le système multi-agent *AGATS* en utilisant d'autres bases de données de trajectoires que les résultats de l'apprentissage d'EVAA, par exemple les roadmaps issues d'algorithmes de planification telles que : les PRM ou le RRT (voir section 4.2). Cela pourrait demander l'ajout de certaines capacités aux agents, par exemple pour segmenter des parties de trajectoires qui sont des courbes, par échantillonnage ou en utilisant les tangentes.

Ensuite, les expérimentations ont soulevées différents points qu'il serait intéressant d'explorer :

— Effectuer des expérimentations sur d'autres benchmarks ou avec des valeurs de r_{Z_p} plus petites afin d'approfondir les hypothèses émises sur cette distance de vision lors de l'étude de sensibilité (section 8.2.4.3). Il est étonnant que le temps de calcul n'évolue pas avec cette distance de vision, et il faudrait vérifier l'hypothèse de la densité d'avion perçu qui évolue peu.

— Les expérimentations d'*AGATS* se font dans un mode semi-asynchrone, qui reste une hypothèse non réaliste, en particulier pour les *Agents EM*, et effectuer des expérimentations purement asynchrones permettraient de vérifier la cohérence des comportements des différents agents d'*AGATS*. Il serait par ailleurs intéressant de tester le déterminisme de *CAAMAS* dans ces deux cas.

De manière générale, ce dernier point existe dans de nombreux systèmes *AMAS*, et il serait intéressant de s'y attaquer plus généralement. Une piste serait du côté de la criticité : la criticité de trajectoire de l'*Agent EM* ($Crit_{i,traj}$) pourrait s'aggraver au fil du temps tant qu'il est défavorisé (tant que les autres agents qui ont agit avant lui ne l'aident pas à revenir à sa trajectoire, visible dans la section 8.2.2), de même la criticité d'agents défavorisés au détriment d'autres pourrait s'aggraver au fil du temps.

Enfin, une problématique intéressante dans les *AMAS* est la validation de l'adéquation fonctionnelle des différentes parties du système, c'est-à-dire des différents agents qui le composent. Les comportements des agents nécessitent d'être vérifiés, et cela implique la mise en situation de l'agent afin d'étudier si sa réponse à la situation dans laquelle il se trouve est adéquate. Certaines situations peuvent être représentées schématiquement, mais il est difficile de représenter toute la complexité des situations rencontrées par un agent. Il serait alors intéressant de définir des caractéristiques qu'un agent doit satisfaire, et qu'un système génère un scénario de test adéquat. Généraliser le système *AGATS* pourrait être un début de réponse pour une telle problématique. Un tel système de génération de scénarios pourrait être combiné à des comportements d'adaptations proposés précédemment pour la criticité et l'organisation afin que le système détecte des situations dans lesquelles il est en difficulté, génère un ensemble de scénarios, et s'entraîne sur ces scénarios pour apprendre à surmonter ces situations.

Bibliographie

[Abdelgawad et al., 2016] Abdelgawad, K., Henning, S., Biemelt, P., Gausemeier, S., and Trächtler, A. (2016). Advanced traffic simulation framework for networked driving simulators. *IFAC-PapersOnLine*, 49(11) :101–108.

[Alam et al., 2007] Alam, S., Shafi, K., Abbass, H. A., and Barlow, M. (2007). Evolving air traffic scenarios for the evaluation of conflict detection models. In *Proc. 6th Eurocontrol Innovative Research Workshop, Eurocontrol Experiment Research Center*.

[Allignol et al., 2013] Allignol, C., Barnier, N., Durand, N., and Alliot, J.-M. (2013). A new framework for solving en-routes conflicts. In *ATM 2013, 10th USA/Europe Air Traffic Management Research and Development Seminar*, pages pp 1–9, Chicago, United States.

[Allignol et al., 2016] Allignol, C., Barnier, N., Durand, N., and Blond, E. (2016). Detect & avoid, uav integration in the lower airspace traffic. In *ICRAT 2016, 7th International Conference on Research in Air Transportation*.

[Allignol et al., 2012] Allignol, C., Barnier, N., Flener, P., and Pearson, J. (2012). Constraint programming for air traffic management : a survey 1 : In memory of pascal brisset. *The Knowledge Engineering Review*, 27(3) :361–392.

[Alliot and de Verdière, 2003] Alliot, J.-M. and de Verdière, D. C. (2003). Atm : 20 ans d'effort et perspectives. In *Symposium de l'Académie Nationale de l'Air et de l'Espace : vers l'automatisation du vol et sa gestion*.

[Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3) :345–405.

[Badeig et al., 2016] Badeig, F., Adam, E., Mandiau, R., and Garbay, C. (2016). Analyzing multi-agent approaches for the design of advanced interactive and collaborative systems. *Journal of Ambient Intelligence and Smart Environments*, 8(3) :325–346.

[Barceló et al., 2010] Barceló, J. et al. (2010). *Fundamentals of traffic simulation*, volume 145. Springer.

[Barraquand and Latombe, 1990] Barraquand, J. and Latombe, J.-C. (1990). A monte-carlo algorithm for path planning with many degrees of freedom. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1712–1717. IEEE.

[Bellman, 1958] Bellman, R. (1958). On a routing problem. *Quarterly of applied mathematics*, 16(1) :87–90.

[Ben-Akiva et al., 2000] Ben-Akiva, M., Davol, A., Toledo, T., Koutsopoulos, H. N., Burghout, W., Andréasson, I., Johansson, T., and Lundin, C. (2000). Mitsimlab for stockholm enhancements, calibration and validation.

[Bicchi and Pallottino, 2000] Bicchi, A. and Pallottino, L. (2000). On optimal cooperative conflict resolution for air traffic management systems. *IEEE Transactions on Intelligent Transportation Systems*, 1(4) :221–231.

[Bilimoria, 2001] Bilimoria, K. D. (2001). Methodology for the performance evaluation of a conflict probe. *Journal of Guidance, Control, and Dynamics*, 24(3) :444–451.

[Boes, 2014] Boes, J. (2014). *Apprentissage du contrôle de systèmes complexes par l'auto-organisation coopérative d'un système multi-agent : application à la calibration de moteurs à combustion*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Bohlin and Kavraki, 2000] Bohlin, R. and Kavraki, L. E. (2000). Path planning using lazy prm. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 521–528. IEEE.

[Bonakdarian et al., 1998] Bonakdarian, E., Cremer, J., Kearney, J., and Willemsen, P. (1998). Generation of ambient traffic for real-time driving simulation. In *In Image Conference*. Citeseer.

[Bonini et al., 2009] Bonini, D., Dupré, C., and Granger, G. (2009). How erasmus can support an increase in capacity in 2020. In *Proceedings of the 7th International Conference on Computing, Communications and Control Technologies : CCCT*, page 6. International Inst. of Informatics and Statistics (IIIS) Orlando, FL.

[Bonjean et al., 2014] Bonjean, N., Mefteh, W., Gleizes, M. P., Maurel, C., and Migeon, F. (2014). Adelfe 2.0. In *Handbook on agent-oriented design processes*, pages 19–63. Springer.

[Bonnet et al., 2015] Bonnet, J., Gleizes, M.-P., Kaddoum, E., Rainjonneau, S., and Flandin, G. (2015). Multi-satellite mission planning using a self-adaptive multi-agent system. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 11–20. IEEE.

[Borghini et al., 2014] Borghini, G., Arico, P., Graziani, I., Salinari, S., Babiloni, F., Imbert, J.-P., Granger, G., Benhacene, R., Napoletano, L., Terenzi, M., and Pozzi, S. (2014). Analysis of neurophysiological signals for the training and mental workload assessment of ATCos. In *SESAR 2014, 4th SESAR Innovation Days*, Madrid, Spain.

[Bouziat, 2017] Bouziat, T. (2017). *A Cooperative Architecting Procedure for Systems of Systems Based on Self-Adaptive Multi-Agent Systems*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Breil, 2017] Breil, R. (2017). *Système multi-agents pour l'auto-structuration du trafic aérien*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Breil et al., 2016] Breil, R., Delahaye, D., Lapasset, L., and Féron, É. (2016). Multi-agent systems for air traffic conflicts resolution by local speed regulation. In *7th International Conference on Research in Air Transportation (ICRAT 2016)*.

[Brun et al., 2009] Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pages 48–70. Springer.

[Camps et al., 1998] Camps, V., Gleizes, M.-P., and Glize, P. (1998). Une théorie des phénomènes globaux fondée sur des interactions locales. *JP. BARTHÈS, V. CHEVRIER et C. BRASSAC, éditeurs : Systèmes multi-agents–de l'interaction à la socialité–Actes des 6èmes JFIADSMA*, pages 207–220.

[Carpin and Pillonetto, 2005] Carpin, S. and Pillonetto, G. (2005). Motion planning using adaptive random walks. *IEEE Transactions on Robotics*, 21(1) :129–136.

[Chaimatanan et al., 2013] Chaimatanan, S., Delahaye, D., and Mongeau, M. (2013). Strategic deconfliction of aircraft trajectories. In *ISIATM 2013, 2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, pages p–xxxx.

[Cheng et al., 2009] Cheng, B. H., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., et al. (2009). Software engineering for self-adaptive systems : A research roadmap. In *Software engineering for self-adaptive systems*, pages 1–26. Springer.

[Chu et al., 2003] Chu, L., Liu, H. X., Oh, J.-S., and Recker, W. (2003). A calibration procedure for microscopic traffic simulation. In *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, volume 2, pages 1574–1579. IEEE.

[De Berg et al., 2008] De Berg, M., Cheong, O., Van Kreveld, M., and Overmars, M. (2008). *Computational Geometry : Introduction*. Springer.

[de Saint-Exupéry et al., 1931] de Saint-Exupéry, A., Gide, A., and De Coster, G. (1931). *Vol de nuit*, volume 3. Gallimard.

[De Wolf and Holvoet, 2004] De Wolf, T. and Holvoet, T. (2004). Emergence versus self-organisation : Different concepts but promising when combined. In *International workshop on engineering self-organising applications*, pages 1–15. Springer.

[Degas et al., 2016] Degas, A., Arcangeli, J.-P., Trouilhet, S., Calvary, G., Coutaz, J., Lavirotte, S., and Tigli, J.-Y. (2016). Opportunistic composition of human-computer interactions in ambiant spaces. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 998–1005. IEEE.

[Delahaye et al., 2010] Delahaye, D., Peyronne, C., Mongeau, M., and Puechmorel, S. (2010). Aircraft conflict resolution by genetic algorithm and b-spline approximation. In *EIWAC 2010, 2nd ENRI International Workshop on ATM/CNS*, pages pp–71.

[DGAC, 2013] DGAC (2013). *Le temps des ingénieurs de la navigation aérienne , Mémoires techniques, 1945-1985*. DGAC.

[DGAC, 2017] DGAC (2017). *60 ans de contrôle aérien en route*. DGAC.

[DGAC, 2018] DGAC (2018). Rapports annuels sur la sécurité aérienne.

[Di Marzo Serugendo et al., 2011] Di Marzo Serugendo, G., Gleizes, M.-P., and Karageorgos, A. (2011). Self-organising software : From natural to artificial adaptation.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1) :269–271.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., Colorni, A., et al. (1996). Ant system : optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B : Cybernetics*, 26(1) :29–41.

[Dougui, 2011] Dougui, N. E. (2011). *Planification de trajectoires avion : approche par analogie lumineuse.* PhD thesis, Université Paul Sabatier-Toulouse III.

[Durand, 1996] Durand, N. (1996). *Optimisation de trajectoires pour la résolution de conflits aériens en route*. PhD thesis, INPT.

[Durand, 2018] Durand, N. (2018). Constant speed optimal reciprocal collision avoidance. *Transportation research part C : emerging technologies*, 96 :366–379.

[Durand and Alliot, 2009] Durand, N. and Alliot, J.-M. (2009). Ant colony optimization for air traffic conflict resolution. In *ATM Seminar 2009, 8th USA/Europe Air Traffic Management Research and Developpment Seminar*.

[Durand and Barnier, 2015] Durand, N. and Barnier, N. (2015). Does atm need centralized coordination ? autonomous conflict resolution analysis in a constrained speed environment. *Air Traffic Control Quarterly*, 23(4) :325–346.

[ECTL, 2018] ECTL (2018). Atfcm users manual.

[ECTL, 2019] ECTL (2019). Intermediate two-year forecast of service units - may 2019.

[El Esawey and Sayed, 2011] El Esawey, M. and Sayed, T. (2011). Calibration and validation of micro-simulation models of medium-size networks. *Advances in Transportation Studies*.

[Elbanhawi and Simic, 2014] Elbanhawi, M. and Simic, M. (2014). Sampling-based robot motion planning : A review. *Ieee access*, 2 :56–77.

[Erdi, 2008] Erdi, P. (2008). Complex systems : The intellectual landscape. *Complexity Explained*, pages 1–23.

[Erzberger, 2006] Erzberger, H. (2006). Automated conflict resolution for air traffic control. *ICAS*.

[Esteoul, 2019] Esteoul, T. (2019). *Prévision de production de parcs éoliens par systèmes multi-agents auto-adaptatifs*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Ferber, 1995] Ferber, J. (1995). Les systèmes multi-agents : vers une intelligence collective. *InterEditions, Paris*, 322.

[Flynn et al., 2003] Flynn, G., Benkouar, A., and Christien, R. (2003). Pessimistic sector capacity estimation. *EEC Note*, 21(03) :1.

[Gajananan et al., 2013] Gajananan, K., Nantes, A., Miska, M., Nakasone, A., and Prendinger, H. (2013). An experimental space for conducting controlled driving behavior studies based on a multiuser networked 3d virtual environment and the scenario markup language. *IEEE Transactions on Human-Machine Systems*, 43(4) :345–358.

[Georgé et al., 2011] Georgé, J.-P., Gleizes, M.-P., and Camps, V. (2011). Cooperation. In Serugendo, G. D. M., Gleizes, M.-P., and Karageorgos, A., editors, *Self-organising software : From natural to artificial adaptation*, page 193. Springer Science & Business Media.

[Girardet, 2014] Girardet, B. (2014). *Trafic aérien : détermination optimale et globale des trajectoires d'avion en présence de vent*. PhD thesis, Toulouse, INSA.

[Gleizes et al., 1999] Gleizes, M.-P., Camps, V., and Glize, P. (1999). A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In *Fourth European Congress of Systems Science*, pages 20–24.

[Glize, 2001] Glize, P. (2001). L'adaptation des systèmes à fonctionnalité émergente par auto-organisation coopérative. *Hdr, Université Paul Sabatier, Toulouse III*.

[Goerzen et al., 2009] Goerzen, C., Kong, Z., and Mettler, B. (2009). A survey of motion planning algorithms from the perspective of autonomous uav guidance. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pages 65–100. Springer.

[Goldstein, 1999] Goldstein, J. (1999). Emergence as a construct : History and issues. *Emergence*, 1(1) :49–72.

[Guespin-Michel, 2015] Guespin-Michel, J. (2015). *Emancipation et pensée du complexe*. Ed. du Croquant.

[Guys, 2014] Guys, L. (2014). *Planification de trajectoires d'avions sans conflit : fonctions biharmoniques et fonction de navigation harmonique*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2) :100–107.

[Heesbeen et al., 2003] Heesbeen, B., Hoekstra, J., and Valenti Clari, M. (2003). Traffic manager-traffic simulation for validation of future atm concepts. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, page 5368.

[Hilburn, 2004] Hilburn, B. (2004). Cognitive complexity in air traffic control : A literature review. *EEC note*, 4(04).

[Hollander and Liu, 2008] Hollander, Y. and Liu, R. (2008). The principles of calibrating traffic microsimulation models. *Transportation*, 35(3) :347–362.

[Hourdakis et al., 2003] Hourdakis, J., Michalopoulos, P. G., and Kottommannil, J. (2003). Practical procedure for calibrating microscopic traffic simulation models. *Transportation Research Record*, 1852(1) :130–139.

[Hsu et al., 1997] Hsu, D., Latombe, J.-C., and Motwani, R. (1997). Path planning in expansive configuration spaces. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2719–2726. IEEE.

[Jha et al., 2004] Jha, M., Gopalan, G., Garms, A., Mahanti, B. P., Toledo, T., and Ben-Akiva, M. E. (2004). Development and calibration of a large-scale microscopic traffic simulation model. *Transportation Research Record*, 1876(1) :121–131.

[Jonathan et al., 2010] Jonathan, R. S., Peter, N., Ernest, D., Jonathan, R. S., and Jonathan, R. S. (2010). Artificial intelligence : a modern approach. vol. 2.

[Kaddoum, 2011] Kaddoum, E. (2011). *Optimization under Constraints of Distributed Complex Problems using Cooperative Self-Organization*. PhD thesis, Paul Sabatier University Toulouse.

[Kavraki, 1994] Kavraki, L. (1994). *Random networks in configuration space for fast path planning*. PhD thesis, Stanford University.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4) :566–580.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1) :90–98.

[Khatib and Le Maitre, 1978] Khatib, O. and Le Maitre, J. (1978). Dynamic controlof manipulators operating in a complex environment. In *On Theory and Practice of Robots and Manipulators, 3rd CISM-IFToMM Symp*, volume 267.

[Kim and Rilett, 2003] Kim, K.-O. and Rilett, L. (2003). Simplex-based calibration of traffic microsimulation models with intelligent transportation systems data. *Transportation Research Record*, 1855(1) :80–89.

[Ksontini et al., 2015] Ksontini, F., Mandiau, R., Guessoum, Z., and Espié, S. (2015). Affordance-based agent model for road traffic simulation. *Autonomous Agents and Multi-Agent Systems*, 29(5) :821–849.

[Kuchar and Yang, 2000] Kuchar, J. K. and Yang, L. C. (2000). A review of conflict detection and resolution modeling methods. *IEEE Transactions on intelligent transportation systems*, 1(4) :179–189.

[Kuffner and LaValle, 2000] Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect : An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE.

[Kupfer et al., 2013] Kupfer, M., Mercer, J., Cabrall, C. D., and Callantine, T. J. (2013). Designing scenarios for controller in the loop air traffic simulations. In *AIAA Modeling and Simulation Technologies (MST) Conference*, page 5156.

[Latombe, 2012] Latombe, J.-C. (2012). *Robot motion planning*, volume 124. Springer Science & Business Media.

[LaValle, 1998] LaValle, S. M. (1998). Rapidly-exploring random trees : A new tool for path planning. *Report No. TR 98-11, Computer Science Department*.

[Lehouillier, 2015] Lehouillier, T. (2015). *Modèles déterministes et stochastiques pour la résolution de conflits entre aéronefs*. PhD thesis, École Polytechnique de Montréal.

[Lehouillier et al., 2017] Lehouillier, T., Omer, J., Soumis, F., and Desaulniers, G. (2017). Two decomposition algorithms for solving a minimum weight maximum clique model for the air conflict resolution problem. *European Journal of Operational Research*, 256(3) :696–712.

[Lin and Lee, 2015] Lin, C. E. and Lee, C.-J. (2015). Conflict detection and resolution model for low altitude flights. In *Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on*, pages 406–411. IEEE.

[Lind, 2001] Lind, J. (2001). *Iterative software engineering for multiagent systems : the MASSIVE method*. Springer-Verlag.

[Lopez et al., 2018] Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and WieBner, E. (2018). Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE.

[Lorenz, 2000] Lorenz, E. (2000). The butterfly effect. *World Scientific Series on Nonlinear Science Series A*, 39 :91–94.

[Lozano-Perez, 1983] Lozano-Perez, T. (1983). Spatial planning : A configuration space approach. *IEEE transactions on computers*, 2 :108–120.

[Lozano-Pérez and Wesley, 1979] Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10) :560–570.

[Ma and Abdulhai, 2002] Ma, T. and Abdulhai, B. (2002). Genetic algorithm-based optimization approach and generic tool for calibrating traffic microscopic simulation parameters. *Transportation research record*, 1800(1) :6–15.

[Maas et al., 2016] Maas, J., Sunil, E., Ellerbroek, J., and Hoekstra, J. (2016). The effect of swarming on a voltage potential-based conflict resolution algorithm. In *submitted to the 7th International Conference on Research in Air Transportation*.

[Mandiau et al., 2008] Mandiau, R., Champion, A., Auberlet, J.-M., Espié, S., and Kolski, C. (2008). Behaviour based on decision matrices for a coordination between agents in a urban traffic simulation. *Applied Intelligence*, 28(2) :121–138.

[Martin, 2013] Martin, C. (2013). *La gestion de la charge mentale des contrôleurs aériens en-route : apports de l'eye-tracking dans le cadre du projet européen SESAR*. PhD thesis, Universite Toulouse 2 Le Mirail (UT2 Le Mirail).

[Mazer et al., 1998] Mazer, E., Ahuactzin, J. M., and Bessiere, P. (1998). The ariadne's clew algorithm. *Journal of Artificial Intelligence Research*, 9 :295–316.

[Mouillet, 2017] Mouillet, V. (2017). User manual for the base of aircraft data (bada) revision 3.14. Technical report, tech. rep., EUROCONTROL Experimental Centre, Brétigny, France.

[Noël, 2012] Noël, V. (2012). *Component-based software architectures and multi-agent systems : mutual and complementary contributions for supporting software development*. PhD thesis.

[OACI, 2012] OACI (2012). rocédures pour les services de la navigation aérienne (pans-atm) : Gestion du trafic aérien, 2012.

[Olive, 2006] Olive, X. (2006). Résolution de conflits par algorithmes stochastiques parallèles. *Mémoire de DEA, École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse*.

[Olstam and Espié, 2007] Olstam, J. and Espié, S. (2007). Combination of autonomous and controlled vehicles in driving simulator scenarios. In *Road Safety and Simulation (RSS2007)*.

[Olstam et al., 2011] Olstam, J., Espié, S., Måardh, S., Jansson, J., and Lundgren, J. (2011). An algorithm for combining autonomous vehicles and controlled events in driving simulator experiments. *Transportation research part C : emerging technologies*, 19(6) :1185–1201.

[Overmars, 1992] Overmars, M. H. (1992). *A random approach to motion planning*, volume 92. Unknown Publisher.

[Paglione et al., 2003] Paglione, M., Oaks, R., and Bilimoria, K. (2003). Methodology for generating conflict scenarios by time shifting recorded traffic data. In *AIAA's 3rd Annual Aviation Technology, Integration, and Operations (ATIO) Forum*, page 6724.

[Pallottino et al., 2002] Pallottino, L., Feron, E. M., and Bicchi, A. (2002). Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE transactions on intelligent transportation systems*, 3(1) :3–11.

[Passos et al., 2011] Passos, L. S., Rossetti, R. J., and Kokkinogenis, Z. (2011). Towards the next-generation traffic simulation tools : a first appraisal. In *6th Iberian Conference on Information Systems and Technologies (CISTI 2011)*, pages 1–6. IEEE.

[Perles, 2017] Perles, A. (2017). *An adaptive multi-agent system for the distribution of intelligence in electrical distribution networks : state estimation*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Peyronne, 2012] Peyronne, C. (2012). *Modélisation mathématique et résolution automatique de conflits par algorithmes génétiques et par optimisation locale continue*. PhD thesis, Université Paul Sabatier-Toulouse III.

[Prandini et al., 2012] Prandini, M., Putta, V., and Hu, J. (2012). Air traffic complexity in future air traffic management systems. *Journal of Aerospace Operations*, 1(3) :281–299.

[Prats et al., 2017] Prats, X., Barrado, C., Vidosavljevic, A., Delahaye, D., Netjasov, F., and Crnogorac, D. (2017). Assessing atm performance with simulation and optimisation tools the apache project. In *Seventh SESAR Innovation Days*.

[Preparata and Shamos, 2012] Preparata, F. P. and Shamos, M. I. (2012). *Computational geometry : an introduction*. Springer Science & Business Media.

[Rantanen and Wickens, 2012] Rantanen, E. M. and Wickens, C. D. (2012). Conflict resolution maneuvers in air traffic control : Investigation of operational data. *The International Journal of Aviation Psychology*, 22(3) :266–281.

[Rantrua, 2017] Rantrua, A. (2017). *Simulation massive de monde virtuelpar système multi-agent auto-adaptatif*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Ratrout and Rahman, 2009] Ratrout, N. T. and Rahman, S. M. (2009). A comparative analysis of currently used microscopic and macroscopic traffic simulation software. *The Arabian Journal for Science and Engineering*, 34(1B) :121–133.

[Reif, 1979] Reif, J. H. (1979). Complexity of the mover's problem and generalizations. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 421–427. IEEE.

[Rimon and Koditschek, 1992] Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on robotics and automation*, 8(5) :501–518.

[Ritchie III, 2019] Ritchie III, J. (2019). Enact–generating aircraft encounters using a spherical model. *USA/Europe ATM R&D Seminar*.

[Rong et al., 2002] Rong, J., Bokadia, S., Shandy, S., and Valasek, J. (2002). Hierarchical agent based system for general aviation cd&r under free flight. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4553.

[Roussos et al., 2010] Roussos, G., Dimarogonas, D. V., and Kyriakopoulos, K. J. (2010). 3d navigation and collision avoidance for nonholonomic aircraft-like vehicles. *International Journal of Adaptive Control and Signal Processing*, 24(10) :900–920.

[Serugendo et al., 2006] Serugendo, G. D. M., Irit, M.-P., and Karageorgos, A. (2006). Self-organisation and emergence in mas : An overview. *Informatica*, 30(1).

[Shandy and Valasek, 2001] Shandy, S. and Valasek, J. (2001). Intelligent agent for aircraft collision avoidance. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4055.

[Signor et al., 2004] Signor, D., Davis, P., Lozito, S., Andre, A., Sweet, D., and Wallace, E. (2004). Efficient air traffic scenario generation. In *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, page 6399.

[Siméon et al., 2000] Siméon, T., Laumond, J.-P., and Nissoux, C. (2000). Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6) :477–493.

[Souissi et al., 2013] Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N., and Feyzeau, P. (2013). Path planning : A 2013 survey. In *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pages 1–8. IEEE.

[Stentz, 1994] Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310–3317. IEEE.

[Svestka and Overmars, 1995] Svestka, P. and Overmars, M. H. (1995). Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 2, pages 1631–1636. IEEE.

[Toledo et al., 2003] Toledo, T., Koutsopoulos, H. N., Davol, A., Ben-Akiva, M. E., Burghout, W., Andréasson, I., Johansson, T., and Lundin, C. (2003). Calibration and validation of microscopic traffic simulation tools : Stockholm case study. *Transportation Research Record*, 1831(1) :65–75.

[Tönnis, 2007] Tönnis, M. (2007). The tangible car-rapid intuitive traffic scenario generation in a hybrid table-top and virtual environment. In *Proceedings of the 4th International Workshop on the Tangible Space Initiative (6th International Symposium on Mixed and Augmented Reality)*.

[Tönnis and Klinker, 2009] Tönnis, M. and Klinker, G. (2009). A collaborative table-top platform for discussion and development of traffic scenarios with human behavior.

[Tran Dac, 2004] Tran Dac, H. (2004). *Sectorisation contrainte de l'espace aérien*. PhD thesis, Compiègne.

[Udupa, 1977] Udupa, S. M. (1977). *Collision detection and avoidance in computer controlled manipulators*. PhD thesis, California Institute of Technology.

[Van Den Berg et al., 2011] Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer.

[Vanaret, 2015] Vanaret, C. (2015). *Hybridation d'algorithmes évolutionnaires et de méthodes d'intervalles pour l'optimisation de problèmes difficiles*. PhD thesis, École Doctorale Mathématiques, Informatique et Télécommunications (Toulouse) ; 142547247.

[Verstaevel et al., 2018] Verstaevel, N., Georgé, J.-P., Bernon, C., and Gleizes, M.-P. (2018). A self-organized learning model for anomalies detection : Application to elderly people. In *2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 70–79. IEEE.

[Wassink et al., 2006] Wassink, I., Van Dijk, B., Zwiers, J., Nijholt, A., Kuipers, J., and Brugman, A. (2006). In the truman show : Generating dynamic scenarios in a driving simulator. *IEEE intelligent systems*, 21(5) :28–32.

[Weiss, 1999] Weiss, G. (1999). *Multiagent systems : a modern approach to distributed artificial intelligence*. MIT press.

[Wiener, 1948] Wiener, N. (1948). *Cybernetics or Control and Communication in the Animal and the Machine*. Technology Press.

[Zeghal, 1994] Zeghal, K. (1994). *Vers une théorie de la coordination d'actions. Application à la navigation aérienne*. PhD thesis, Paris 6.

Liste des Figures

Liste des Tables

Liste des Algorithmes

Glossaire

Nomenclature Aéronautique

NM nautical mile. 8, 19, 144, 145, 155, 158, 159, 161, 171, 211

waypoint Point de croisement en français, point défini par une position géographique, des coordonnées de latitude, de longitude et dans la plupart des cas d'altitude, et correspond à un point de passage pour les routes aéronautiques. 11, 13, 15, 27, 28, 29, 103, 144, 209

hPa hectopascal. 13

ft feet. 13, 19, 178, 179

Flight Plan Plan de vol d'un avion, voir 1.2.2. 13, 17, 28, 29

conflit aéronautique Perte de séparation réglementaire entre deux avions. 144, 145, 146, 163, 212

ACC Area Control Center. 7, 8

VFR Visual Flight Rules. 9, 15

IFR Instrument Flight Rules. 9, 156

ICAO International Civil Aviation Organization. 9

CRNA Centre en Route de la Navigation Aérienne. 9, 10, 11, 209

UTA Upper Traffic Area. 9, 11, 209

SNA Services de la Navigation Aérienne. 9, 10, 209

ASM Air Space Management. 13, 14, 16, 17

ATM Air Traffic Management. 14, 42

ATS Air Traffic Services. 14, 16

ATFM Air Traffic Flow Management. 14, 16

BADA Base of Aircraft DAta. 28, 30, 31, 34

ASTOR Aircraft Simulation for Traffic Operations Research. 28, 31, 34

CPA Closest Point of Approach. 30, 69

ADS-B système de surveillance dans lequel chaque avion détermine sa position gps, et la diffuse par radio avec d'autres données qui le caractérisent (identifiant, vitesse,entre autres). 144, 145

Nomenclature de Planification de trajectoire

C espace de configuration. viii, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 74, 209, 220

M_i entité mobile numéro i. 51, 53, 220

M ensemble des n mobiles. 51, 143

$CObs$ sous-partie de l'espace de configuration C contenant uniquement les obstacles. 52, 209

q_{init} configuration initiale dans l'espace de configuration. 52, 53, 58, 59, 60, 61, 209

q_{goal} configuration voulue dans l'espace de configuration. 52, 53, 58, 59, 60, 61, 209

W espace dans lequel peut évoluer le mobile, *Workspace* en anglais. 52

O ensemble des m obstacles. 52, 55, 56

O_j obstacle numéro j. 52

q_i une configuration du mobile M_i de C_i. 52, 54, 58

$Cfree$ sous-partie de l'espace de configuration C ne contenant pas d'obstacles. 52, 53, 54, 55, 56, 58, 59, 60, 62

C_i espace de configuration du mobile M_i. 220

PRM Probabilistic RoadMap Tree. 58, 59, 60, 198

RRT Rapidly exploring Random Tree. 60, 198

MILP Mixed-Integer Linear Programing. 65

Nomenclature AMAS

AGATS Autonomous GenerAtion of Traffic Simulations. 1, 2, 3, 78, 93, 94, 95, 96, 97, 98, 106, 120, 134, 139, 140, 143, 144, 146, 148, 159, 173, 175, 177, 178, 179, 180, 186, 187, 190, 191, 192, 195, 196, 197, 198, 210, 213, 217

CAAMAS Collision Avoidance Adaptive Multi-Agent System. 1, 3, 140, 143, 144, 148, 151, 152, 153, 156, 157, 158, 166, 170, 171, 172, 173, 174, 175, 188, 190, 191, 192, 196, 198, 211, 215

AGEAS Autonomous GEneration of trAffic Scenarios. 2, 3, 140, 177, 178, 182, 183, 186, 187, 188, 190, 191, 192, 196, 212

AMAS4Opt Adaptive Multi-Agent System for Optimisation. 2

EVAA Environnement Virtuel Auto-Adaptatif. 2, 78, 89, 90, 91, 94, 95, 104, 110, 139, 144, 179, 183, 184, 195, 198, 210, 217

AMAS Adaptive Multi-Agent System. 78, 85, 86, 87, 88, 89, 90, 93, 190, 192, 193, 194, 198

SNC Situation de Non Coopération d'un agent. 87, 134, 136, 137, 138, 180

ADELFE Atelier de Développement de Logiciels à Fonctionnalité Emergente. 88

Nomenclature d'AGATS

$Crit_{char,i}$ Criticité de la i caractéristique de l'*Agent Situation*. 98, 99, 100

$Crit_{sit}$ Criticité de l'*Agent Situation*. 99, 105

$Crit_{traj,comp}$ Criticité de l'*Agent Trajectoire* en fonction de la complétude de sa trajectoire. 101, 103, 107, 108, 193, 210

$Crit_{part,comp}$ Criticité de l'*Agent Partie* en fonction de la complétude de sa Partie de Trajectoire. 103, 104, 106, 108

$Crit_{traj,traff}$ Criticité de l'*Agent Trajectoire* en fonction du réalisme comportemental de sa trajectoire. 103, 109, 210

$Crit_{part,traff}$ Criticité de l'*Agent Partie* en fonction du réalisme comportemental de sa Partie de Trajectoire. 103, 104, 106, 107, 109, 116

$Crit_{traj,feas}$ Criticité de l'*Agent Trajectoire* en fonction de la faisabilité de sa trajectoire. 103

$Crit_{part,feas}$ Criticité de l'*Agent Partie* en fonction de la faisabilité de sa Partie de Trajectoire. 103, 104, 109, 119, 120

$Crit_{traj}$ Criticité de l'*Agent Trajectoire*. 103, 105

$Crit_{part}$ Criticité de l'*Agent Partie*. 104, 109, 111, 113, 114

$Crit_{extr}$ Criticité de l'*Agent Extrémité*. 115

$Crit_{extr,plan}$ Criticité de l'*Agent Extrémité* par rapport à son planning. 115, 116

$Crit_{extr,char}$ Criticité de l'*Agent Extrémité* par rapport à ses caractéristiques. 115, 116

$Crit_{extr,char,i}$ Criticité de l'*Agent Extrémité* par rapport à sa $i$ème caractéristique. 116, 211

Nomenclature de l'Agent EM

M_i *Agent entité mobile* M_i. 117, 118, 126, 128, 129, 143, 149, 221, 222

Zp_i Zone de perception de M_i. 117, 118, 128, 129, 130, 132, 144, 221, 222

M_j Entité mobile dans la zone de perception de l'*agent entité mobile* M_i. $M_j \in Zp_i$. 126, 128, 129, 130, 221, 222

Ac_i Ensemble des n actions de l'*agent entité mobile* M_i. $Ac_i = \left\{Ac_{i,k}\right\}_{0<k\leq n}$. 117, 130, 146, 211, 221

$Ac_{i,k}$ Action numéro k de l'*agent entité mobile* M_i. Elle fait partie de l'ensemble des actions Ac_i. 117, 118, 119, 120, 121, 124, 126, 132, 133, 135, 149, 150, 221, 222

$\overrightarrow{p_{i,k}}$ vecteur position du mobile M_i s'il effectue l'action $Ac_{i,k}$. 121, 122, 124, 126, 127, 128, 147

$\overrightarrow{v_{i,k}}$ vecteur vitesse du mobile M_i s'il effectue l'action $Ac_{i,k}$. 122, 123, 124, 125, 127, 128, 147

$\overrightarrow{p_j}$ vecteur position du mobile $M_j \in Zp_i$. 126, 127, 128, 134, 147, 149

$\overrightarrow{v_j}$ vecteur vitesse du mobile $M_j \in Zp_i$. 127, 128, 134, 147, 149

$Ac_{i,k_{decided}}$ Action $Ac_{i,k}$ décidée par l'*agent entité mobile* M_i. Elle fait partie de l'ensemble des actions Ac_i. 118, 133

$d_{min,i,j,k}$ distance minimale entre un mobile M_i et un mobile M_j si le mobile M_i effectue l'action $Ac_{i,k}$. 126, 127, 128, 147

$t_{min,i,j,k}$ temps pour lequel la distance entre un mobile M_i et un mobile M_j si le mobile M_i effectue l'action $Ac_{i,k}$ est minimale. 126, 127, 128, 129, 147, 148

$Crit_i$ Criticité de l'*agent entité mobile* M_i. 117, 119, 120, 135, 149, 150

$Crit_{i,coll}$ Criticité de l'*agent entité mobile* M_i par rapport à la collision. 119, 120

$Crit_{i,k,coll}$ Criticité de l'*agent entité mobile* M_i par rapport à la collision s'il effectue l'action $Ac_{i,k}$. 119, 120, 124, 126, 133, 134, 149

$Crit_{i,j,k,coll}$ Criticité de collision d'un *agent entité mobile* M_i avec une entité mobile M_j s'il effectue l'action $Ac_{i,k}$. 119, 124, 126, 134, 149, 222

$Crit_{1,j,k}$ Première composante de $Crit_{i,j,k,coll}$ par rappor à la distance. 119, 126, 128, 129, 147, 149, 170, 211, 222

$Crit_{2,j,k}$ Deuxième composante de $Crit_{i,j,k,coll}$ par rapport au temps. 119, 126, 128, 129, 147, 148, 149, 158, 161, 170, 211

$Crit_{i,traj}$ Criticité de l'*agent entité mobile* M_i par rapport au suivi de trajectoire. 119, 120, 149, 198

$Crit_{i,k,traj}$ Criticité de l'*agent entité mobile* M_i par rapport au suivi de trajectoire s'il effectue l'action $Ac_{i,k}$. 119, 120, 133, 134, 149, 193, 222

$Crit_{3,k}$ Première composante de $Crit_{i,k,traj}$. 119, 120, 121, 122, 149

$Crit_{4,k}$ Deuxième composante de $Crit_{i,k,traj}$. 119, 120, 121, 122, 149

$Crit_{5,k}$ Troisième composante de $Crit_{i,k,traj}$. 119, 120, 121, 122, 149

$Crit_j$ Criticité de l'*agent entité mobile* $M_j \in Zp_i$. 134, 135, 149, 150

$Crit_{i,k}$ Criticité de l'*agent entité mobile* M_i s'il effectue l'action $Ac_{i,k}$. 135, 150

$Crit_{1,j,k,hor}$ Composante horizontale de $Crit_{1,j,k}$. 147, 149, 158, 159, 160, 161

$Crit_{1,j,k,ver}$ Composante verticale de $Crit_{1,j,k}$. 147, 149

Agent EM Agent Entité Mobile. 95, 97, 101, 103, 104, 105, 109, 114, 115, 116, 117, 118, 126, 132, 134, 135, 136, 138, 139, 140, 143, 144, 145, 148, 150, 151, 152, 153, 155, 157, 158, 161, 162, 164, 165, 166, 170, 171, 172, 173, 174, 175, 179, 180, 186, 188, 190, 191, 192, 193, 194, 195, 196, 197, 198

Agent EM$_i$ Agent Entité Mobile i. 117, 118, 119, 120, 121, 122, 123, 124, 126, 127, 129, 130, 131, 132, 133, 135, 137, 138, 143, 144, 145, 146, 148, 149, 150, 211, 217

Agent EM$_j$ Agent Entité Mobile j, perçu par l'Agent EM$_i$. 117, 118, 119, 125, 126, 127, 130, 132, 133, 134, 135, 137, 138, 148, 149, 150

Module Perception Module Perception de l'Agent EM. 130, 132, 133, 136, 195, 196, 217

Module Voisin Module Voisin de l'Agent EM. 130, 131, 132, 133, 136, 137, 138, 148, 217

Module Action Module Action de l'Agent EM. 130, 132, 133, 134, 137, 138, 148, 149, 158, 170, 171, 191, 196, 217

Module Décision Module Décision de l'Agent EM. 130, 132, 133, 134, 135, 137, 138, 148, 149, 150, 217