UNIVERSITE DE NICE-SOPHIA ANTIPOLIS
ECOLE DOCTORALE STIC
Sciences et Technologies de l'Information et de la Communication

T H È S E

pour obtenir le titre de

Docteur ès Sciences
de l'université de Nice-Sophia Antipolis

Discipline : Informatique

présentée et soutenue par **Michaël SOULIGNAC**

Planification de trajectoire en présence de courants

Application aux missions de drones

Thèse dirigée par **Michel Rueher**
Soutenue le 10 Février 2009

Jury :

M. Michel Rueher	Directeur de thèse	Professeur Université de Nice Sophia Antipolis
M. Malik Ghallab	Rapporteur	Délégué Général à la Recherche et au Transfert pour l'Innovation INRIA
M. Luc Jaulin	Rapporteur	Professeur ENSIETA
M. Jean-Paul Laumond	Rapporteur	Directeur de recherche LAAS/CNRS
M. Jean-Pierre Merlet	Examinateur	Directeur de recherche INRIA
M. Patrick Taillibert	Examinateur	Ingénieur Expert THALES Aerospace

À Aurélie,
À Amandine.

Remerciements

Tout d'abord, je voudrais remercier MM. Patrick Taillibert et Michel Rueher qui ont cru en moi, et qui, il y a un peu plus de trois ans, ont décidé de vivre cette aventure scientifique avec moi. Les nombreux échanges que nous avons pu avoir ont fortement participé à améliorer le contenu de cette thèse, en termes de rigueur et de pédagogie. Patrick a aussi su me "challenger" (comme il le dit si bien !) tout au long de ce périple, et m'a constamment poussé à innover, dans des moments où j'aurais pu me satisfaire d'un état moins avancé de ces travaux.

Je profite de ces remerciements pour faire un petit clin d'oeil à l'équipe du laboratoire d'Intelligence Artificielle de THALES, à laquelle j'ai eu le plaisir d'appartenir pendant la durée de ma thèse. Les différents stagiaires et doctorants qui s'y sont succédés m'ont tous aidé à leur manière, certains en apportant simplement leur bonne humeur, d'autres en faisant preuve d'un esprit critique (mais toujours constructif) qui m'a permis de progresser.

Ensuite, je voudrais témoigner toute ma reconnaissance envers les membres du jury, et particulièrement les rapporteurs, pour le temps et la confiance qu'ils m'ont accordés, et pour la qualité de leurs commentaires.

Enfin, je voudrais chaleureusement exprimer toute ma gratitude à ma femme, sans qui cette thèse n'aurait certainement jamais abouti. Son soutien et sa bienveillance constante m'ont permis de tenir bon dans les moments de doute, et aller de l'avant. Et comme le veut la tradition, je tiens à la remercier pour ce beau (mais quelque peu agité !) bébé, qui se reconnaîtra peut être un jour en lisant ces quelques lignes.

Egalement merci à la société THALES et l'ARNT qui ont co-financé ces travaux de thèse dans le cadre d'une contrat CIFRE.

Table des matières

Introduction

1 Problématique

Depuis les années 1980, des robots, de type très divers, sont utilisés de façon grandissante pour assister l'homme dans des tâches répétitives, longues ou dangereuses. Ils se divisent en deux grandes catégories :

- Les robots mobiles, utilisés pour des tâches d'exploration (fig. 1a), de surveillance (fig. 1b), d'assistance à la personne (fig. 1c) ou encore de loisirs (fig. 1d). Généralement de petite taille, ils disposent d'une réserve d'énergie limitée, de capacités de perception (capteurs de proximité, caméra, etc.) et éventuellement d'un modèle de l'environnement.
- Les robots industriels, utilisés dans des chaînes de production, pour effectuer des tâches nécessitant précision ou puissance, généralement de façon répétée. Ils se présentent le plus souvent sous la forme de bras manipulateurs de taille imposante, ayant un grand nombre de degrés de libertés. Dans cette catégorie se trouvent les robots de type *Kuka*, très répandus dans le domaine automobile (fig. 1e).

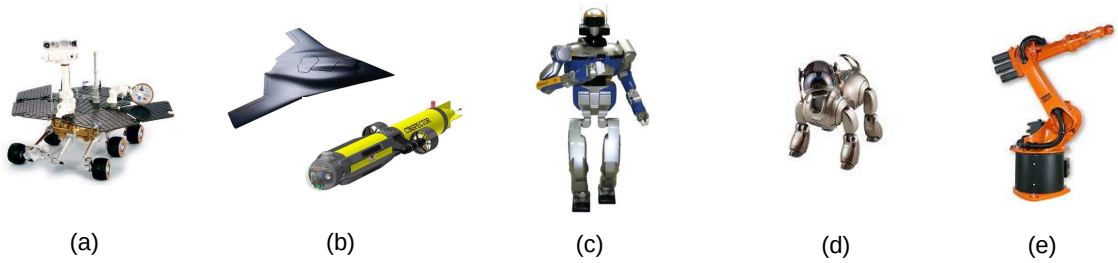

FIG. 1 – Quelques applications de la robotique
(a) robots d'exploration (*Spirit* de la NASA) ; (b) drones : aériens (*Neuron* de Dassault), sous-marins (*C'inspector* de Kongsberg) ; (c) robots humanoïdes (*HRP-2* de Kawada) ; (d) robots-compagons (*Aibo* de Sony) ; (e) robots industriels (*Kuka*).

Quelle que soit sa nature, un robot peut être modélisé comme un mobile ponctuel évoluant dans l'*espace des configurations* [LP83], noté $\mathcal{C}$. Cet espace est de dimension m, chaque dimension étant associée à un degré de liberté du robot, comme illustré ci-après.

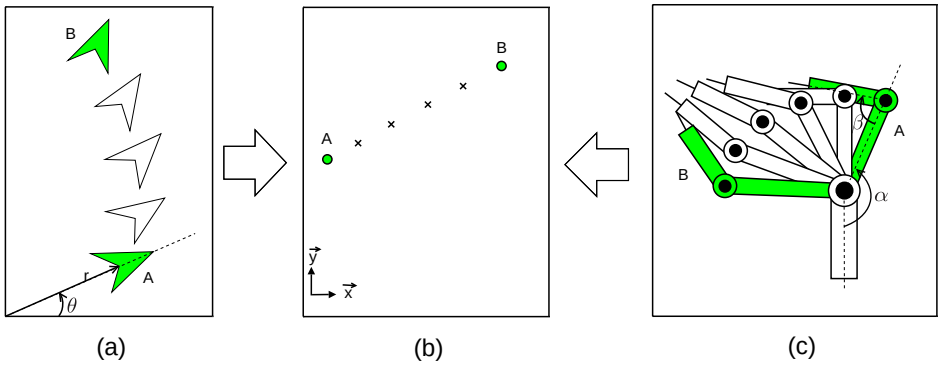

FIG. 2 – L'espace des configurations
(b) Espace des configurations associé à : (a) un drone volant à altitude constante et (c) un robot manipulateur à deux articulations. Dans (a), les dimensions x et y représentent les coordonnées polaires r et θ du drone. Dans (c), elles représentent les positions angulaires α et β des segments du bras.

Dans cette thèse, nous nous intéressons uniquement au cas $m = 2$. Même dans le cas d'une dimension aussi faible, le concept d'espace des configurations permet de modéliser des problèmes de nature très différente (aussi bien le mouvement d'un bras manipulateur à deux articulations que celui d'un drone).

Certaines configurations de $\mathcal{C}$ sont physiquement inatteignables. Ces configurations forment des régions interdites appelées *obstacles*. Elles peuvent matérialiser une limite associée à un degré de liberté (angle de braquage maximal, par exemple), ou une collision avec un autre objet de l'environnement. Si ce dernier est également en mouvement, on parle d'obstacle mobile, sinon d'obstacle fixe. A tout instant, le sous-ensemble de $\mathcal{C}$ regroupant tous les obstacles est noté $\mathcal{C}_{obst}$. Son complémentaire est appelé espace valide et noté $\mathcal{C}_{valide}$.

Dans $\mathcal{C}$, on peut également définir un courant $\vec{w}$ s'ajoutant à la vitesse du robot (provoquant un phénomène de dérive). Ce courant peut par exemple matérialiser un vent (pour les robots aériens) ou un courant marin (pour les robots marins ou sous-marins) présent dans l'environnement initial, éventuellement variable dans le temps ou l'espace.

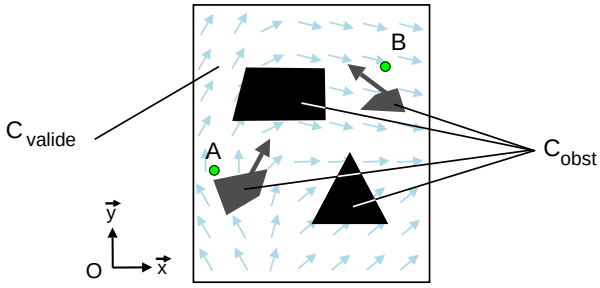

FIG. 3 – Ajout d'obstacles et de courants
Espace des configurations de la figure 2, auquel a été ajouté deux obstacles fixes (en noir), deux obstacles mobiles (en gris) et des courants (en bleu).

Grâce à cette abstraction, la planification d'une tâche pour un robot, aussi variée soit-elle (de la mission d'exploration à l'usinage d'une pièce), peut être reformulée en une planification de trajectoire.

Plus précisément, étant données une configuration initiale A (l'état initial du robot) et une configuration finale B (l'état souhaité du robot), le problème de planification de trajectoire consiste à calculer simultanément un chemin dans $\mathcal{C}_{valide}$ reliant A et B et la vitesse du mobile, à tout instant, sur ce chemin. Cette trajectoire doit minimiser un critère, qui varie énormément selon le domaine d'application. Les critères les plus classiques sont le temps écoulé, la distance parcourue ou encore l'énergie consommée. Mais on peut également avoir des critères très spécifiques au domaine. Pour les missions de surveillance impliquant des drones, par exemple, on peut chercher à minimiser la probabilité de détection de l'engin, en limitant l'exposition de celui-ci dans des zones dites "à risque" (portée d'un radar, par exemple).

Assez naturellement, le problème de planification de trajectoire est très étudié depuis les débuts de la robotique. Par conséquent, de très nombreuses approches ont été proposées. Ces approches, résumées dans la première partie de cette thèse, sont pour la plupart très performantes et garantissent des temps de réponse très court. Toutefois, la prise en compte des courants n'a été que très récemment étudiée (depuis les années 2000). De par la relative jeunesse de cette nouvelle branche, les approches proposées dans la littérature sont peu nombreuses et assez limitées.

Or, une classe de robots est particulièrement sensible à la présence de courants : les drones, ou avions sans pilote. Cette classe représente une part de plus en plus importante dans la robotique mobile, tant dans les investissements que dans la multiplicité de leurs applications.

Historiquement, les drones ont d'abord été utilisés dans un contexte militaire, pour des missions de reconnaissance et de surveillance. Ils permettent en effet de ne pas engager la vie de pilotes dans des zones hostiles, mais aussi d'obtenir de bien meilleures performances dans la conduite de la mission, en terme de d'endurance ou de précision. Ces avantages stratégiques présagent que les drones vont être, dans un moyen terme, massivement déployés sur les champs de d'opérations, comme l'illustre la figure 4.

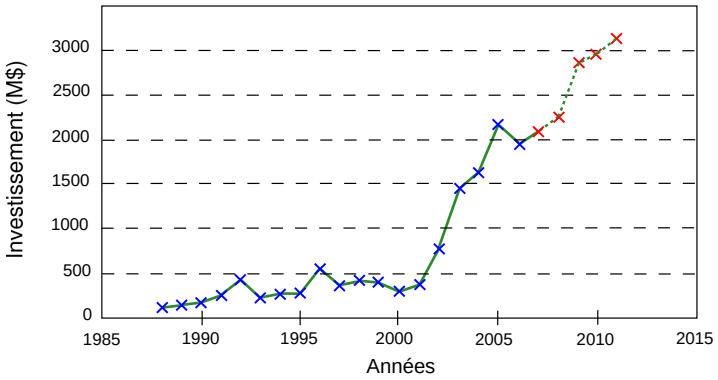

FIG. 4 – Investissement grandissant dans le domaine des drones
Budget (croix bleues = effectif, croix rouges : prévisionnel) de la défense américaine dans le domaine des drones, tiré de [Zal06].

Aujourd'hui, l'utilisation des drones s'étend largement au domaine civil. A titre illustratif, pendant des incendies de grande ampleur en Californie, en octobre 2007, la NASA a déployé des drones de type Ikhana ("intelligent" en Indien) pour apprécier la progression des fronts de flammes et effectuer différentes mesures [Sch08]. Les résultats étaient communiqués en temps réel sur Internet, et visualisables sur le logiciel GoogleEarth. Et cette application civile est loin d'être la seule. De façon non exhaustive, nous pouvons notamment citer : la surveillance côtière (douane, immigration), l'inspection d'infrastructures (pipelines, lignes électriques), la cartographie (recherche de minerais, d'hydrocarbures), le contrôle du trafic, etc.

Dans toutes ces missions, de par leur faible vitesse et leur petite taille, les drones sont particulièrement sensibles aux courants.

En effet, les courants peuvent significativement ralentir le drone, augmentant ainsi sa consommation d'énergie ou son temps de parcours :

- Une consommation d'énergie excessive peut conduire à une rupture de ressources en pleine mission, et donc à l'immobilisation (voire au crash) du drone en terrain hostile.
- Une augmentation importante du temps de parcours peut quant à elle compromettre les conditions d'atterrissage du drone ou la réalisation d'objectifs temporellement contraints.

Ces deux problèmes sont considérablement amplifiés dans le cas de missions de longue durée, où il devient crucial d'anticiper les changements de courants dans le temps.

De plus, si les courants deviennent forts (c'est à dire plus rapides que le drone lui-même), ceux-ci peuvent mener à une dérive incontrôlable du drone. Une première conséquence est de rendre certaines zones de l'environnement inaccessibles, notamment des zones que le drone devait survoler, ce qui peut aboutir à l'échec d'une partie voire de la totalité de la mission. Une conséquence plus désastreuse concerne la survie même du drone : il est envisageable que le courant porte le drone vers un obstacle, entraînant une collision inévitable.

Toutefois, l'ensemble de ces problématiques est aujourd'hui assez peu traité dans la littérature. Comme nous le verrons tout au long de cette thèse, les quelques méthodes existantes ne sont pas satisfaisantes en termes de fiabilité ou de performances. Pour ces raisons, nous avons décidé de focaliser cette thèse sur ces aspects.

2 Organisation de la thèse

Cette thèse traite du problème de planification *prévisionnelle* de trajectoire entre deux points d'un environnement de dimension 2 contenant des courants. L'aspect prévisionnel de la planification correspond au fait que l'environnement est supposé totalement connu à l'avance, ces connaissances étant issues de prévisions météorologiques (pour les courants) ou de trafic (pour les obstacles). Cette type de planification s'oppose à la planification *réactive* d'un robot qui découvrirait les caractéristiques de l'environnement au fur et à mesure de son déplacement.

Le cadre applicatif de cette thèse est la préparation de missions pour véhicules aériens autonomes, ou drones.

L'organisation de cette thèse est la suivante :

1. **Etat de l'art**

   Cette partie résume l'ensemble des résultats en planification de mouvements depuis les années 60 jusqu'à aujourd'hui (les derniers résultats recensés datent de mi-2008). Elle est divisée en trois chapitres, qui retracent l'évolution des techniques de planification, de la plus simple à la plus évoluée.

   - Le **chapitre 1** présente les techniques résolvant le problème le plus simple, la planification de chemin, qui ne tient compte que des obstacles fixes.
   - Le **chapitre 2** complexifie un peu le problème en introduisant des obstacles mobiles. On parle alors de planification de trajectoire.
   - Enfin, le **chapitre 3** ajoute la présence de courants variables, dans le temps et dans l'espace. Ce chapitre met en évidence que les méthodes existantes ne sont pas satisfaisantes dans le cas de courants forts ou variables dans le temps.

2. **Contributions principales**

   Cette partie contient les contributions majeures de la thèse. Les techniques introduites dans cette partie viennent pallier les manques ou les limitations des techniques existantes, présentées dans de chapitre 3. Plus précisément :

– Le **chapitre 4** introduit une méthode de planification de trajectoire capable de gérer les courants forts, c'est à dire les courants plus rapides que le drone. En présence de tels courants, les techniques proposées dans la littérature ne sont pas fiables. En effet, elles peuvent fournir soit des chemins invalides (c'est à dire physiquement irréalisables), soit aucun chemin, même si des solutions valides existent.

  Notre nouvelle approche, appelée la *propagation d'onde coulissante* [STR08a] permet d'augmenter significativement le succès de planification (c'est à dire la capacité à trouver une solution valide si elle existe) par rapport aux techniques existantes. La figure 5 ci après illustre ce point.

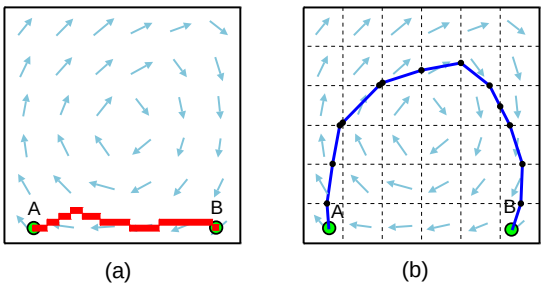

FIG. 5 – Planification de trajectoire en présence de courants forts
(a) Chemin (en rouge) obtenu par une méthode de la littérature (Fast Marching, proposée dans [PPPL05]). Ce chemin va contre des courants qui sont beaucoup trop forts, ce qui est physiquement irréalisable ; (b) Chemin (en bleu foncé) obtenu par la propagation d'onde coulissante. Ce chemin, contournant le tourbillon, est intégralement valide.

– Le **chapitre 5** introduit une méthode de planification de trajectoire capable de gérer les courants variables dans de temps, ce qui n'est possible, à notre connaissance, avec aucune méthode de la littérature.

  Si nous faisons l'analogie avec la planification d'itinéraire routier, nous retrouvons une problématique similaire : quelle est la meilleure heure de départ, compte tenu des prévisions de trafic ? Nous sommes régulièrement confrontés à cette question, mais pour le moment, nous réalisons cette tâche de planification manuellement. C'est également généralement le cas des opérateurs dans le domaine des drones, c'est pourquoi nous proposons dans ce chapitre une technique pour l'automatiser.

  Cette nouvelle technique, appelée *propagation d'onde symbolique* [STR09], calcule la meilleure date de départ pour le drone pour minimiser son temps de parcours, en anticipant les changements de courants dans le temps. La figure 6 fournit un exemple de problème ainsi que la solution calculée par notre approche.

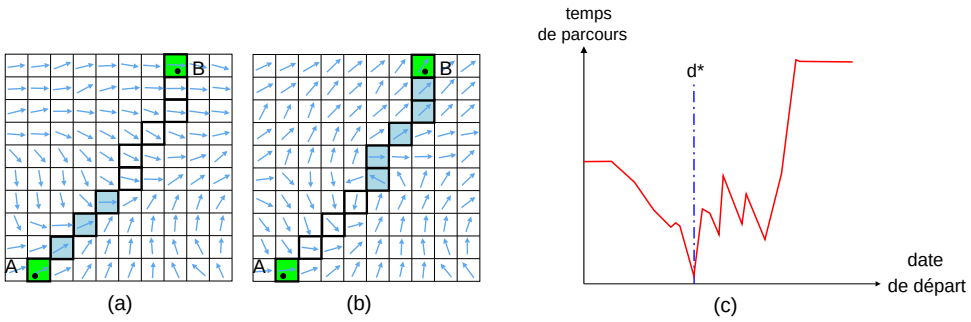

FIG. 6 – Planification de trajectoire en présence de courants variables dans le temps
Etant données un ensemble de cartes de courants (a) et (b), la propagation d'onde symbolique trouve la date de départ d^* minimisant le temps de parcours entre A et B, représenté en (c). Dans chaque carte des courants, le drone exécute la partie de la trajectoire représentée en bleue, tirant au mieux parti des courants.

3. Autres contributions

Cette partie présente deux autres contributions un peu plus mineures dans le domaine de la planification de trajectoire :

– Le **chapitre 6** introduit le concept de *collision d'ondes* [ST07], qui permet la planification de multiples chemins (entre plusieurs points à visiter) de façon efficace. Nous montrons que ce concept peut diviser le temps de calcul jusqu'à 4, comparé à l'utilisation répétée (pour chaque chemin) de techniques existantes.

– Le **chapitre 7** propose d'utiliser la programmation par contraintes pour moduler la vitesse du drone [STR07]. La modulation de vitesse consiste à générer un profil de vitesse que doit suivre le drone sur un chemin prédéfini. Ce profil a pour but d'éviter les obstacles mobiles (en accélérant ou en ralentissant).

  Si nous prenons de nouveau l'analogie avec préparation d'itinéraires routiers, nous pouvons constater que les planificateurs actuels (Mappy ou ViaMichelin, par exemple) fournissent à l'automobiliste un plan de route global, mais c'est à l'automobiliste de localement accélérer ou décélérer pour tenir compte des autres véhicules (maintien d'une distance de sécurité, dépassement d'un véhicule trop lent), des piétons, des feux, etc. C'est la même idée pour les drones, avec des contraintes spécifiques (présence des courants notamment).

Enfin, dans la conclusion, nous ferons un bilan sur l'ensemble des contributions de cette thèse. En particulier, nous présenterons les applications actuelles de ce travail, ainsi qu'une liste de points qu'il serait intéressant d'améliorer ou d'approfondir. A partir de cette analyse, nous donnerons quelques pistes pour d'éventuels travaux de recherche dans la continuité de cette thèse.

3 Dépendances entre chapitres

La figure ci-après présente les dépendances entre les chapitres de cette thèse.

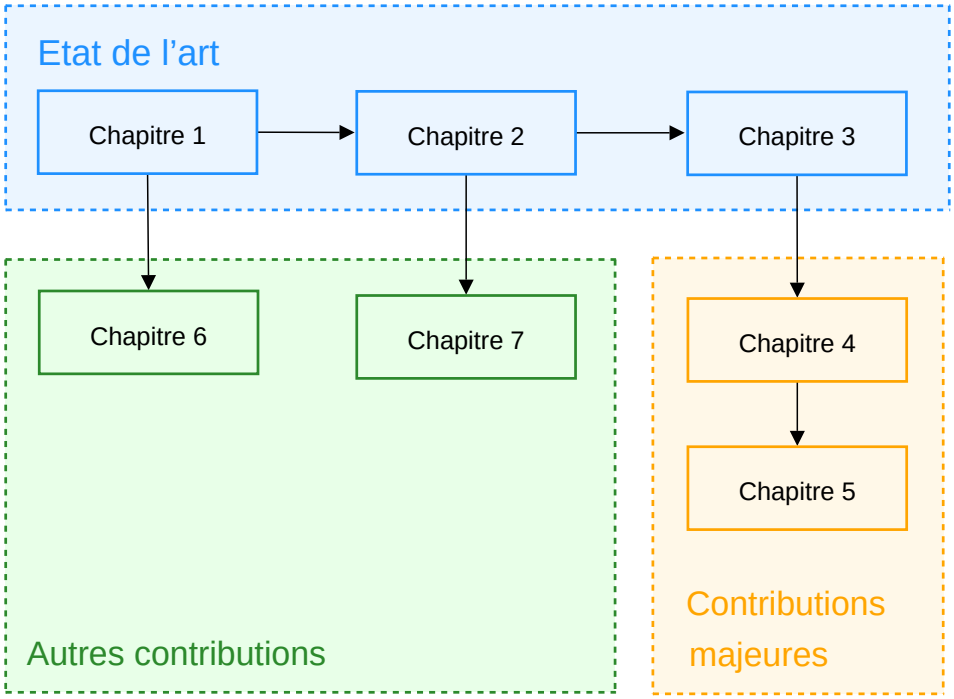

FIG. 7 – Guide de lecture
La notation Chapitre $i \rightarrow$ Chapitre j signifie : "les concepts introduits dans le chapitre i sont nécessaires pour une bonne compréhension du chapitre j".

4 Notations

Cette partie décrit les notations valables dans l'ensemble de cette thèse. Toutes les autres notations sont locales, c'est à dire uniquement valables pour le chapitre courant.

4.1 Espace des configurations

$\mathcal{C}$	Espace des configurations du robot = espace Euclidien de dimension 2
$\mathcal{C}_{obst}$	Partie de $\mathcal{C}$ occupée par des obstacles
$\mathcal{C}_{valide}$	Partie libre de $\mathcal{C}$, i.e. dépourvue d'obstacles. Egale à $\mathcal{C} \setminus C_{obst}$.
$\mathcal{R} = (O, \vec{x}, \vec{y})$	Repère orthonormé associé à $\mathcal{C}$
$\vec{u} = (u_x, u_y)$	Un vecteur dans $\mathcal{R}$
A	Configuration initiale du robot
B	Configuration finale du robot
$\vec{w}$	Champ vectoriel de dimension 2, définissant le vecteur-vitesse du courant en tout point de $\mathcal{C}$
$v^{\vec{r}}$	Vitesse de déplacement du robot par rapport à $\mathcal{R}$
$v^{\vec{w}}$	Vitesse de déplacement du robot par rapport à $\vec{w}$

4.2 Opérateurs

$\{E_i\}_{i\in[1,n]}$	Ensemble de n éléments E_1, E_2, ..., E_n
$\cup$	Union ensembliste
$\cap$	Intersection ensembliste
$u = \|\|\vec{u}\|\|$	norme de $\vec{u}$
$\vec{u} \cdot \vec{v}$	Produit scalaire entre $\vec{u}$ et $\vec{v}$
$\vec{u} \wedge \vec{v}$	Produit vectoriel entre $\vec{u}$ et $\vec{v}$
$\widehat{\langle \vec{u}, \vec{v} \rangle}$	Angle formé par $\vec{u}$ et $\vec{v}$

Première partie

Etat de l'art

Introduction

Cette partie présente et compare des techniques de planification de mouvement qui ont été proposées dans de domaine de la robotique, des années 1980 à aujourd'hui. La plupart d'entre elles sont particulièrement bien détaillées dans l'ouvrage de Latombe [Lat91], qui fait référence pour la période 1980-1990. Pour les techniques plus récentes, le lecteur intéressé pourra consulter le livre de référence de LaValle [Lav06].

Nous avons expliqué dans l'introduction de cette thèse que n'importe quel robot pouvait être modélisé comme un mobile ponctuel évoluant dans l'*espace des configurations* [LP83], noté $\mathcal{C}$. Chaque dimension de cet espace représente un degré de liberté du robot. Dans cet espace, l'état initial du robot est représenté par la configuration A, et l'état voulu (le but) par la configuration B.

Au fil des années, les éléments présents dans $\mathcal{C}$ se sont peu à peu complexifiés. Dans les années 80, l'espace était uniquement occupé par un ensemble d'*obstacles fixes* (figure 8a). Ces obstacles modélisent des régions de l'environnement inatteignables par le robot, dues à la présence d'objets (du meuble au bâtiment), la topologie du terrain (du talus à la montagne), ou encore à des limitations technologiques (rayon de braquage, quantité énergie embarquée).

Puis a été envisagée la possibilité que ces obstacles deviennent *mobiles* (figure 8b), c'est à dire que leur position varie dans le temps. Les obstacles mobiles peuvent modéliser un environnement dynamique (des portes qui s'ouvrent et se ferment par exemple), ou d'autres objets (d'autres robots, par exemple) qui se déplacent dans cet environnement et qu'il faut éviter.

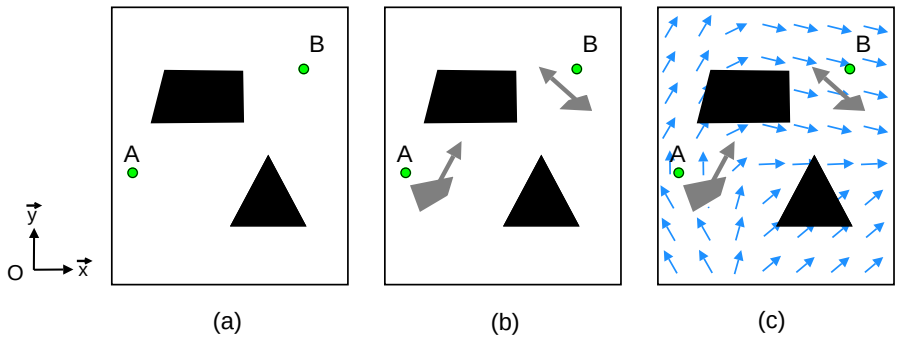

FIG. 8 – Des espaces des configurations de plus en plus complexes
La figure illustre l'ajout progressif d'éléments dans l'espace des configurations du robot : (a) obstacles fixes (en noir) ; (b) obstacles mobiles (en gris) avec leur vecteur-vitesse instantané (flèches) ; (c) courants (flèches bleues).

Enfin ont été ajoutés des courants dans $\mathcal{C}$ (figure 8c). Comme mentionné dans l'introduction de cette thèse, toute une catégorie de robots (aériens et marins) est particulièrement sensible à la présence de courants. Les ignorer peut avoir des conséquences très importantes sur la consommation d'énergie du robot, voire sur sa survie. En effet, si les courants deviennent trop forts, le robot peut dériver de façon incontrôlée, éventuellement jusqu'à heurter un obstacle.

Ainsi, selon les éléments présents dans l'espace des configurations, le problème de planification de mouvement de robots a été formulé comme étant, par ordre de difficulté croissante :

– **Une planification de chemin**, si $\mathcal{C}$ contient uniquement des obstacles fixes (figure 8a).

  Ce problème consiste à trouver une courbe de longueur finie, appelée *chemin*, entièrement contenue dans l'espace valide $\mathcal{C}_{valide}$ et reliant A et B. Il peut être résolu en un temps polynomial en fonction du nombre d'obstacles. Les méthodes les plus évoquées dans la littérature sont :

  – Les méthodes de décomposition, discrétisant l'espace afin d'appliquer des algorithmes de type A^* [HNR68],
  – Les méthodes probabilistes, procédant à un échantillonnage aléatoire de l'espace : Probabilistic RoadMap [KL94], Rapid Random Trees [LaV98], Probabilistic Cell Decomposition [Lin04],
  – Les potentiels artificiels [Kha80], s'inspirant des phénomènes électromagnétiques,
  – Les métaheuristiques, s'inspirant du monde du vivant : algorithmes génétiques [Hol75], essaims particulaires [KE95], colonies de fourmis [CDM91].

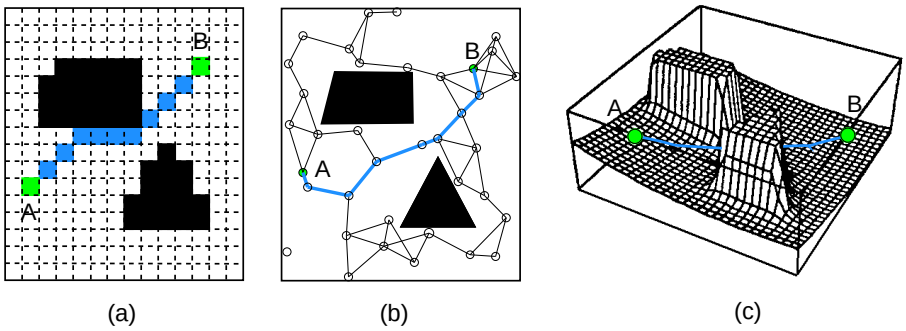

FIG. 9 – Planification de chemin
La figure illustre les chemins calculés dans l'espace de la figure 8a en appliquant les méthodes suivantes : (a) algorithme A^* sur des cellules régulières ; (b) Probabilistic RoadMap ; (c) potentiels artificiels.

– **Une planification de trajectoire**, si $\mathcal{C}$ contient en plus des obstacles mobiles (figure 8b).

  Ce problème consiste à déterminer simultanément deux choses : un chemin reliant A et B, comme précédemment, et la vitesse du mobile, à tout instant, sur ce chemin.

Il a été démontré que ce problème était NP-difficile dans un espace de dimension 2 [Can88], même dans le cas d'obstacles extrêmement simples (polygonaux, se déplaçant de façon rectiligne uniforme). Devant ce constat, deux branches de méthodes approximatives sont apparues : les méthodes globales et les méthodes locales :

1. Les méthodes *globales* tiennent compte de l'intégralité du mouvement des obstacles pour planifier une trajectoire. Elles se divisent en deux catégories :

   (a) Les méthodes *directes*, qui planifient la trajectoire du mobile en une seule fois. Elles consistent en grande majorité à construire l'espace des configurations-temps [ELP86], de dimension 3 (les 2 dimensions initiales + 1 dimension temporelle). Dans celui-ci, tous les obstacles sont représentés par des volumes statiques.

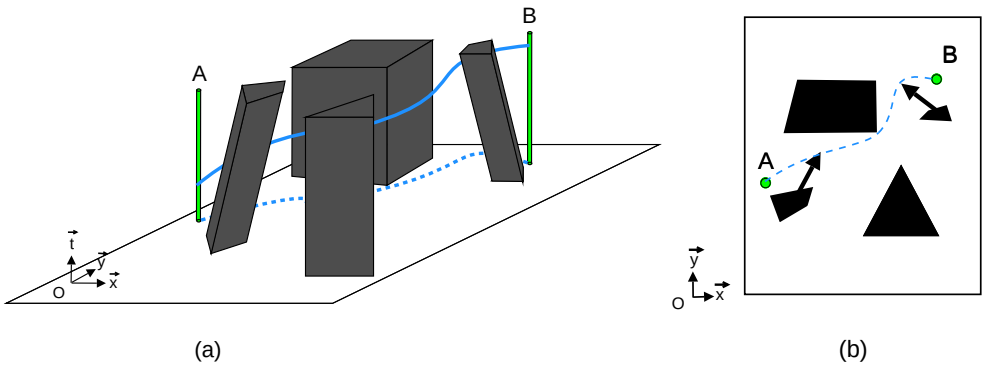

FIG. 10 – Planification globale directe

En ajoutant une dimension supplémentaire à l'espace initial de la figure 8b, on obtient un l'espace des configurations-temps (a) dans lequel tous les éléments sont des volumes. Le chemin planifié est illustré en bleu et sa projection dans le plan $(O, \vec{x}, \vec{y})$ est présentée dans la partie (b).

   Cette représentation est plus commode, car elle permet d'avoir une vue globale sur la dynamique de l'environnement. Toutefois, une discretisation déterministe (à base des méthodes de décomposition) de cet espace entraîne généralement un nombre important d'éléments. En pratique, pour conserver de bonnes performances, les méthodes procédant à un échantillonnage aléatoire de l'espace (principalement Probabilistic RoadMap [HKLR02] et les algorithmes génétiques [GCFR98]) sont majoritairement utilisées.

   (b) Les méthodes *indirectes*, qui procèdent en deux phases : une planification de chemin et une modulation de vitesse sur ce chemin [KZ86]. Cette dernière peut être vue comme une deuxième planification de chemin, dans un espace-temps de dimension 2. Cette décomposition permet de reformuler le problème de planification de trajectoire en deux planifications de chemin et donc de le traiter en un temps polynomial. Ceci se fait au prix de la complétude : certaines situations aboutiront à un échec alors qu'il existait une solution.

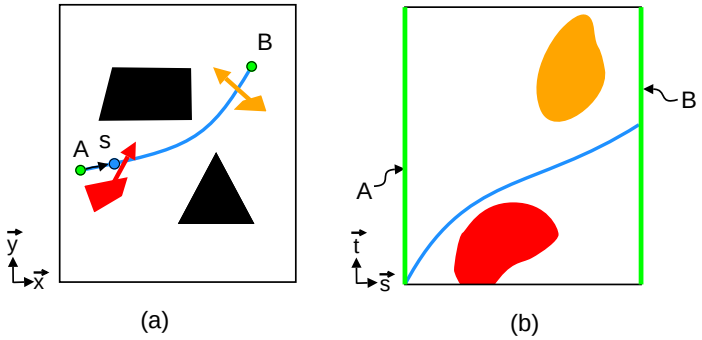

FIG. 11 – Planification globale indirecte

La planification comporte deux phases : (a) le calcul d'un chemin évitant uniquement les obstacles fixes et (b) la modulation de vitesse du mobile sur ce chemin, permettant d'éviter les obstacles mobiles, dans un espace-temps de dimension 2 (l'abscisse curviligne s et le temps t). Cet espace comporte des régions interdites (en rouge et en orange) matérialisant des conflits avec les obstacles mobiles (de la même couleur).

2. Les méthodes *locales* tiennent compte de l'état des obstacles à intervalles réguliers. Dans une méthode d'ordre n, l'état d'un obstacle est constitué des n dérivations successives de sa position [FS98]. Ainsi :

   (a) Les méthodes d'ordre 0 tiennent uniquement compte de la position des obstacles. Purement réactives, elles mettent en jeu des techniques classiquement utilisées dans un contexte temps réel pour des tâches de navigation : les réseaux de neurones [Jan04], la logique floue [SN85] et la déformation de chemin [QK93]. Elles sont donc adaptées pour éviter les collisions à court terme. Pour les utiliser dans un contexte de planification, les perceptions de l'environnement par le mobile sont simulées.

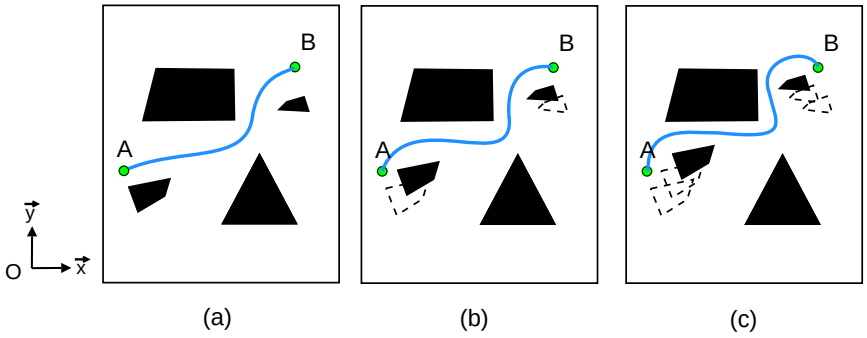

FIG. 12 – Planification locale d'ordre 0

La figure illustre la réaction locale à la présence d'obstacles par déformation de chemin. Les positions successives des obstacles mobiles sont illustrées en pointillés.

   (b) Les méthodes d'ordre 1 tiennent non seulement compte de la position des obstacles, mais aussi de leur vecteur-vitesse. Ceci permet d'anticiper leur déplacement et d'éviter les collisions à plus long terme. Dans cet esprit, ont été proposés le concept d'obstacle-vitesse [FS93] et une adaptation des potentiels artificiels [GC02].

3. **Une planification de trajectoire en présence de courants**, si $\mathcal{C}$ contient en plus des courants (figure 8c).

   Le passage de la planification de chemin à la planification de trajectoire est assez naturelle, puisque la planification de trajectoire repose principalement sur des méthodes de planification de chemin, soit telles quelles (en temps que première étape de calcul), soit les adaptant (en ajoutant une dimension temporelle).

   Toutefois, l'adaptation de ces techniques à un environnement contenant des courants est beaucoup moins aisée. Sur la quantité très importante d'algorithmes de planification de trajectoire, une infime partie a été étendue à la présence de courants : les algorithmes génétiques [RK03] et les potentiels artificiels [GAO05]. Plus anecdotiquement, on trouve quelques méthodes spécifiques, comme l'optimisation de B-splines [ISM05] et les champs de vitesses [NBMB06].

Cette progression dans la complexité des problèmes est assez représentative de la progression historique dans les algorithmes de planification de mouvement. C'est pourquoi nous avons décidé de retracer cette progression au fil des chapitres :

- Le chapitre 1 présente les méthodes de planification de chemin,
- Le chapitre 2 décrit ensuite les méthodes de planification de trajectoire,
- Le chapitre 3 étend enfin les méthodes précédentes à la présence de courants.

Chaque chapitre se réfère aux méthodes décrites dans des chapitres précédents. Ils peuvent toutefois être lus de façon indépendante. A titre illustratif, les méthodes présentées sont successivement appliquées aux espaces de configurations de la figure 8 (8a dans le chapitre 1, 8b dans le chapitre 2 et 8c dans la chapitre 3). Des détails sur les algorithmes associés (complexité, déterminisme, complétude, optimalité, etc.) sont fournis.

Chapitre 1

La planification de chemin

Sommaire

Comme énoncé en introduction, étant données deux configurations A et B, le problème de planification de chemin consiste à trouver une courbe de longueur finie L, appelée *chemin*, entièrement contenue dans l'espace valide C_{valide}, reliant A et B.

Formellement parlant, le chemin recherché est défini par le graphe d'une fonction continue $\gamma : l \in [0, L] \mapsto (x, y) \in \mathcal{C}_{valide}$, vérifiant $\gamma(0) = A$ et $\gamma(L) = B$. La fonction γ^* minimisant la longueur L est appelée chemin optimal.

Les méthodes les plus évoquées dans la littérature pour résoudre ce problème ont été répertoriées dans le diagramme de la figure 1.1. Ces méthodes se divisent en 4 grandes classes : les méthodes de décomposition, les méthodes probabilistes, les potentiels artificiels et les métaheuristiques. Chaque classe est décrite dans une section.

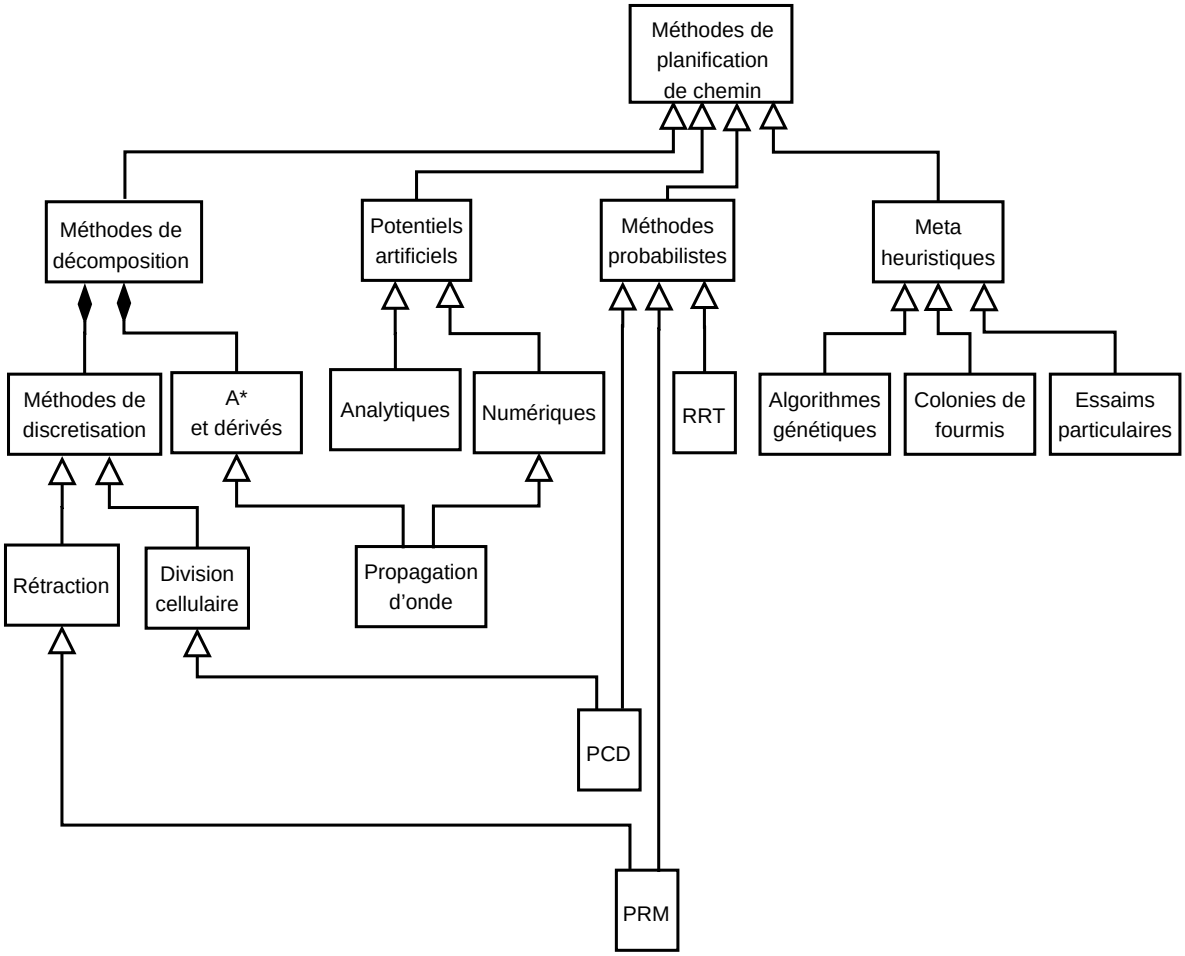

FIG. 1.1 – Méthodes de planification de chemin
Les triangles blancs symbolisent des liens d'héritage, et les losanges noirs des liens de composition.

1 Les méthodes de décomposition

Les méthodes de décomposition consistent à discrétiser l'espace initial, continu, en un ensemble fini d'éléments. Ces éléments sont ensuite représentés au sein d'un graphe sur lequel il sera possible d'appliquer des algorithmes de recherche de plus court chemin, de type A^*.

1.1 La discrétisation de l'espace

L'espace est principalement discrétisé de deux manières différentes : par division en cellules ou par rétraction de chemins.

1. La **division en cellules** consiste à décomposer l'espace valide (i.e. dépourvu d'obstacles) en un ensemble de régions disjointes. Cette division peut être exacte ou approchée.

   – Une division *exacte* aboutit à un recouvrement complet de l'espace valide. Les cellules sont des polygones quelconques, dont la forme dépend de la position des sommets des obstacles. La division trapézoïdale, proposée par Chazelle [Cha87], est la plus simple. Elle consiste à tracer une série de segments parallèles à un des axes ($\vec{y}$ dans la figure 1.2), passant par les

sommets des obstacles. Si n est le nombre total de sommets, le nombre de polygones est en $O(n)$, et le temps de calcul en $O(n \log n)$.

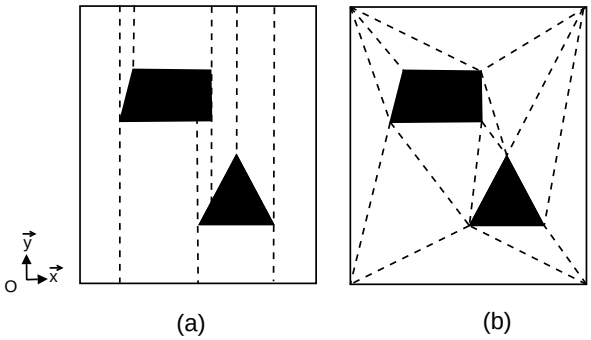

FIG. 1.2 – Division cellulaire exacte
La partie (a) présente une division à base de trapèzes et la partie (b) à base de triangles. La forme des cellules est conditionnée par la position des sommets des obstacles.

– Une division *approchée* aboutit à un recouvrement partiel de l'espace valide. La forme des cellules est prédéfinie, généralement rectangulaire.
La division de type *quadtree* est un cas particulier de la division 2^m-tree (où m désigne la dimension de l'espace) avec $m = 2$. Elle permet de concentrer l'effort de division au niveau des zones critiques, i.e. aux environs des obstacles.
Initialement développée pour des tâches de graphisme [JT80], elle consiste à diviser récursivement les cellules appartenant à un obstacle en 2^m sous-cellules. La condition d'arrêt est souvent un nombre maximal r de subdivisions, appelé *résolution*. Dans ce cas, le nombre de cases peut s'élever à 2^{mr}. En d'autres termes, à m fixé, le temps de calcul augmente donc de façon exponentielle avec r.

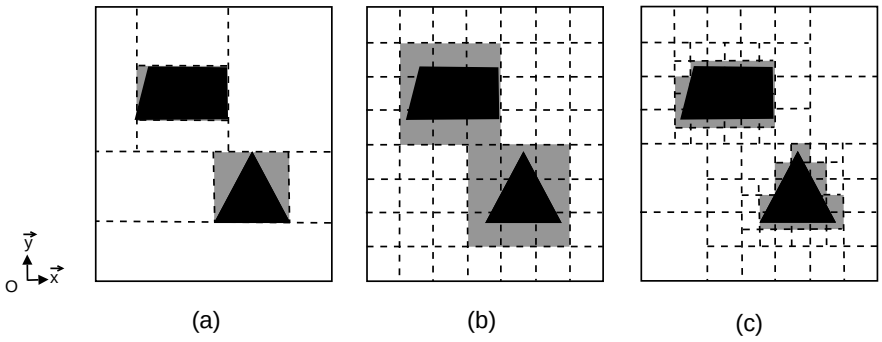

FIG. 1.3 – Division cellulaire approchée
La partie (a) présente une division à base de rectangles de taille variable, la partie (b) à base de cellules régulières, la partie (c) utilisant un *quadtree* de résolution 1. Les cellules grises sont considérées comme occupées par un obstacle.

Il est important de noter que les divisions approchées sont sources d'incomplétude. Si les cellules sont trop grossières, certains passages étroits peuvent se retrouver bouchés, comme on peut l'observer sur la figure 1.3b. Dans le pire des cas, ce phénomène peut éliminer le seul chemin possible.

2. La **rétraction de chemins** consiste à extraire de l'espace un réseau de courbes unidimensionnelles. Dans la littérature anglophone, on parle de *roadmap*.

   Le *graphe de visibilité*, introduit par Nilsson [Nil69], s'inscrit dans une volonté de minimisation de la distance parcourue. Il exploite l'idée selon laquelle la manière optimale de contourner un ensemble d'obstacles est de passer par leurs sommets. Dans la version la plus simple, les sommets sont reliés deux à deux par un segment, si celui-ci n'intersecte pas d'obstacles. Les algorithmes les plus efficaces (de cet algorithme de base[1]) construisent ces segments en un temps en $O(n^2)$ (où n est le nombre de sommets) [Wel85].

   Le *diagramme de Voronoi*, introduit par O'Dunlaing et Yap, est destiné à minimiser les risques de collision avec les obstacles. Il est constitué par l'ensemble des courbes équidistantes aux obstacles. Ce diagramme peut être construit en un temps $O(n \log n)$, en utilisant le principe de balayage introduit par Fortune [For86] et représenté figure 1.5.

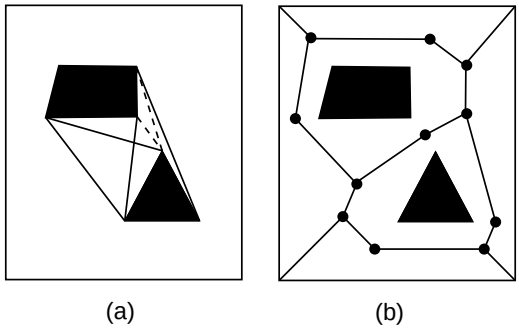

FIG. 1.4 – Rétraction de chemins (*roadmap*)
La partie (a) illustre le graphe de visibilité. Dans la version réduite, les segments en pointillés sont supprimés. La partie (b) illustre le diagramme de Voronoi.

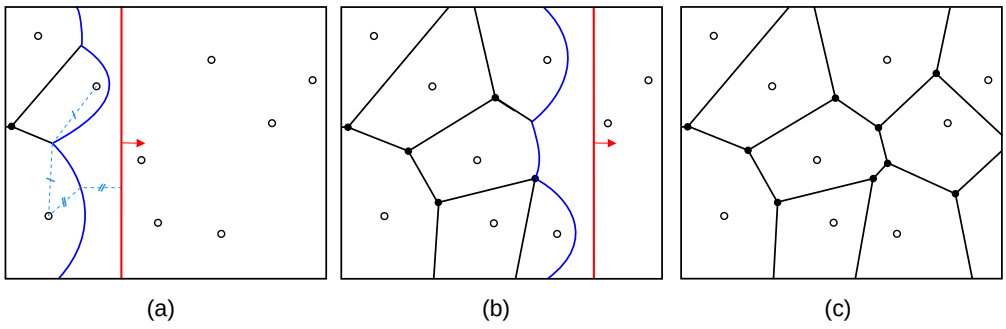

FIG. 1.5 – Algorithme de Fortune
La figure illustre la construction du diagramme du Voronoi dans un environnement contenant 10 sommets, par application de l'algorithme de Fortune. L'algorithme utilise une droite D (en rouge) qui balaie l'espace de gauche à droite. Le front F dessiné en bleu, équidistant de D et des sommets, dessine progressivement les arcs du diagramme (segments noirs).

[1] Une version améliorée existe, appelée *graphe de visibilité réduit*. Dans celle-ci, seuls les segments tangents aux obstacles sont conservés. La construction de segments peut alors être réalisée en un temps en $O(\log n)$ [Roh86]. Notons qu'un segment S est tangent à un obstacle O au point M si et seulement si dans un voisinage V de M l'intérieur de O tient entièrement d'un coté de D.

En complément à ces méthodes de discrétisation déterministes, des méthodes probabilistes ont été proposées : PRM (Probabilistic RoadMap) et PCD (Probabilistic Cell Decomposition). Elles sont décrites en détail dans la section 2.

1.2 La recherche de plus court chemin

A la fin de la phase de discrétisation, nous pouvons construire un graphe reflétant les déplacements possibles dans l'espace. Ce graphe, appelé *graphe de connectivité*, est un couple $(\mathcal{N}, \mathcal{A})$, avec :

1. $\mathcal{N} = \{N_i\}$ l'ensemble des noeuds, leur nature dépendant de la méthode de discrétisation utilisée :
   - Dans le cas d'une division cellulaire, un noeud représente une cellule de l'environnement. Les point de départs et d'arrivée sont représentés par les cellules qui les contiennent.
   - Dans le cas d'une rétraction de chemin, un noeud représente un point de passage de la *roadmap*. A ces noeuds sont ajoutés deux noeuds supplémentaires : les points de départ et d'arrivée.
2. $\mathcal{A} = \{a_{ij} = (i, j, c_{ij})\}$ l'ensemble des arcs, matérialisant le voisinage entre les noeuds : $a_{ij} \in \mathcal{A}$ si et seulement si les noeuds N_i et N_j sont voisins. Le coût c_{ij} entre les noeuds i et j est donné par une métrique $\mathcal{M}_{i,j}$ spécifique au problème.Il est toujours positif.

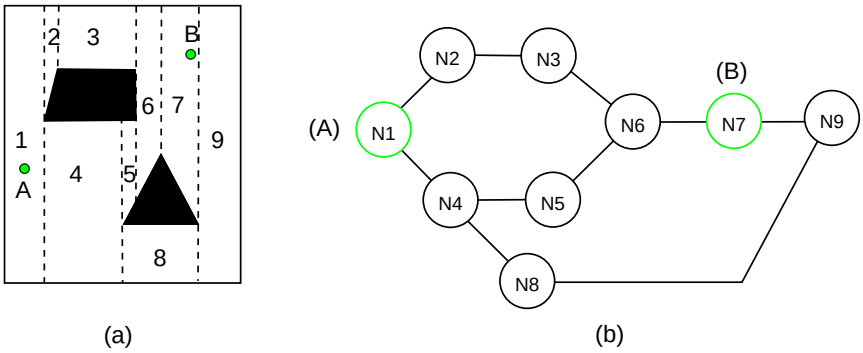

FIG. 1.6 – Graphe de connectivité

La figure présente le graphe de connectivité (b) correspondant à la discrétisation trapézoïdale (a). Les noeuds verts correspondent aux cellules de départ et d'arrivée.

Cette notion permet d'abstraire le problème initial : la recherche de plus court chemin ne se fait plus dans l'espace initial, mais dans un graphe.

L'algorithme le plus utilisé pour cette tâche est l'algorithme A^*, introduit par Hart, Nilsson et Raphael [HNR68]. L'algorithme A^* est une amélioration de l'algorithme de plus court chemin proposé par Dijkstra [Dij59]. Toute l'astuce de l'A^* réside dans l'ajout d'une partie heuristique, qui permet de guider la recherche dans le graphe vers le noeud but, et donc de limiter le nombre de noeuds évalués.

L'algorithme A^* manipule une liste de noeuds évalués appelée *OPEN*. Au début, la liste contient uniquement le noeud de départ. Puis, à chaque étape de la recherche, l'algorithme extrait le noeud de plus faible évaluation de *OPEN* et le développe. Développer un noeud N consiste à évaluer ses successeurs S et de les ajouter dans *OPEN*. N est quant à lui supprimé de *OPEN*, et exclu de toute évaluation ultérieure.

L'évaluation d'un successeur S du noeud N consiste à :

1. Etablir un *pointeur de précédence* de S vers N, mémorisant le fait que N soit le prédécesseur optimal de S.
2. Evaluer S, en utilisant la fonction f définie par :

$$f(S) = g(S) + h(S) \tag{1.1}$$

où :
- $g(S)$ est le coût réel du noeud de départ au noeud S. On a $g(S) = g(N) + c(N, S)$, où $c(X, Y)$ désigne le coût nécessaire pour se déplacer de X à Y.
- $h(S)$ est une estimation du coût nécessaire pour atteindre un noeud d'arrivée à partir de S.

g est appelée *fonction de coût* et h *fonction heuristique*. Le but de cette dernière est de guider la recherche vers un noeud d'arrivée.

La recherche s'arrête quand le noeud d'arrivée est atteint (c'est à dire quand le noeud d'arrivée est sélectionné pour être développé). Le chemin entre le départ et l'arrivée est alors déduit en partant du noeud d'arrivée et en parcourant, par pointages successifs, la liste des prédécesseurs. La propriété très intéressante de l'A^* est que le chemin trouvé est assuré d'être optimal si la fonction h est *admissible* (i.e. si elle ne surestime jamais le coût restant).

On peut noter deux cas triviaux de fonctions admissibles :

- La fonction constante $h = 0$. Dans ce cas, le résultat (et le temps de calcul) de l'A^* coïncide exactement avec celui de l'algorithme de Dijkstra.
- La distance Euclidienne au but, correspondant à un déplacement direct (i.e. en ligne droite) du noeud courant au noeud but. Cette heuristique, très utilisée, ignore totalement les contournements dus aux obstacles.

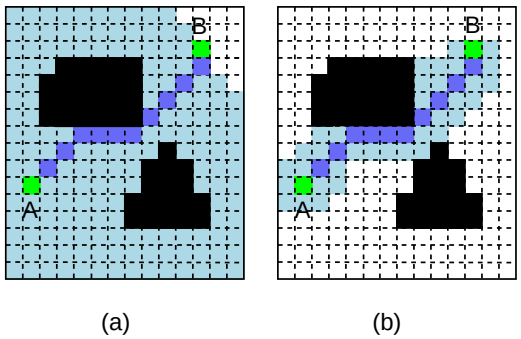

FIG. 1.7 – Influence de la fonction heuristique dans l'A^*

La figure présente (en bleu clair) les cellules évaluées par l'algorithme A^* avec les fonctions heuristiques suivantes : (a) fonction constante nulle et (b) distance Euclidienne au but. Dans (a), les cellules sont évaluées de façon isotrope (formant un disque) ; dans (b), de façon privilégiée vers le but (formant une bande). Dans les deux cas, le chemin trouvé est le même, illustré en bleu foncé.

Beaucoup de variantes de l'A^* ont été introduites par la suite, pour minimiser le nombre de noeuds explorés en cas de replanification dynamique (dans le cas d'un environnement changeant ou partiellement connu). La première du genre fut le D^* (pour *Dynamic* A^*), proposée par Stentz

[Ste94]. Depuis, un grand nombre d'extensions ont été proposées, la plus récente étant le *Field D** [FS07], réduisant des erreurs commises due à la discrétisation en cases.

Toutes ces techniques ont en commun d'utiliser les informations des recherches précédentes pour accélérer la recherche courante. Toutefois, celles-ci sortant du thème global de cette thèse (la planification prévisionnelle), elles ne seront pas détaillées ici.

2 Les méthodes probabilistes

Les méthodes probabilistes ont initialement été proposées pour permettre la planification de chemin dans des espaces de dimensions élevées, dans lesquels les méthodes de décomposition deviennent extrêmement coûteuses[2]. Toutefois, elles sont régulièrement utilisées dans des espaces de faible dimension, dans toutes les applications où la rapidité de planification est prioritaire sur l'optimalité du chemin trouvé.

Basées sur un échantillonnage aléatoire de l'espace, elles ne sont pas complètes, mais *asymptotiquement complètes*. Cela signifie que, si un chemin existe, la probabilité de le trouver tend vers 1 si le temps de calcul tend vers l'infini !

2.1 Probabilistic RoadMap (PRM)

Dans les méthodes de rétraction classiques, des points d'intérêt sont prélevés dans l'espace en fonction de la position et de la forme des obstacles (diagramme de Voronoi et graphe de visibilité, présentés page 25).

Dans la méthode PRM, des points, appelés *échantillons*, sont prélevés aléatoirement. Puis, chaque échantillon est relié à ses voisins en utilisant un planificateur local. Le planificateur le plus utilisé tente simplement de relier des échantillons en ligne droite, en vérifiant les collisions avec les obstacles. On peut également envisager d'utiliser des méthodes à bases de potentiels artificiels continus, décrits dans la section 3.

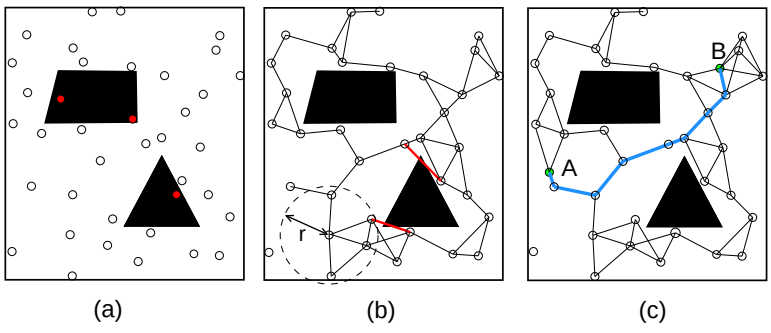

FIG. 1.8 – Méthode PRM
La méthode procède en deux étapes : (a) échantillonnage aléatoire de l'espace et (b) la connexion locale des échantillons (en utilisant un voisinage de rayon r), formant ainsi un graphe. Dans chaque étape, les éléments dessinés en rouge, intersectant des obstacles, sont supprimés. La partie (c) présente, en bleu, le plus court chemin entre A et B à dans le graphe.

[2]Quelle que soit la méthode utilisée, le nombre de cellules augmente exponentiellement avec la dimension n de l'espace. Par exemple, a une résolution r fixée, la méthode 2^m-tree génère un nombre de hypercubes en $O(2^{mr})$.

Comme après toute méthode de rétraction, nous disposons d'un graphe dans lequel nous pouvons réaliser une recherche de plus court chemin.

La méthode PRM rencontre des difficultés quand les uniques solutions empruntent un passage étroit, la probabilité qu'une succession d'échantillons soit tirée à l'intérieur de ce passage étant très faible.

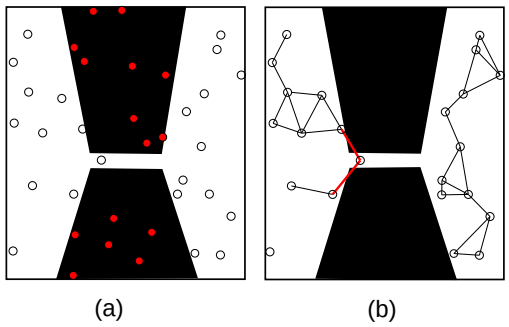

FIG. 1.9 – Impact d'un passage étroit sur la méthode PRM
La figure reprend l'échantillonnage de la figure 1.8 en présence d'un passage étroit. Ce passage provoque la suppression d'un grand nombre d'échantillons (a) et d'arcs (b). Il en résulte que les parties gauche et droite de l'espace ne sont pas connectées.

Pour y remédier, plusieurs extensions de la méthode PRM ont été proposées, dont le principe est de "guider" la génération des échantillons, plutôt que de la laisser complètement aléatoire. Les stratégies suivantes en sont deux exemples :

- Kavraki *et al.* [KL94] ont introduit une étape supplémentaire, dite d'*enrichissement*, ajoutant des échantillons aux environs de ceux ayant peu de voisins. La probabilité qu'un échantillon soit sélectionné pour enrichissement est proportionnelle à $1/(1+v)$, où v est son nombre de voisins.
- Amato *et al.* [ABD+98] ont utilisé les échantillons en collision avec les obstacles comme sources de rayons. Puis, de nouveaux échantillons sont générés, au bord des obstacles. Leur position est déterminée par une recherche dichotomique. Ce principe est la base de la méthode OBPRM (Obstacle Based PRM).

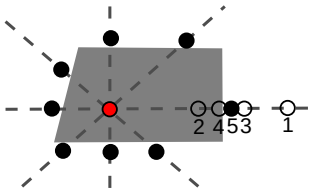

FIG. 1.10 – Principe de la méthode OBPRM
La figure illustre la génération d'échantillons utilisée dans la méthode OBPRM (Obstacle Based PRM). Chaque échantillon entrant en collision avec un obstacle (en rouge) génère des rayons à angles réguliers. Sur ces rayons sont déterminés de nouveaux échantillons (en blanc) au bord de l'obstacle.

2.2 Probabilistic Cell Decomposition (PCD)

PCD est une méthode de division cellulaire probabiliste, introduite par Lingelbach [Lin04], mélangeant les concepts de la méthode PRM (l'échantillonnage aléatoire) et du 2^m-tree (la subdivision de cellules).

Au départ, l'espace contient une unique cellule, considérée comme libre. Puis, une série de divisions cellulaires est réalisée comme suit :

1. Un échantillon E est tiré au hasard dans l'espace,
2. La cellule C contenant E est divisée en deux sous-cellules[3] : C_1, contenant encore E, et C_2. La nature de la sous-cellule C_1 est déterminée par la nature de E (faisant partie d'un obstacle ou non). La nature de C_2 est inchangée.

Les critères d'arrêt envisageables sont de même nature que ceux utilisés pour le 2^m-tree. On peut par exemple choisir un nombre maximal de subdivisions.

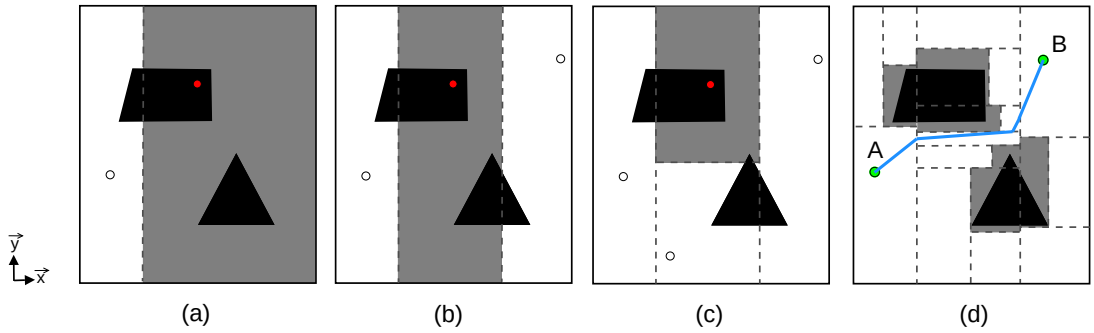

FIG. 1.11 – Méthode PCD
Les cellules sont subdivisées par tirage successifs d'échantillons. La sous-cellule contenant le nouvel échantillon est de même nature que celui-ci : valide (en blanc) ou occupée par un obstacle (en rouge). Les parties (a), (b) et (c) illustrent les trois premières subdivisions. La partie (d) présente (en bleu) le chemin trouvé entre A et B après 18 subdivisions.

Grâce au concept de cellule, une vaste zone peut être couverte par un seul échantillon. Ceci permet de résoudre des problèmes possédant des passages étroits avec un nombre d'échantillons restreint.

2.3 Rapid Random Trees (RRT)

Développée par LaValle [LaV98], la méthode RRT est une alternative probabiliste aux algorithmes de type A^*. Cette fois, l'échantillonnage aléatoire n'est pas utilisé pour discrétiser l'espace, mais directement pour construire l'arbre de recherche.

Etant donné un arbre $\mathcal{A}$, celui-ci est étendu comme suit :

1. Un point M est tiré de façon aléatoire dans l'espace, la loi de probabilité utilisée étant souvent biaisée de façon à favoriser l'exploration des zones non couvertes,

[3] Étant donné un échantillon E, une subdivision consiste à détecter l'échantillon F le plus proche, et à tracer une droite passant par le milieu du segment $[EF]$, parallèle à un des axes du repère.

2. Le noeud N le plus proche (en terme de distance) est sélectionné dans $\mathcal{A}$,
3. Un nouveau noeud N' est placé sur le segment $[NM]$, à un petite distance ε de N,
4. Si le mouvement en ligne droite $N \rightarrow N'$ est valide (i.e. si aucune collision n'est détectée), le noeud N' est ajouté à l'arbre. Sinon on retourne en 1.

L'expansion s'arrête quand le but est atteignable par le noeud le plus proche, en effectuant un mouvement de longueur prédéfinie (souvent ε).

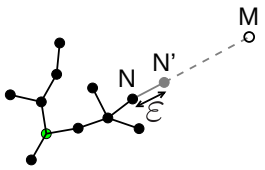

FIG. 1.12 – Extension d'arbre de recherche dans la méthode RRT
L'arbre existant est étendu par l'intermédiaire d'un nouvel échantillon M, via le noeud le plus proche N. Le nouveau noeud N' est placé une petite distance ε de N sur le segment $[NM]$.

Plusieurs variantes ont été proposées, pour améliorer les performances ou favoriser les multiples planifications dans un même environnement.

Pour le premier point, Kuffner et Lavalle ont proposé, dans leur méthode RRT-Connect [KL00], de développer simultanément deux arbres : l'un partant du noeud de départ, et l'autre de l'arrivée. Quand ceux-ci sont suffisamment proches, et si cela est possible, ceux-ci sont connectés. On parle alors de *RRT bidirectionnel*.

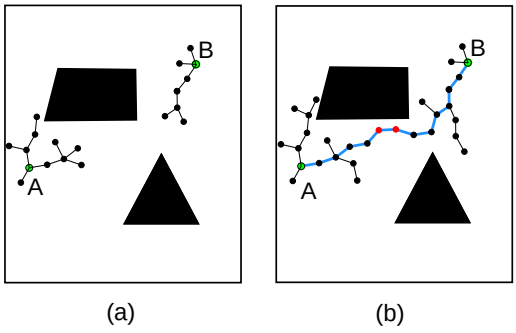

FIG. 1.13 – RRT bidirectionnel
Deux arbres sont développés simultanément, partir des points de départ A et d'arrivée B. Quand ceux-ci sont suffisamment proches, ils sont connectés. Le chemin entre A et B (en bleu) est alors obtenu en parcourant, à partir des points de connexion (en rouge), les prédécesseurs successifs dans les deux arbres.

Pour le deuxième point, Bruce et Veloso ont suggéré, dans ce qu'ils ont nommé ERRT (Extented RRT) [BV02], de procéder à l'exploration de l'espace avec une probabilité p, et de réutiliser les noeuds présents sur le chemin précédemment calculé avec une probabilité $1-p$. La probabilité p est fixée selon la dynamique de l'environnement, éventuellement par apprentissage.

3 Les potentiels artificiels

Les méthodes précédemment évoquées se ramènent toutes à un graphe pour modéliser les déplacements possibles du mobile. Khatib [Kha80] a proposé une toute autre approche, inspirée de la physique. Elle considère le mobile comme une particule sous l'influence d'un champ de potentiel U, dont les variations locales reflètent la structure de l'espace.

Les méthodes à base de potentiels se divisent en deux catégories : analytiques et numériques.

3.1 Méthodes analytiques

Les approches analytiques manipulent l'expression explicite du champ de potentiel U. Le plus souvent, le potentiel $U(M)$ d'un point M est la somme de potentiels élémentaires :

$$U(M) = U_{attr}(M) + \sum_{i=1}^{n} U^i_{rep}(M) \tag{1.2}$$

- U_{attr} est un champ de potentiel *attractif*, associé au point but, augmentant avec la distance, généralement de façon linéaire ou quadratique. Kathib a par exemple proposé des fonctions de la forme :

  $$U_{attr}(M) = \frac{1}{2} \cdot \varepsilon \cdot d(M,B)^2 \tag{1.3}$$

  où ε est une constante positive, et $d(M,B)$ la distance Euclidienne de M au but B.

- U^i_{rep} est un champ de potentiel répulsif, associé à l'obstacle i. L'idée de ce champ est de créer une barrière aux environs de l'obstacles, de façon à le rendre infranchissable, tout en ayant une influence négligeable ailleurs.
  Une possibilité est de prendre la forme inverse de U_{attr}, en procédant au changement de variable $d \rightarrow 1/d$:

  $$U^i_{rep}(M) = \frac{1}{2} \cdot \eta \cdot \frac{1}{d(M,O_i)^2} \tag{1.4}$$

  η désignant une constante positive et O_i l'$i$ème l'obstacle de l'espace.

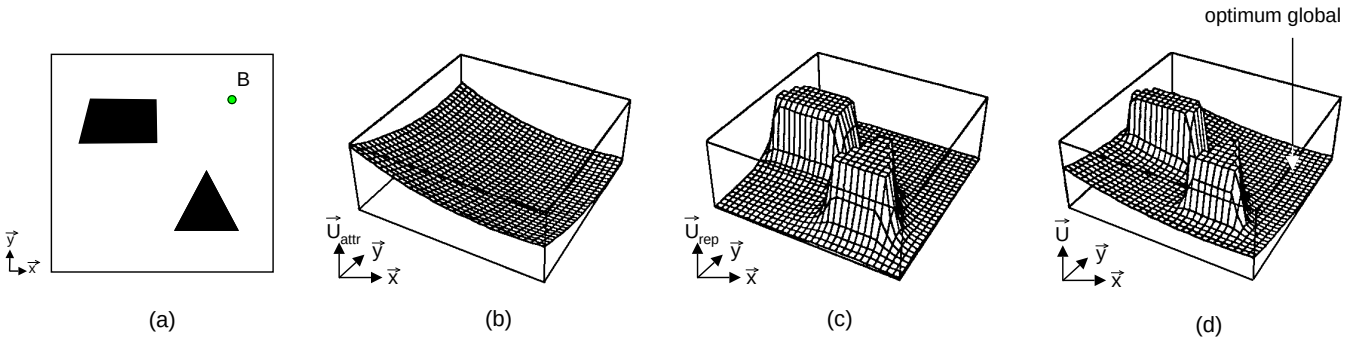

FIG. 1.14 – Construction du champ de potentiel
La figure (tirée de [Lat91]) illustre la construction du champ de potentiel artificiel reflétant la structure de l'espace (a), contenant le point but B. La partie (b) représente le champ attractif associé à B, (c) les champs répulsifs associé aux obstacles et (d) le champ total.

La fonction U ainsi construite possède un minimum global (de valeur proche de 0) au point but B, et des valeurs tendent vers l'infini aux environs des obstacles. Un chemin entre un point A et B peut donc être construit en relevant les étapes successives de la minimisation de U en partant de A.

L'algorithme le plus simple pour effectuer cette minimisation est la *descente du gradient* (ou l'un de ses dérivés, présentés dans [Min83]). Elle consiste à effectuer une série de déplacements infinitésimaux dans la direction $-\vec{\nabla}U$, où $\vec{\nabla}$ désigne le gradient[4] de U.

Le chemin obtenu est alors la courbe de moindre énergie dans l'espace de la figure 1.14d. Pour illustrer le propos, le chemin est analogue à celui que suivrait une bille, sans frottements, en présence d'un champ gravitationnel.

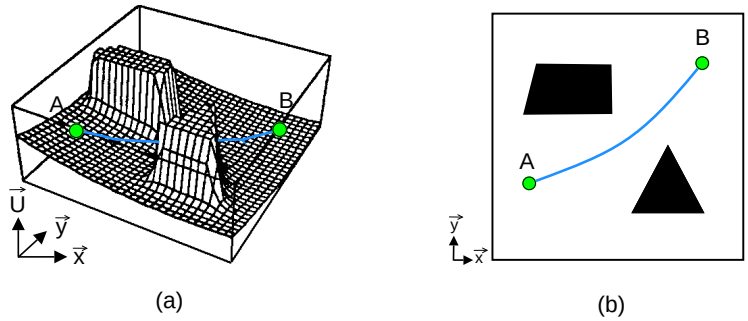

FIG. 1.15 – Minimisation du champ de potentiel
La partie (a) présente (en bleu) la succession de positions empruntées pendant la minimisation du champ de potentiel de la figure 1.14d, en partant du point A. Le chemin correspondant dans l'espace initial est fourni dans la partie (b).

Le principal défaut des potentiels artificiels analytiques réside dans le fait que l'atteinte d'un minimum n'est pas forcement équivalent à l'atteinte du but. C'est notamment le cas :

- Quand les obstacles sont proches les uns des autres, donnant naissance à des minima locaux : les algorithmes d'optimisation, supposant des fonctions convexes, peuvent alors converger vers un de ces minima. Dans la figure 1.15, par exemple, le rapprochement ou l'agrandissement des obstacles obstruerait le passage central emprunté par le robot, qui se retrouverait bloqué.

- Quand des obstacles sont présents aux environs du but : l'optimum global de la fonction est alors décalé, ce qui provoque l'arrêt du robot aux environs du but, sans l'atteindre. Ce dernier problème est connu sous le nom de GNRON (*Goal Non Reachable with Obstacles Nearby*).

Ces problèmes ont été atténués, mais non résolus, principalement de deux façons différentes : en jouant sur l'expression des fonctions U_{attr} et U_{rep} [CBW90][GC00] ou en ayant recours à des mouvements aléatoires pour débloquer le robot [BL90].

[4]Les composantes $\vec{\nabla}U$ sont les dérivées partielles de la fonction U par rapport à ses variables.

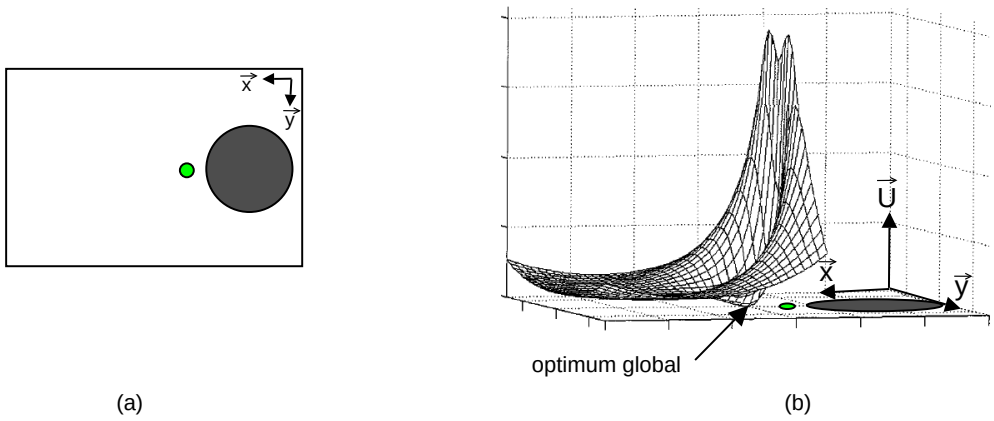

FIG. 1.16 – Le problème GNRON

Dans l'espace (a), un obstacle (en gris) est présent aux environs du point but (en vert). La présence de cet obstacle décale l'optimum global du champ de potentiel, illustré en (b), qui n'est alors plus situé au niveau du point but. Ce problème est tiré de [GC00].

3.2 Méthodes numériques

Une autre façon de pallier les problèmes évoqués ci-avant est de ne calculer que le potentiel attractif associé au but, limité à l'espace valide. C'est le concept de *fonction de navigation* [Kod87]. Cependant, l'expression analytique d'une telle fonction n'est pas toujours calculable.

Suite aux travaux de Jarvis [Jar85], Barraquand *et al.* ont donc proposé une méthode de calcul numérique [BL91], appelée *propagation d'onde*, indépendante du nombre et de la forme des obstacles.

Dans celle-ci, l'espace est discrétisé au moyen d'une grille, puis un front d'onde est propagé comme suit : la cellule de départ A est affectée à 0. Puis, chaque voisin (de Manhattan[5]) de A est affecté à 1 ; puis, chaque voisin (non évalué) des cellules de valeur 1 est affectée à 2 ; et ainsi de suite.

Au cours de cette propagation, les cellules appartenant aux obstacles sont ignorées. Le potentiel ainsi calculé représente la distance de Manhattan[6] nécessaire pour rejoindre la cellule but, en contournant les obstacles.

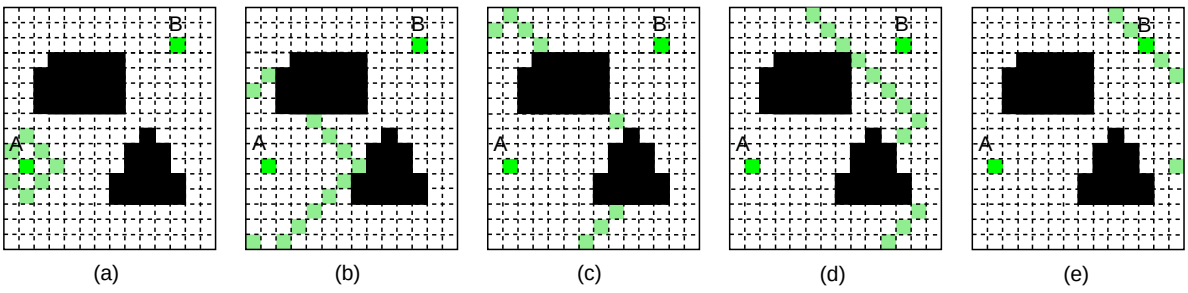

FIG. 1.17 – Propagation d'onde utilisant le voisinage et la distance de Manhattan

Cet algorithme de base a été généralisé par Dorst [DT88] à n'importe quel type de voisinage

[5]Les voisins de Manhattan d'une case C contient les cases adjacentes à C situées aux 4 points cardinaux.
[6]La distance Manhattan représente le nombre total de déplacements horizontaux et verticaux entre deux cases.

et de distance. A chaque itération de la propagation, la case de coût minimal est extraite du front d'onde et ses voisins, fournis par une fonction $\mathcal{V}$, sont évalués à l'aide d'une métrique $\mathcal{M}$ (opération d'évaluation). Si un voisin V avait déjà été évalué au préalable, la valeur minimale est conservée (opération de comparaison).

L'ensemble de l'algorithme est fourni dans la figure ci-après. Il constituera la base des contributions majeures de cette thèse.

PROPAGATION_ONDE($\widetilde{A}, \widetilde{B}, G, d$)
- ▷ **Entrée :** $\widetilde{A}, \widetilde{B}$: cases de départ et d'arrivée
- ▷ **Entrée :** G : grille
- ▷ **Entrée :** d : date de départ en $\widetilde{A}$
- ▷ **Locale :** F : front d'onde courant
- ▷ **Locale :** D : cases développées par le front d'onde
- ▷ **Locale :** H : tête de F (case de moindre coût)
- ▷ **Locale :** V : un voisin de H
- ▷ **Locale :** c_V^{temp} : evaluation temporaire de V

1 **Début**
2 $F \leftarrow \{\widetilde{A}\},\ D \leftarrow \emptyset$
3 $c_X \leftarrow +\infty, \forall X \in G \setminus \{\widetilde{A}\}$
4 $c_{\widetilde{A}} \leftarrow 0$
5 **faire**
6 $H \leftarrow \arg\min\{c_X, X \in F\}$
7 **pour chaque** $V \in (\mathcal{V}(H) \setminus D)$ **faire**
8 $c_V^{temp} \leftarrow c_H + \mathcal{M}_{H,V}(c_H + d)$ //opération d'évaluation
9 $c_V \leftarrow \min\{c_V, c_V^{temp}\}$ //opération de comparaison
10 $F \leftarrow F \cup \{V\}$
11 $F \leftarrow F \setminus \{H\}$
12 $D \leftarrow D \cup \{H\}$
13 **tantque** $H \neq \widetilde{B}$
14 **Fin**

FIG. 1.18 – L'algorithme la propagation d'onde, adapté de [DT88].

La figure 1.19 reprend l'exemple de la figure 1.17, en utilisant cette fois le voisinage de Moore[7] et la distance Euclidienne comme métrique. On peut constater une nette différence dans progression du front d'onde entre les deux figures.

De manière générale, jouer sur le voisinage et/ou la métrique permet de contrôler la manière dont le front d'onde se propage dans l'environnement. Il est notamment possible de freiner l'expansion du front d'onde dans certaines parties de l'environnement (que l'on veut éviter) et de l'accélérer dans d'autres (que l'on veut privilégier). On parle alors d'expansion *non isotrope*[8].

[7] Les voisins de Moore d'une case C contient les 8 cases adjacentes à C.

[8] C'est grâce à ce principe d' anisotropie qu'il est possible tenir compte de l'influence de courants dans l'environnement, comme nous le verrons dans le chapitre 3.

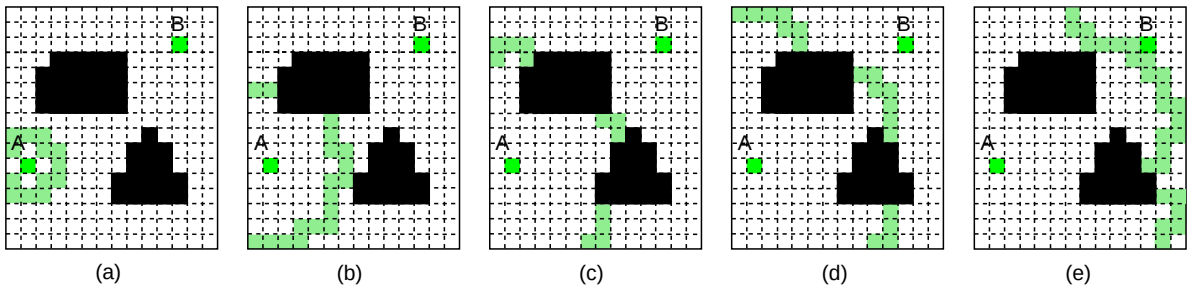

FIG. 1.19 – Propagation d'onde utilisant le voisinage de Moore et la distance Euclidienne

Dans la littérature, on retrouve le concept de propagation d'onde sous d'autres noms, dans des domaines totalement différents de la robotique. Par exemple, on peut la voir apparaître sous la dénomination *transformée de distance* en infographisme [Jar85], ou encore *Fast Marching* dans des applications de physique [Set96]. On peut également la voir comme une simple application de l'algorithme de plus court chemin de Dijkstra [Dij59].

S'il existe quelques différences dans leur propriétés, tous ces algorithmes ont strictement la même complexité temporelle (dans le pire des cas), en $O(N \log N)$, si N est le nombre de cases dans la grille.

Dans les applications pratiques, qui peuvent nécessiter des résolutions de grille importantes, N peut rapidement devenir élevé (de l'ordre de plusieurs millions), ce qui peut aboutir à des temps de calcul importants. Pour conserver de bonnes performances, une solution consiste à implémenter directement la propagation d'onde sur des systèmes matériels.

Glasius *et al.* [GKG95] ont été les premiers à implémenter la propagation d'onde sous la forme de cartes de Kohonen physiques. Cette idée a énormément été reprise par la suite. Les publications [RWHK97][YM01][LSR05] en sont quelques exemples. D'autres auteurs, comme Behring *et al.* [BBCM00] ont quant à eux opté pour des automates cellulaires.

4 Les métaheuristiques

Dans certaines méthodes évoquées ci-dessus, des *heuristiques* sont parfois utilisées. Il s'agit de règles empiriques privilégiant, en utilisant astucieusement les spécificités du problème, certaines directions dans l'espace de recherche. Une heuristique bien choisie permet de réduire, parfois de manière très importante, le nombre d'éléments explorés cet espace. L'ajout d'une heuristique dans un algorithme de recherche peut donc améliorer significativement ses performances. L'exemple le plus connu est l'évolution de l'algorithme de plus court chemin Dijkstra vers l'algorithme A^*, présenté page 27.

Il est important de ne pas les confondre avec les *métaheuristiques*, qui sont elles-mêmes des algorithmes de recherche à part entière. Leurs caractéristiques principales sont d'être stochastiques (i.e. basées sur un processus aléatoire) et génériques (i.e. conçues pour pouvoir être appliquées sur une large gamme de problèmes différents, sans nécessiter de changements profonds). Elles s'inspirent souvent de phénomènes naturels, exploitant des connaissances issues de la biologie de l'évolution (algorithmes génétiques) ou de l'éthologie (colonies de fourmis, essaims particulaires).

4.1 Les algorithmes génétiques

Les algorithmes génétiques s'inspirent du principe de sélection naturelle proposé par Darwin, selon lequel au fil du temps, les gènes conservés au sein d'une population donnée sont ceux qui sont les plus adaptés aux besoins de l'espèce vis à vis de son environnement.

Dans cette simulation informatique, proposée par Holland [Hol75], chaque individu de la population est une chaîne finie de gènes, représentant une solution du problème à résoudre. La population initiale est générée aléatoirement, puis des générations successives sont créées à l'aide du processus cyclique représenté ci-après.

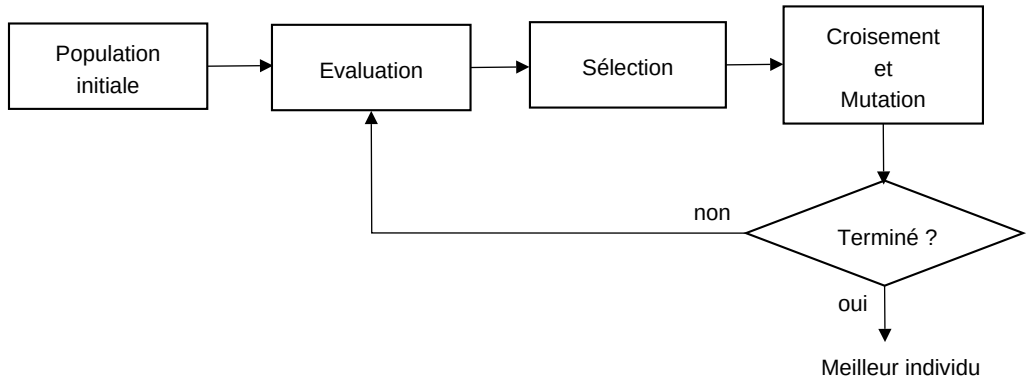

FIG. 1.20 – Organigramme d'un algorithme génétique standard

1. Dans la phase d'*évaluation*, les individus sont notés selon la qualité des solutions qu'ils représentent. Cette note est appelée *adaptation*.
2. Dans la phase de *sélection*, les meilleurs individus sont extraits de la population. La stratégie de sélection la plus ancienne est appelée "roue de la fortune". Elle consiste à attribuer à chaque individu une probabilité de sélection proportionnelle à son adaptation.
   Depuis, d'autres stratégies ont été proposées, comme la sélection par rang ou la sélection par tournoi.
3. Dans la phase de *croisement*, deux individus sont sélectionnés avec une probabilité p_c, (les parents), puis leurs gènes sont échangés pour donner naissance à deux nouveaux individus (les enfants). Le croisement le plus simple consiste à découper aléatoirement les individus en deux parties puis à échanger les moitiés. Cette phase est destinée à maintenir, voire enrichir la diversité de la population.
4. Dans la phase de *mutation*, les gènes des individus sont modifiés, avec une probabilité p_m, par l'intermédiaire d'une perturbation aléatoire. Cette phase a pour rôle de favoriser l'exploration de nouvelles régions de l'espace de recherche.

Le critère d'arrêt est généralement un nombre maximal de générations. La probabilité de croisement varie énormément selon les problèmes, alors que celle de mutation est généralement très faible (moins de 10%). Notons que seule la phase de mutation est nécessaire pour garantir la convergence vers une solution[9] (les croisements permettent de l'accélérer).

[9] On pourra ainsi noter l'existence d'une approche très similaire (et antérieure) aux algorithmes génétiques, appelée *stratégie d'évolution* [Rec73], est basée uniquement sur des mutations.

Dans le contexte spécifique de planification de chemin, un gène représente le plus souvent un point (x_i, y_i) de l'espace et un individu un chemin, reliant un nombre variable de gènes par des segments de droite. Un individu est noté $I = \langle (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) \rangle$.

L'adaptation d'un individu doit favoriser les chemins permettant d'atteindre le but en parcourant une longueur minimale et en évitant les obstacles. Pour cela, on peut par exemple utiliser une adaptation a de la forme [GFC05] :

$$a(I) = \frac{1}{\alpha \cdot l_{totale} + \beta \cdot d_{but} + \gamma \cdot n_{inter}} \tag{1.5}$$

où α, β, γ sont des coefficients de pondération, l_{totale} la longueur totale du chemin, d_{but} la distance Euclidienne au but, et n_{inter} le nombre de segments entrant en collision avec des obstacles.

Pour les opérateurs génétiques, on trouve dans la littérature deux grandes tendances : utiliser les opérateurs standards, travaillant sur les bits [SS97] (en représentant les chemins par des chaînes binaires) ou développer des opérateurs spécifiques. Dans cette dernière catégorie, on trouve notamment les opérateurs suivants [RKPC02][Tor04] :

- Croisement : un point est sélectionné au hasard dans les deux chemins et les parties situées d'un côté du point sont interverties,
- Mutation : un point du chemin est déplacé, ajouté ou supprimé.

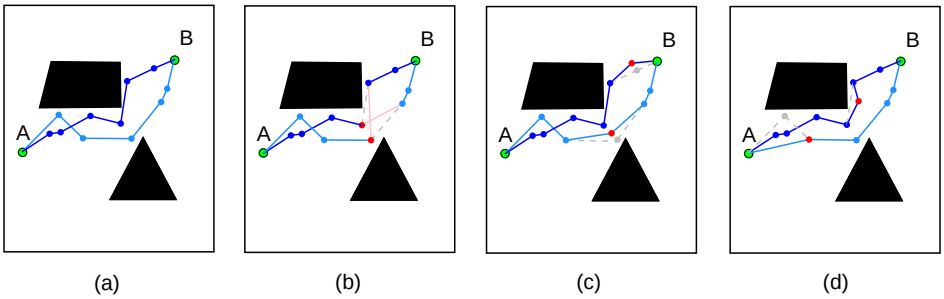

FIG. 1.21 – Opérateurs génétiques spécifiques aux chemins
La figure illustre l'effet des opérateurs génétiques sur deux chemins dessinés dans la partie (a). Partie (b) : effet de l'opérateur de croisement. Partie (c) : effet de l'opérateur de mutation, par déplacement de points de passage. Partie (d) : effet de l'opérateur de mutation, par ajout/suppression de points de passage.

Les algorithmes génétiques sont très appréciés pour leur flexibilité. En effet, ils ne tiennent compte de l'environnement que par le concept d'adaptation. Il suffit donc, a priori, d'en modifier l'expression pour intégrer de nouvelles contraintes.

Cependant, agréger le critère à optimiser et les contraintes dans une seule valeur entraîne une perte d'information importante, qui rend difficile le maintien des solutions dans l'espace valide[10]. En effet, la violation des contraintes peut être compensée par une très faible valeur du critère à optimiser.

[10] Dans un cadre plus général, ce problème se pose à chaque fois qu'il faut faire un choix parmi plusieurs objets en fonction d'un certain nombre de caractéristiques. Il est connu sous le nom de *décision multicritère*.

Par exemple, selon l'expression de l'adaptation donnée ci-dessus avec des paramètres égaux à 1, un chemin de longueur 10, atteignant le but sans collision aura une adaptation de $1/(10+0+0) = 0.1$, alors qu'un autre chemin, deux fois moins long, atteignant le but avec 2 collisions aura une adaptation supérieure, égale à $1/(5+0+2) \approx 0.14$. Ainsi, le deuxième chemin, invalide, a plus de chances de survivre que le premier !

Pour atténuer ce problème, deux approches ont été proposées :

1. Associer des sous-populations à chaque critère puis les combiner. C'est la philosophie de l'approche *VEGA* (*Vector Evaluated Genetic Algorithm*) [Sch85].

2. Utiliser la notion de Pareto-dominance, issue de l'économie [Mie99]. En considérant que la fonction f_i fournit une évaluation du critère i (parmi n), un individu I_1 domine I_2 si et seulement si :

$$\forall i \in [1,n] : \ f_i(I_1) \leq f_i(I_2) \ et \ \exists j \in [1,n] : f_j(I_1) < f_i(I_2) \tag{1.6}$$

   Les individus non dominés, formant le *front de Pareto*, sont considérés comme les plus adaptés. Notons que la construction de ce front nécessite l'évaluation de nombreux individus, et peut donc s'avérer particulièrement coûteuse.

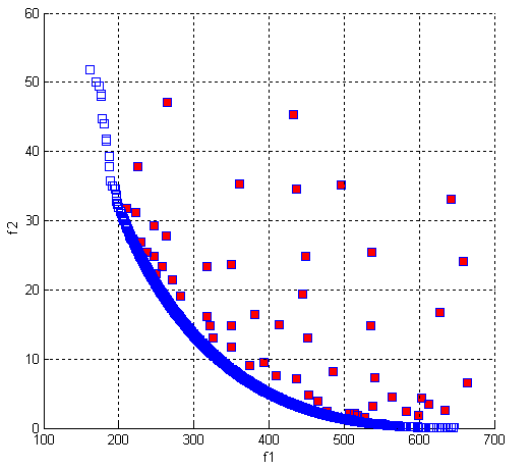

FIG. 1.22 – Front de Pareto
La figure illustre le front de Pareto obtenu pour deux critères (la longueur du chemin et le nombre de collisions), mesurés par $f1$ et $f2$. Les solutions dominées sont représentées en rouge et non dominées en bleu. Ces dernières constituent le front de Pareto.

   Dans cet esprit, les approches *NPGA* (*Niched Pareto Genetic Algorithm*) [HNG94] et *NSGA* (*Non-dominated Sorting Genetic Algorithm*) [SD94] ont été proposées.

Le deuxième problème majeur des algorithmes génétiques est la convergence prématurée vers un minimum local. Pour le prévenir, les améliorations suivantes peuvent être utilisées [Mic98] :

1. L'introduction d'une espérance de vie.

   Les individus sont considérés comme mortels, leur espérance de vie augmentant avec leur adaptation. Ceci permet d'assurer le renouvellement de la population.

2. Les îlots.

   Des sous-populations (appelées îlots ou dèmes) évoluent en parallèle avec des paramètres différents, potentiellement vers des solutions différentes. A une fréquence fixée (exprimée en nombre de générations, par exemple), certains individus migrent d'une sous-population à l'autre. Ceci permet de limiter les convergences prématurées tout en explorant l'espace de recherche dans plusieurs directions.

3. L'élitisme.

   A chaque génération, l'application des opérateurs peut altérer les meilleurs individus de la population. Afin de les préserver, le modèle élitiste les écarte du processus d'évolution. Concrètement, ils sont copiés avant les phases de mutation et de croisement, puis réinjectés dans la population.

4. Les niches.

   Une notion de distance entre individus est introduite, permettant d'identifier les individus présents dans un même voisinage, appelé *niche*. Plus il y a d'individus dans une même niche, plus ils sont dégradés. Ceci permet de pénaliser les individus trop semblables.

4.2 Les essaims particulaires

Alors que les algorithmes génétiques reposent sur un principe de compétition, les essaims particulaires s'appuient sur un principe de collaboration. Proposée par Kennedy et Ebenhart [KE95], cette technique met en jeu des entités élémentaires (les *particules*), qui s'influencent les unes les autres pour faire converger le groupe (l'*essaim*) vers un optimum.

Ce principe est inspiré du comportement des oiseaux se déplaçant en groupe vers des lieux d'intérêt (un point d'eau, par exemple).

Au départ de l'algorithme, sont attribuées aux particules (de façon aléatoire ou non) une position et un vecteur-vitesse. Puis, à chaque itération t (correspondant à un pas de temps), chaque particule P se déplace en fonction des composantes suivantes :

1. Sa position et sa vitesse précédente, $x(t-1)$ et $\vec{v}(t-1)$,
2. Sa meilleure position[11] occupée x^*,
3. La meilleure position occupée par l'essaim sans son voisinage x_g^*

[11] De façon analogue aux algorithmes génétiques, une note est attribuée à une position par une fonction d'évaluation. Des exemples seront donnés par la suite.

Ceci se traduit par les équations de mouvement suivantes :

$$\begin{aligned} \vec{v}(t) &= \omega \cdot \vec{v}(t-1) + \alpha \cdot r_1(t) \cdot (x^* - x(t-1)) + \beta \cdot r_2(t) \cdot (x_g^* - x(t-1)) \\ x(t) &= x(t-1) + \vec{v}(t) \end{aligned} \tag{1.7}$$

où ω, α et β sont des coefficients de pondération. ω définit l'inertie de la particule, tandis que α et β dosent l'importance de la mémoire individuelle par rapport à l'influence (locale) du groupe[12] . Les nombres $r_i(t)$ sont quant à eux des nombres aléatoires tirés dans l'intervalle $[0, 1]$.

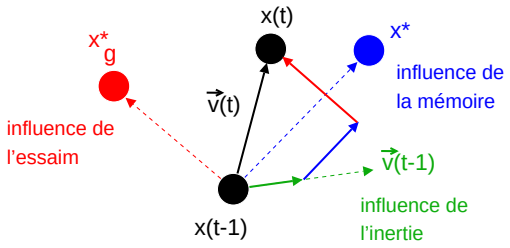

FIG. 1.23 – Décomposition du vecteur-vitesse d'une particule

Le vecteur-vitesse courant $\vec{v}(t)$ d'une particule P est influencée par son vecteur vitesse précédent $\vec{v}(t-1)$, sa meilleure position occupée x^* ainsi que par la meilleure position occupée par les particules voisines x_g^*.

Cette technique a été appliquée pour la première fois dans le cadre de la planification de chemin par Qin *et al* [QSLC04]. Dans cette application, une particule P est formée de n noeuds N_i se déplaçant librement sur des segments de droite, tracés perpendiculairement à l'axe Départ-Arrivée, à intervalles réguliers.

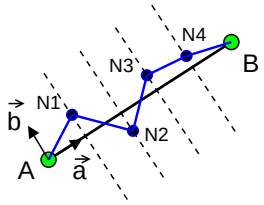

FIG. 1.24 – Particule pour la planification de chemin

La figure représente, en bleu, une modélisation possible pour une particule dans un contexte de planification de chemin.

Si on note A le point de départ et B le point d'arrivée, il est possible de construire un repère orthonormé local $(A, \vec{a}, \vec{b})$, vérifiant $\vec{a} = \overrightarrow{AB}/||\overrightarrow{AB}||$, $\vec{a} \perp \vec{b}$ et $||\vec{b}|| = 1$. Dans ce repère, la position x une particule peut être totalement représentée par la liste des ordonnées b_i des noeuds N_i :

$$x = [b_1, b_2, ..., b_n] \tag{1.8}$$

L'évaluation de la position x est quant à elle donnée par la fonction f définie par :

$$f(x) = \frac{d_{A,B}}{\sqrt{\frac{d_{A,B}^2}{n+1} + \sum_{i=1}^{n-1} (b_{i+1} - b_i)^2}} \tag{1.9}$$

[12] En d'autres termes, ces poids définissent un compromis entre les tendances sociales et cognitives de la particule.

où $d_{A,B}$ représente la distance la distance Euclidienne entre A et B (calculée une seule fois, au début de l'algorithme). Le dénominateur représente la longueur du chemin associé à la particule P.

Cette représentation est astucieuse car elle permet de représenter les particules de façon très compacte, diminuant ainsi les données stockées en mémoire et le nombre de calculs. Toutefois, des tests de collision (et des rectifications éventuelles) doivent être effectués à chaque déplacement de particule.

Pour supprimer ce problème, Zhao et Yan [ZY05] ont proposé d'utiliser une décomposition cellulaire exacte (technique présentée page 24) pour générer les supports des particules : ceux-ci sont les côtés des cellules n'appartenant pas à des obstacles.

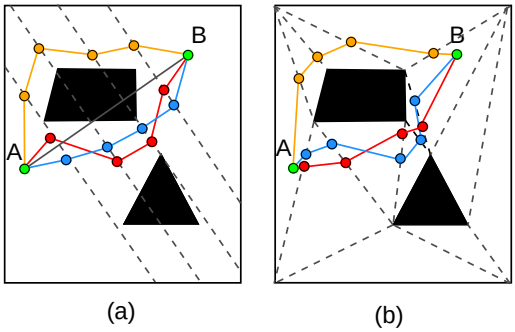

FIG. 1.25 – Différents types de particules
La figure compare illustre 3 particules dans un même environnement, en utilisant les modélisations de Qin (a) et Zhao (b). Dans (b), les tests de collision avec les obstacles sont inutiles.

Grâce à la représentation de Zhao, les tests de collision deviennent inutiles, puisque les particules évoluent uniquement dans l'espace valide, et les cellules sont de forme convexe (ce qui garantit l'appartenance des segments entre particules à l'espace valide).

De plus, il est possible de faire un prétraitement pour identifier les segments les plus susceptibles d'être le support du plus court chemin. Celui-ci consiste à construire le graphe de connectivité associé aux segments, puis d'appliquer un algorithme de type A^*, comme expliqué dans la section 1.2.

Notons enfin que des tests [ES98] réalisés sur un panel de fonctions de référence en optimisation ont permis de constater que les essaims particulaires nécessitaient un effort calculatoire moins important que les algorithmes génétiques pour obtenir des solutions de qualité équivalente. Cependant, il n'existe pas, à notre connaissance, de comparaison de ce type dans le cas particulier de la planification de chemin.

4.3 Les colonies de fourmis

Dans les algorithmes génétiques et les essaims particulaires, la solution optimale est directement représentée par un individu. Le principe des algorithmes de recherche est alors de faire tendre toute une population vers cet individu idéal, par compétition ou par collaboration.

Colorni *et al.* [CDM91] ont proposé une approche d'un autre genre, appelé optimisation par colonies de fourmis. Dans celle-ci, la solution optimale *émerge* progressivement de l'espace de recherche, par l'action des individus.

Cette approche s'inspire du comportement des fourmis recherchant un chemin allant de leur colonie à une source de nourriture, illustré dans la figure ci-après.

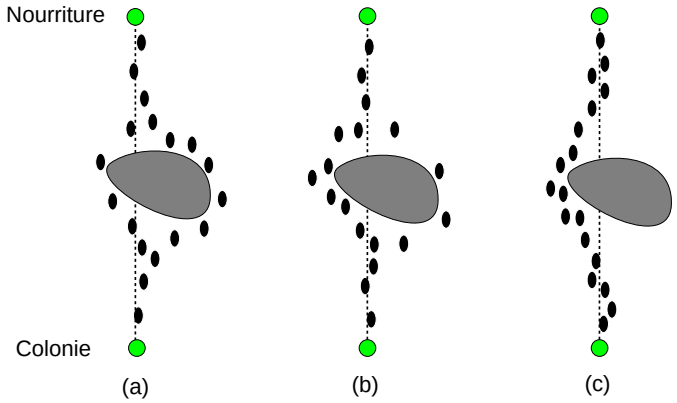

FIG. 1.26 – Construction d'un chemin par une colonie de fourmis
La figure détaille l'établissement progressif d'un chemin entre la colonie et une source de nourriture. Au départ, la distribution de phéromones est uniforme, l'obstacle est contourné de façon indifférente. Au fil du temps, le chemin le plus court est progressivement renforcé, et devient le seul utilisé.

1. Au départ, les fourmis explorent aléatoirement l'environnement autour de la colonie. Si elles trouvent une source de nourriture, elles rentrent au nid (en utilisant des repères), en prenant soin de laisser sur leur chemin une piste de phéromones.

2. Ces phéromones étant attractives, les fourmis passant à proximité auront tendance à suivre cette piste, de façon plus ou moins directe.

3. Si plusieurs pistes sont possibles pour atteindre la même source de nourriture, la plus courte sera, dans le même temps, celle parcourue par plus de fourmis. Ainsi, cette piste sera la plus renforcée par des phéromones.

4. Avec le temps, la piste la plus courte devient la plus attractive. Les autres finissent par disparaître, les phéromones étant volatiles.

Le premier algorithme basé sur ce principe, *Ant System* [DMC96], a été proposé dans le but de réaliser une recherche plus court chemin dans un graphe, trop vaste pour appliquer les méthodes classiques (l'algorithme A^* notamment). A chaque noeud de ce graphe est associé une quantité de phéromones.

Au départ, la même quantité de phéromones est déposée sur tous les noeuds, et un nombre fini de fourmis est placé sur le noeud de départ. Ensuite, les fourmis se déplacent au sein du graphe, par relation de voisinage, en respectant les règles suivantes :

– Chaque noeud ne peut être visité qu'une seule fois.

– La probabilité de sélection d'un noeud non visité augmente avec la quantité de phéromones déposée, et diminue avec la distance. Formellement, au noeud i, la probabilité de sélectionner un noeud j est donnée par :

$$p_{ij} = \frac{\tau_j^\alpha / d_{ij}^\beta}{\sum_{k \neq j} \tau_k^\alpha / d_{ik}^\beta} \tag{1.10}$$

α et β étant des paramètres, d_{ij} la distance Euclidienne entre i et j et τ_j la quantité de phéromones associée sur le noeud j.

Lorsqu'une fourmi atteint le noeud d'arrivée, une quantité de phéromones, inversement proportionnelle à la distance parcourue, est déposée sur les noeuds visités. La fourmi est quant à elle replacée au noeud de départ. L'évaporation des phéromones au cours du temps est simulée à l'aide d'un paramètre $\rho \in [0, 1]$:

$$\forall i : \ \tau_i(t+1) = \rho \cdot \tau_i(t) \tag{1.11}$$

Zlochin *et al.* [ZBMD04] ont montré que les algorithmes de ce type pouvaient être assimilés à une variante de la descente du gradient, dite *stochastique*.

Pour les appliquer au problème de planification de chemin, il suffit de discrétiser l'espace initial. Le plus souvent, cette discrétisation est réalisée à l'aide de cellules régulières, comme dans la figure 1.27, mais toutes les méthodes décrites dans les sections 1.1 ou 2 sont utilisables.

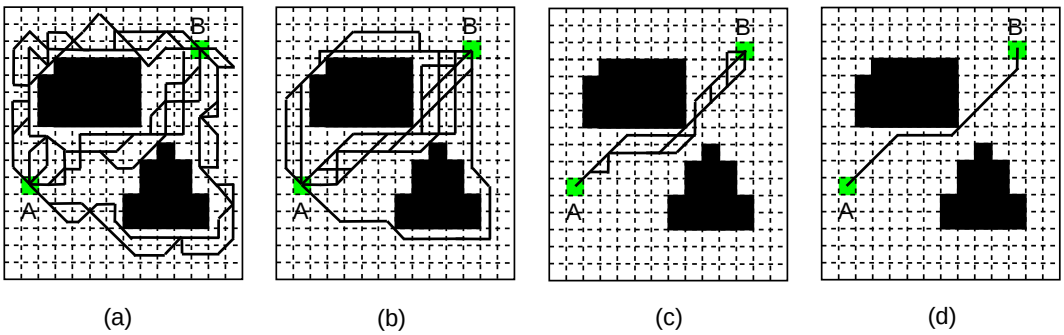

FIG. 1.27 – Évolution d'une colonie de fourmis sur des cellules régulières
La figure présente l'évolution au cours du temps, des chemins empruntés par 8 fourmis dans un espace discrétisé à l'aide de cellules régulières, formant une grille de taille 16×14. Le chemin présenté figure (d) finit par être le seul emprunté.

Les algorithmes d'optimisation à base de colonies de fourmis n'ont pas le problème lié à la présence de plusieurs critères. D'abord, parce les fourmis ne se déplacent que dans l'espace valide ; ensuite parce que seuls les chemins permettant d'arriver au point but sont renforcés.

Grâce à ces deux propriétés, seule la longueur du chemin est prise en compte dans la quantité de phéromones déposée. Il n'y a plus de critère lié au nombre de collisions, ni à la distance au point but (tous deux égaux à 0).

Tout comme les algorithmes génétiques, les algorithmes d'optimisation à base de colonies de fourmis peuvent souffrir d'une vitesse de convergence faible. Pour y remédier, différentes stratégies ont été envisagées :

- Fan *et al.* [FLY+03] ont ajouté, dans une version dite *intensifiée*, une étape de simplification des chemins, consistant remplacer des successions de mouvements élémentaires par des mouvements directs en ligne droite.
- Karimifar [KJ04] a introduit le concept de *conversation* permettant aux fourmis de combiner des chemins partiels pour en produire de nouveaux. Cette démarche rappelle les opérateurs de croisement présents dans les algorithmes génétiques.
- Mei *et al.* [MTZ05] ont suggéré de ne renforcer que les meilleurs chemins parmi n (forme d'élitisme) et d'étendre ce renforcement à leur voisinage (renforçant ainsi une bande centrée sur le chemin).

5 Résumé

Le tableau ci-après résume les caractéristiques de l'ensemble des méthodes de planification de chemin évoquées dans ce chapitre, dans un espace $2D$ contenant des obstacles de forme polygonale.

Une méthode est qualifiée de *déterministe* si elle planifie à coup sur le même chemin dans deux espaces identiques.

Elle est dite *optimale* si le chemin retourné est de longueur minimale dans son espace de travail. Par exemple, les méthodes de décomposition effectuent une recherche de plus court chemin dans un espace discrétisé. Le chemin retourné est bien de longueur minimale dans cet espace, mais pas forcement dans l'espace initial.

Enfin, les *paramètres* indiqués concernent les algorithmes de base pour chaque méthode et non leurs extensions, qui peuvent éventuellement en introduire d'autres.

Méthode		Déterministe	Optimale	Paramètres
Méthodes de décomposition	Exacte	oui	oui	aucun
	Approchée	oui	oui	résolution grille
Méthodes probabilistes	PRM	non	non	nb échantillons, taille voisinage
	PCD	non	non	nb échantillons
	RRT	non	non	longueur pas
Potentiels artificiels	Analytiques	oui	non	coeffs champs
	Numériques	oui	oui	résolution grille
Métaheuristiques	Algorithmes génétiques	non	non	nb générations types opérateurs, taille population,
	Colonies de fourmis	non	non	nb itérations nb fourmis, coeffs phéromones
	Essaims particulaires	non	non	nb particules voisinage, coeffs vect. vitesse

Dans ce tableau, nous pouvons voir deux grandes catégories : les méthodes déterministes, le plus souvent discrètes, et les méthodes non déterministes, le plus souvent continues.

– Les méthodes déterministes correspondent principalement aux méthodes de décomposition. Elles sont optimales (dans l'espace discrétisé) et exécutables en un temps polynomial.

  Pour celles-ci, il faut faire deux choix : le type de discrétisation (exacte ou approchée) et éventuellement la résolution choisie. Ces choix dépendent beaucoup de la distribution et des caractéristiques géométriques des obstacles (forme, taille). En effet, il serait totalement incongru de discrétiser un espace ne contenant qu'un seul obstacle rectangulaire au moyen d'une

grille. Par contre, le procédé semble plutôt avantageux pour 50 obstacles de forme complexe. La résolution de la grille va alors principalement être fonction de l'éloignement moyen des obstacles. Comme expliqué dans la section 1.1, une résolution trop grossière peut obstruer un grand nombre de passages, voire la totalité.

De plus, ces méthodes étant basées sur un algorithme de recherche de type A^*, un travail important peut être nécessaire pour trouver une heuristique efficace. Notamment, la recherche d'une heuristique consistante peut s'avérer particulièrement ardue (voire impossible) pour certains problèmes.

– Les méthodes non déterministes regroupent les méthodes probabilistes et les métaheuristiques. Ces méthodes présentent deux avantages : elles sont peu sensibles aux dimensions de l'espace de recherche (puisqu'elles procèdent à un échantillonnage et non à une discrétisation) et peuvent être stoppées à tout moment (garantissant un temps de réponse fixé). Toutefois, elles n'apportent aucune garantie sur la qualité ni même sur la validité de la solution fournie. De plus, la plupart présentent un temps de convergence long vers la solution optimale et nécessitent le réglage d'un grand nombre de paramètres. Ce réglage peut être effectué de façon empirique ou éventuellement par apprentissage. Dans les deux cas, cette tâche est particulièrement fastidieuse. De plus, elle est généralement à recommencer en cas de changements (même minimes) sur les caractéristiques du problème.

En résumé, pour la planification de chemin dans un espace de dimension 2, les méthodes de décomposition semblent les plus adaptées.

Toutefois, une augmentation du nombre de dimensions de l'espace entraîne une explosion du nombre d'éléments générés par l'étape de discrétisation, et donc du temps de planification. Or, dans le contexte d'une planification de trajectoire, une dimension supplémentaire est ajoutée aux deux dimensions spatiales initiales : la dimension temporelle.

Dès lors, les méthodes de décomposition deviennent peu efficaces pour résoudre le problème en totalité, et sont généralement utilisées en résoudre une partie (par exemple, pour moduler la vitesse du mobile sur un chemin pré-établi).

Les méthodes non déterministes prennent donc tout leur intérêt, puisqu'elles sont capables de traiter le problème en intégralité en un temps raisonnable. C'est ce que nous allons voir dans le chapitre suivant.

Chapitre 2

La planification de trajectoire

Sommaire

Etant données deux configurations A et B, le problème de planification de trajectoire consiste à trouver simultanément :

- Un chemin γ, entièrement contenu dans l'espace valide $\mathcal{C}_{valide}$ et reliant A et B,
- A chaque instant t, la position du mobile sur γ.

Formellement parlant, la trajectoire recherchée est définie par le graphe d'une fonction continue $\vartheta : t \in [0, T] \mapsto (x, y) \in \mathcal{C}_{valide}$ vérifiant $\vartheta(0) = A$ et $\vartheta(T) = B$. La fonction ϑ^* minimisant la durée T est appelée trajectoire optimale.

Les méthodes les plus évoquées dans la littérature pour résoudre ce problème ont été répertoriées dans le diagramme de classes de la figure 2.1. Ces méthodes se divisent en deux grandes classes : les méthodes globales et les méthodes locales. Dans ce chapitre, une section est consacrée à chacune de ces classes.

1 Les méthodes globales

Les méthodes globales tiennent compte du mouvement des obstacles dans leur intégralité pour planifier une trajectoire, en une seule passe (méthodes directes) ou en plusieurs passes (méthodes indirectes).

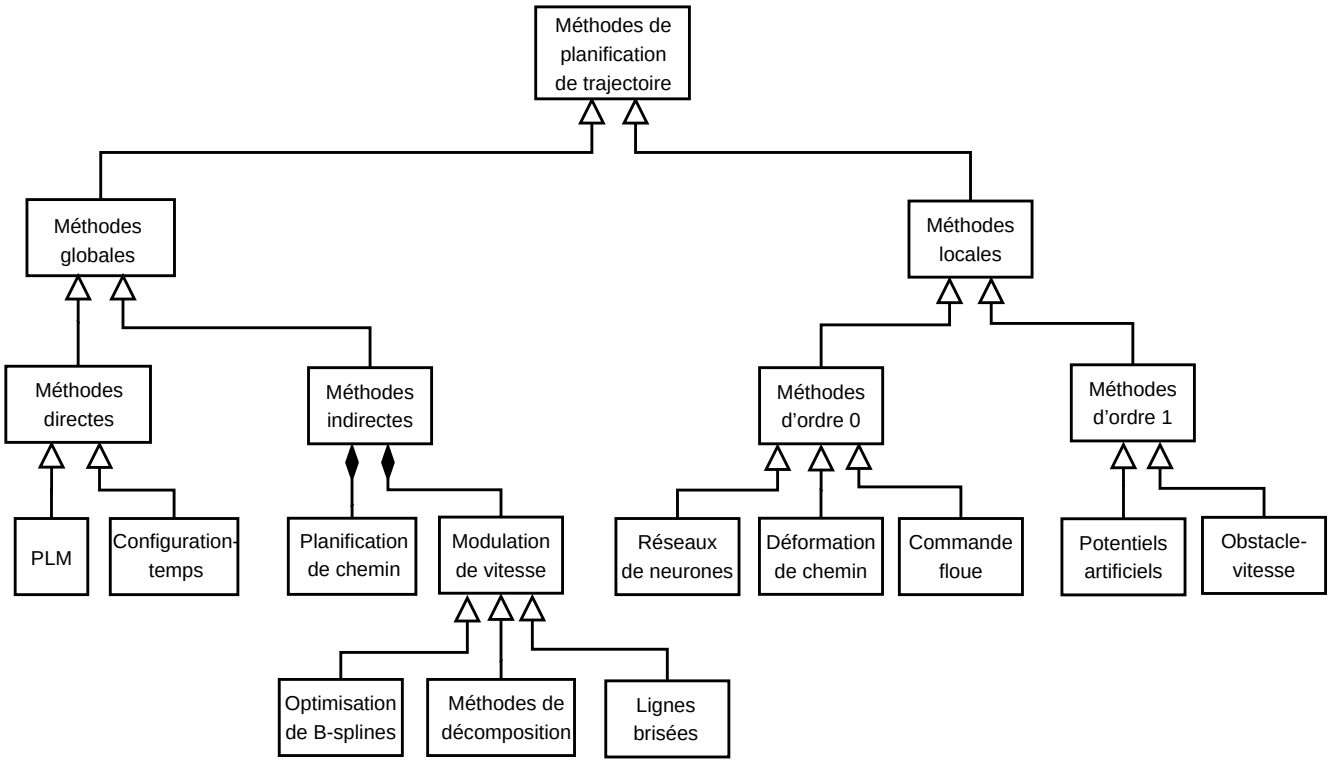

FIG. 2.1 – Méthodes de planification de trajectoire
Les triangles blancs symbolisent des liens d'héritage, et les losanges noirs des liens de composition.

1.1 Les méthodes directes

1.1.1 L'approche configuration-temps

L'approche configuration-temps, introduite dans [ELP86], consiste à construire l'espace des configurations-temps, possédant 3 dimensions : les deux dimensions de l'espace des configurations initial, plus une dimension temporelle.

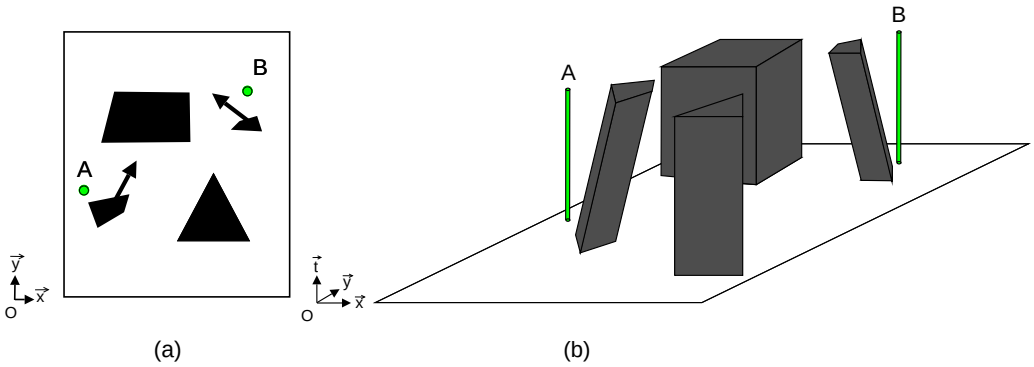

FIG. 2.2 – Espace des configurations-temps
En ajoutant une dimension supplémentaire à l'espace des configurations initial (a), de dimension 2, on obtient l'espace des configurations-temps (b), de dimension 3, contenant uniquement des volumes statiques.

Cette représentation est très commode, car tous les éléments de l'environnement, fixes et mobiles, sont modélisés par des volumes. Toutefois, une discretisation déterministe (à base des méthodes de décomposition) de cet espace entraîne généralement un nombre important d'éléments.

Ainsi, on trouve essentiellement dans la littérature l'utilisation de métaheuristiques ou de méthodes probabilistes, procédant toutes deux à un échantillonnage aléatoire de l'espace des configurations-temps. Parmi elles, on peut citer la méthode PRM [HKLR02] et les algorithmes génétiques [GCFR98][FXGC05].

Pour pouvoir appliquer ces méthodes, les fonctions de coût sont modifiées pour intégrer la notion de durée. Par exemple, dans [GCFR98], le coût d'une trajectoire représentée par l'individu $I = \langle (x_1, y_1, t_1), ..., (x_n, y_n, t_n) \rangle$ est donné par :

$$C(I) = \alpha \cdot d((x_n, y_n), B) + \beta \cdot t_n \tag{2.1}$$

où $d(X, B)$ désigne la distance entre un point X et le point but B ; α et β sont des coefficients de pondération.

1.1.2 La programmation linéaire mixte

En alternative à l'approche précédente, Richards et How [RH02] ont proposé de reformuler le problème de planification de trajectoire en un Programme Linéaire Mixte (PLM). Leur démarche est la suivante :

Une durée maximale, appelée *horizon* et notée T, est fixée pour atteindre le point d'arrivée. L'intervalle $[0, T]$ est divisé en utilisant un pas constant ρ. A chaque pas de temps i sont associés les variables suivantes :

- $p_i = (x_i, y_i) \in \mathbb{R}^2$, indiquant la position du mobile,
- $v_i = (v_x^i, v_y^i) \in \mathbb{R}^2$ indiquant la vitesse du mobile,
- $d_i \in \{0, 1\}$ indiquant si le mobile se déplace (le mobile étant à l'arrêt quand le point d'arrivée est atteint).

Le problème consiste à minimiser le nombre de pas de temps où le mobile se déplace. Le PLM en résultant est donc :

$$\min_{p_i, v_i} \sum_{i=1}^{T} d_i \tag{2.2}$$

Sous les contraintes suivantes :

1. Conditions de départ :
   Si $A = (x_A, y_A)$ désigne le point de départ, on a :

$$\begin{array}{rcl} x_1 & = & x_A \\ y_1 & = & y_A \\ d_1 & = & 1 \end{array} \tag{2.3}$$

   Ces contraintes forcent le mobile à démarrer en A.

2. Conditions d'arrivée :
   Si $B = (x_B, y_B)$ désigne le point d'arrivée, on a :

$$\begin{array}{rrcl} \forall i \in [0,T] : & x_i & \leq & x_B + M \cdot d_i \\ & x_i & \geq & x_B - M \cdot d_i \\ & y_i & \leq & y_B + M \cdot d_i \\ & y_i & \geq & y_B - M \cdot d_i \end{array} \tag{2.4}$$

   où M désigne un grand nombre positif.
   Ces contraintes traduisent l'équivalence $p_i = B \Leftrightarrow d_i = 0$, c'est à dire que le mobile est à l'arrêt au point d'arrivée.

3. Evitement des obstacles :
   Les obstacles sont modélisés par des rectangles R_j, définis par deux coins : inférieur $C^j_{inf} = (x^j_{inf}, y^j_{inf})$ et supérieur $C^j_{sup} = (x^j_{sup}, y^j_{sup})$. L'évitement de R_j est imposé par les contraintes suivantes :

$$\begin{array}{rrcl} \forall k \in [0,4] : & r^k_j & \in & \{0,1\} \\ \forall i \in [0,T] : & x_i & \leq & x^j_{inf} + M \cdot r^1_j \\ & -x_i & \leq & x^j_{sup} + M \cdot r^2_j \\ & y_i & \leq & y^j_{inf} + M \cdot r^3_j \\ & -y_i & \leq & y^j_{sup} + M \cdot r^4_j \\ & \sum_{k=1}^{4} r^k_j & \leq & 3 \end{array} \tag{2.5}$$

   où M désigne un grand nombre positif.

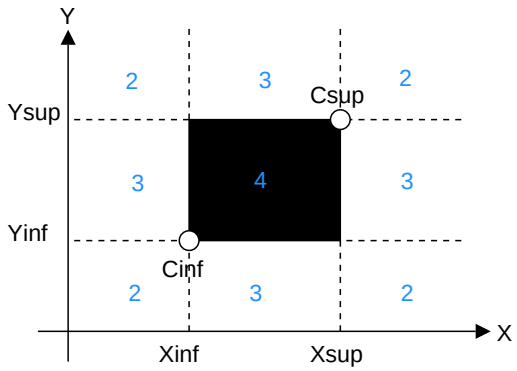

FIG. 2.3 – Modélisation d'un obstacle dans un PLM
La figure présente, en noir, un obstacle rectangulaire (défini par ses coins C_{inf} et C_{sup}) et en bleu le nombre de relaxations r nécessaires dans les contraintes définies dans l'équation 2.5, selon la position du mobile. On a $r \leq 3$ dans les 8 zones extérieures à l'obstacle, et $r > 3$ dans la zone intérieure. La condition $r \leq 3$ permet donc à garantir l'évitement de l'obstacle.

   Les variables binaires r^k_j permettent de relaxer les contraintes auxquelles elles appartiennent. (si $r^k_j = 1$, la contrainte correspondante est toujours vérifiée). Pour chaque obstacle, 3 relaxations sont autorisées, ce qui garantit que le mobile est à l'extérieur de l'obstacle dans au moins une direction.

4. Vitesse maximale :

La vitesse v_i du mobile possède une borne supérieure v_{max}. Cette contrainte se traduit naturellement par :

$$\forall i \in [0,T] : {v_x^i}^2 + {v_y^i}^2 \leq v_{max}^2 \tag{2.6}$$

Géométriquement parlant, cette contrainte se traduit par l'appartenance du point $v_i = (v_x^i, v_y^i)$ au disque de rayon v_{max}. Cependant, cette contrainte n'est pas linéaire. Les auteurs proposent donc de linéariser le cercle par un polygone à M facettes, ce qui correspond aux contraintes suivantes :

$$\forall i \in [0,T] \forall m \in [1,M] \quad v_x^i \sin\left(\tfrac{2\pi n}{M}\right) + v_y^i \cos\left(\tfrac{2\pi n}{M}\right) \leq v_{max} \tag{2.7}$$

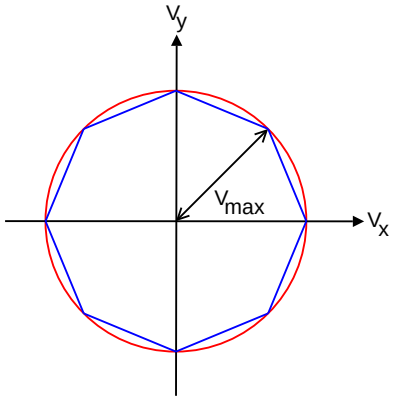

FIG. 2.4 – Linéarisation de la contrainte de vitesse
La figure représente la linéarisation du cercle de rayon v_{max}, matérialisant la limite de vitesse du mobile, par un octogone. L'octogone ainsi obtenu constitue une approximation intérieure du cercle.

Cette reformulation en PLM est particulièrement commode, puisqu'elle permet :

1. L'utilisation des logiciels d'optimisation du marché (CPLEX notamment), évitant l'implémentation d'un planificateur spécifique ainsi que sa maintenance,
2. L'ajout aisé de nouvelles contraintes dans le modèle : limite sur l'angle de braquage, coordination de plusieurs mobiles, etc.

Toutefois, la résolution d'un PLM est en général un problème NP-difficile [Vaz01]. Cette difficulté vient du fait que certaines variables doivent être entières. Cette contrainte n'est pas naturelle pour les programmes linéaires, qui sont classiquement résolus dans des domaines continus. Pour la respecter, les domaines de ces variables sont relaxés à des domaines continus, puis progressivement réduits. Toutefois, chaque réduction s'accompagne de la résolution d'un programme linéaire, ce qui peut s'avérer particulièrement coûteux en temps de calcul. Les principales méthodes utilisées sont le branch & bound [LD60] et les plans de coupe [Dan60]. Ce caractère NP-difficile n'est pas surprenant, puisque le problème initial l'était, et que l'écriture d'un PLM n'en est qu'une simple reformulation. Il peut rendre cette méthode très lente sur des exemples pathologiques.

Un autre inconvénient de l'approche est son manque d'expressivité. En effet, elle impose d'utiliser des contraintes linéaires uniquement, ce qui restreint considérablement description les caractéristiques du robot et de son environnement. Notamment, le profil de consommation, est généralement modélisé par des fonctions quadratiques.

Pour ces raisons, Wei *et al.* ont récemment proposé dans [WZD08] d'utiliser la Programmation Quadratique Séquentielle (PQS) plutôt que la PLM. L'approche apporte gain important en expressivité, en dégradant toutefois sensiblement les performances. Sur des instances similaires à Richards et How, les auteurs obtiennent en effet des temps de calcul environ 50% supérieurs.

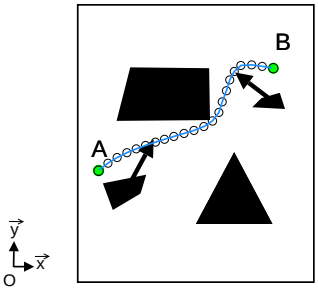

FIG. 2.5 – Trajectoire obtenue par résolution d'un PLM
La figure présente la trajectoire obtenue en modélisant le problème sous la forme d'un PLM, en utilisant 20 pas de temps (cercles noirs).

La trajectoire ci-avant étant quasi-optimale (aux approximations linéaires près), nous l'utiliserons comme référence pour juger (de façon qualitative) de la qualité des solutions fournies par les autres méthodes.

1.2 Les méthodes indirectes

Les méthodes indirectes sont dues à Kant et Zucker [KZ86], qui ont proposé pour la première fois de décomposer le problème de planification de trajectoire en deux sous-problèmes : une planification de chemin et une modulation de la vitesse du mobile sur le chemin planifié.

Cette décomposition permet de résoudre le problème de planification de trajectoire en un temps polynomial, puisque les deux sous-problèmes peuvent l'être[1]. Ceci se fait au prix de la complétude : certaines situations aboutiront à un échec alors qu'il existait une solution.

Par exemple, si le chemin planifié conduit à un face à face entre mobile et obstacle mobile, il est possible que les vitesses empruntables ne permettent pas d'éviter la collision (notamment, certains engins volants ne peuvent pas s'arrêter), comme illustré dans la figure ci-après.

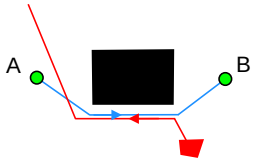

FIG. 2.6 – Exemple non soluble par modulation de vitesse
Dans cet exemple, le mobile, se déplaçant (avec une vitesse minimale non nulle) sur sa trajectoire bleue, fait face à un obstacle mobile de manière prolongée (de trajectoire rouge). La modulation de vitesse ne permet pas d'éviter la collision. Une replanification du chemin initial (contournant l'obstacle fixe noir par le haut, par exemple), est nécessaire.

[1] En effet, les deux sous-problèmes consistent à planifier un chemin dans un espace 2D (le premier dans l'espace des configurations, le deuxième dans un espace-temps).

Les méthodes indirectes associent une étape de calcul à chaque sous-problème :

1. L'étape de planification de chemin consiste à ignorer les obstacles mobiles et à appliquer l'une des méthodes présentées dans le chapitre 1,

2. L'étape de modulation de vitesse consiste à construire un espace-temps à deux dimensions dans lequel sont majoritairement appliquées des méthodes de décomposition. Plus anecdotiquement, on trouve aussi l'utilisation de méthodes spécifiques, comme l'optimisation de B-splines [Bor88] ou l'algorithme des *lignes brisées* [ST06].

La partie planification de chemin ayant déjà été traitée dans le chapitre 1, cette section décrit uniquement les méthodes de modulation de vitesse. Pour ce faire, nous commençons par présenter l'espace-temps qui leur sert de support.

1.2.1 La construction de l'espace-temps

Construire un espace-temps permet de reformuler le problème de modulation de vitesse en un problème de planification de chemin.

Cet espace-temps possède deux dimensions :

1. L'abscisse curviligne s parcourue sur le chemin, évoluant dans l'intervalle $[0, L]$, où L est la longueur du chemin,

2. Le temps écoulé t, évoluant dans l'intervalle $[0, T]$, T étant la date maximale pour atteindre le point d'arrivée (imposée par la quantité d'énergie embarquée ou par un impératif opérationnel, par exemple).

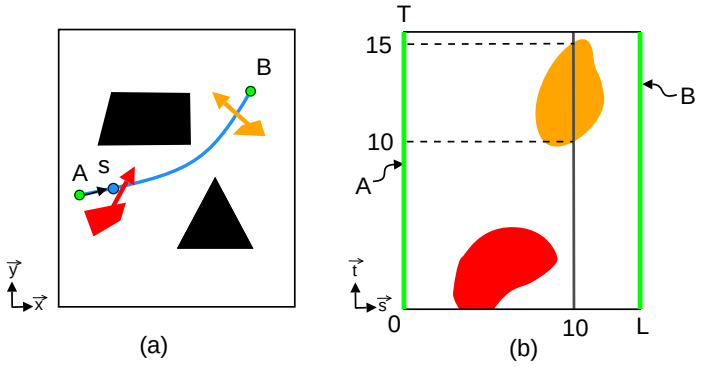

FIG. 2.7 – Construction d'un espace-temps pour la modulation de vitesse
Dans cet espace temps à deux dimensions (l'abscisse curviligne s parcourue sur le chemin et le temps écoulé t), les points de départ et d'arrivée génèrent les colonnes vertes, et les obstacles mobiles les surfaces rouge et orange.

Dans cet espace-temps :

– Chaque position sur le chemin est représentée par une colonne d'épaisseur infinitésimale. En particulier, les points de départ A et d'arrivée B, sont représentés par des colonnes situées aux extrêmes gauche et droite (i.e à $s = 0$ et $s = L$),

– Les obstacles mobiles sont représentés par des surfaces fermées de forme quelconque. Ces surfaces définissent les instants interdits pour chaque abscisse curviligne sur le chemin[2]. Par exemple, dans la figure ci-après, l'abscisse curviligne 10 est interdite entre les dates 10 et 15.

Le but de la modulation de vitesse est alors de déterminer un chemin entre la colonne de départ et la colonne d'arrivée en évitant les zones interdites, que nous appellerons obstacles. Pour ce faire, les méthodes de décomposition sont majoritairement utilisées, principalement basées sur un graphe de visibilité ou des cellules régulières. Quelques méthodes spécifiques ont également été proposées. L'ensemble de ces méthodes est décrit dans les sections suivantes.

1.2.2 Les méthodes de décomposition

Les méthodes de décomposition présentées dans la section 1 du chapitre 1 sont utilisables dans l'espace-temps introduit ci-avant, avec quelques modifications :

– A tout moment, les points accessibles appartiennent au *secteur des vitesses valides* (représenté en bleu dans la figure 2.8a), délimité par les vitesses minimale et maximale du mobile. Ceci modifie la notion de voisinage d'un élément de l'espace (qui se retrouve restreint).

– Une fonction de coût spécifique à l'espace-temps doit être développée[3], puisque la seule notion de durée ne suffit pas à discriminer les chemins. Par exemple, dans la figure 2.8b, les 3 chemins représentés sont temporellement équivalents (puisqu'ils permettent au mobile d'arriver au même moment), mais ils ne le sont pas d'un point de vue énergétique (par exemple, contrairement aux deux autres, le chemin bleu ne présente pas de variations de vitesse, généralement coûteuses en énergie).

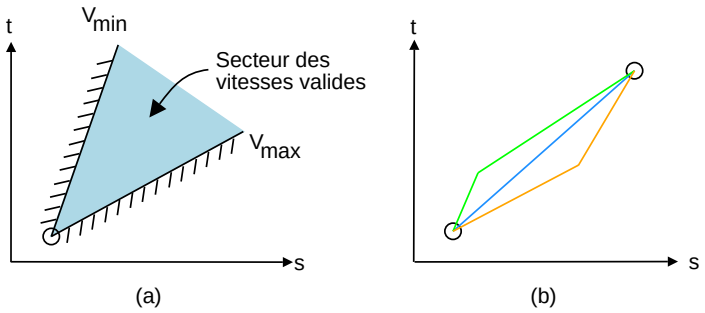

FIG. 2.8 – Particularités liées à l'espace-temps
(a) secteur des vitesses valides, délimité par les vitesses minimale et maximale du mobile (notées (v_{min} et v_{max} respectivement) ; (b) équivalence temporelle des chemins (les 3 chemins permettent d'arriver au même moment, mais avec des dépenses énergétiques différentes)

Par exemple, dans [JLH02], le coût c entre deux points $M_i = (s_i, t_i)$ et $M_{i+1} = (s_{i+1}, t_{i+1})$ est un compromis entre arrivée au plus tôt et le maintien d'une vitesse constante de la vitesse :

$$c(M_i, M_{i+1}) = w_1 \cdot t_{i+1} + w_2 \cdot (v_{i+1} - v_i) \tag{2.8}$$

où w_1 et w_2 sont des coefficients de pondération et v_i représente la vitesse du mobile au point M_i ($v_i = (s_i - s_{i-1})/(t_i - t_{i-1})$).

[2]Une abscisse curviligne est interdite si elle est occupée en même temps par le mobile et l'obstacle.
[3]Ce problème est également présent dans l'approche configuration-temps, présenté dans la section 1.1.1 page 50.

Les méthodes de décomposition les plus utilisées sont basées sur un graphe de visibilité [FS90][HCL99] ou sur des cellules régulières [Fra93][vdBO05].

Dans le cas du graphe de visibilité, les obstacles sont grossièrement approximés par des rectangles. Comparé à un graphe de visibilité classique, un grand nombre d'arcs est éliminé : tous les arcs se dirigeant vers le bas, ainsi que tous ceux partant d'un sommet uniquement atteignable par un déplacement horizontal (vitesse infinie).

Dans le cas des cellules régulières, l'abscisse curviligne et le temps sont discrétisés en utilisant des pas constants (δs et δt respectivement). Les seuls déplacements possibles d'une case à une autre consistent à attendre (vecteur $(0, \delta t)$), avancer (vecteur $(\delta s, \delta t)$) ou éventuellement reculer (vecteur $(-\delta s, \delta t)$). Ce nombre limité de déplacements aboutit à des chemins de moins bonne qualité (i.e. arrivant plus tard) que le graphe de visibilité. Pour obtenir une qualité équivalente, il faut choisir un pas de temps δt infinitésimal, ce qui aboutit à une explosion du nombre de cellules.

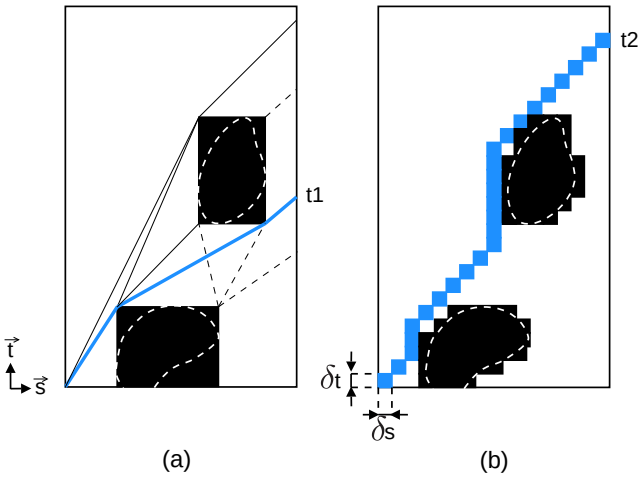

FIG. 2.9 – Modulation de vitesse par des méthodes de décomposition
La figure présente les chemins obtenus dans l'espace-temps de la figure 2.7 par deux méthodes de décomposition : (a) le graphe de visibilité (élagué des arcs en pointillés), permettant d'arriver à la date t_1 et (b) les cellules régulières, permettant d'arriver à la date t_2. On constate que $t_2 > t_1$. Notons que $t_2 \to t_1$ si $\delta t \to 0$.

1.2.3 L'optimisation de B-splines

Étant donnés m réels t_j dans l'intervalle $[0, 1]$, appelés *noeuds* et vérifiant $0 \leq t_1 \leq ... \leq t_m$, une B-spline S est une combinaison linéaire de n splines de base $b_{i,d}$ de degré d, pondérée par des *points de contrôle* P_i :

$$S(t) = \sum_{i=1}^{n} P_i \cdot b_{i,d}(t) \tag{2.9}$$

Les splines de base $b_{i,d}$, de degré $d = m-n$, sont définies par les relations de récurrence suivantes :

$$b_{i,0}(t) = \begin{cases} 1 & si \quad t_j \leq t < t_{j+1} \\ 0 & sinon \end{cases} \tag{2.10}$$

$$b_{i,d}(t) = \frac{t - t_i}{t_{i+p} - t_i} \cdot b_{i,d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}} \cdot b_{i+1,d-1}(t) \tag{2.11}$$

Le plus souvent, on utilise des splines cubiques uniformes. La dénomination *cubique* traduit que $d = 3$ et *uniforme* que les noeuds t_j sont équidistants dans l'intervalle $[0, 1]$. Elles sont représentées ci-dessous pour $n = 6$ points de contrôle (impliquant $m = 9$ noeuds).

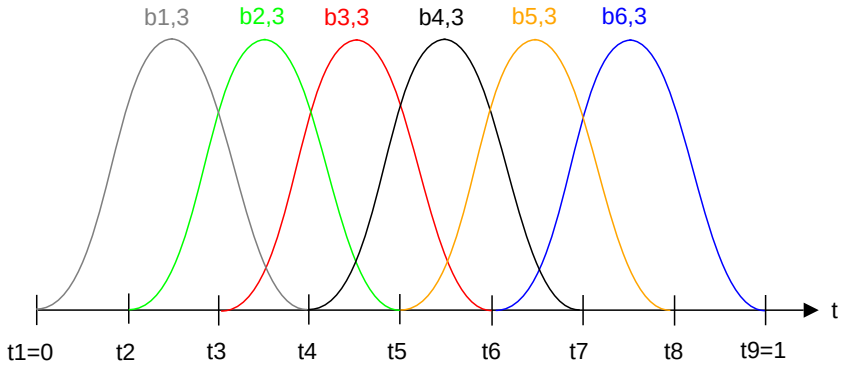

FIG. 2.10 – Splines de base cubiques uniformes pour 6 points de contrôle
L'aspect cubique donne une allure de gaussienne aux splines. L'aspect uniforme entraîne qu'elles soient toutes identiques : le graphe de la spline $b_{i+1,3}$ peut être déduit par une translation vers la droite d'une distance $2/m$ du graphe de la spline $b_{i,3}$.

Pour une valeur t donnée, seules d splines de base sont non nulles. Par conséquent, la position de chaque point de la B-spline n'est influencée que par les d points de contrôle les plus proches. Les points de contrôle P_i ne modifient donc que localement la forme de la B-spline.

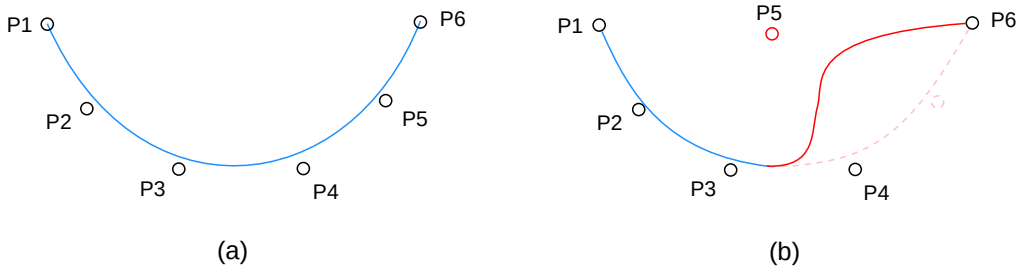

FIG. 2.11 – Déformation d'une B-spline
Le déplacement du point de contrôle P_5 ne déforme que la partie rouge de la B-spline, la partie bleue ne dépendant que des points de contrôle de P_1 à P_4.

Borrow [Bor88] a proposé d'utiliser une B-spline cubique uniforme pour la modulation de vitesse, le graphe de la B-spline représentant un chemin dans l'espace-temps défini dans la section 1.2.1[4]. La position des n points de contrôle P_i est la solution du problème d'optimisation suivant :

$$\min_{P_i} \int_{t=0}^{L} S(t)dt \tag{2.12}$$

[4]En réalité, il s'agit d'un espace-vitesse de dimensions (s, v), où s est l'abscisse curviligne du mobile et v sa vitesse, mais ces deux modélisations sont équivalentes.

Sous les contraintes :

$$v_{min} \leq \frac{dS(t)}{dt} \leq v_{max} \tag{2.13}$$

$$\forall i \in [1,n], \forall j \in [1,q] : \ \delta_{min} \leq d(P_i, Z_j) \tag{2.14}$$

où $d(P_i, Z_j)$ désigne la distance entre le point de contrôle P_i et l'obstacle Z_j.

En d'autres termes, les points de contrôle doivent minimiser la surface sous la B-spline en vérifiant les contraintes suivantes :

- Respecter les bornes v_{min} et v_{max} sur la vitesse du mobile (autrement dit, rester dans le secteur des vitesses valides représenté dans la figure 2.8a)
- Maintenir une distance minimale δ_{min} avec les obstacles. Il faut noter que cette dernière contrainte n'est pas suffisante pour garantir l'évitement des obstacles par la B-spline (figure 2.12a). Toutefois, les risques de collision peuvent être diminués en augmentant le nombre n de points de contrôle (figure 2.12b).

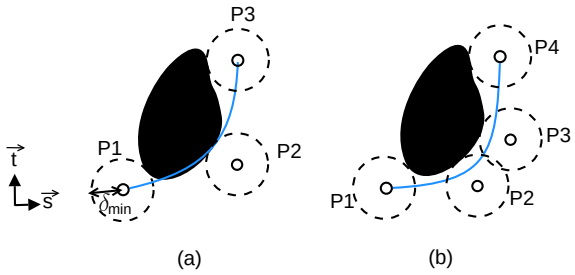

FIG. 2.12 – Évitement d'un obstacle par une B-spline

La partie (a) montre l'insuffisance des contraintes 2.14 pour éviter les obstacles : tous les points de contrôle P_i sont distants de δ_{min} mais pas forcement la B-spline, ce qui aboutit à une collision. La partie (b) montre l'amélioration apportée par l'ajout de points de contrôle.

Le problème d'optimisation est résolu en utilisant des techniques de point intérieur[5]. La figure ci-après montre le résultat obtenu en utilisant 4 points de contrôle.

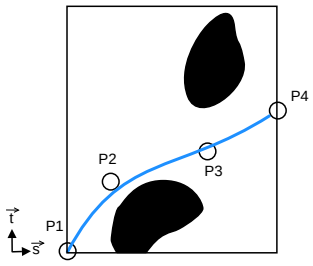

FIG. 2.13 – Modulation de vitesse par optimisation de B-spline

La figure présente le chemin obtenu dans l'espace-temps de la figure 1.2.1, par optimisation d'une B-spline définie par 4 points de contrôle.

[5]Ces techniques consistent à intégrer les contraintes dans la fonction objectif sous la forme de pénalités. Pour plus de détails, le lecteur pourra se référer à [Min83].

Plus récemment, des auteurs ont proposé d'utiliser des fonction polynomiales par morceaux plutôt qu'une unique B-Spline, avec des conditions de raccordement (continuité, dérivabilité, etc.). Utiliser plusieurs morceaux permet notamment un évitement plus aisé des obstacles. Le lecteur intéressé pourra consulter [BR05].

1.2.4 L'algorithme des lignes brisées

L'algorithme des *lignes brisées*, que nous avons nous-mêmes introduit dans [ST06], a pour but d'atteindre au plus tôt une colonne C de l'espace-temps à partir d'un point $P = (s_p, t_p)$ tout en en limitant les variations de vitesse. Son principe est le suivant :

1. Dans la colonne C, le point Q le plus bas (correspondant à une arrivée au plus tôt) est calculé. Si C est placée à l'abscisse curviligne $s_c > s_p$, Q a pour coordonnées $(s_c, (s_c - s_p)/v_{max})$, où v_{max} désigne la vitesse maximale du mobile.
2. Les intersections entre le segment $[PQ]$ et les obstacles sont relevées dans l'ensemble I. Si $I = \emptyset$, l'algorithme s'arrête. Sinon, I est de la forme $\{I_1 = (s_1, t_1), ..., I_n = (s_n, t_n)\}$ avec $s_1 \leq ... \leq s_n$. La première intersection I_1 aboutit à deux appels récursifs :
   (a) Le premier appel a pour but de relier le point A à la colonne C_1 d'abscisse curviligne s_1. Celui-ci le point d'arrivée $Q_1 = (s_1, t_1)$ dans la colonne C_1.
   (b) Le deuxième appel a pour but de relier le point Q_1 à la colonne B. Celui-ci renvoie le point d'arrivée $Q' = (s_c, t'_c)$ dans la colonne C (on a $t'_c > t_c$).

   Géométriquement, ces deux appels récursifs peuvent être perçus comme la brisure du segment $[PQ]$ en deux sous-segments : $[PQ_1]$ et $[Q_1Q']$.

Dans l'espace-temps, une ligne droite correspond à une vitesse constante. L'exécution de l'algorithme peut être interprétée ainsi : au départ, l'algorithme tente d'atteindre la colonne C à partir de P en imposant au mobile de garder la même vitesse (sa vitesse maximale). A chaque conflit avec un obstacle mobile, une variation de vitesse est introduite. La succession de lignes brisées ainsi construite est très similaire au chemin obtenu par l'optimisation de B-splines, comme illustré dans la figure ci-après[6] (à comparer avec avec la figure 2.13 page 59).

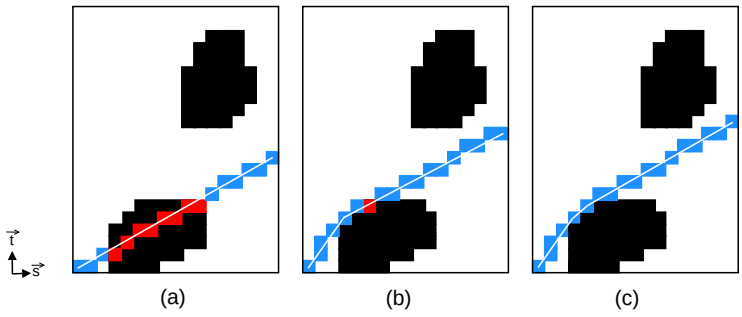

FIG. 2.14 – Modulation de vitesse par application des lignes brisées
Cette figure montre les différentes étapes de l'exécution de l'algorithme des lignes brisées, dans l'espace-temps discrétisé de la figure 2.9b. A chaque étape, le segment intersectant un obstacle est brisé en deux parties.

[6] Dans cette figure, l'environnement a été discrétisé à l'aide de cellules régulières pour permettre de détecter facilement les intersections avec les obstacles, de forme complexe.

2 Les méthodes locales

Dans les méthodes locales, la trajectoire est le résultat d'une succession de déplacements effectués par le mobile à intervalles réguliers. Chaque déplacement pouvant être vu comme une planification de chemin élémentaire, ces méthodes locales sont équivalentes, d'un point de vue combinatoire, à un nombre fini de planifications de chemins. Elles peuvent donc être exécutées en un temps polynomial. Toutefois, ceci se fait au prix de l'optimalité : l'absence d'une vision globale peut entraîner des détours inutiles, notamment l'exploration d'impasses.

A chacun de ses déplacements, le mobile tient compte de l'état des obstacles. Selon Fiorini et Schiller [FS98], l'état $e_O(t)$ d'un obstacle O contient les n dérivées successives de sa position $P(t)$:

$$e_O(t) = (P(t), P'(t), ..., P^{(n)}(t)) \tag{2.15}$$

où $^{(n)}$ symbolise n dérivations successives par rapport au temps t.

La valeur de n définit l'*ordre* de la méthode locale. La plupart des méthodes décrites dans la littérature sont d'ordre 0 (i.e. tenant compte de la position des obstacles). On trouve des méthodes d'ordre 1 (tenant compte en plus de leur vecteur-vitesse), mais il n'existe pas, à notre connaissance, de méthodes d'ordre 2 (tenant compte de l'accélération) ou plus.

2.1 Les méthodes d'ordre 0

Les méthodes d'ordre 0, purement réactives, sont classiquement utilisées pour des tâches de navigation dans un environnement inconnu. Elles sont toutefois utilisables pour la planification de trajectoire, en simulant l'écoulement du temps ainsi que les perceptions du mobile. Ceci présente l'avantage de réduire l'écart entre la trajectoire planifiée et son exécution réelle (problème récurrent en robotique) puisque les méthodes utilisées sont homogènes dans les deux situations.

2.1.1 Les Réseaux de Neurones Artificiels (RNA)

Les RNA, dûs à Mac Culloch et Pitts, tentent de mimer les interactions entre des neurones biologiques. Ils sont constitués d'unités de calcul identiques, appelées *neurones*, reliés entre eux par des liaisons pondérées et unidirectionnelles, appelées *synapses*.

Chaque neurone N_i possède n entrées et une seule sortie. Il effectue un traitement élémentaire qui consiste à :

1. Calculer la somme de ses entrées x_j^i, pondérée par les poids synaptiques w_j^i, et biaisée par θ_i (potentiellement négatifs) ;
2. Appliquer une fonction d'activation φ (la même pour tous les neurones). Cette fonction permet d'introduire une non-linéarité, souvent par l'intermédiaire d'un seuil. La plus utilisée est la *sigmoïde*, d'expression $1/(1 + e^{-\lambda x})$. Le paramètre λ permet de jouer sur la pente du seuil : plus λ augmente, plus le seuil est raide.

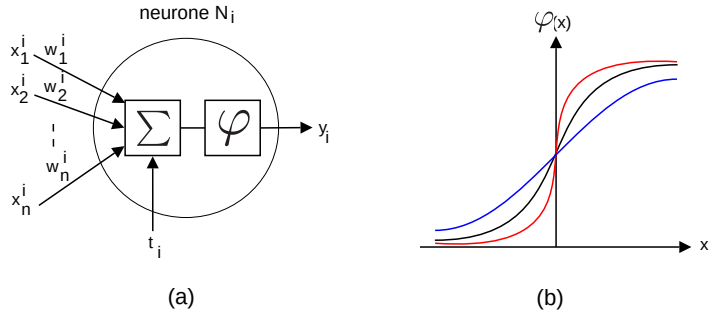

FIG. 2.15 – Neurone artificiel

La partie (a) représente le principe de fonctionnement d'un neurone artificiel. Les synapses sont représentées par des flèches et (b) la fonction sigmoïde φ pour différentes valeurs de λ ($\lambda = 1$ en noir, 0.5 en bleu et 2 en rouge).

La sortie y_i du neurone N_i est donc donnée par :

$$y_i = \varphi\left(\sum_{j=1}^{n} w_j^i \cdot x_j^i + \theta_i\right) \tag{2.16}$$

Un RNA peut être vu comme un graphe orienté dont les noeuds sont les neurones et les arcs les synapses. La topologie de ce graphe définit le type de RNA.

Un des types les plus utilisés est le Perceptron Multi Couches (PMC), introduit par Rumelhart[7] [Rum86]. Les noeuds d'un PMC sont structurés en une série d'hyperplans appelés *couches*.

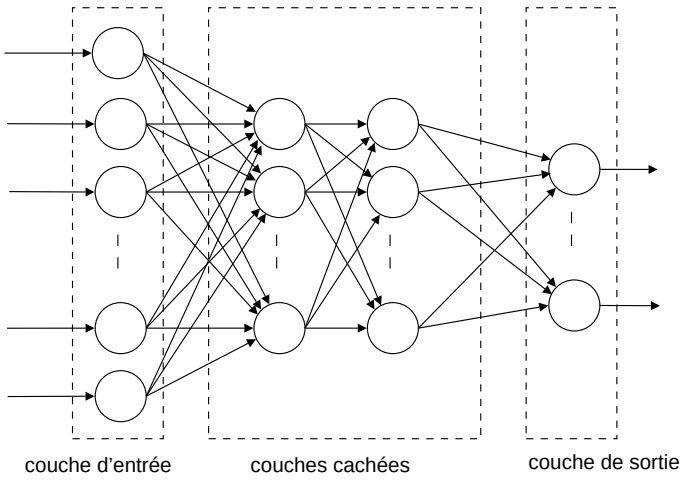

FIG. 2.16 – Réseau de neurones de type PMC

Les neurones sont représentés par des cercles et les synapses par des flèches. Chaque couche (colonne) alimente la suivante.

[7]Le PMC a été proposé en réponse à un livre écrit par Minsky et Papert [MP69] mettant en évidence certaines limitations du perceptron monocouche (notamment l'impossibilité d'apprendre la fonction logique XOR) responsable d'une période noire d'une quinzaine d'années dans le domaine des RNA.

La première couche est appelée *couche d'entrée*, les couches intermédiaires *couches cachées* et la dernière *couche de sortie*. Les sorties d'une couche i sont utilisées comme entrées de la couche $i+1$ (on parle d'architecture *feed-forward*). Souvent, le nombre de neurones décroît au fur et à mesure des couches.

Il a été démontré [Cyb89] que les PMC pouvaient être utilisés comme des approximateurs universels de fonctions. Plus précisément, n'importe quelle fonction peut être approximée avec une précision arbitraire grâce à un PMC à 3 couches (i.e. à 1 couche cachée).

Or, nous avons vu en introduction de ce chapitre (page 49) qu'une trajectoire pouvait être modélisée par une fonction $\vartheta : [0, T] \mapsto (x, y) \in \mathcal{C}_{valide}$.

Cette fonction peut être vue comme la composition de fonctions $e \circ d \circ p$, avec :

- $p : t \mapsto (p_1, ..., p_n)$ les perceptions de l'environnement par le mobile, provenant de ses capteurs,
- $d : (p_1, ..., p_n) \mapsto (c_1, ..., c_m)$ la décision du mobile, correspondant à la commande la plus adaptée aux perceptions,
- $e : (c_1, ..., c_m) \mapsto (x, y)$ l'exécution de la commande par le mobile, fournissant une position dans l'espace valide.

Les fonctions p et e sont connues. Elles correspondent aux entrées/sorties d'informations entre le mobile et l'environnement. On peut voir ces fonctions comme des interfaces, dont l'expression dépend des choix des technologies utilisées dans le mobile.

La fonction d correspond à la partie réellement cognitive du mobile. Elle est inconnue et à priori complexe. Il semble donc astucieux d'utiliser un PMC pour l'approximer.

Une des principales difficultés dans cette tâche est de judicieusement choisir la modélisation des entrées et des sorties du réseau : type de repère (relatif, absolu), système de coordonnées (cylindrique, cartésien), nature des variables (continues, discrètes), etc. Une mauvaise modélisation peut considérablement ralentir (voire faire échouer) l'apprentissage.

Prenons l'exemple du type de repère. Pour faciliter l'apprentissage, il semble préférable d'utiliser un repère lié au mobile (relatif) plutôt qu'un repère lié à l'environnement (absolu). En effet, une même commande (tourner de 20°, par exemple) dans deux situations différentes sera modélisée par la même sortie dans un repère relatif mais pas dans un repère absolu.

Janglova [Jan04] propose de découper l'entourage du mobile en secteurs angulaires réguliers par l'intermédiaire de n segments S_i (un angle de $2\pi/n$ sépare deux segments).

Les n entrées du PMC sont les distances aux obstacles d_i détectés sur les segment S_i, les n sorties sont les distances d'_i que le mobile doit parcourir sur le segment S_i. Cette modélisation est très favorable à l'apprentissage car elle représente les entrées et les sorties de façon homogène (même dimension, même nature).

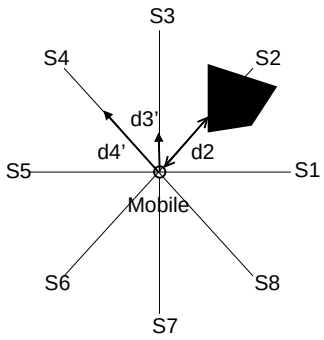

FIG. 2.17 – Entrées/Sorties d'un PMC pour la navigation

Dans [Jan04] n segments S_i placés à intervalles angulaires réguliers sont utilisés pour définir les entrées/sorties d'un PMC à des fins de navigation : les distances aux obstacles d_i et les distances à parcourir d'_i sur les segments.

Vient ensuite le problème de l'apprentissage. Il peut être de deux types : supervisé ou semi-supervisé.

– L'apprentissage *supervisé* consiste à utiliser une base d'exemples, généralement de taille importante. Un exemple est un couple (x, y^*) où x est une entrée et y^* la sortie désirée.

  Les entrées x sont présentées au réseau, qui les propage jusqu'à la couche de sortie. L'erreur entre la sortie ainsi calculée $\widetilde{y}$ et la sortie désirée $y*$ permet de corriger les poids du réseau. La règle la plus simple pour réaliser cette correction est la *rétropropagation* [Wer74]. Dans celle-ci, les poids synaptiques sont mis à jour couche par couche par[8] :

$$\delta w_j^{(i)} = \lambda \cdot \delta y^{(i)} \cdot x^{(i-1)} \tag{2.17}$$

  L'exposant $^{(i)}$ sert à désigner la couche courante. Il varie de la couche de sortie vers la couche d'entrée, autrement dit dans le sens inverse de la propagation d'informations dans le réseau (d'où le nom de la méthode). Le facteur $\lambda \in [0, 1]$ représente le taux d'apprentissage. Son réglage est délicat : s'il est trop faible, l'apprentissage est lent ; s'il est trop élevé, le réseau est instable et l'apprentissage inexistant.

  Cependant, utiliser un apprentissage supervisé suppose que l'on dispose déjà d'une méthode de planification de trajectoire (pour fournir les sorties désirées). On peut alors mettre en doute l'intérêt d'utiliser un RNA. Si la méthode dont on dispose est lente, il y a sans doute un moyen spécifique d'en approximer le résultat, sans avoir recours à un RNA. Par exemple, si la méthode est à base de cases, en augmenter les dimensions est un moyen d'obtenir une solution plus grossière, et donc plus rapide à calculer.

– L'apprentissage *semi-supervisé* évalue non pas les sorties du réseau, mais leur effet au bout d'un temps fixé (longueur parcourue, énergie dépensée, etc.). Cependant, cette évaluation n'est pas directement utilisable pour la mise à jour des poids, car l'évaluation et la sortie du réseau sont de nature hétérogène. L'évaluation doit donc agir implicitement sur les poids du réseau.

[8] Il est possible d'ajouter un terme d'inertie, appelé *momentum*, pour minimiser les risques de convergence vers un minimum local. A chaque itération, on réinjecte une partie du $\delta w_j^{(i)}$ calculé précédemment.

Pour ce faire, Fogel *et al.* [FFP90] ont proposé l'approche Evolving Neural Networks (ENN). Elle consiste à considérer un RNA comme un individu $W = \langle w_1, ..., w_n \rangle$ contenant tous les poids (la topologie du réseau est implicite). Une population de réseaux est générée aléatoirement, puis son évolution est simulée par un algorithme génétique[9].

La figure ci-après montre la trajectoire obtenue par cette méthode, en utilisant la modélisation de Janglova et la fonction d'évaluation décrite dans l'équation 1.5 page 39. Elle présente un détour assez important par rapport à la trajectoire de référence présentée page 54.

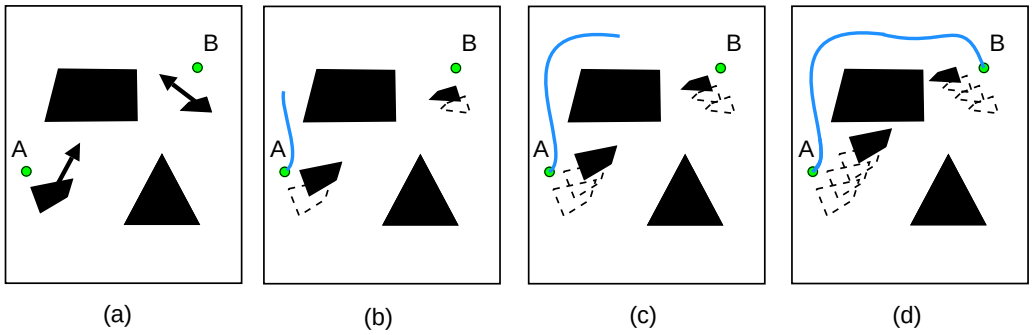

FIG. 2.18 – Construction d'une trajectoire par un PMC entraîné via l'approche ENN
La figure illustre la construction progressive d'une trajectoire (en bleu) par un RNA de type PMC, dont les poids ont été optimisés via l'approche ENN. Le mobile, ayant tendance à fuir les obstacles, atteint le point but par le haut.

L'inconvénient majeur des RNA est qu'ils peuvent paraître "opaques". Pour des problèmes complexes, il n'est pas toujours aisé de donner une signification aux poids du réseau, même avec une certaine expérience dans le domaine. Ceci pose les problèmes suivants :

- Le dimensionnement du réseau (nombre de couches et de neurones par couche) n'est pas une tâche triviale[10]. Un nombre trop faible de neurones peut aboutir à une approximation trop grossière alors qu'un nombre trop élevé peut aboutir à un un surapprentissage : le réseau apprend les exemples "par coeur" (c'est à dire qu'il les stocke implicitement dans les poids synaptiques), sans parvenir à en tirer une généralisation.
- Nous ne sommes jamais totalement sûrs de ce que le réseau a réellement appris. Il est en effet possible que son comportement soit conforme aux attentes sur un grand nombre d'exemples, mais tout à fait inattendu sur quelques exemples isolés.

  Pour illustrer ce problème, prenons un contexte analogue : l'approximation de fonctions par des séries entières. La fonction sinus peut être approximée, avec une précision arbitraire, par la série entière suivante :

$$S_n(x) = \sum_{i=0}^{n} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!} \tag{2.18}$$

[9] Un certain nombre de précautions sont prises dans la définition opérateurs génétiques. Par exemple, l'opérateur de croisement ne doit pas mélanger les poids de couches différentes, car ils n'ont pas la même signification.

[10] De nombreuses techniques tentent d'automatiser cette tâche, par complexifications (NEAT [SM02]) ou par simplifications (OBD [CDS90], OBS [HSW93]) successives d'un réseau initial.

Plus il y a de termes dans la somme (i.e. plus n est élevé), plus l'intervalle sur lequel la fonction et sa série sont proches est étendu. Pour $n = 7$, le graphe de $sin(x)$ et $S_n(x)$ sont quasiment confondus sur l'intervalle $I = [-3, 3]$, mais en dehors de cet intervalle, ces graphes divergent totalement, comme on peut le voir dans la figure ci-après.

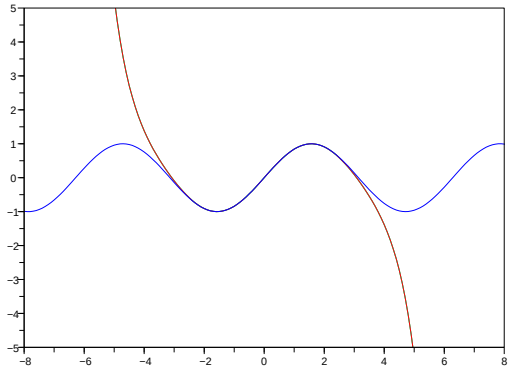

FIG. 2.19 – Approximation de la fonction sinus par une série entière
La figure présente les graphes de la fonction sinus (en bleu) et de sa série entière (en noir) définie par l'équation 2.18, avec $n = 7$.

Ainsi, si on se contentait de prendre des exemples dans l'intervalle I, on aboutirait à la conclusion que la série est une excellente approximation de la fonction, et on serait considérablement surpris par le résultat fourni pour $x = 10$: -1307.46 au lieu de - 0.54 !

2.1.2 La logique floue

La logique floue, introduite par Zadeh [Zad65], permet de réaliser des raisonnements symboliques sur des données numériques. Une des principales applications de la logique floue est la commande floue. Elle consiste à trouver la commande la plus adaptée à un système en fonction des données numériques provenant de capteurs et de connaissances du domaine, modélisées sous la forme de règles. Le schéma de principe de la commande floue est fourni ci-après.

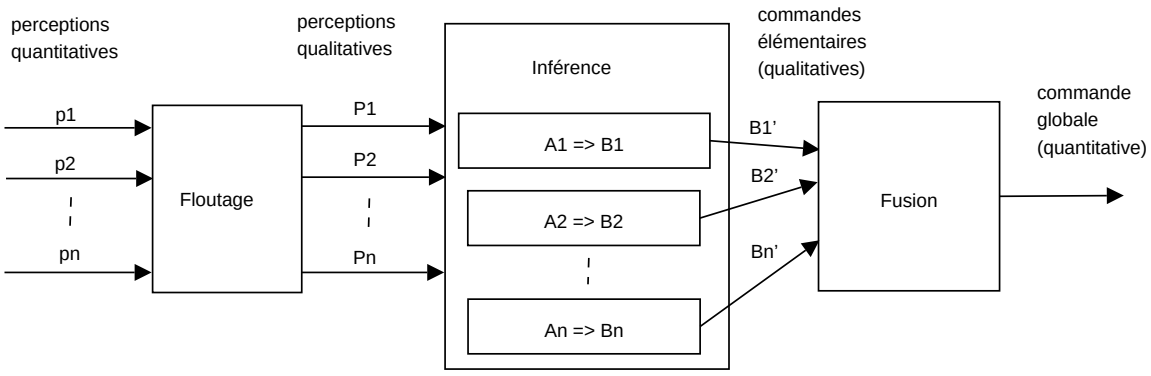

FIG. 2.20 – Commande floue
Les perceptions quantitatives p_i sont transformées en perceptions qualitatives P_i par floutage. Ces dernières produisent, par l'intermédiaire de règles floues de type $A_i \Rightarrow B_i$, des commandes élémentaires B'_i, fusionnées pour déterminer la commande globale.

1. L'étape de floutage permet de traduire des perceptions quantitatives p_i (mesures provenant de capteurs : angles, distances) en perceptions qualitatives P_i (ensemble de symboles : proche, éloigné, gauche, droite). Elle est réalisée par l'intermédiaire d'ensembles flous, généralement définis par des trapèzes.

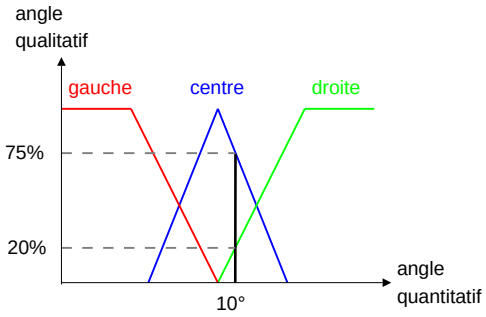

FIG. 2.21 – Floutage d'un l'angle
La figure illustre le floutage d'un angle par 3 ensembles flous : gauche, centre et droite. La mesure 10° vérifie le concept gauche à 0%, centre à 75% et droite à 20%.

2. L'étape d'inférence consiste à appliquer un ensemble de règles floues. Ces règles sont des implications du type $A_i \Rightarrow B_i$.

   Les conditions A_i correspondent à des conjonctions de perceptions qualitatives P_j : $condition = P_1 \wedge P_2 \wedge ... \wedge P_n$.

   Les actions B_i sont partiellement exécutées, selon l'intensité des perceptions P_i. Cette exécution partielle est matérialisée par un tronquage de l'ensemble flou B_i, donnant lieu à un ensemble B'_i.

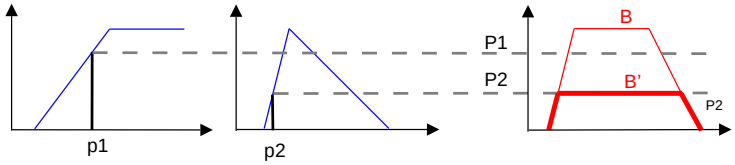

FIG. 2.22 – Inférence floue
La figure illustre le déclenchement de la règle floue $P_1 \wedge P_2 \Rightarrow B$. L'ensemble initial de A est tronqué par le niveau minimal des perceptions P_i, donnant lieu à l'ensemble B'.

3. L'étape de fusion consiste à agréger et à déflouter les actions élémentaires B'_i pour obtenir un déplacement. La méthode la plus utilisée pour le défloutage est la centroïde, qui consiste à calculer, de façon imagée, le centre de gravité de l'ensemble flou.

   Il est important de noter que selon l'ordre des opérations d'agrégation et de défloutage, les résultats peuvent être différents. Autrement dit, ces operations ne sont pas commutatives. La figure 2.23 illustre ce point : pour des ensembles flous initiaux identiques, l'ordre défloutage-aggrégation donne une valeur de 3.25, alors que l'ordre inverse donne plus du double !

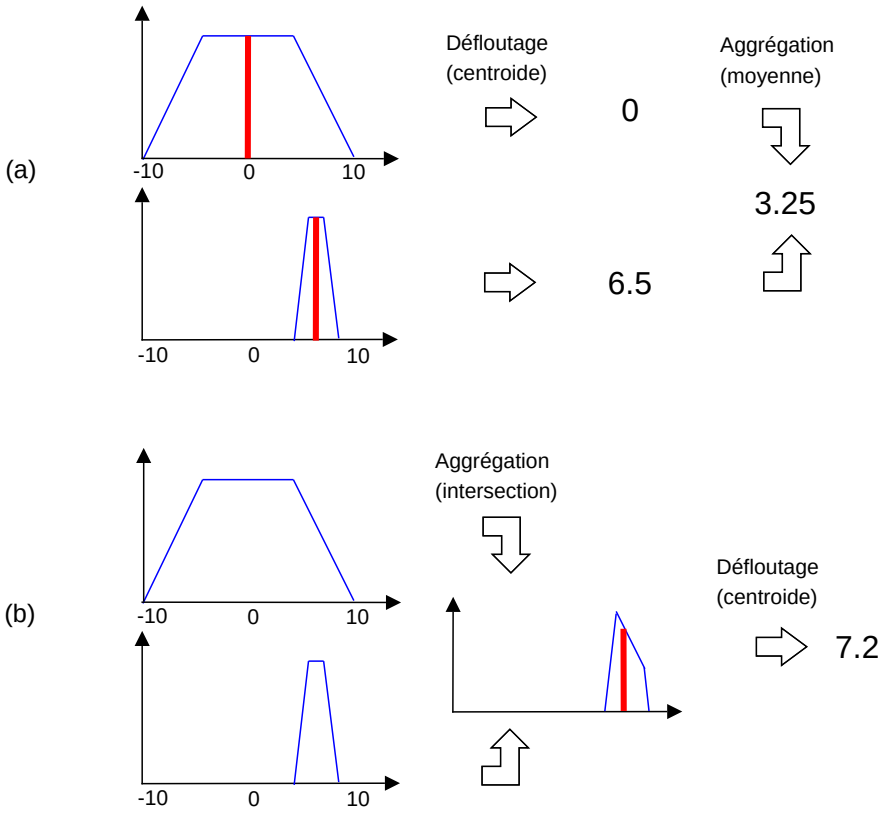

FIG. 2.23 – Fusion d'actions
La figure présente la fusion de deux actions floues. La partie (a) correspond à un défloutage puis agrégation et la partie (b) à l'ordre inverse. Dans cet exemple, les résultats sont très différents.

Sugeno et Nishida [SN85] ont proposé pour la première fois d'utiliser le principe de commande floue pour la navigation d'un mobile dans un environnement dynamique. Par rapport aux réseaux de neurones (présentés dans la section 2.1.1), la commande floue possède, par nature, deux avantages :

- Le raisonnement est transparent (par opposition à l'opacité des réseaux de neurones). En effet, celui-ci est basé sur un ensemble de règles, généralement simples, du type :

  $(\text{angle\_obstacle} = \text{droite}) \wedge (\text{distance\_obstacle} = \text{proche}) \Rightarrow \text{direction} = \text{gauche}$

  Ces règles sont aisément compréhensibles et modifiables, et rendent possible l'intégration des connaissances d'un expert du domaine.

- La commande est robuste. Elle tient compte des incertitudes, tant sur la connaissance de l'environnement que sur l'exécution réelle de la trajectoire.

Cette technique de navigation peut être utilisée pour planifier des trajectoires, en simulant l'évolution de l'environnement au cours du temps, comme cela est fait pour les réseaux de neurones.

Ce contexte de simulation permet d'optimiser les règles floues. Au départ, les règles sont fixées aléatoirement ou avec l'aide d'un expert. Elles sont ensuite ajustées en effectuant un grand nombre d'essais dans des environnements différents.

Pour effectuer cet ajustement, on trouve deux principales approches :

– L'apprentissage par renforcement [YY99] :

Pour un même ensemble de conditions C_i, plusieurs règles probabilistes de la forme $A_i \overset{p_j}{\Rightarrow} B_j$ sont définies, où p_j désigne la probabilité d'exécuter l'action B_j.

La navigation du mobile est simulée pendant une durée T fixée (une collision avec un obstacle interrompt la simulation). Une fois la simulation achevée, une récompense $R(T)$ est attribuée au mobile, selon son état à l'instant T. On peut par exemple opter pour une récompense $R(T)$ de la forme $1/d_B$, où d_B est la distance au but : plus le mobile a réussi à approcher du but, plus il est récompensé (dans le cas d'une collision, le mobile est fortement pénalisé en fixant $R(T) = 0$).

La récompense $R(T)$ est ensuite rétropropagée aux pas de temps précédents, en utilisant une expression de la forme :

$$R(t) = \sum_{i=1}^{T-t} \gamma^t \cdot R(t+j) \tag{2.19}$$

$\gamma \in [0, 1]$ est le facteur de dépréciation. Il permet de régler l'importance que l'on donne aux récompenses futures par rapport aux récompenses immédiates. Un facteur γ proche de 1 correspond à une vision à long terme : toutes les récompenses ont une importance équivalente ; γ est proche de 0 correspond à une vision court terme : seules les récompenses immédiates ont de l'importance, les récompenses futures sont négligées.

Les probabilités $p_j(t)$ sont ensuite ajustées proportionnellement aux $R(t)$.
A la fin de l'étape d'apprentissage, pour chaque ensemble de conditions A_i, la règle ayant la plus forte probabilité d'exécution est conservée.

– Les algorithmes génétiques [JH00] :

Les actions B_i sont considérées comme des gènes. A partir de ces gènes, on forme des individus $I = \langle B_1, ..., B_n \rangle$, contenant toutes les actions présentes dans les règles floues. Ces individus entrent dans le processus d'évolution décrit dans le chapitre 1 (page 38) : évaluation, sélection, mutation et croisement.

L'évaluation d'un individu I consiste, comme dans l'apprentissage par renforcement, à simuler le déplacement du mobile pendant une durée T en appliquant les actions présentes dans I, puis à attribuer une note selon l'état du mobile. Pour les autres phases de l'évolution, les opérateurs standards sont utilisés.

A la fin du processus d'évolution, le meilleur individu est sélectionné pour définir les actions des règles floues.

2.1.3 La déformation de chemin

Dans les réseaux de neurones et la logique floue, la trajectoire est le résultat d'un processus décisionnel : étant donné une situation, le mobile choisit l'action qui lui semble la plus appropriée.

Dans l'approche proposée par Quinlan et Khatib, *Elastic Bands* [QK93], la trajectoire est le résultat d'un processus de déformation de chemin.

Au départ, un premier chemin est planifié, en utilisant l'une des méthodes présentées dans le chapitre 1. Ensuite, ce chemin est assimilé à un élastique soumis à deux types de forces :

- Des forces internes, tendant à le comprimer
- Des forces externes, dues à la répulsion des obstacles, tendant à l'étirer

Pour simuler l'action de ces forces, les auteurs proposent d'utiliser des *bulles* ayant les propriétés suivantes :

- Le centre d'une bulle est situé sur le chemin,
- La surface d'une bulle est entièrement contenue dans l'espace valide,
- Deux bulles voisines doivent se toucher[11].

La force de contraction interne d'une bulle B_i de centre M_i dépend de ses voisines B_{i-1} et B_{i+1} :

$$f_c^i = k_c \cdot \left(\frac{M_{i-1} - M_i}{d(M_{i-1}, M_i)} + \frac{M_i - M_{i+1}}{d(M_i, M_{i-1})} \right) \tag{2.20}$$

où $d(X, Y)$ désigne la distance entre X et Y, et k_c le coefficient de contraction des bulles, analogue au coefficient de raideur utilisé pour les ressorts.

La force de répulsion de chaque obstacle O_j sur la bulle B_i est de la forme :

$$f_r^{i,j} = k_r \cdot d(B_i, O_j) \tag{2.21}$$

où k_r désigne le coefficient de répulsion des obstacles.

La position du centre M_i de la de bulle B_i est mise à jour par :

$$M_i(t+1) = M_i(t) + \alpha \cdot \left(f_c^i(t) + \sum_j f_r^{i,j} \right) \tag{2.22}$$

La figure 2.24 montre une déformation d'un chemin initialement calculé par la méthode PRM (en ignorant les obstacles mobiles).

Des méthodes analogues ont été proposées par la suite, avec des modélisations différentes. Bortoff [Bor00] a modélisé les chemins par des chaînes (dont les maillons jouent un rôle analogue aux bulles), Lamireaux *et al.* [LBL04] par des fonctions continues.

Enfin, notons que cette technique a récemment été étendue à la déformation de trajectoire. Le principe est analogue à la déformation de chemin, sauf qu'une dimension supplémentaire (temporelle) est ajoutée. Le lecteur intéressé pourra consulter [KF07].

[11] Cette condition permet de garantir que le chemin est constamment contenu dans des bulles, et donc dans l'espace valide.

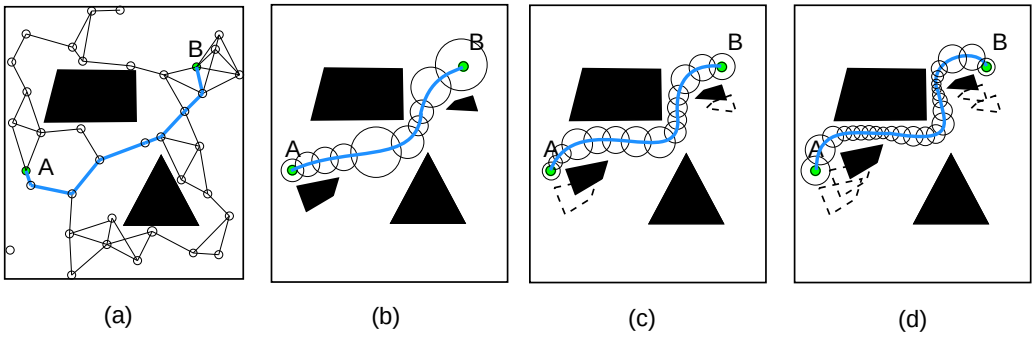

FIG. 2.24 – Déformation de chemin
La partie (a) illustre le chemin entre A et B ne tenant compte que des obstacles fixes (calculé par la méthode PRM). Les parties (b) à (d), les déformations successives du chemin en fonction de l'évolution des obstacles mobiles. Des bulles sont progressivement ajoutées pour maintenir l'ensemble en contact.

2.2 Les méthodes d'ordre 1

Les méthodes d'ordre 0 appliquent une stratégie à court terme : le mobile ne modifie sa trajectoire que dans le cas d'une proximité jugée trop importante avec un obstacle. Les déplacements du mobile sont parfois inappropriés, comme l'illustre la figure ci-après.

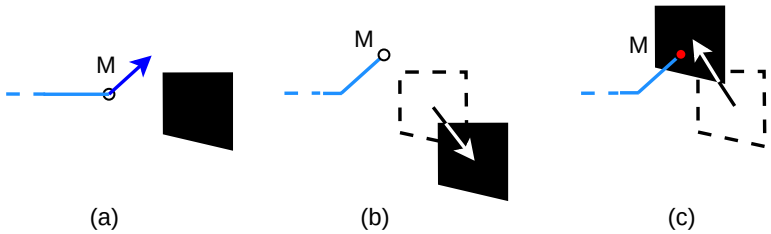

FIG. 2.25 – Limites des méthodes d'ordre 0
Le déplacement effectué en (a) par le mobile M, résultant d'une méthode d'ordre 0, ne tient compte que de la position de l'obstacle. Il est inutile dans la situation (b), et insuffisant dans la situation (c).

Ce problème peut être atténué en optant pour une stratégie à plus long terme : tenir compte du vecteur-vitesse des obstacles pour anticiper leurs déplacements. C'est l'esprit des méthodes qualifiées de *premier ordre*. Dans cette catégorie sont principalement mentionnées la notion d'obstacle-vitesse [FS93] et l'extension des potentiels artificiels [GC02].

2.3 Les obstacles-vitesses

Considérons un obstacle fixe O, modélisé par un cercle, et un mobile M, ponctuel, se déplaçant par rapport à lui avec un vecteur-vitesse $\vec{v}_{M,O}$. Le maintien de certains vecteurs-vitesses provoque à coup sûr une collision entre M et O.

Ces vecteurs pointent tous à l'intérieur d'un cône, appelé *cône de collision* et noté $c_{M,O}$. Il est délimité par les deux tangentes à O issues de M, comme l'illustre la figure ci-dessous.

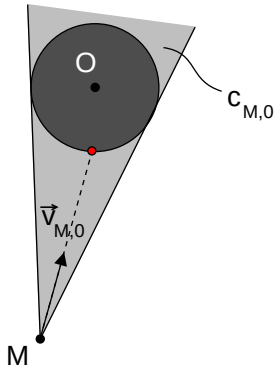

FIG. 2.26 – Cône de collision
Le maintien de tout vecteur-vitesse $\vec{v}_{M,O}$ (du mobile M par rapport à l'obstacle O) pointant dans le cône $c_{M,O}$ (en gris) provoquera une collision entre M et O (en rouge).

Fiorini et Schiller [FS93] ont étendu ce raisonnement à des obstacles mobiles ayant un vecteur-vitesse connu, grâce à l'approche des *obstacles-vitesses*. Le principe de cette approche est le suivant :

Soient $\vec{v}_O$ et $\vec{v}_M$ les vecteurs-vitesses absolus de l'obstacle O et du mobile M à un instant t. Par principe de composition des vitesses, ces vecteurs sont liés par la relation :

$$\vec{v}_M = \vec{v}_{M,O} + \vec{v}_O \tag{2.23}$$

Ainsi, l'ensemble des vecteurs-vitesses $\vec{v}_M$ dangereux pour le mobile (i.e. aboutissant à une collision avec l'obstacle) peuvent être déduits en translatant le cône de collision $c_{M,O}$ du vecteur $\vec{v}_O$. Cet ensemble est appelé obstacle-vitesse associé à O et noté $vo(O)$.

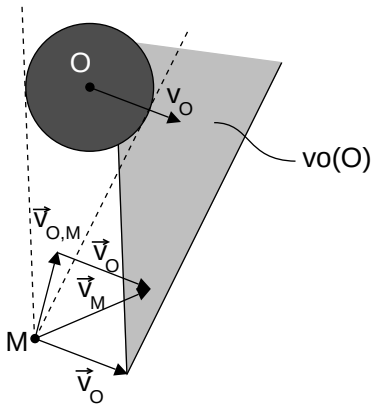

FIG. 2.27 – Obstacle-vitesse
La figure illustre l'obstacle-vitesse $vo(O)$ associé à l'obstacle O se déplaçant à vitesse constante $\vec{v}_O$. L'ensemble $vo(O)$ est obtenu par une translation de vecteur $\vec{v}_O$ du cône de collision $c_{M,O}$ (en pointillés). En d'autres termes, si $\vec{v}_{M,O}$ pointe à l'intérieur de $c_{M,O}$, alors $\vec{v}_M$ pointe à l'intérieur de $vo(O)$.

Par opposition, tous les vecteurs-vitesses $\vec{v}_M$ pointant en dehors de $vo(O)$ sont sûrs pour le mobile, i.e. garantis sans collision. Ils constituent l' *Ensemble des Vitesses de Contournement (EVC)*.

Si la vitesse du mobile n'était pas contrainte, l'EVC serait de taille infinie. Cependant, la plupart du temps, le mobile modélise un système physique réel, avec des limitations énergétiques et technologiques. Sa vitesse possède donc une borne supérieure v_{max}, et éventuellement une borne inférieure v_{min} (pour les engins volants, par exemple). L'EVC est donc réduit à un anneau délimité par les cercles de rayon v_{min} et v_{max}, appelé *Ensemble Vitesses de Contournement Atteignables (EVCA)*.

La construction de cet ensemble est aisément généralisable en présence de n obstacles O_i ($i \in [1, n]$), en excluant cette fois l'union des obstacles-vitesses $vo(0_i)$.

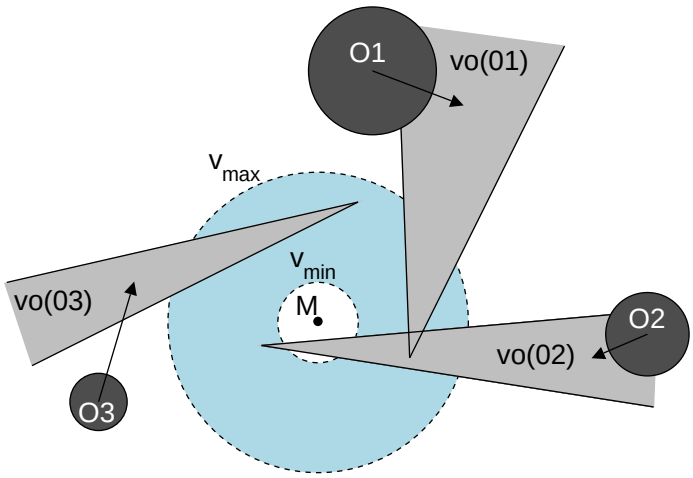

FIG. 2.28 – Ensemble des Vitesses de Contournement Atteignables (EVCA)
La figure illustre l'EVCA en présence de 3 obstacles. Cet ensemble est représenté par l'anneau bleu, délimité par les cercles de rayon v_{min} et v_{max}, auquel est retranché la surface couverte par les obstacles.

Le calcul de l'EVCA à intervalles réguliers δt permet de planifier la trajectoire du mobile dans un environnement contenant des obstacles mobiles quelconques. Les obstacles sont modélisés par des cercles les englobant, et l'hypothèse des vecteurs-vitesses constants fournit une approximation linéaire de leur mouvement pendant la durée δt.

A chaque pas de temps, le vecteur-vitesse du mobile est déterminé de la façon suivante :

1. L'EVCA est discrétisé en utilisant un maillage régulier. Ceci permet de gérer la forme complexe de cet ensemble (pouvant être constitué de plusieurs morceaux, mêlant lignes droites et arrondis) et de réduire le choix à un nombre fini de vitesses.
2. La vitesse du mobile est choisie de manière heuristique parmi les noeuds de la maille. On peut par exemple choisir la vitesse qui permet de se rapprocher le plus du point but.

Cette démarche suppose que le vecteur-vitesse des obstacles évolue lentement dans le temps. Si l'évolution est au contraire rapide et brutale, l'anticipation effectuée par le mobile sera totalement inappropriée, ce qui aboutit aux mêmes défauts que les méthodes d'ordre 0 (présentés page 71).

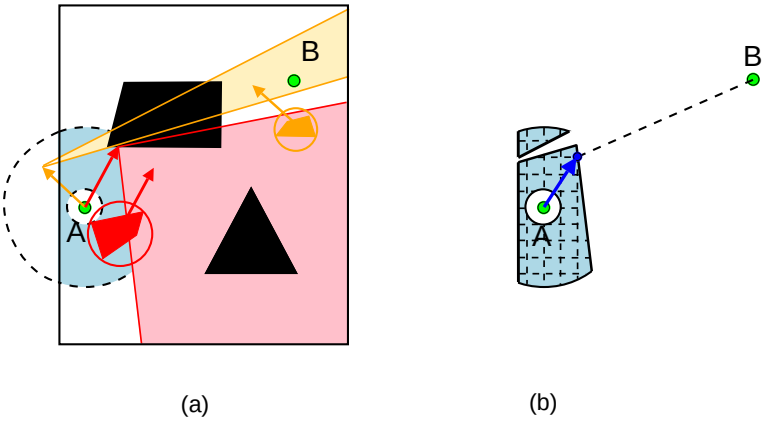

FIG. 2.29 – Choix de la meilleure vitesse de contournement
L'EVCA est d'abord construit dans la partie (a), de la même manière que dans la figure 2.28. La meilleure vitesse de contournement est ensuite déterminée dans la partie (b) : l'EVCA est discrétisé au moyen d'une maille, et le vecteur-vitesse pointant sur le noeud le plus proche du point but est sélectionné.

2.4 Les potentiels artificiels

A priori, les potentiels artificiels utilisés pour la planification de chemin (présentés page 33) peuvent être utilisés pour la planification de trajectoire en tant que méthodes locales d'ordre 0 (le potentiel répulsif associé aux obstacles ne tenant compte que de leur position).

Cependant, le mobile devant prendre des décisions à intervalles réguliers et courts, les potentiels doivent être calculés très rapidement. Pour cela, l'algorithme permettant de propager la valeur des potentiels est directement implémenté sur des systèmes matériels. Notamment, l'implémentation sous la forme de cartes de Kohonen physiques est régulièrement utilisée depuis son introduction par Glasius [GKG95]. On peut entre autres citer [RWHK97][YM01][LSR05].

Toutefois, les problèmes d'optimalité et de complétude inhérents aux méthodes d'ordre 0 (évoqués page 71) se posent toujours. Ces problèmes peuvent être atténués en exploitant la généricité des potentiels artificiels. L'expression des potentiels peut en effet être remodelée à loisir afin d'intégrer de nouveaux éléments. Notamment, les potentiels répulsifs liés aux obstacles peuvent être étendus pour inclure l'information liée à leurs vecteurs-vitesses (connus ou estimés), devenant ainsi la base d'une méthode d'ordre 1.

Les premiers potentiels de ce type furent proposés par Hussien [Hus89]. Cependant, ceux-ci ne prenaient en compte que le vecteur-vitesse des obstacles, et pas celui du mobile. Or, les deux sont nécessaires pour anticiper les collisions.

Ge et Cui [GC02] ont donc proposé des potentiels basés sur la vitesse relative du mobile par rapport aux obstacles (la même base que les obstacles-vitesses). Leur raisonnement est le suivant :

Soit $\vec{a}$ le vecteur unitaire dirigé du mobile M vers un obstacle O et $\vec{b}$ le vecteur unitaire orthogonal à $\vec{a}$. Le vecteur-vitesse de M par rapport à O, noté $\vec{v}_{M,O}$, peut être exprimé dans le repère local $\mathcal{R}_M = (M, \vec{a}, \vec{b})$.

On a ainsi :

$$\vec{v}_{M,O} = \vec{v}^{a}_{M,O} \cdot \vec{a} + \vec{v}^{b}_{M,O} \cdot \vec{b} \tag{2.24}$$

Les composantes $\vec{v}_{M,O}$ dans $\mathcal{R}_M$ sont utilisées pour calculer les composantes de la force répulsive $\vec{F}_O$ associée à O, de la manière suivante :

$$\begin{aligned} \vec{F}^{a}_{O} &= -\alpha \cdot \vec{v}^{a}_{M,O} \\ \vec{F}^{b}_{O} &= \beta \cdot \vec{v}^{b}_{M,O} \end{aligned} \tag{2.25}$$

où α et β sont des constantes positives.

$\vec{F}^{a}_{O}$ est la composante d'*éloignement* : elle incite le mobile à rester à distance de l'obstacle. La composante $\vec{F}^{b}_{O}$ est la composante de *contournement* : elle incite le mobile à se diriger vers une région plus sûre.

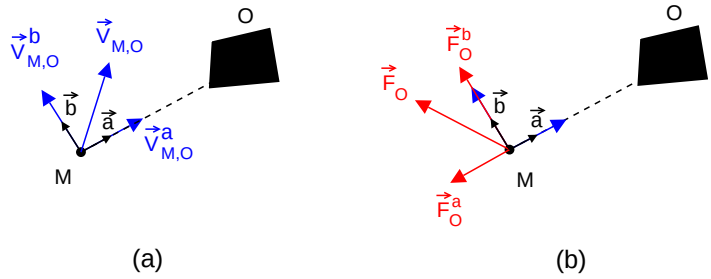

FIG. 2.30 – Force répulsive d'ordre 1

La figure illustre les étapes de la construction de la force répulsive $\vec{F}_O$ d'ordre 1 associée à un obstacle O : (a) projection de $\vec{v}_{M,O}$, vecteur-vitesse relatif de M par rapport à un O (en bleu) ; (b) calcul des composantes de F_O (en rouge) dans le repère $(M, \vec{a}, \vec{b})$ à partir de celles de $\vec{v}_{M,O}$.

Le potentiel répulsif U_O associé à $\vec{F}_O$ est ensuite déduit en utilisant la relation :

$$\vec{F}_O = -\vec{\nabla} U_O \tag{2.26}$$

où $\vec{\nabla} x$ désigne le gradient de x.

Le potentiel attractif U_B lié au but B est toujours d'ordre 0, i.e. de la forme $\gamma \cdot d(M,B)^2$ (où $d(M,B)$ désigne la distance Euclidienne entre M et B, et γ une constante positive).

En absence de variations brusques dans les vitesses des obstacles mobiles, les trajectoires calculées en utilisant les potentiels artificiels attractifs se rapprochent de celles obtenues par les méthodes globales[12], comme on peut le voir dans la figure ci-après.

[12] Ceci est le cas si les coefficients présents dans l'expression des potentiels sont convenablement choisis. Une étude analytique étant très difficile, ces coefficients sont déterminés de façon empirique par les auteurs.

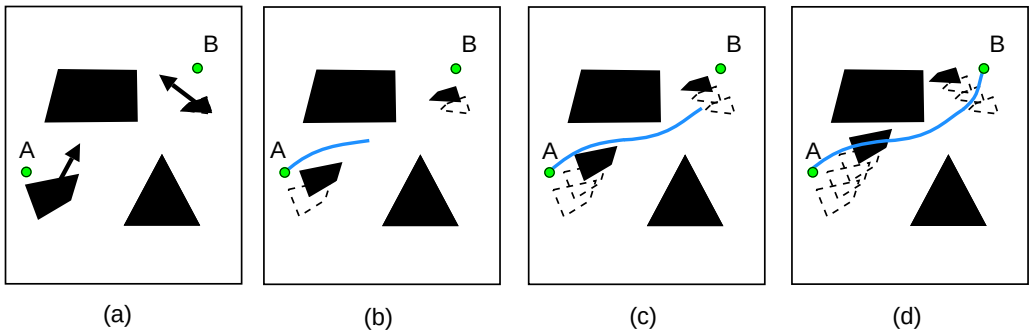

FIG. 2.31 – Construction d'une trajectoire en utilisant les potentiels artificiels d'ordre 1
Les obstacles ayant un mouvement rectiligne uniforme (donc très prévisible), la trajectoire générée (en bleu) est très similaire à celle obtenue par la méthode globale basée sur un PLM (figure 2.5 page 54).

3 Résumé

Le tableau ci-après résume les caractéristiques de l'ensemble des méthodes de planification de trajectoire évoquées dans ce chapitre, dans un espace $2D$ contenant des obstacles de forme polygonale, effectuant une série de mouvements rectilignes uniformes.

Méthode		Déterministe	Optimale	Paramètres
Méthodes globales directes	PRM	non	non	nb échantillons, taille voisinage
	Algorithmes génétiques	non	non	nb générations types opérateurs, taille population
	PLM	oui	oui	nb pas de temps nb facettes
Méthodes globales indirectes	Méthodes de décomposition	oui	oui	aucun
	Optimisation de B-Splines	oui	non	nb points contrôle dist. aux obstacles
	Algorithme des Lignes Brisées	oui	non	aucun
Méthodes locales d'ordre 0	Réseaux de neurones	oui	non	nb neurones nb couches
	Commande floue	oui	non	nb règles
	Déformation de chemins	oui	non	coeffs raideur
Méthodes locales d'ordre 1	Obstacles-vitesses	oui	non	taille maille
	Potentiels artificiels	oui	non	coeffs potentiels

Le problème de planification de trajectoire étant NP-difficile, il n'est pas surprenant de voir que la quasi-totalité des méthodes ne sont pas optimales.

La seule méthode optimale[13] est basée sur une modélisation sous la forme d'un Programme Linéaire Mixte (PLM). Elle nous a permis d'obtenir une trajectoire de référence pour ce chapitre.

Les principaux avantages de cette méthode sont : sa flexibilité (car elle permet de facilement changer les caractéristiques du problème), ainsi que sa facilité de mise en oeuvre (car il suffit d'utiliser des resolveurs du marché).

Le principal défaut de l'approche concerne la maîtrise du temps de calcul. Même si ceux-ci demeurent raisonnables sur des problèmes de taille modérée, ceux-ci peuvent augmenter exponentiellement quand la taille de ces problèmes augmente (nombre d'obstacles, nombre de points de passage définissant la trajectoire, etc.). Ce phénomène, prévisible par le caractère NP-difficile de la résolution d'un PLM, a récemment été mis en évidence de façon expérimentale dans [YELP08].

A priori, les méthodes globales aboutissent à des trajectoires de meilleure qualité (i.e. atteignant le but plus tôt) que les méthodes locales, car elles possèdent une vue d'ensemble du mouvement des obstacles. Cependant, les seules méthodes globales utilisables en pratique reposent sur un échantillonnage aléatoire de l'espace, ce qui ne fournit aucune garantie sur la qualité (ni même l'existence) d'une solution. D'un point de vue pratique, les méthodes globales et locales (d'ordre 1) fournissent des trajectoires globalement équivalentes.

L'aspect purement réactif des méthodes locales d'ordre 0 les rendent moins intéressantes que les autres d'un point de vue qualité, mais elles ont l'avantage d'être les plus rapides à l'exécution.

Sur le nombre important des algorithmes de planification de trajectoire cités ci-dessus, une partie très limitée a été étendue afin de gérer la présence de courants dans l'environnement. C'est ce que nous allons voir dans le chapitre suivant.

[13] Les méthodes de décomposition permettent une modulation de vitesse optimale, mais la décomposition planification de chemin/modulation aboutit à une trajectoire non-optimale.

Chapitre 3

Extensions à la présence de courants

Sommaire

Dans les chapitres 1 et 2 de cet état de l'art, nous avons progressivement introduit, dans l'espace des configurations $\mathcal{C}$ du robot, des obstacles fixes puis mobiles.

Dans ce chapitre, nous allons encore complexifier l'espace des configurations en ajoutant un courant W. Ce courant peut être qualifié de réel ou d'artificiel :

- Un courant *réel* correspond à la représentation directe d'un flux physiquement présent dans l'environnement du mobile (vent, courant marin, etc.).

- Un courant *artificiel* permet de prendre en compte des éléments de nature plus quelconque, mais ayant des effets similaires à un courant réel. C'est notamment le cas du relief sur un mobile terrestre : les pentes descendantes peuvent être assimilées à des courants favorables, et ascendantes à des courants défavorables. Une modélisation de ce type pourra être trouvée dans [TCJ03].

L'effet de W peut être perçu comme l'ajout d'un biais $\vec{w}$ à la vitesse $\vec{v^w}$ du mobile en l'absence de courants. Formellement parlant, W peut être défini par une fonction du type :

$$W : \vec{v^r} \mapsto \vec{v_w} = \vec{v^r} + \vec{w} \tag{3.1}$$

Une autre interprétation équation ci-dessous est la composition des vitesses : $\vec{v^r}$ est la vitesse relative du mobile par rapport sol et $\vec{v^w}$ est la vitesse relative du mobile par rapport aux courant W. En tout point de l'espace des configurations, la différence vectorielle de ces deux grandeurs est $\vec{w}$.

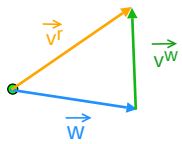

FIG. 3.1 – Principe de composition des vitesses

Le biais $\vec{w}$ peut être vu comme un champ vectoriel de dimension 2. Il est généralement défini de façon discrète (i.e. sur les noeuds d'un maillage), par l'intermédiaire de mesures ou de prévisions, comme illustré dans la figure 3.2.

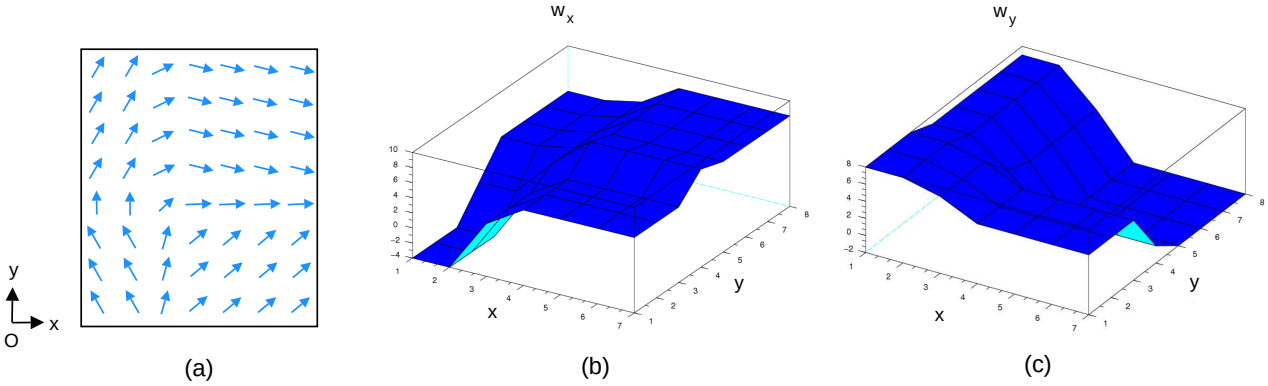

FIG. 3.2 – Représentation des courants dans l'espace des configurations
(a) Les courants sont modélisés par un champ vectoriel $\vec{w}$, connu sur les noeuds d'une maille (ici régulière). Les parties (b) et (c) représentent les projections de $\vec{w}$ sur les axes x et y, respectivement notées w_x et w_y.

La présence du champ $\vec{w}$ modifie les coûts associés aux déplacements effectués entre deux configurations, selon la région de l'espace. Par exemple, dans une zone où $\vec{w}$ pointe vers la droite, tous les déplacements effectués de la gauche vers droite verront leurs coûts diminués, cette diminution traduisant le fait que le courant porte le mobile. Les coûts de déplacement peuvent également dépendre de la date de départ, si le champ est variable dans le temps.

La variation des coûts, dans le temps comme dans l'espace, rend inadaptées les méthodes de planification de trajectoire présentées dans le chapitre 2. A notre connaissance, ce n'est qu'assez récemment (depuis 2003 [RK03]) que les premières réponses à ce problème ont été apportées, donnant naissance à une nouvelle branche dans le domaine de la planification de mouvements, la plus complexe : la planification de trajectoires en présence de courants.

Comme on peut le voir dans la figure ci-après, cette nouvelle branche a uniquement donné lieu à des méthodes globales. Ce constat n'est pas surprenant, car il semble peu pertinent de ne tenir compte que localement du courant. Une stratégie locale est acceptable pour tenir compte des obstacles mobiles, puisque ceux-ci sont généralement peu nombreux et de petite taille, ce qui limite leur influence sur la trajectoire. Toutefois, ce n'est pas le cas pour les courants : les régions qu'ils couvrent peuvent être relativement étendues, et leur intensité d'ordre comparable à la vitesse du mobile.

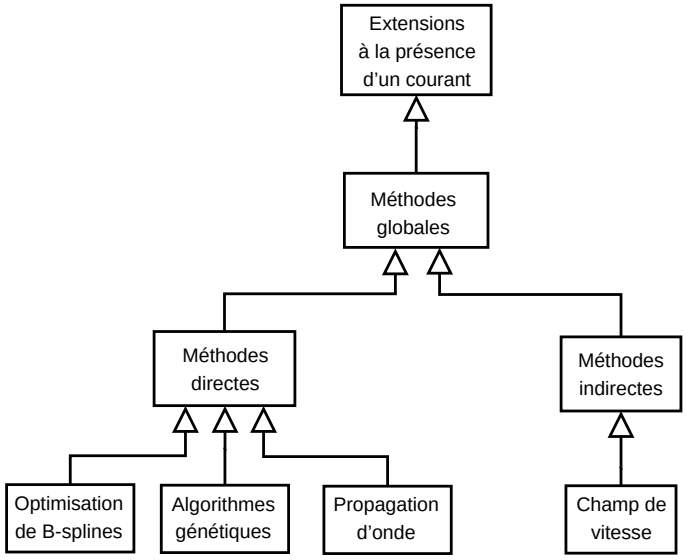

FIG. 3.3 – Extensions à la présence d'un courant

Les triangles blancs symbolisent des liens d'héritage.

1 L'optimisation de B-splines

Nous avons vu dans le chapitre 2 1.2.3 (page 57) que l'utilisation d'une B-spline permet de reformuler une planification de trajectoire en problème d'optimisation continue :

- La trajectoire est représentée par une B-spline, entièrement déterminée par le degré d des splines de base $b_{i,d}$ (généralement fixé à 3) et par un ensemble de points de contrôle P_i (jouant le rôle d'attracteurs). A tout instant t, la position $p(t)$ du mobile est donnée par :

$$p(t) = \sum_{i=1}^{n} P_i \cdot b_{i,d}(t) \tag{3.2}$$

  avec $p(0) = A$ et $p(T) = B$, où A et B désignent les points de départ et d'arrivée du mobile, et T le temps de parcours total.

- La position des points de contrôle P_i est déterminée en minimisant un critère, par exemple T.

En l'absence de courant, la vitesse $\vec{v^r}$ du mobile par rapport au sol est liée à p par la contrainte :

$$\frac{dp(t)}{dt} = \vec{v^r}(t) \tag{3.3}$$

En appliquant le principe de composition de vitesses vu en introduction (équation 3.1), la contrainte précédente devient :

$$\frac{dp(t)}{dt} = \vec{v^w}(t) + \vec{w}(t) \tag{3.4}$$

où $\vec{v^w}$ est la vitesse du mobile par rapport au courant, c'est à dire par rapport au champ $\vec{w}$. Notons que la norme de $\vec{v^w}$ est considérée comme constante, et appelée *vitesse de croisière*.

Par nature, $\vec{w}$ est connu sous la forme de données discrètes, à intervalles de temps et d'espace réguliers (généralement issues de mesures ou de prévisions). En d'autres termes, l'expression analytique du champ vectoriel définissant $\vec{w}$ est inconnue, ce qui rend les méthodes d'optimisation continue inutilisables.

Afin de régler ce problème, Inanc *et al.* [ISM05] ont procédé comme suit :

1. Dans chaque intervalle de temps (généralement fixé à une heure), le champ $\vec{w}$ est considéré comme invariant,
2. Des B-splines sont utilisées pour approximer l'expression de $\vec{w}$ par interpolation.

Le premier point permet de décomposer le problème initial en k sous-problèmes : un premier entre t_0 et t_1, un deuxième entre t_1 et t_2, et ainsi de suite, jusqu'à t_{k-1}. Les conditions finales du problème i sont utilisées comme conditions initiales du problème $i+1$.

Concernant le deuxième point, nous avons vu que le champ $\vec{c}$ possédait deux composantes, w_x et w_y, correspondant à ses projections respectives sur les axes x et y.

La surface de chaque composante w_φ ($\varphi = x$ ou y) peut être approximée par des B-splines, en utilisant les valeurs connues de w_φ, notées $\widetilde{w_\varphi}$, comme points de contrôle.

Ainsi, l'expression de w_φ est façon analogue à l'équation 3.4, la somme simple étant remplacée par une somme double, et les points de contrôle P_i par les valeurs $\widetilde{w_\varphi}(i,j)$:

$$w_\varphi(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} \widetilde{w_\varphi}(i,j) \cdot b_{i,d}(x) \cdot b_{j,d}(y) \tag{3.5}$$

Les interpolations des données représentées dans les figures 3.2b et 3.2c sont fournies ci-après.

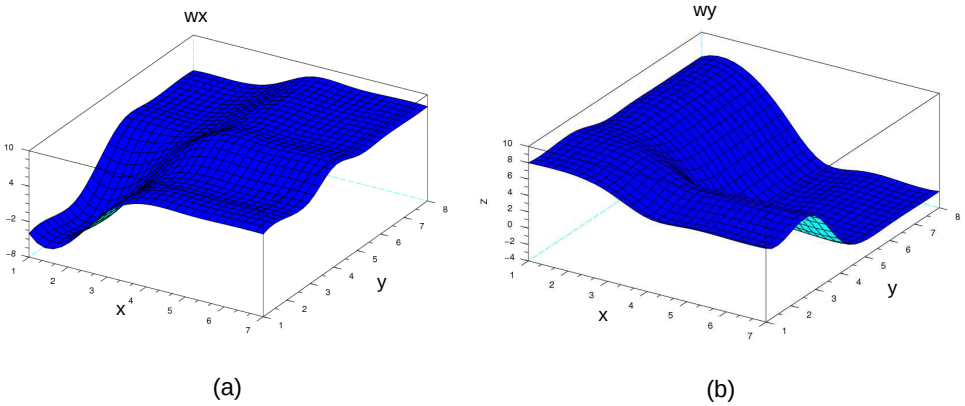

FIG. 3.4 – Interpolation d'un champ vectoriel par des B-splines
Les parties (a) et (b) présentent respectivement les interpolations des composantes w_x et w_y illustrées dans les figures 3.2b et 3.2c, en utilisant des splines cubiques uniformes.

L'approche possède l'avantage de modéliser de façon unifiée la trajectoire et les courants (par des B-splines), ce qui permet d'appliquer les techniques classiques d'optimisation continue sous contraintes. Bien que les contraintes soient non linéaires, le processus d'optimisation est relativement rapide pour chaque intervalle de temps (l'auteur obtient des temps de calcul inférieurs à 10 secondes sur des données réelles de courant).

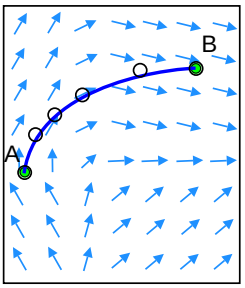

FIG. 3.5 – Trajectoire obtenue par optimisation à base de B-splines
Dans cet exemple, le courant a été considéré comme constant pendant tout le déplacement du mobile. La trajectoire obtenue (en bleu) a été calculée en utilisant 6 points de contrôle (en noir), dont 2 fixés (aux points de départ et d'arrivée).

Récemment l'approche a été améliorée dans [ZIOM08], en modélisant les courants par des B-splines 3D (deux dimensions spatiales, comme précédemment, plus une dimension temporelle). Ainsi, le problème d'optimisation n'est plus résolu sur plusieurs intervalles de temps successifs, mais en une seule passe. L'auteur mentionne de substantiels gains sur la qualité de la solution calculée (le coût de la trajectoire est en effet de l'ordre de 15 fois plus faible sur les exemples fournis !).

Toutefois, même si la solution nous paraît de "bonne" qualité, nous n'avons aucune idée de la distance de cette solution par rapport à la solution optimale. En effet, dans la plupart des cas, les composantes du champ $\vec{w}$ ne sont pas convexes. Dans la figure 3.4, par exemple, on peut observer de nombreux minima locaux. Or, le principe des techniques d'optimisation sous contraintes est de les ajouter sous la forme de pénalités dans la fonction objectif. Cet objectif a donc de fortes chances de contenir à son tour des minima locaux.

Ainsi, il n'y a aucune garantie que le minimum retourné par l'algorithme soit global (en particulier, il n'est pas a priori évident que la trajectoire illustrée dans la figure 3.5 soit effectivement de durée minimale).

Un autre défaut de l'approche est l'absence d'obstacles dans le modèle, qui limite considérablement son champ d'application. Il pourrait être envisageable de les ajouter, de deux manières différentes :

– Maintenir une distance minimale δ_{min} entre les points de contrôle P_i et les obstacles, par l'intermédiaire de contraintes proposées par Borrow [Bor88]. Comme nous l'avons vu page 59, ce type de contrainte ne fournit aucune garantie quand à l'évitement effectif de ces obstacles. En effet, même si les points de contrôle sont évitent les obstacles, ce n'est pas forcement le cas de la B-spline.

– Approximer grossièrement les obstacles par des rectangles, et énumérer les différentes possibilités de contournement (gauche, droite, haut, bas), comme l'a proposé How [RH02]. Comme mentionné page 51, cette énumération introduit des variables binaires dans le modèle. Or, imposer des valeurs binaires à des variables dans des méthodes d'optimisation continue nécessite d'utiliser des procédés de type branch and bound ou plans de coupe qui, comme nous l'avons vu dans le chapitre précédent, peuvent aboutir à des temps de calcul relativement importants.

2 Les algorithmes génétiques

De par leur généricité, les algorithmes génétiques permettent de simplement intégrer la présence de courant.

Considérons par exemple des individus de la forme $I = \langle X_1, ..., X_n \rangle$, avec $X_i = (x_i, y_i, t_i)$ pour représenter des trajectoires. Pour les évaluer, nous pouvons utiliser l'équation présentée page 51, rappelée ci-dessus :

$$C(I) = \alpha \cdot d((x_n, y_n), B) + \beta \cdot t_n \tag{3.6}$$

La seule différence réside dans le calcul des dates t_i. Les dates concernant deux points consécutifs X_i et X_{i+1} sont liées par :

$$t_{i+1} = t_i + \Delta t_{i,i+1} \tag{3.7}$$

Le temps de parcours $\Delta t_{i,i+1}$ dépend des k zones de courant traversées. On a en effet :

$$\Delta t_{i,i+1} = \sum_{j=1}^{k} \delta t_j \tag{3.8}$$

avec δt_j le temps passé dans la zone de courant j. Pour le calculer, on peut par exemple utiliser la fonction de coût proposée par Garau [GAO05] :

$$\delta t_j = \frac{d_j}{||v^{\vec{w}_j} + \vec{w_j}||} \tag{3.9}$$

avec δd_j la distance parcourue dans la zone de courant j, $\vec{w_j}$ la vitesse du courant et dans cette zone, et $v^{\vec{w}_j}$ la vitesse relative du mobile par rapport à $\vec{w_j}$. Quel que soit j, la norme $v^{\vec{w}_j}$ est considérée comme constante (comme pour l'optimisation de B-splines).

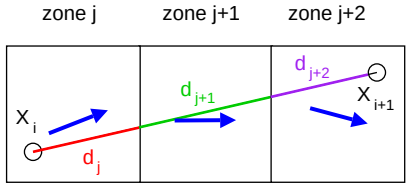

FIG. 3.6 – Calcul du temps de parcours entre 2 points en présence de courants
Le temps de parcours $\Delta t_{i,i+1}$ entre les points X_i et X_{i+1} se décompose en temps élémentaires δt_j (un par zone homogène de courant). δt_j dépend de la distance d_j parcourue dans chaque zone, et du vecteur-vitesse $\vec{w_j}$ du vent (flèches oranges).

Sur le même principe, d'autres fonctions de coût ont été proposées, par exemple la quantité d'énergie dépensée [ACO04].

Comme pour l'optimisation de B-splines, l'intégration des obstacles, fixes comme mobiles, pose quelque peu problème. En effet, en pénalisant les individus à chaque collision, nous pouvons favoriser les trajectoires évoluant uniquement dans l'espace valide (cf. équation 1.5 page 39). Mais nous n'avons aucune garantie que ces trajectoires éviterons effectivement les obstacles à la fin du processus d'évolution. De plus, nous retrouvons tous les problèmes liés à l'utilisation d'algorithmes génétiques : évaluation multicritère, convergence prématurée, etc. (cf. pages 39 et suivantes).

Des applications aux drones aériens et sous-marins pourront respectivement être trouvées dans [RK03] et [RVR04].

3 Méthode des champs de vitesses

Dans les méthodes précédentes, le courant fait partie intégrante du processus d'optimisation de la trajectoire. La trajectoire ainsi calculée est un compromis entre deux objectifs :

1. Minimiser la distance parcourue,
2. Se faire porter par le courant.

Si le vent a une faible intensité (inférieure à 50% de la vitesse du mobile, selon les auteurs), le deuxième objectif peut être considéré comme négligeable, au profit du premier. Selon cette idée, Nelson *et al.* [NBMB06] ont proposé de procéder en deux étapes (de manière analogue à Kant et Zucker [KZ86] pour la modulation de vitesse) :

1. Planifier un chemin de longueur minimale,
2. Maintenir le mobile sur le chemin planifié.

La première étape consiste à appliquer l'une des méthodes mentionnées dans le chapitre 1. Par exemple, les auteurs ont eu recours à l'approche RRT, décrite dans la section 2.3 (page 31).

La deuxième étape correspond à une tâche classique en automatique : le suivi de consigne en présence de perturbations. Ici, le chemin joue le rôle de la consigne et les variations de courant celui des perturbations.

Ce suivi est réalisée par l'intermédiaire d'un *champ de vitesses*. Ce champ a un rôle comparable à celui d'un champ de potentiel attractif : diriger le mobile vers une région R de l'espace. Cependant, ces champs sont de nature différente :

- Dans le cas d'un champ de vitesses, R est une courbe (le chemin planifié). Le champ représente le vecteur-vitesse que doit utiliser le mobile pour rejoindre R. Les valeurs de ce champ sont directement utilisables pour commander le mobile.
- Dans le cas d'un champ de potentiel, R est un point (le point but). Le champ représente le coût pour rejoindre R. Une technique d'optimisation continue (telle que la descente du gradient) doit être utilisée pour déduire les déplacements du mobile à partir de ce champ.

La notion de champ de vitesses a été définie pour des chemins composés de segments de droite[1]. Ce champ est construit en trois étapes :

1. Calcul de champs élémentaires $\vec{c_i}$, en l'absence de courant.Pour chaque segment de droite $S_i = [A_i, B_i]$, un champ de vitesses $\vec{c_i}$ est calculé comme suit :

   Soit $M = (x, y)$ la position du mobile. Depuis A_i, le mobile forme un angle angle α_i avec S_i égal à :

   $$\alpha_i = \left\langle \widehat{\overrightarrow{A_iB_i}, \overrightarrow{MB_i}} \right\rangle = \arccos\left(\frac{\overrightarrow{A_iB_i} \cdot \overrightarrow{MB_i}}{||\overrightarrow{A_iB_i}||\ ||\overrightarrow{MB_i}||} \right) \tag{3.10}$$

   La distance d_i entre le mobile et S_i est donnée par :

   $$d_i = \frac{||\overrightarrow{A_iB_i} \wedge \overrightarrow{A_iM}||}{||\overrightarrow{A_iB_i}||} \tag{3.11}$$

   A partir de ces informations, l'angle β_i du vecteur-vitesse $\vec{c_i}$ par rapport à S_i est déterminé par :

   $$\beta_i = \left\langle \widehat{\overrightarrow{A_iB_i}, \vec{c_i}} \right\rangle = \alpha_i + s \cdot \phi \cdot \left(\frac{d_i}{d_{max}} \right)^K \tag{3.12}$$

   où $\phi \in [0, \pi/2]$ et $K \geq 1$ sont des paramètres, matérialisant respectivement l'angle d'approche à l'infini et un gain. d_{max} dénote la distance maximale entre le mobile et le chemin. Enfin, s matérialise le côté du point par rapport à S_i ($s = -1$ si M est à gauche de S_i, $s = 1$ sinon).

   Le vecteur-vitesse $\vec{c_i}$ est finalement déduit par :

   $$\vec{c_i} = v^w \cdot \begin{pmatrix} \cos\alpha_i \\ \sin\beta_i \end{pmatrix} \tag{3.13}$$

   Comme dans les approches précédentes, v^w est une constante représentant la vitesse de croisière du mobile (par rapport au courant).

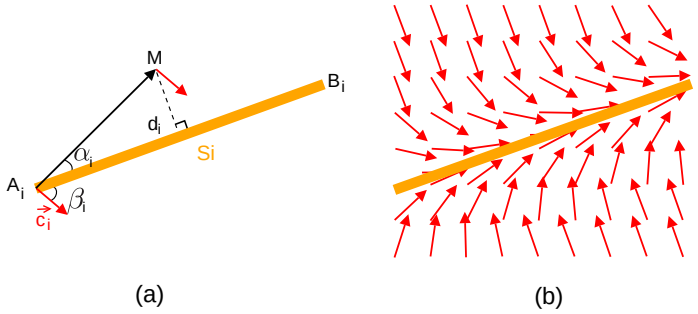

FIG. 3.7 – Champ de vitesses élémentaire

La figure illustre la construction d'un champ élémentaire avec $\phi = \pi/2$, $K = 2$ et $s = -1$.

[1] C'est le cas pour l'ensemble des méthodes présentées dans le chapitre 1, sauf pour les potentiels artificiels analytiques, pour lesquels une étape de linéarisation du chemin est nécessaire.

2. Calcul du champ global $\vec{c}$.

   L'expression de $\vec{c}$ est le champ moyen des n champs élémentaires v_i (un champ élémentaire pour chacun des n segments de droite constituant le chemin) :

$$\vec{c} = \frac{1}{n} \sum_{i=1}^{n} \vec{c_i} \tag{3.14}$$

3. Calcul du champ global corrigé $c^{\vec{w}}$, intégrant le courant.

   Cette correction consiste à ajouter les composantes du champ w matérialisant le courant à celles de v, selon un principe analogue à la composition de vitesse :

$$c^{\vec{w}} = \vec{c} + \vec{w} \tag{3.15}$$

   Si $\vec{w}$ n'est connu qu'en un ensemble restreint de points, des techniques d'interpolation sont utilisables, notamment celle à base de B-splines expliquée dans la section 1 page 81.

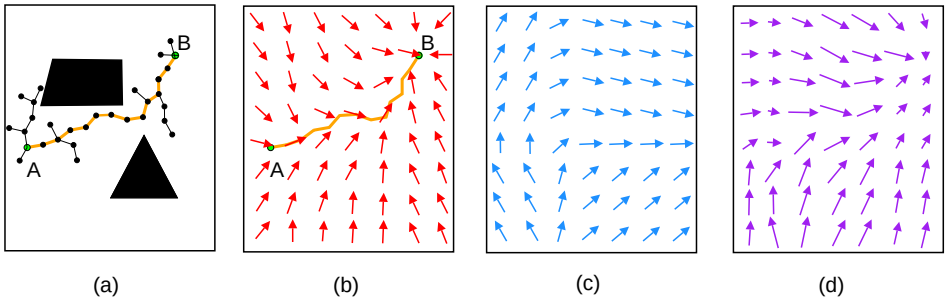

FIG. 3.8 – Champ de vitesses

La figure illustre la construction progressive d'un champ de vitesses : (a) chemin planifié par la méthode RRT ; (b) champ de vitesses $\vec{c}$ associé à ce chemin ; (c) champ de courant $\vec{w}$; (d) champ de vitesses corrigé $c^{\vec{w}}$, intégrant le courant (c).

L'approche à base de champ de vitesses constitue certainement la méthode de planification de trajectoire la plus rapide en présence de courants. Toutefois, elle n'a d'intérêt que pour des courants modérés, ce qui peut consituer une limite dans les possibilités d'application.

De plus, comme la philosophie de l'approche est de ne pas modifier le chemin calculé, les obstacles mobiles ne pourront être évités que par correction locale (par modulation de vitesse ou déformation de chemin). Ceci aboutit aux problèmes d'incomplétude évoqués dans ce même chapitre : dans certaines situations, la seule solution pour éviter l'obstacle est de le contourner ; par exemple si le robot et l'obstacle sont face à face et vont dans ses sens opposés. Ainsi, l'approche sera incapable de trouver une solution, alors qu'il en existait une.

4 La propagation d'onde

Dans la section 3.2 (page 35), nous avons présenté une technique appelée propagation d'ondes, initialement proposée pour palier les problèmes de minima locaux inhérents aux méthodes de potentiels artificiels. Cette propagation est généralement réalisée dans une grille régulière. Le front d'onde, constitué d'un ensemble de cases, est successivement étendu par relation de voisinage : le front d'onde à l'instant $t+1$ est constitué des cases voisines au front t. La différence de coût entre les deux fronts d'onde est donnée par une métrique $\mathcal{M}$.

Pour tenir compte de la présence d'un courant dans les coût, il suffit d'opter pour une métriques adaptée. Par exemple, en utilisant la métrique proposée par Garau [GAO05], citée précédemment, le coût entre deux cases C_1 et C_2 est :

$$\mathcal{M}_{C_1,C_2} = \frac{d_1}{||\vec{v^w} + \vec{w_1}||} + \frac{d_2}{||\vec{v^w} + \vec{w_2}||} \tag{3.16}$$

où d_i et $\vec{w_i}$ représentent respectivement la distance parcourue et le courant dans la case C_i. v^w est la vitesse de croisière du robot (constante).

D'autres métriques, au principe similaire, ont été proposées par Pêtrès [PPPL05] et par nous-mêmes [Sou07].

En utilisant une métrique de ce type, la propagation du front d'onde n'est plus isotrope. Le front s'étend en effet de façon privilégiée dans la direction du courant, comme on peut le voir dans la figure ci-après (à comparer à la figure 1.17 page 35). On peut notamment constater que la partie basse de l'environnement, contenant des courants défavorables, n'a pas été explorée.

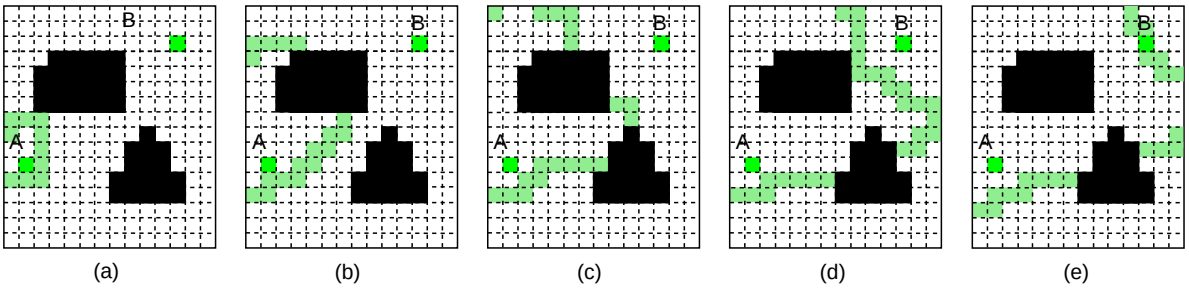

FIG. 3.9 – Propagation d'onde tenant compte du courant

Les obstacles fixes sont depuis longtemps gérés dans propagation d'onde, de façon très simple, par un processus d'exclusion des cases faisant partie de ces obstacles (auxquelles sont affectées un coût infini).

Les obstacles mobiles peuvent également être gérés en intégrant dans l'expression de $\mathcal{M}$ les éventuels ralentissements provoqués l'obstruction passagère des cases de la grille par ces obstacles. Ce principe, proposé par Kimmel *et al*, est décrit dans [KKB98].

Ainsi, avec une métrique appropriée (mélangeant les résultats de Garau et de Kimmel, par exemple), la propagation d'onde est capable de gérer la présence de courants et d'obstacles, fixes comme mobiles, et cela un temps polynomial, tout en garantissant l'absence de minima locaux.

5 Résumé

Le tableau ci-après résume les caractéristiques de l'ensemble des extensions évoquées dans ce chapitre.

La dénomination *gestion obstacles mobiles* correspond à la capacité de l'approche à permettre l'évitement d'obstacles mobiles, compte tenu des adaptations existantes (i.e. proposées dans la littérature). Les adaptations potentielles de l'approche ne sont pas considérées.

Méthode	Déterministe	Optimale	Paramètres	Gestion obstacles mobiles
Optimisation de B-Splines	oui	non	nb points contrôle	non
Algorithmes génétiques	non	non	nb générations types opérateurs taille population	non
Champs de vitesse	oui	non	non angle d'approche gain	non
Potentiels artificiels	oui	oui	résolution grille	oui

La méthode d'optimisation à base de B-splines est très élégante, car elle modélise l'ensemble des données du problème à travers une fonction unique. Toutefois, l'intégration des obstacles, particulièrement des obstacles mobiles, semble délicate. Selon les contraintes choisies, le temps de calcul n'est plus maîtrisé, ou la validité de la solution n'est plus garantie. Autre point faible de l'approche : elle fournit aucune garantie sur la qualité de la solution fournie (distance par rapport à la solution optimale).

Les algorithmes génétiques sont très lents par nature, et tous les problèmes liés au réglage des paramètres subsistent (fastidieux et spécifique au problème).

Les champs de vitesse imposent une hypothèse forte sur l'intensité des courants (qui doit être modérée). Ceci rend le champ d'application de cette méthode assez limité. D'autre part, l'introduction d'obstacles mobiles serait source d'incomplétude, puisque seule la vitesse du robot est modifiée (et jamais le chemin sur lequel il se déplace). Or, de nombreux cas, notamment tous les cas de face-à-face, imposent des contournements spatiaux et donc des modifications locales du chemin.

La propagation d'onde semble donc la plus appropriée pour tenir compte des courants : elle ne nécessite pas le réglage fastidieux de paramètres, et permet une intégration aisée des obstacles (fixes comme mobiles). De plus, il s'agit de la seule approche globalement optimale en un temps polynomial.

Toutefois, nous allons voir dans les chapitres suivants que cette technique, utilisée en tant que telle, n'est pas satisfaisante dans deux cas particuliers de courants : les courants forts (i.e. plus rapides que le robot) et les courants variables dans le temps. C'est pourquoi nous introduisons dans cette thèse deux extensions de la propagation d'onde : la propagation d'onde coulissante (dans le chapitre 4) et la propagation d'onde symbolique (dans le chapitre 4). Ces deux extensions constituent les contributions majeures de la thèse.

Deuxième partie

Principales contributions

Introduction

En introduction de cette thèse, nous avons vu en que les robots mobiles étaient utilisés de façon grandissante pour réaliser des missions pouvant être répétitives, longues ou dangereuses (reconnaissance, surveillance, sauvetage, etc.). Dans cette partie de la thèse, nous nous intéressons à une catégorie particulière de ces robots mobiles : les drones, ou avions sans pilote. De par leur faible vitesse et leur petite taille, les drones sont particulièrement sensibles aux courants, ce qui peut avoir des conséquences très importantes sur la réussite de la mission, voire sur la survie du drone.

En effet, les courants peuvent significativement ralentir le drone, augmentant ainsi sa consommation d'énergie ou son temps de parcours. Une consommation d'énergie excessive peut conduire à une rupture de ressources en pleine mission, et donc à l'immobilisation (voire au crash) du drone en terrain hostile. Une augmentation importante du temps de parcours peut quant à elle compromettre les conditions d'atterrissage du drone ou la réalisation d'objectifs temporellement contraints. Ces deux problèmes sont particulièrement présents dans le cas de missions de longue durée, où il devient crucial d'anticiper les changements de courants dans le temps.

De plus, si les courants deviennent forts (c'est à dire plus rapides que le drone lui-même), ceux-ci peuvent mener à une dérive incontrôlable du drone. Une première conséquence est de rendre certaines zones de l'environnement inaccessibles, notamment des zones que le drone devait survoler, ce qui peut aboutir à l'échec d'une partie voire de la totalité de la mission.

Pour répondre à ces problématiques, nous présentons dans cette partie deux nouveaux algorithmes de planification de trajectoire en présence de courants. Ces deux algorithmes, qui constituent les contributions majeures de cette thèse, sont des extensions propagation d'onde [DT88] présentée à la page 35 de l'état de l'art.

Tout d'abord, le chapitre 4 introduit une méthode de planification de trajectoire capable de gérer les courants forts, c'est à dire les courants plus rapides que le drone. En présence de tels courants, les techniques proposées dans la littérature ne sont pas fiables. En effet, elles peuvent fournir soit des chemins invalides (c'est à dire physiquement irréalisables), soit aucun chemin, même si des solutions valides existent. Notre nouvelle approche, appelée la *propagation d'onde coulissante* [STR08a] permet d'augmenter significativement le succès de planification (c'est à dire la capacité à trouver une solution valide si elle existe) par rapport aux techniques existantes.

Ensuite, le chapitre 5 introduit une méthode de planification de trajectoires capable de gérer les courants variables dans de temps, ce qui n'est possible, à notre connaissance, avec aucune méthode de la littérature. Cette nouvelle méthode, appelée *propagation d'onde symbolique* [STR09], calcule la meilleure date de départ pour le drone pour minimiser son temps de parcours, en anticipant les changements de courants dans le temps.

Chapitre 4

Planification de trajectoire en présence de courants forts

Sommaire

1 Introduction

Dans de nombreuses applications, des robots mobiles peuvent être amenés à évoluer dans des environnements soumis à des courants. Notamment, des véhicules autonomes aériens, ou drones, sont de plus en plus utilisés dans des contextes de surveillance maritime (contrôles de douane, de pêche, de pollution), d'inspection ou de réparation d'infrastructures (câbles, pipelines, etc).

Ces drones, de taille et de poids souvent réduits sont particulièrement sensibles aux courants. En particulier, si les courants deviennent *forts*, c'est à dire plus rapides que le drone, alors le drone se fera inévitablement entraîner dans la direction des courants (quelque soit la commande de son moteur). Dans le meilleur des cas, cela peut rendre certaines zones inaccessibles (le point d'arrivée, notamment) ce qui aboutira à l'échec de la mission. Dans le pire des cas, le courant peut porter le drone vers un obstacle et entraîner une collision.

Dans ces conditions, nous allons voir que les algorithmes existants (décrits dans le chapitre 3) ne sont pas satisfaisants, en termes de fiabilité, de qualité de la solution ou de performances. Nous proposons donc dans ce chapitre une nouvelle approche appelée *propagation d'onde coulissante*. Il s'agit d'une extension de la propagation d'onde (décrite page 35) dans le domaine continu. Celle-ci utilise un nouveau type de support, appelé *curseur*, à la place des traditionnelles cases pour propager les coûts.

2 Formulation du problème

Le problème consiste à trouver la trajectoire permettant à un robot de relier deux points A et B en un temps minimal, dans un environnement planaire contenant des courants, comme illustré dans la figure 4.1.

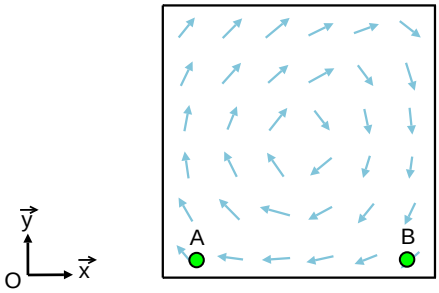

FIG. 4.1 – Un problème de planification de trajectoire en présence de courants
A et B dénotent les points de départ et d'arrivée et les flèches bleues les échantillons de courant (vecteurs vitesses connus en un nombre fini de points).

Nous rappelons que le robot est modélisé comme un objet ponctuel évoluant dans un espace Euclidien $\mathcal{C}$ de dimension 2, de repère orthonormé $\mathcal{R} = (O, \vec{x}, \vec{y})$.

Les courants sont modélisés par un champ vectoriel de dimension 2, noté $\vec{w}$. Ce champ est connu en un nombre fini de points de $\mathcal{C}$, appelés *échantillons* (flèches bleues dans la figure 4.1). Ces échantillons correspondent à des prévisions météorologiques.

La vitesse du robot est notée $\vec{v^r}$ si elle est relative à $\mathcal{R}$ et $\vec{v^w}$ si elle est relative au champ $\vec{w}$. Notons que par principe de composition des vitesses, ces deux quantités sont liées par la relation :

$$\vec{v^r} = \vec{v^w} + \vec{w} \tag{4.1}$$

Notre problème est caractérisé par les deux hypothèses suivantes sur les modules des vitesses :

- $v^w = constante \neq 0$. Cela signifie que le robot se déplace à une vitesse fixée et non nulle par rapport aux courants, appelée *vitesse de croisière*. Cette vitesse est uniquement due à la commande des moteurs du robot.

- w peut être supérieur à v^w. Cela signifie que les courants peuvent plus rapides que le robot. C'est dans ce contexte particulier que les méthodes existantes sont mises en défaut, ce qui en justifie une nouvelle.

3 Limitations des méthodes existantes

3.1 Identification des méthodes applicables au problème

Comme nous l'avons vu dans première partie de cette thèse, un grand nombre d'approches ont été proposées, depuis les années 80, dans le cadre de la planification de chemin (chapitre 1) puis de la planification de trajectoire (chapitre 2). Cependant, l'ajout de courants dans l'environnement réduit considérablement le nombre de méthodes applicables. Dans le chapitre 3, nous avons répertorié les méthodes suivantes :

- l'optimisation de B-Splines,
- les algorithmes génétiques,
- les champs de vitesses,
- la propagation d'onde.

Dans le contexte de ce chapitre, nous pouvons éliminer d'amblée les champs de vitesses, puisque ceux-ci ne fonctionnent qu'en présence de courants modérés (50% de la vitesse du robot au plus). Or, dans ce chapitre nous savons, par la définition du problème ci-dessus, que les courants peuvent être plus rapides que le robot.

L'optimisation de B-splines, bien que très élégante, ne permet pas de traiter aisément la présence d'obstacles. Ceux-ci ne sont pas inclus dans le modèle proposé, et les quelques pistes que nous avons identifiées pour le faire ne sont pas satisfaisantes.

Une première piste consisterait à maintenir une distance minimale δ_{min} entre les points de contrôle de la B-spline et les obstacles, en utilisant les contraintes proposées par Borrow [Bor88] (cf. page 59). Toutefois, ce type de contrainte est assez faible, dans la mesure où l'évitement des obstacles par les points contrôle n'implique pas systématiquement l'évitement des obstacles par B-spline. Ce défaut peut être limité en introduisant beaucoup de points de contrôle, mais aucune garantie ne peut être fournie.

Une deuxième piste consisterait à approximer les obstacles par excès, en les encadrant par des rectangles, puis énumérer à l'aide de variables binaires, les différentes possibilités de contournement (gauche, droite, haut, bas). Cette approche, utilisée par How [RH02] dans le cadre de la Programmation Linéaire Mixte (cf. page 51) peut aboutir à un temps de calcul important sur des exemples pathologiques. En effet, il est en théorie en $O(exp(n))$ pour n obstacles dans le pire des cas.

Les algorithmes génétiques souffrent de problèmes similaires concernant les obstacles. En effet, dans ce type d'approche, les chemins sont modélisés par des individus, dont la probabilité de survie dépend du nombre de collisions avec des obstacles : plus le nombre de collisions est grand, moins l'individu a de chances de survivre. Toutefois, si des individus invalides (c'est à dire intersectant des obstacles) sont de très bonne qualité vis-à-vis du critère à optimiser (par exemple le temps de parcours), ceux-ci peuvent tout à fait l'emporter sur des individus valides ! En plus de ces problèmes de fiabilité, les algorithmes génétiques sont assez laborieux à paramétrer. En effet, le nombre de paramètres est assez important (taille de la population, types d'opérateurs et probabilités associées, nombre de générations, etc.), et leur valeur dépend fortement du problème considéré. Ainsi, il est notamment possible que la valeur des paramètres dépende de l'intensité des courants : les paramètres adaptés à des courants modérés ne le seront pas forcement pour des courants forts.

Enfin, il faut mentionner que les algorithmes génétiques, tout comme l'optimisation de B-splines, n'offrent aucune garantie de trouver l'optimum global du problème, puisque les fonctions manipulées ne sont pas convexes. De nombreuses approches peuvent être utilisées pour "décoincer" les algorithmes d'optima locaux (notamment les mouvements aléatoires proposés par Barraquand [BL90]), mais elles ne font que rallonger un temps de calcul qui est déjà important.

Reste donc la propagation d'onde. Cette technique ne possède pas les inconvénients cités ci-dessus, en effet :

- La fonction de coût construite par propagation a la propriété fondamentale de ne pas posséder de minimum local. Dans ces conditions, le chemin construit par hill climbing (l'équivalent discret de la descente du gradient) est garanti d'être globalement optimal (dans l'espace discrétisé qu'est la grille).
- L'intégration des obstacles, fixes comme mobiles, est aisée. Pour en tenir compte, la fonction de coût est localement modifiée. Un coût infini (très grand dans la pratique) est associé aux cases appartenant aux obstacles fixes, ce qui provoque leur évitement systématique. Pour les obstacles mobiles, une adaptation a été proposée dans [KKB98], consistant à intégrer la fonction de coût les éventuels ralentissements provoqués l'obstruction passagère des cases.
- Le temps de calcul dans le pire des cas est connu et raisonnable. Il a l'énorme avantage de ne pas augmenter avec le nombre d'obstacles. Ainsi, le temps de calcul n'est sensible qu'au nombre de cases. Si la grille possède N cases, il est en $O(N \log N)$. Cette caractéristique fait de la propagation d'onde l'une des méthodes les plus utilisées aujourd'hui dans les applications de robotique mobile, où les temps de réponse doivent être extrêmement courts.

La propagation d'onde semble donc être le meilleur candidat pour traiter les courants forts. Dans le chapitre 3, nous avons mentionné que trois extensions de la propagation d'onde avaient été proposées pour tenir compte des courants : deux proposées en 2005, par Pêtrès [PPPL05] et Garau [GAO05], puis une proposée en 2007 par nous-mêmes [Sou07]. Si celles-ci fonctionnent de façon tout à fait convenable en présence de courants modérés, ce n'est plus le cas en présence de courants forts.

En effet, les deux premières extensions (Pêtrès et Garau) souffrent de problèmes d'*incorrection*, c'est à dire qu'elles sont susceptibles de renvoyer des trajectoires qui ne sont pas physiquement réalisables par le robot. Quand à la notre, elle souffre de problèmes d'*incomplétude*. Cette fois, les trajectoires retournées sont toujours valides, mais il peut arriver que l'algorithme échoue à trouver une solution, même s'il en existe. En d'autres termes, les deux premières extensions sont trop optimistes et la dernière trop pessimiste. C'est ce que nous allons voir ci-après.

3.2 Problèmes d'incorrection

La philosophie des extensions proposées par Pêtrès et Garau est la même : modifier la fonction de coût (qui représente généralement le temps de parcours) pour y intégrer l'influence des courants. Schématiquement, le temps de parcours va se trouver diminué si le robot suit les courants (puisqu'il se fait porter), et augmenté dans le cas contraire.

Dans cette partie, nous montrons que ces fonctions de coût sont *invalides*. C'est à dire qu'elles identifient des cases comme atteignables, alors que le mouvement à effectuer pour atteindre ces cases est physiquement irréalisable.

Pour illustrer ce propos, considérons l'exemple suivant : le robot, ayant une vitesse de croisière v^w de $100km/h$, doit effectuer un mouvement unitaire $\vec{d}$ selon $\vec{x}$ dans un courant opposé, légèrement plus rapide que lui, noté $\vec{w}^{opp} = -120\vec{x}$. Bien évidemment, ce mouvement est impossible, puisque la vitesse du robot par rapport au sol est $\vec{v^r} = -20\vec{x}$.

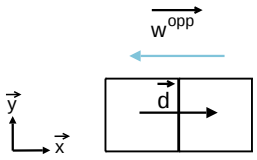

FIG. 4.2 – Exemple de mouvement impossible

Le courant $w^{\vec{o}pp}$, plus rapide que le robot, fait dériver le robot dans le sens inverse du mouvement $\vec{d}$. Ce mouvement n'est donc pas physiquement réalisable.

Voyons ce que nous obtenons en utilisant les fonctions de coût de Pêtrès et Garau.

Tout d'abord, Pêtrès a proposé dans [PPPL05] la fonction composite suivante :

$$\tau_1 = \tau_1^{dist} + \gamma\tau_1^{cur} \tag{4.2}$$

Cette fonction peut être interprétée comme suit : la quantité τ_1^{dist} mesure la distance parcourue et τ_1^{cur} la différence angulaire entre la direction de déplacement du robot et celle du courant. Ainsi, τ_1 correspond à un compromis entre minimiser la distance parcourue et pointer dans la même direction que le courant. Ce compromis est réglé par le paramètre γ. Il fixe le seuil à partir duquel le robot va effectuer un détour pour aller chercher des courants favorables.

Dans notre exemple, nous avons :

$$\begin{cases} \tau_1^{dist} &= d \\ \tau_1^{cur} &= 1 - \tau_1\vec{x} \cdot w^{\vec{o}pp}/Q \end{cases} \tag{4.3}$$

où $Q = (d + 2\gamma)w^{max}$ est un coefficient de normalisation, w^{max} l'intensité maximale du courant l'environnement, γ un gain positif et $\cdot$ le produit scalaire.

En utilisant ces informations, l'équation 4.2 devient :

$$\tau_1 = \frac{d + \gamma}{1 + (\gamma/Q)w_x^{opp}} \tag{4.4}$$

Enfin, sachant que $w^{max} = w^{opp}$ et que $d = 1$, le coût du mouvement $\vec{d}$ sans courant est :

$$\tau_1 = 1 + \gamma \tag{4.5}$$

Alors qu'en présence de $w^{\vec{o}pp}$ ce coût devient :

$$\tau_1' = \frac{1 + \gamma}{1 - \frac{\gamma}{(1+2\gamma)}} = k_1 \cdot \tau_1 \tag{4.6}$$

En d'autres termes, la présence de $w^{\vec{o}pp}$ pénalise le coût de $\vec{d}$ par un facteur k_1 égal à :

$$k_1 = \frac{1}{1 - \frac{\gamma}{1+2\gamma}} \tag{4.7}$$

Comme expliqué précédemment, γ est un gain positif. $\gamma = 0$ donne $k_1 = 1$, et $\gamma \to +\infty$ donne $k_1 \to 2$. Ainsi, quelque soit la valeur de γ, le mouvement est au plus pénalisé par un facteur 2.

Ainsi, en utilisant τ_1, un chemin invalide (i.e. physiquement irréalisable) de longueur l sera potentiellement préféré à un chemin valide de longueur $2l$. Dans ces conditions, un planificateur basé sur τ_1 pourra renvoyer des chemins qui ne seront pas physiquement exécutables par le robot.

C'est notamment le cas pour notre exemple de la figure 4.1. Le résultat obtenu en appliquant l'approche de Pêtrès (avec $\gamma = 1$) est présenté dans la figure 4.3. La partie (a) présente la propagation d'onde obtenue. On peut constater que cette propagation n'est pas isotrope. Ceci est normal, car le front d'onde explore en premier les régions de moindre coût, c'est à dire ici les régions où le courant "pousse" le robot. La partie (b) présente ensuite le chemin obtenu par hill climbing dans la fonction τ_1. Ce chemin est totalement invalide (matérialisé par la couleur rouge).

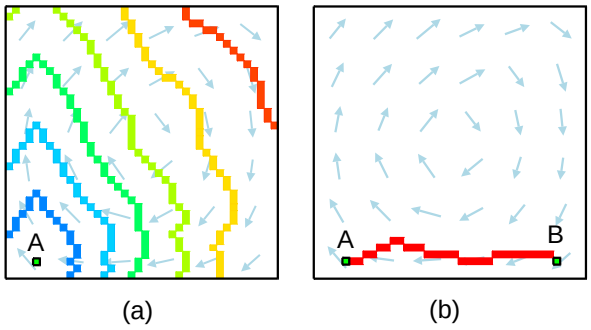

FIG. 4.3 – Résultat obtenu en utilisant la fonction de coût τ_1
(a) propagation d'onde (bleu=coût faible, rouge=coût élevé) ; (b) chemin obtenu (rouge=invalide). Les vitesses des courants et du robot sont respectivement $w = 120$ km/h et $v^w = 100$ km/h.

Cette invalidité est due à la nature même de τ_1. En effet, cette fonction n'a pas de sens physique particulier. Le compromis entre suivi des courants et détour est réglé de façon empirique par l'utilisateur, par l'intermédiaire du gain γ, mais ce réglage n'est pas forcement fidèle à l'impact réel des courants sur le robot.

L'approche proposée par Garau est un pas vers plus de réalisme. Dans son article [GAO05], Garau a introduit la fonction de coût suivante :

$$\tau_2 = \frac{d}{||\vec{v^w} + \vec{w^{opp}}||} \tag{4.8}$$

où $||\cdot||$ représente la norme Euclidienne.

L'idée de cette fonction de coût est de directement refléter l'impact des courants sur la vitesse effective du robot, sans impliquer l'utilisateur dans une quelconque pondération. Un courant porteur va globalement augmenter cette vitesse, alors qu'un courant opposé aura tendance à la diminuer. Ceci est matérialisé par le dénominateur de τ_2.

Si nous reprenons le mouvement $\vec{d}$ de la figure 4.2, son coût est $\tau_2 = 1/100$ sans courant, et $\tau_2' = 1/20 = 5\tau_2$ en présence de $w^{\vec{o}pp}$. Autrement dit, la présence du courant $w^{\vec{o}pp}$ est ici pénalisée par un facteur $k_2 = 5$. Cette pénalisation est plus sévère que précédemment, mais n'interdit pas à coup sûr ce mouvement, qui est pourtant impossible. Dans certaines situations, certaines parties de chemins peuvent donc demeurer invalides.

C'est ce que l'on peut observer sur notre exemple de la figure 4.1. La figure 4.4 présente le résultat obtenu en utilisant l'approche de Garau. Premier constat en observant la partie (a) : la propagation est non isotrope, comme précédemment, mais semble mieux "épouser" la forme du tourbillon formé par les courants. Cela signifie que l'impact des courants sur le front d'onde, et donc sur les chemins, paraît plus important. C'est pour cela que dans la partie (b), le chemin obtenu est beaucoup plus dévié que précédemment (par rapport au chemin direct, sans courant : la ligne horizontale). Toutefois, cette déviation n'est encore pas suffisante, puisque certaines parties du chemin demeurent invalides (les diagonales, en rouge).

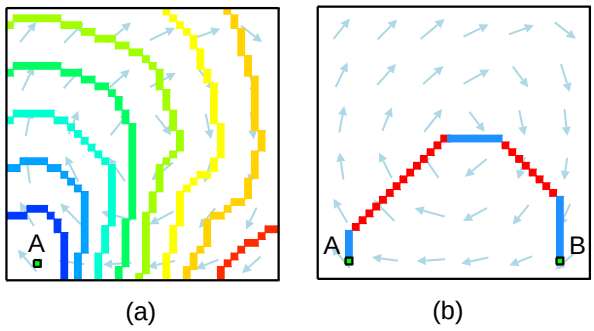

FIG. 4.4 – Résultat obtenu en utilisant la fonction de coût τ_2
(a) propagation d'onde (bleu=coût faible, rouge=coût élevé) ; (b) chemin obtenu (bleu=valide, rouge=invalide). Les vitesses des courants et du robot sont respectivement $w = 120$ km/h et $v^w = 100$ km/h.

Pour résumer, l'utilisation de fonctions de coût dites *invalides*, c'est à dire ne rejetant pas catégoriquement les mouvements impossibles, peut aboutir à des chemins non réalisables. Pêtrès et Garau sont deux exemples donnés à titre illustratif, mais le principe des fonctions est toujours le même : pénaliser ou favoriser, à l'aide un coefficient k, les régions de l'environnement selon la configuration des courants.

Si ce mécanisme empirique fonctionne dans le cas de courants modérés, il n'est plus satisfaisant en cas de courants forts. En effet, le seul moyen de minimiser les risques de chemins invalides est de pénaliser très fortement les chemins qui ne vont pas dans le même sens que les courants, en choisissant une très grande valeur pour k. Mais ceci pose le problème de privilégier le suivi des courants, au détriment de la distance parcourue. Autrement dit, le robot pourra être amené à faire des détours très importants pour se faire porter par des courants à l'autre bout de l'environnement, alors que cela n'en vaut pas forcement la peine.

Pour éviter ces problèmes, il semble plus pertinent que la fonction τ représente directement le critère que nous souhaitons minimiser, c'est à dire le temps de parcours. Mais il faut aussi que τ reflète de façon explicite les mouvements impossibles : il faut un temps infini pour rejoindre une case non atteignable.

C'est dans cette optique que nous avons proposé la fonction τ_3 présentée ci-après. Toutefois, si l'usage de τ_3 règle les problèmes d'incorrection, elle en introduit de nouveaux : des problèmes d'incomplétude. C'est à dire que l'algorithme peut échouer à fournir une solution, même s'il en existe. C'est ce que nous allons développer ci-dessous.

3.3 Problèmes d'incomplétude

Dans [Sou07], nous avons introduit une nouvelle fonction de coût τ_3, représentant le temps de parcours *réel* du robot, c'est à dire intégrant l'influence des courants. τ_3 a été obtenue en exploitant le principe de composition de vitesses introduit précédemment. Ce raisonnement est expliqué ci-après.

Considérons un mouvement $\vec{d}$ dans un courant $\vec{w}$. Le temps de parcours τ_3 associé à $\vec{d}$ vérifie :

$$\frac{\vec{d}}{\tau_3} = \vec{v^r} = \vec{w} + \vec{v^w} \tag{4.9}$$

Cette relation traduit deux choses :
- Les vecteurs $\vec{d}$ et $\vec{v^r}$ sont colinéaires,
- La norme de $\vec{v^r}$ est plus faible que celle de $\vec{d}$. Le facteur entre ces deux normes est τ_3.

Ces propriétés sont visibles dans la figure 4.5.

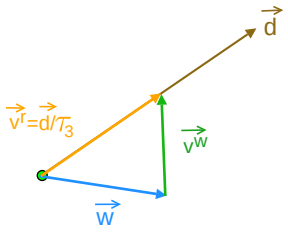

FIG. 4.5 – Illustration de la composition des vitesses

Le temps de parcours τ_3 peut être vu comme un gain entre les vecteurs $\vec{d}$ et $\vec{v^r}$.

Contrairement à Garau (équation 4.8), nous n'allons pas directement appliquer la norme Euclidienne l'équation 4.9, mais nous allons la projeter sur les axes $\vec{x}$ et $\vec{y}$, ce qui donne, après réécriture :

$$\begin{cases} v_x^w = \frac{d_x}{\tau_3} - w_x \\ v_y^w = \frac{d_y}{\tau_3} - w_y \end{cases} \tag{4.10}$$

Nous savons également que la norme de $\vec{v^w}$ vérifie :

$$v^{w2} = v_x^{w2} + v_y^{w2} \tag{4.11}$$

En combinant les équations 4.10 et 4.11, il vient :

$$(d_x - w_x \cdot \tau_3)^2 + (d_y - w_y \cdot \tau_3)^2 = v^{w2} \cdot \tau_3^2 \tag{4.12}$$

Géométriquement parlant, l'expression 4.12 est l'équation d'un cône oblique, appelé *cône d'accessibilité* et illustré dans la figure 4.6. L'axe de ce cône a pour vecteur directeur $\tau_3 \cdot \vec{w}$. Notons que dans le cas particulier de l'absence de courants ($\vec{w} = \vec{0}$), on obtient un cône droit.

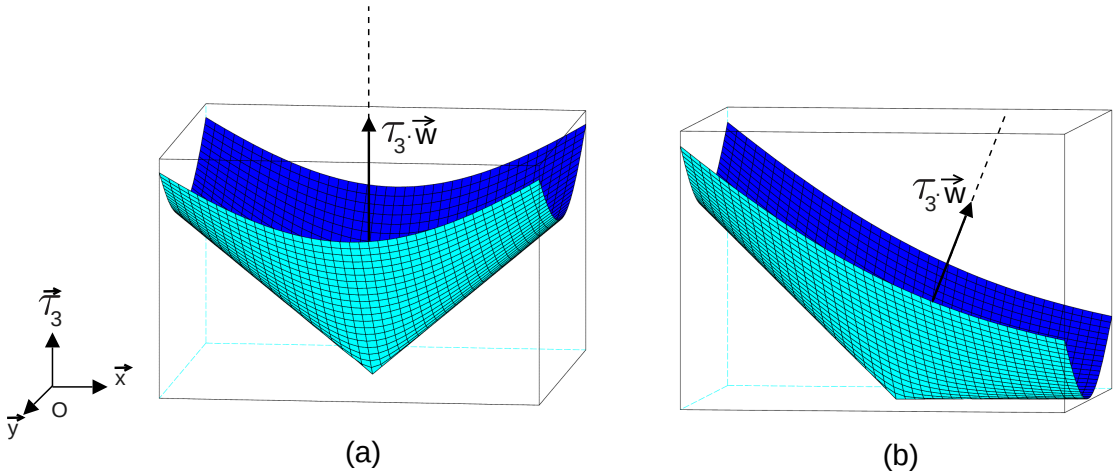

FIG. 4.6 – Le cône d'accessibilité
(a) en absence de courants ($\vec{w} = \vec{0}$) ; (b) en présence d'un courant latéral $\vec{w} = 70\vec{x}$. La vitesse de croisière du robot est $v^w = 100km/h$.

Analytiquement parlant, l'expression 4.12 est un équation du second degré en τ_3, que l'on peut réécrire de la façon suivante :

$$(w^2 - v^{w2}) \cdot \tau_3^2 - 2 \cdot \vec{d} \cdot \vec{w} \cdot \tau_3 + d^2 = 0 \tag{4.13}$$

τ_3 est donc la plus petite racine positive de cette équation. On montre que τ_3 est donné par :

$$\tau_3 = \frac{\sqrt{\Delta} - \vec{d} \cdot \vec{w}}{v^{w2} - w^2} \tag{4.14}$$

avec $\Delta = v^{w2} \cdot (d_x^2 + d_y^2) - (w_x \cdot d_y - w_y \cdot d_x)^2$.

Notons que dans le cas particulier de $v^w = w$, nous avons :

$$\tau_3 = \frac{d^2}{2\vec{d} \cdot \vec{w}} \tag{4.15}$$

PREUVE

Supposons tout d'abord que $v^w \neq w$.

Si $\Delta \geq 0$, l'équation 4.13 possède deux racines τ_3^1 et τ_3^2 (potentiellement confondues) données par :

$$\tau_3^1 = \frac{-\vec{d} \cdot \vec{w} - \sqrt{\Delta}}{v^{w2} - w^2} \quad \text{et} \quad \tau_3^2 = \frac{-\vec{d} \cdot \vec{w} + \sqrt{\Delta}}{v^{w2} - w^2}$$

Ces racines sont liées par la relation suivante :

$$\tau_3^1 \cdot \tau_3^2 = \frac{d^2}{w^2 - v^{w2}}$$

Il y a alors deux cas à considérer, selon le signe de $w^2 - v^{w2}$.

▷ 1er cas : $w^2 - v^{w2} < 0$ (c'est à dire $w < v^w$: le courant est moins rapide que le robot)

Les racines sont de signe opposé.

Comme illustré dans la figure 4.7, la racine positive correspond à la solution à notre problème : $\vec{v^r}$ est dans le sens du mouvement $\vec{d}$ et le temps croissant. La racine négative correspond quant à elle à un mouvement opposé à $\vec{d}$, combiné à un temps décroissant. Si cette solution a une existence mathématique, elle n'est pas réaliste au sens physique.

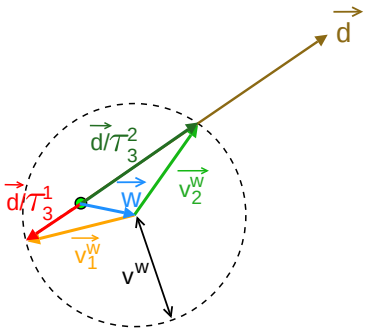

FIG. 4.7 – Composition des vitesses, avec $w < v^w$

Les racines τ_3^1 et τ_3^2 correspondent aux l'intersections du cercle de rayon v^w avec la droite de vecteur directeur $\vec{d}$. Seul le mouvement associé à τ_3^2 (en vert foncé) est physiquement valide. En effet, le mouvement associé à τ_3^1 (en rouge) est dans le sens contraire à $\vec{d}$ et suppose un temps négatif.

Nous savons que $\sqrt{\Delta} \geq 0$ et que $v^{w2} - w^2 > 0$, donc $\tau_3^2 \geq \tau_3^1$. Dès lors, τ_3^2 est nécessairement la racine positive (car si τ_3^2 était négative, τ_3^1 serait aussi négative, ce qui est en contradiction avec l'opposition de signe).

La quantité τ_3 recherchée est donc donnée par τ_3^2.

▷ 2ème cas : $w^2 - v^{w2} > 0$ (c'est à dire $w > v^w$: le courant est plus rapide que le robot)

Les deux racines sont de même signe. La question est de savoir si elles sont toutes deux négatives (dans ce cas, le problème est impossible), ou toutes deux positives (le problème a deux solutions).

Nous savons que $w^2 - v^{w2} > 0$. Si nous exploitons cette inégalité, nous obtenons :

$$
\begin{aligned}
& w^2 - v^{w2} > 0 \\
\Leftrightarrow\ & (w^2 - v^{w2}) \cdot d^2 \geq 0 \\
\Leftrightarrow\ & [(w_x^2 + w_y^2) - v^{w2}] \cdot (d_x^2 + d_y^2) \geq 0 \\
\Leftrightarrow\ & d_x^2 \cdot w_x^2 + d_y^2 \cdot w_y^2 \geq v^{w2} \cdot (d_x^2 + d_y^2) - w_x^2 \cdot d_y^2 - w_y^2 \cdot d_x^2 \\
\Leftrightarrow\ & d_x^2 \cdot w_x^2 + d_y^2 \cdot w_y^2 + 2 \cdot w_x \cdot w_y \cdot d_x \cdot d_y \geq v^{w2} \cdot (d_x^2 + d_y^2) - w_x^2 \cdot d_y^2 - w_y^2 \cdot d_x^2 + 2 \cdot w_x \cdot w_y \cdot d_x \cdot d_y \\
\Leftrightarrow\ & (d_x \cdot w_x + d_y \cdot w_y)^2 \geq v^{w2} \cdot (d_x^2 + d_y^2) - (w_x \cdot d_y - w_y \cdot d_x)^2 \\
\Leftrightarrow\ & (\vec{d} \cdot \vec{w})^2 \geq \Delta \geq 0 \\
\Leftrightarrow\ & \vec{d} \cdot \vec{w} \geq \sqrt{\Delta} \\
\Leftrightarrow\ & \frac{\sqrt{\Delta} - \vec{d} \cdot \vec{w}}{v^{w2} - w^2} \geq 0 \\
\Leftrightarrow\ & \tau_3^2 \geq 0
\end{aligned}
$$

τ_3^2 est positive, donc les deux racines sont positives.

Comme l'illustre la figure 4.8, τ_3^1 et τ_3^2 correspondent à deux façons différentes de faire le mouvement $\vec{d}$: une façon lente (en luttant contre le courant) et une façon rapide (en s'aidant du courant).

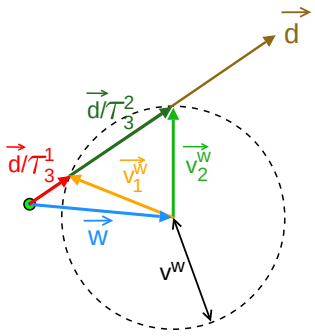

FIG. 4.8 – Composition des vitesses, avec $w > v^w$

Les mouvements associés à τ_3^1 et τ_3^2 sont tous deux physiquement valides (dans le sens de $\vec{d}$). Toutefois, le mouvement associé à τ_3^1 (en rouge) tire beaucoup moins parti du courant que le mouvement associé à τ_3^2 (en vert foncé).

Comme nous cherchons à minimiser le temps de parcours, nous souhaitons utiliser la plus petite des racines. Nous savons que $\sqrt{\Delta} \geq 0$ et que $v^{w2} - w^2 < 0$, donc $\tau_3^2 \leq \tau_3^1$.

La quantité τ_3 recherchée est donc une nouvelle fois donnée par τ_3^2.

Dans les deux cas, la solution à notre problème est la racine τ_3^2.

Supposons à présent que $v^w = w$. Dans ces conditions, le facteur $w^2 - v^{w2}$ de l'équation 4.13 devient nul. Dès lors, l'équation n'est que du premier degré. Nous avons en effet :

$$-2 \cdot \vec{d} \cdot \vec{w} \cdot \tau_3 + d^2 = 0$$

D'où le résultat de l'équation 4.15.□

Comme les fonctions de coût proposées par Pêtres et Garau, τ_3 incite le robot à suivre les courants. En effet, si le robot et le courant vont dans la même direction, alors le produit scalaire $\vec{d} \cdot \vec{w}$, et donc τ_3 diminue. Les régions où le courant pousse le robot sont donc favorisées.

Mais τ_3 indique également les régions que le robot peut atteindre en tout point de l'environnement, selon le courant $\vec{w}$ en présence. En effet, l'équation 4.12 peut être vue comme l'équation d'un cercle dont le centre et le rayon évolue dans le temps. Ce cercle délimite à tout moment la région accessible par le robot. Le centre du cercle se déplace à la vitesse w, et son rayon augmente à la vitesse v^w. Nous avons plusieurs cas, selon les valeurs respectives de w et v^w :

1. Si $w < v^w$ (fig. 4.9a), le centre du cercle se déplace plus lentement que son rayon ne grossit. Ainsi, l'ensemble des points atteignables par le robot est donc tout le plan.
2. Si $w = v^w$ (fig. 4.9b), le centre du cercle se déplace à la même vitesse que son rayon ne grossit. L'ensemble des points atteignables par le robot est donc le demi-plan dans situé du côté de $\vec{w}$, délimité par la droite normale à $\vec{w}$.
3. Enfin, si $w > v^w$ (fig. 4.9c), le centre du cercle se déplace plus vite que son rayon ne grossit. L'ensemble des points atteignables par le robot est donc un secteur angulaire $\Phi = [-\varphi, \varphi]$, appelé *secteur d'accessibilité*, avec :

$$\varphi = \arccos\left(\frac{\sqrt{w^2 - v^{w2}}}{w}\right) \tag{4.16}$$

Notons que si $w \to v^w$, on a $\varphi \to \pi/2$. On retrouve le cas 2.

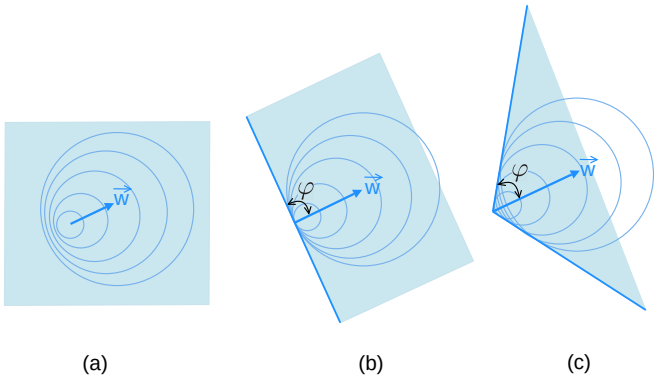

FIG. 4.9 – Secteur d'accessibilité
Plus le courant s'intensifie, plus la zone accessible par le robot, appelé *secteur d'accessibilité* et illustrée en bleu clair, se réduit. (a) $w = 80km/h$; (b) $w = 100km/h$; (c) $w = 120km/h$. La vitesse de croisière du robot est $v^w = 100km/h$.

PREUVE

Si $w \neq v^w$, τ_3 est donné par l'équation 4.14. Cette expression n'est définie que pour $\Delta \geq 0$. Si nous exploitons cette condition, il vient :

$$
\begin{aligned}
& \Delta \geq 0 \\
\Leftrightarrow \quad & v^{w2} \cdot (d_x^2 + d_y^2) - (w_x \cdot d_y - w_y \cdot d_x)^2 \geq 0 \\
\Leftrightarrow \quad & v^{w2} \cdot d^2 - w_x^2 \cdot d_y^2 - w_y^2 \cdot d_x^2 \geq -2 \cdot w_x \cdot w_y \cdot d_x \cdot d_y \\
\Leftrightarrow \quad & v^{w2} \cdot d^2 - w_x^2 \cdot d_y^2 - w_y^2 \cdot d_x^2 - w_y^2 \cdot d_x^2 - w_x^2 \cdot d_y^2 \geq -2 \cdot w_x \cdot w_y \cdot d_x \cdot d_y - w_y^2 \cdot d_x^2 - w_x^2 \cdot d_y^2 \\
\Leftrightarrow \quad & v^{w2} \cdot d^2 - (w_x^2 + w_y^2) \cdot (d_x^2 + d_y^2) \geq -(w_x \cdot d_x + w_y \cdot d_y)^2 \\
\Leftrightarrow \quad & (\vec{d} \cdot \vec{w})^2 \geq (w^2 - v^{w2}) \cdot d^2
\end{aligned}
$$

Il y a alors deux cas, selon le signe de $v^{w2} - w^2$.

▷ 1er cas : $w^2 - v^{w2} < 0$ (c'est à dire $w < v^w$: le courant est moins rapide que le robot)

Comme $(\vec{d} \cdot \vec{w})^2 \geq 0$, la condition $(\vec{d} \cdot \vec{w})^2 \geq (w^2 - v^{w2}) \cdot d^2$ est toujours vérifiée, quel que soit le vecteur $\vec{d}$. Autrement dit, le déplacement $\vec{d}$ n'est pas contraint : tous les points du plan sont accessibles par le robot.

▷ 2ème cas : $w^2 - v^{w2} > 0$ (c'est à dire $w > v^w$: le courant est plus rapide que le robot)

Dans ce cas, on a :

$$\vec{d} \cdot \vec{w} \geq \sqrt{(w^2 - v^{w2})} \cdot d$$

Par définition $\cos \widehat{\left\langle \vec{d}, \vec{w} \right\rangle} = \frac{\vec{d} \cdot \vec{w}}{d \cdot w}$. D'où

$$\cos \widehat{\left\langle \vec{d}, \vec{w} \right\rangle} \geq F \tag{4.17}$$

avec $F = \frac{\sqrt{w^2 - v^{w2}}}{w}$.

Si on considère que sans contrainte, $\widehat{\left\langle \vec{d}, \vec{w} \right\rangle} \in [-\pi, \pi]$, nous pouvons faire le raisonnement suivant :

Sur $[0, \pi]$, la fonction cosinus est décroissante. Ainsi, $\cos \widehat{\left\langle \vec{d}, \vec{w} \right\rangle} \geq F \Leftrightarrow \widehat{\left\langle \vec{d}, \vec{w} \right\rangle} \leq \arccos F$.

Sur $[-\pi, 0]$, comme la fonction cosinus est symétrique, nous obtenons $\widehat{\left\langle \vec{d}, \vec{w} \right\rangle} \geq -\arccos F$.

Ainsi, nous avons bien $\widehat{\left\langle \vec{d}, \vec{w} \right\rangle} \in [-\varphi, \varphi]$, avec $\varphi = \arccos F$, comme illustré dans la figure 4.10.

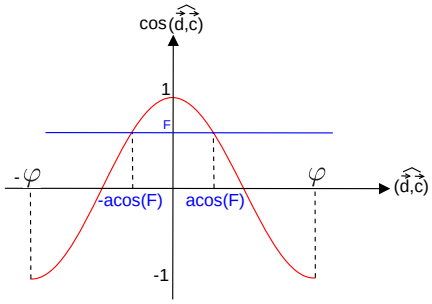

FIG. 4.10 – Illustration de l'équation 4.17

Si à présent $w = v^w$, τ_3 est donné par l'équation 4.15. Comme $\tau_3 \geq 0$, on en déduit $\vec{d} \cdot \vec{w} \geq 0$.

Par définition $\cos \widehat{\langle \vec{d}, \vec{w} \rangle} = \frac{\vec{d} \cdot \vec{w}}{d \cdot w}$. Ainsi nous avons $\cos \widehat{\langle \vec{d}, \vec{w} \rangle} \geq 0$, ce qui est un cas particulier de l'équation 4.17 avec $F = 0$. On en déduit $\widehat{\langle \vec{d}, \vec{w} \rangle} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. □

Grâce à l'équation 4.16, en cas de courants forts (c'est à dire quand $w \geq v^w$), nous pouvons formellement identifier les mouvements $\vec{d}$ valides. En effet, $\vec{d}$ est valide si et seulement si l'angle $\widehat{\langle \vec{d}, \vec{w} \rangle}$ appartient à l'intervalle $\Phi = [-\varphi, \varphi]$.

La fonction τ_3 reflète bien ce fait. En effet, si $\widehat{\langle \vec{d}, \vec{w} \rangle}$ tend vers les bornes de Φ, alors τ_3 tend vers l'infini. Ceci est particulièrement visible dans l'équation 4.15 : $\widehat{\langle \vec{d}, \vec{w} \rangle} \to \pm\pi/2 \Rightarrow \vec{d} \cdot \vec{w} \to 0^+ \Rightarrow \tau_3 \to +\infty$.

Par convention, pour tout angle $\widehat{\langle \vec{d}, \vec{w} \rangle} \notin \Phi$ on fixe $\tau_3 = +\infty$.

Grâce à cette propriété fondamentale, nous avons la garantie que tous les chemins calculés en utilisant τ_3 seront entièrement valides. Nous avons donc résolu les problèmes d'incomplétude mentionnés ci avant. Toutefois, d'autres problèmes apparaissent, dus cette fois à la discrétisation.

En effet, il est possible que le problème ait une solution dans le domaine continu, mais que cette solution ne soit pas représentée dans le domaine discret. Pour illustrer ce propos, considérons que nous nous trouvons en une case C de la grille, en présence d'un unique courant fort $\vec{w}$. En considérant que le voisinage de Moore soit utilisé, C a 8 voisins, notés de V_1 à V_8 dans la figure 4.11. Chaque case est représentée par un point unique, généralement le centre de cette case. Le coût entre deux cases est donc représenté par le coût entre leurs deux centres.

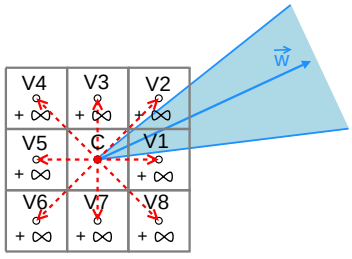

FIG. 4.11 – Problèmes d'incomplétude dus à la discrétisation

Même si des déplacements valides existent dans le domaine continu (secteur bleu), ce n'est pas forcement le cas dans le domaine discret. Ici, aucun des centres des cases voisines à C ne sont atteignables. Le robot est donc bloqué en C.

Dans la figure 4.11, la région accessible depuis le centre de C (le secteur d'accessibilité, en bleu) est non vide. Il y a donc des déplacements possibles depuis C, contenant potentiellement l'unique solution au problème. Seulement dans ce cas précis, les centres des voisins de V_1 à V_8 sont tous en dehors du secteur d'accessibilité. L'algorithme en déduit donc, à juste titre, qu'aucun de ces voisins n'est accessible. Mais cette conclusion, qui ne concerne que les 8 centres des cases, est étendue à l'ensemble de l'espace recouvert par les cases[1]. Ainsi, sur la base de quelques échantillons (mal placés), algorithme aboutit à la conclusion qu'aucun déplacement n'est possible.

Si nous utilisons τ_3 dans l'exemple de la figure 4.12, on peut constater que le front d'onde est "gelé" sans atteindre le point d'arrivée, car il finit par être uniquement composé de cases non atteignables (en rouge). La conclusion, dans le domaine discret, est qu'il n'existe pas de chemin valide entre A et B. Mais cette conclusion est fausse dans le domaine continu, dans lequel il existe une infinité de solutions valides (passant au dessus du tourbillon). La solution représentée en bleu dans la figure 4.18 (page 115) en est un exemple.

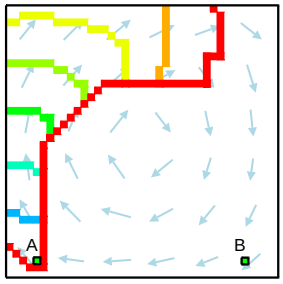

FIG. 4.12 – Résultat obtenu en utilisant la fonction de coût τ_3
Le front d'onde (bleu=coût faible, rouge=coût infini) se bloque avant d'atteindre le point d'arrivée B. Les vitesses des courants et du robot sont respectivement $w = 120$ km/h et $v^w = 100$ km/h.

Pour résoudre ce problème, une première idée consisterait à utiliser de plus petites cases. Mais en réalité, quelle que soit la taille des cases, le nombre de voisins (et donc la précision angulaire) reste le même. Ainsi, si nous reprenons l'exemple de la figure 4.11, le fait d'augmenter la résolution de la grille ne change rien au nombre de directions de propagation : il y en aura toujours 8, espacées de 45°. Et par conséquent, le centre des cases sera toujours en dehors du secteur d'accessibilité.

Une deuxième solution consisterait à augmenter le nombre de voisins. Dans ce domaine, deux grandes techniques ont été envisagées :

1. **Augmenter la portée du voisinage** [DZ94].
   Notons L_i le voisinage de rayon i, exprimée en distance de Manhattan (nombre de déplacements verticaux et horizontaux). En utilisant cette notation, L_1 correspond au voisinage de Manhattan. Il permet 4 directions de propagation (les 4 directions cardinales). L_2 est le voisinage de Moore et en permet 8 directions de propagation (tous les 45°). Ensuite, on arrive à des voisinages moins utilisés : L_3 permet 16 directions (tous les 22,5°), L_4 en permet 32 (tous les 16,25°) et ainsi de suite.

[1] Ceci est normal, puisque c'est le principe même des approches à base de cellules : représenter un espace infini par un nombre fini d'éléments, sur lesquels nous basons ensuite notre raisonnement.

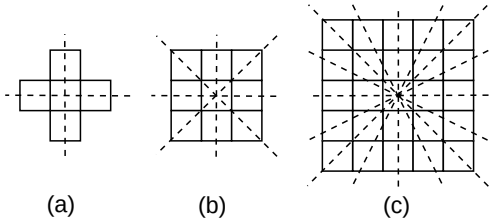

FIG. 4.13 – Influence de la portée du voisinage
La figure présente les voisinages L_i de rayon i (mesuré en distance de Manhattan) et le nombre de directions de propagation D associé. (a) L_1 ($D = 4$) ; (b) L_2 ($D = 8$), (c) L_3 ($D = 8$).

2. **Diviser les bords des cases** [YSSB98].
   Cette approche, plus récente, consiste à graduer les bords des cases à la manière d'une règle. Ici, c'est la résolution de ces graduations (exprimée en nombre de divisions) qui définit le nombre de directions de propagation. Pour une résolution r, jusqu'à $3r + 1$ directions sont utilisables pour propager les coûts.

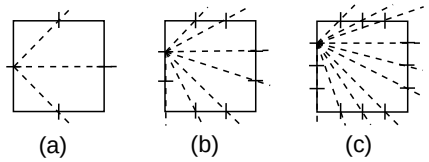

FIG. 4.14 – Influence de la résolution des graduations
La figure présente les voisinages à base de graduations, pour différentes résolutions r (nombre de divisions) et le nombre de directions de propagation D associé. (a) $r = 1$ ($D = 3$) ; (b) $r = 2$ ($D = 7$), (c) $r = 3$ ($D = 10$).

Cette deuxième solution présente deux inconvénients. D'une part, elle provoque une augmentation non négligeable du temps de calcul. Par exemple, passer d'un voisinage L_i à L_{i+1} double le nombre de voisins, et donc le temps de calcul. D'autre part, cette démarche limite les effets du problème sans s'attaquer à sa source : la discrétisation. En effet, c'est le fait même de travailler dans un domaine discret (autrement dit, d'échantillonner l'espace) qu'il faut remettre en question, car c'est la cause profonde des problèmes d'incomplétude.

Nous proposons donc dans la suite de ce chapitre un nouveau type de propagation d'onde, où les coûts ne sont plus propagés dans le domaine discret, mais dans le domaine continu. Cette nouvelle variante, appelée *propagation d'onde coulissante*, est décrite ci-après.

4 La propagation d'onde coulissante

Afin de présenter notre nouvelle variante de la propagation d'onde, nous allons introduire deux concepts : les *Zones Elémentaires de Courant* (ZEC), qui étendent la valeur des échantillons de courant dans des zones homogènes, et les *curseurs*, des points de passage mobiles qui peuvent coulisser sur le bord des ZEC.

4.1 Les Zones Elémentaires de Courant (ZEC)

Classiquement, avant d'appliquer les algorithmes de propagation d'onde, l'environnement est discrétisé à l'aide d'une grille régulière. Toutefois, en présence de courants, cette division n'est plus pertinente, car la répartition des échantillons ne coïncide pas forcement avec la répartition des cases de la grille.

D'une part, les échantillons ne sont pas forcement placés de façon régulière dans l'environnement (comme les cases) et d'autre part, la résolution des échantillons ne correspond pas forcement à celle de la grille.

Dans ces conditions, utiliser une grille régulière peut amener à un sous échantillonnage du courant (si les cases sont trop grossières) ou à un sur-échantillonnage (si les cases sont trop fines). Dans le premier cas, il y a perte d'information (des valeurs de courant sont fusionnées) ; dans le deuxième, il y a redondance (plusieurs cases auront la même valeur).

Pour éviter ces problèmes, il faut que les cellules "s'adaptent" aux échantillons, c'est à dire qu'il y ait exactement une cellule par échantillon, et que la taille de ces cellules s'ajuste selon les distances entre les échantillons. Pour ce faire, nous proposons de construire le diagramme de Voronoï (présenté page 25) entre les échantillons. Ce diagramme est constitué d'un ensemble de segments équidistants des échantillons. Ces segments forment des polygones convexes que nous appelons les Zones Elémentaires de Courant (ZEC), représentées ci-après, pour différentes distributions.

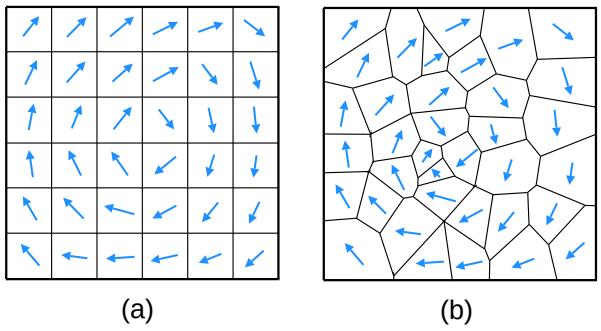

FIG. 4.15 – Les Zones Elémentaires de Courant (ZEC)
La figure présente les ZEC calculées à l'aide d'un diagramme de Voronoï pour deux distributions des échantillons de courants : (a) uniforme; (b) non-uniforme. Notons que le cas uniforme aboutit à une grille régulière.

Dans chaque ZEC, le courant est considéré comme homogène. La valeur du courant n'est autre que la valeur de l'échantillon présent dans la ZEC. En d'autres termes, chaque ZEC étend la valeur d'un échantillon à son voisinage. La taille moyenne du voisinage dépend de la résolution des échantillons.

Il est clair que les ZEC (tout comme les cases précédemment) introduisent des discontinuités dans les courants. Quand nous passons d'une ZEC à une autre, la valeur du courant (tant en intensité qu'en direction) change brutalement, sans phase de transition. Ce choix permet d'obtenir une fonction de coût convexe, comme il sera démontré dans la section suivante. Cette propriété permet

de garantir que la solution retournée par la propagation d'onde coulissante est un minimum global en terme de temps de parcours.

D'autre part, il n'apparaît pas pertinent de calculer de fines transitions entre les échantillons, puisque la distance entre les échantillons correspond à la précision du modèle de prévision. Ainsi, considérer des évolutions de courant en dessous de cette distance n'a pas vraiment de sens. Une autre justification pragmatique des ZEC est la nature même des échantillons : leur aspect prévisionnel implique que valeur inclut déjà une erreur. Si les échantillons sont très rapprochés, l'impact de cette erreur peut être plus significatif sur la qualité de la solution que l'approximation apportée par les ZEC.

4.2 Les curseurs

Dans le contexte des interfaces graphiques, un curseur est constitué d'un repère qui peut être translaté sur une règle. L'utilisateur peut faire coulisser ce repère de façon continue, jusqu'aux extrémités de la règle, comme illustré dans la figure 4.16a.

Nous allons utiliser le même principe dans notre approche. Dans la propagation d'onde coulissante, un curseur est un point de passage mobile, reliant deux ZEC adjacentes ZEC_i et ZEC_j. Ce point matérialise la transition entre ZEC_i et ZEC_j. C'est par ce point que le robot sortira de ZEC_i, et entrera dans ZEC_j. Tout comme le composant graphique, le curseur peut coulisser sur son support (ici le bord commun entre ZEC_i et ZEC_j) jusqu'à ses extrémités, comme illustré dans la figure 4.16b.

D'un point de vue plus formel, un curseur est noté C_i, avec $i \in [1, N]$, où N est le nombre total de bords des ZEC (autrement dit, le nombre de segments contenus dans le diagramme de Voronoï). A chaque curseur C_i sont associées les propriétés suivantes :

- Un bord B_i, d'équation $a_i \cdot x + a_i \cdot y + c_i = 0$, modélisant la règle ;
- Un couple $Z_i = \{ZEC_1, ZEC_2\}$, contenant les deux ZEC ayant pour bord commun B_i.
- Deux points $B_i^- = (x_i^-, y_i^-)$ and $B_i^+ = (x_i^+, y_i^+)$, modélisant les extrémités de la règle ;
- Un point de passage $P_i = (x_i, y_i)$, modélisant le repère ;
- Une variable l_i, l'abscisse curviligne sur B_i, modélisant la position du repère ;
- Un domaine D_i pour l_i : $D_i = [0, l_i^+]$. La valeur 0 correspond à l'extrémité inférieure B_i^- et l_i^+ à l'extrémité supérieure B_i^+.

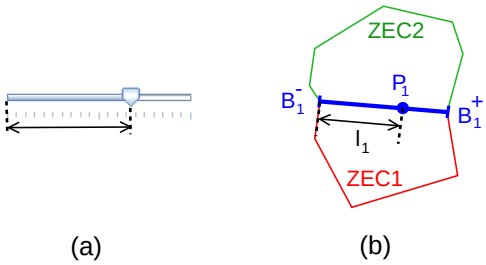

Fig. 4.16 – Les curseurs
(a) le composant graphique ; (b) le point de passage mobile utilisé dans la propagation d'onde coulissante.

Il est important de comprendre que les variables x_i et y_i, matérialisant la position absolue de P_i dans l'espace $\mathcal{C}$, sont en réalité liées par l'équation de droite du bord B_i. Ce lien modélise le fait que le curseur n'a qu'un seul degré de liberté : une translation suivant le bord. Pour cette raison, nous pouvons modéliser la position de P_i par une unique variable, l_i, représentant le déplacement *local* du repère sur sa règle (en terme de longueur parcourue). l_i peut varier de 0 (repère en butée inférieure) à l_i^+ (repère en butée inférieure), l_i^+ représentant la longueur de la règle.

x_i et y_i sont facilement substituables par l_i en utilisant les équations suivantes :

$$\begin{cases} x_i = x_i^- \pm \sqrt{l_i^2/(1+(a_i/b_i)^2)} \\ y_i = y_i^- + (-a_i \cdot x_i - c_i)/b_i \end{cases} \tag{4.18}$$

Le signe $\pm$ de la première ligne dépend du signe de la différence $x_i^+ - x_i^-$: $+$ si la différence est positive, $-$ sinon.

Preuve :

La distance l_i est définie par :

$$l_i^2 = (x_i - x_i^-)^2 + (y_i - y_i^-)^2 \tag{4.19}$$

De plus, d'après l'équation du bord B_i, nous savons que tout point de ce bord (le point de passage P_i mais aussi ses extrémités) vérifient : $a_i \cdot x + a_i \cdot y + c_i = 0$. L'équation précédente devient :

$$l_i^2 = (x_i - x_i^-)^2 + \left(\frac{-a_i \cdot x_i^- - c_i}{b_i} - \frac{-a_i \cdot x_i - c_i}{b_i}\right)^2 = \left(1 + (a_i/b_i)^2\right) \cdot (x_i - x_i^-)^2$$

Il y a alors deux cas, selon le signe de $x_i - x_i^-$. Or, le signe de $x_i - x_i^-$ est le même que celui de $x_i^+ - x_i^-$, car P_i est entre B_i^- et B_i^+. En effet :

$$P_i \in [B_i^-, B_i^+] \Leftrightarrow \exists k \in [0,1] \ / \ \overrightarrow{B_i^- P_i} = k \cdot \overrightarrow{B_i^- B_i^+}$$

En projetant cette propriété sur l'axe $\vec{x}$ on a :

$$(x_i^- - x_i) = k \cdot (x_i^- - x_i^+)$$

Ainsi l'expression de l_i dépend du signe de $x_i^+ - x_i^-$.

$\triangleright$ 1er cas : $x_i^+ - x_i^- > 0$.

On a alors $x_i - x_i^- > 0$. L'équation 4.19 devient :

$$l_i = \sqrt{l_i^2/(1+(a_i/b_i)^2)} \cdot (x_i - x_i^-)$$

On en déduit :

$$x_i = x_i^- + \sqrt{l_i^2/(1+(a_i/b_i)^2)}$$

▷ 2ème cas : $x_i^+ - x_i^- < 0$

On a alors $x_i - x_i^- < 0$. L'équation 4.19 devient :

$$l_i = \sqrt{l_i^2/(1+(a_i/b_i)^2)} \cdot (x_i^- - x_i)$$

On en déduit :

$$x_i = x_i^- - \sqrt{l_i^2/(1+(a_i/b_i)^2)} \qquad \square$$

Notons enfin que les points de départ et d'arrivée sont deux points de passage particuliers, qui n'ont eux aucun degré de liberté. D'une manière plus formelle, le domaine D caractérisant leur règle est le singleton $\{0\}$ (les points peuvent se déplacer d'une longueur 0). Par convention, on notera $V_0 = A$.

4.3 Le nouveau processus de propagation

Dans la propagation d'onde coulissante, les coûts ne sont plus propagés au sein de cases, mais au sein de curseurs. Autrement dit, le curseur devient l'élément de base du front d'onde.

La principale différence entre les cases et les curseurs réside dans la notion de mobilité. Dans le cas des cases, les points de passage sont statiques, c'est à dire fixés à l'avance, généralement au centre des cases. L'utilisation des curseurs introduit un degré de liberté : les points de passage peuvent à présent coulisser sur le bord des ZEC.

Cette mobilité permet de résoudre les problèmes d'incomplétude mentionnés dans la section précédente. En effet, à partir d'un point de passage P_i, l'ensemble des directions de propagation est maintenant utilisable. Il n'y a plus un choix restreint de directions comme précédemment. En particulier, les directions de propagation peuvent être arbitrairement proches des bords du secteur d'accessibilité, ce qui permet de ne pas "perdre" des solutions qui seraient à la limite de l'invalidité.

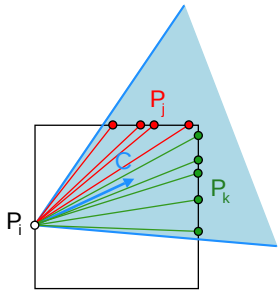

FIG. 4.17 – Modèle continu de propagation des coûts
A partir d'un point de passage P_i, l'ensemble des directions appartenant au secteur d'accessibilité (en bleu) sont utilisables, car ses successeurs potentiels P_j et P_k peuvent se déplacer continûment sur leurs bords respectifs.

La figure 4.18 illustre cette capacité : grâce à la notion de curseur, la propagation d'onde trouve une solution dans d'environnement de la figure 4.1, là où la propagation d'onde classique échouait.

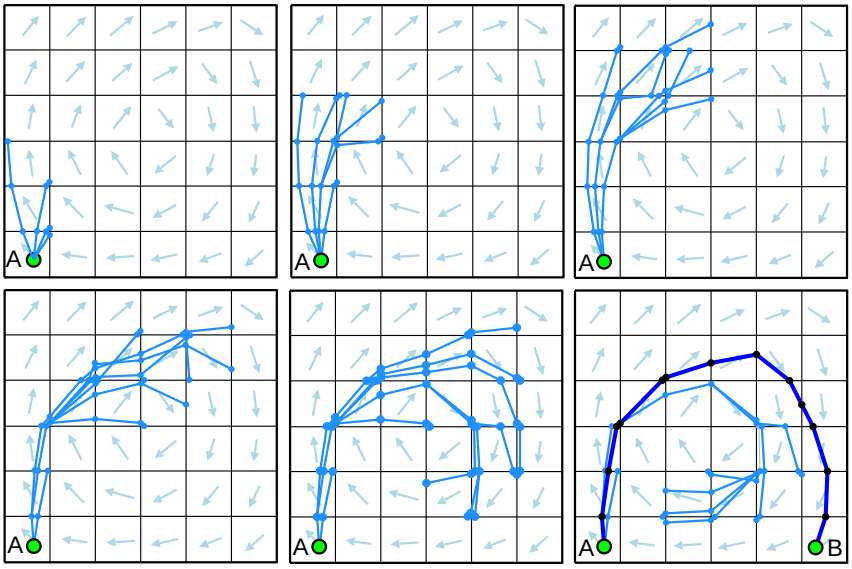

FIG. 4.18 – La propagation d'onde coulissante
Différents stades de la propagation d'onde coulissante, dans les ZEC de la figure 4.15a. Les chemins potentiels sont illustrés en bleu clair, et le chemin final en bleu foncé.

4.4 Le nouveau processus d'évaluation des coûts

Comme on peut le voir dans la figure 4.18 (en bleu clair), un certain nombre de branches optimales sont construites au fur et à mesure de la propagation. Ces branches correspondent à des trajectoires candidates entre A et B.

A chaque itération du processus de propagation, le coût de chaque branche est minimisé en calculant la position optimale des points de passage qu'elle traverse.

Considérons par exemple qu'une branche se termine par le point de passage P_i, dont les prédécesseurs sont P_j ($j \in [0, i-1]$). Le coût associé à cette branche est :

$$T_i = \min_{l_j} \tau_3^i = \min_{l_j} \sum_{j=0}^{i-1} \tau_3^{j,j+1} \tag{4.20}$$

où τ_3^i est le temps de parcours global pour atteindre P_i à partir de A, et $\tau_3^{j,j+1}$ est le temps de parcours local entre deux points de passage successifs P_j et P_{j+1}.

L'expression de $\tau_3^{j,j+1}$ est obtenue en utilisant les équations 4.14 et 4.15. Dans ces équations, nous remplaçons d_x par $x_{j+1} - x_j$, d_y par $y_{j+1} - y_j$ et w par le courant régnant dans la ZEC commune entre C_j and C_{j+1} (donnée par $Z_j \cap Z_{j+1}$). Ensuite, nous substituons les variables x_j et y_j par les variables l_j, en utilisant l'équation 4.18.

En résumé, le coût T_i associé à P_i est calculé en minimisant une fonction à i variables. Ces variables sont $l_1, l_2, ..., l_i$, c'est à dire la position locale de chaque point de passage P_j ($j \in [1, i]$) sur leur support.

Il est démontré ci-après que les fonction locales $\tau_3^{j,j+1}$ sont convexes. τ_3^i étant la somme de fonctions convexes, elle est donc également convexe. Ainsi, nous avons la garantie que le minimum représenté par T_i est global.

PREUVE : Chaque fonction de coût locale $\tau_3^{j,j+1}$ est convexe.

Il y a deux cas possibles, selon les valeurs respectives de v^w et w :

▷ 1er cas : $v^w \neq w$.

$\tau_3^{j,j+1}$ est donnée par l'équation 4.14 :

$$\tau_3^{j,j+1} = \frac{-(w_x \cdot d_x + w_y \cdot d_y) + \sqrt{\Delta}}{v^{w2} - w^2} = \frac{\sqrt{\Delta} - \vec{d} \cdot \vec{w}}{v^{w2} - w^2}$$

avec $d_x = x_{j+1} - x_j$, $d_y = y_{j+1} - y_j$ et $\Delta = v^{w2} \cdot (d_x^2 + d_y^2) - (w_x \cdot d_y - w_y \cdot d_x)^2$.

La fonction $\tau_3^{j,j+1}$ est convexe si et seulement si sa matrice Hessienne H est semi-définie positive, c'est à dire si et seulement si les valeurs propres de H sont positives.

Calculons tout d'abord la matrice Hessienne H. Cette matrice contient toutes les dérivées secondes de $\tau_3^{j,j+1}$, disposées comme suit :

$$H = \begin{pmatrix} \frac{\partial^2 \tau_3^{j,j+1}}{\partial d_x^2} & \frac{\partial^2 \tau_3^{j,j+1}}{\partial d_x \partial d_y} \\ \frac{\partial^2 \tau_3^{j,j+1}}{\partial d_y \partial d_x} & \frac{\partial^2 \tau_3^{j,j+1}}{\partial d_y^2} \end{pmatrix}$$

Calculons donc toutes les dérivées secondes de $\tau_3^{j,j+1}$. Nous avons tout d'abord :

$$\frac{\partial \tau_3^{j,j+1}}{\partial d_x} = \frac{1}{v^{w2} - w^2} \cdot \left(-w_x + \frac{v^{w2} \cdot d_x + w_y \cdot (w_x \cdot d_y + w_y \cdot d_x)}{\sqrt{\Delta}} \right)$$

Il vient :

$$\frac{\partial^2 \tau_3^{j,j+1}}{\partial d_x^2} = \frac{1}{v^{w2} - w^2} \cdot \left(\frac{v^{w2} - w_y^2}{\sqrt{\Delta}} - \frac{\left(v^{w2} \cdot d_x + w_y \cdot (w_x \cdot d_y - w_y \cdot d_x)\right)^2}{2 \cdot \sqrt{\Delta}^3} \right) = \frac{v^{w2} \cdot d_y^2}{\sqrt{\Delta}^3}$$

De la même manière, on obtient :

$$\frac{\partial^2 \tau_3^{j,j+1}}{\partial d_y^2} = \frac{v^{w2} \cdot d_x^2}{\sqrt{\Delta}^3}$$

Et :

$$\frac{\partial^2 \tau_3^{j,j+1}}{\partial d_y \partial d_x} = \frac{\partial^2 \tau_3^{j,j+1}}{\partial d_x \partial d_y} = \frac{-v^{w2} \cdot d_x \cdot d_y}{\sqrt{\Delta}^3}$$

La matrice Hessienne H de $\tau_3^{j,j+1}$ est donc égale à :

$$H = \frac{v^{w2}}{\sqrt{\Delta}^3} \cdot \begin{pmatrix} d_y^2 & -d_x \cdot d_y \\ -d_x \cdot d_y & d_x^2 \end{pmatrix}$$

Les valeurs propres de H sont les racines du polynôme caractéristique défini par :

$$P(\lambda) = \det(\lambda \cdot Id - H) = \frac{v^{w2}}{\sqrt{\Delta}^3} \cdot \begin{vmatrix} \lambda - d_y^2 & d_x \cdot d_y \\ d_x \cdot d_y & \lambda - d_x^2 \end{vmatrix}$$

où Id désigne la matrice identité et $|\cdot|$ le déterminant.

Il vient :

$$P(\lambda) = \frac{v^{w2}}{\sqrt{\Delta}^3} \cdot [(\lambda - d_y^2)(\lambda - d_x^2) - d_x^2 \cdot d_y^2] = \frac{v^{w2}}{\sqrt{\Delta}^3} \cdot \lambda \cdot (\lambda - d^2)$$

Comme $\frac{v^{w2}}{\sqrt{\Delta}^3} > 0$, ce polynôme possède les deux racines suivantes :

$$\lambda_1 = 0 \qquad \text{et} \qquad \lambda_2 = d^2$$

Ces deux racines sont toutes deux positives, donc la fonction $\tau_3^{j,j+1}$ est convexe.

$\triangleright$ 2ème cas : $v^w = w$.

$\tau_3^{j,j+1}$ est donnée par l'équation 4.15 :

$$\tau_3^{j,j+1} = \frac{d^2}{2\vec{d} \cdot \vec{w}}$$

Le calcul des dérivées partielles de $\tau_3^{j,j+1}$ aboutissent à :

$$H = \frac{w^2}{(\vec{d} \cdot \vec{w})^3} \cdot \begin{pmatrix} d_y^2 & -d_x \cdot d_y \\ -d_x \cdot d_y & d_x^2 \end{pmatrix}$$

Ce qui est en fait un cas particulier du Hessien précédemment calculé. En effet, la matrice des dérivées partielles est strictement la même, seul le facteur devant cette matrice est potentiellement différent. Dans le cas 1, nous avions un facteur $v^{w2}/\sqrt{\Delta}^3$ devant la matrice. Or, $v^w = w$, aboutit à $v^{w2} = w^2$ et à $\Delta = (\vec{d} \cdot \vec{w})^2$. En substituant ces valeurs, on obtient bien le facteur $w^2/(\vec{d} \cdot \vec{w})^3$.

La conclusion du cas 1 est donc valable pour le cas 2 : la fonction $\tau_3^{j,j+1}$ est convexe. Ceci est normal, car le cas 2 correspond à la limite du cas 1 quand $v^w \to w$.$\square$

Ainsi, la branche $A \to P_1 \to P_2 \to ... \to P_{i-1} \to P_i$ donne le trajectoire de temps minimal pour atteindre le point de passage P_i. C'est en particulier le cas pour le point B.

Compte tenu de cette propriété, si le front d'onde atteint le point B, nous pouvons affirmer que la branche reliant A et B minimise, de façon globale, le temps de parcours entre ces deux points, ce qui répond à notre problème initial.

Pour réaliser la minimisation de l'équation 4.20, de nombreuses techniques sont applicables, selon les préférences de l'utilisateur, ou des outils disponibles. Pour notre part, comme les seules contraintes sur les variables l_j sont des contraintes de bornes, nous avons utilisé une technique appelée *gradient projeté* [CM87]. Cette technique est analogue à la descente du gradient, mais maintient en permanence les variables dans le domaine valide par un processus de projection. Ce processus est illustré dans la figure 4.19.

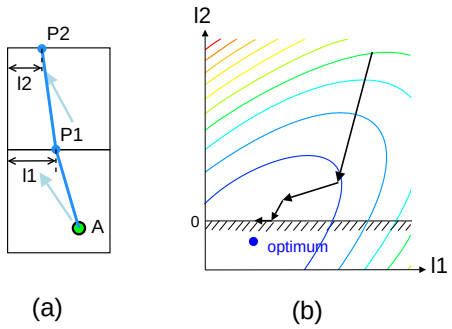

FIG. 4.19 – Algorithme du gradient projeté
(a) zoom autour du point de départ, dans la figure 4.18. Le coût T_2 de la branche $A \rightarrow P_1 \rightarrow P_2$ dépend des variables l_1 et l_2 ; (b) optimisation de T_2 par application du gradient projeté, sur la surface de τ_3^2. Lors de la descente vers l'optimum, les mouvements sont maintenus dans le domaine valide (au dessus des hachures) par un mécanisme de projection.

En général, la descente du gradient converge vers l'optimum, sans forcement l'atteindre. Formellement parlant, dans l'équation 4.20, l'atteinte de l'optimum est matérialisée par la condition $\vec{\nabla}\tau_3^i = \vec{0}$, où $\vec{\nabla}$ représente l'opérateur gradient. Mais la seule certitude que nous ayons est que, au fur et à mesure de la descente du gradient, nous avons $\vec{\nabla}\tau_3^i \rightarrow \vec{0}$.

Cette convergence pouvant être très lente dans certains cas (notamment si la surface de la fonction est très "plate"), celle-ci est souvent stoppée en pratique. C'est ce que nous avons fait dans nos tests, en limitant le nombre d'itérations de la descente du gradient à κ. Beaucoup d'autres conditions sont utilisables, comme par exemple $\nabla\tau_3^i < \varepsilon$.

4.5 L'algorithme

4.5.1 Description

Tout comme la propagation d'onde classique, la propagation d'onde coulissante peut être vue comme une variante de l'algorithme de plus court chemin de Dijkstra [Dij59]. Au départ, le front d'onde contient uniquement le point de départ A. Ensuite, les points du front d'onde P_i sont développés par ordre croissant de coût. Développer un point consiste à extraire P_i du front d'onde, évaluer tous ses voisins P_j et ajouter ceux-ci dans le front d'onde, si nécessaire.

L'algorithme de la propagation d'onde coulissante est fourni ci-après :

PROPAGATION_ONDE_COULISSANTE(A, B, P, κ)
▷ **Entrée :** A,B : points de départ et d'arrivée
▷ **Entrée :** P : liste des N points de passage
▷ **Entrée :** κ : nombre maximal d'itérations dans la descente du gradient
▷ **Locale :** F : front d'onde courant
▷ **Locale :** D : points développés par le front d'onde
▷ **Locale :** P_i : point de passage, de fonction de coût τ_3^i, de valeur minimale T_i

```
Début
    F ← {A}
    faire
        P_i ← arg min{T_k, P_k ∈ F}
        pour chaque P_j ∈ V(P_i, P) \ D faire
            k ← 0
            faire
                PAS_DESCENTE_GRADIENT(P_j)
                MEILLEUR_PREDECESSEUR(P_j)
                k ← k + 1
            tantque k ≤ κ
            AJOUTER_PREDECESSEUR(P_j, P_i)
            F ← F ∪ P_j
        F ← F \ {P_i}
        D ← D ∪ {P_i}
    tantque P_i ≠ B
Fin
```

La fonction $\mathcal{V}(P_i, P)$ renvoie les voisins du point de passage P_i, parmi les N points de passage de l'environnement. Nous rappelons qu'un point de passage est associé à chaque bord des ZEC. Si P_i fait le lien entre deux ZEC notées ZEC_i et ZEC_j et que ZEC_i a déjà été visitée, alors les voisins de P_i sont tous les points de passage P_j associés à tous les bords de ZEC_j (en excluant le bord sur lequel se trouve P_i). Ceci est illustré dans la figure 4.20.

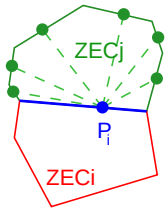

FIG. 4.20 – Voisinage d'un point de passage P_i

Ensuite, la fonction PAS_DESCENTE_GRADIENT effectue un pas de la descente du gradient dans la fonction de coût τ_3^j associée à P_j. Un pas correspond à un déplacement sur la surface de τ_3^j selon la direction $-\vec{\nabla}\tau_3^j$ (illustré par les flèches noires dans la figure 4.19).

Enfin, les fonctions MEILLEUR_PREDECESSEUR et AJOUTER_PREDECESSEUR permettent d'identifier le prédécesseur optimal d'un point P_j.

Ces fonctions sont nécessaires car le prédécesseur optimal d'un point P_j peut varier au cours de la propagation, ce qui n'était pas le cas dans la propagation d'onde classique. Ceci est du à la capacité des points de passage à coulisser sur leur support. Selon leur position sur ce bord, leur prédécesseur optimal peut changer. Ceci est illustré dans la figure 4.21a.

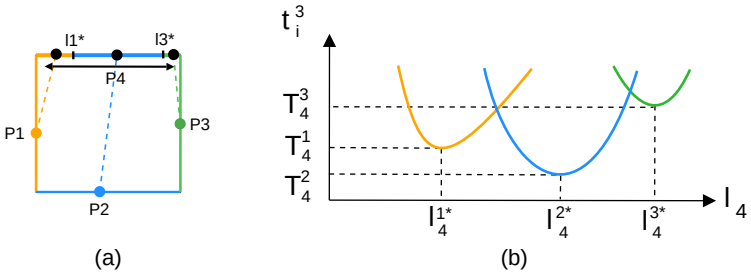

FIG. 4.21 – Variation du prédécesseur optimal
Selon la position l_4 du point P_4, le prédécesseur optimal de P_4 change : il s'agit de P_1 si $l_4 \leq l_4^{1*}$, P_2 si $l_4^{1*} < l_4 < l_4^{3*}$, et P_3 si $l_4 \leq l_4^{3*}$.

Ces fonctions fonctionnent en deux temps :

1. Quand un voisin P_j du point P_i est évalué, P_j est ajouté à la liste des prédécesseurs potentiels de P_j, par l'intermédiaire de la fonction AJOUTER_PREDECESSEUR.

2. Quand P_j sera lui-même développé, le meilleur prédécesseur sera choisi dans la liste des prédécesseurs potentiels par la fonction MEILLEUR_PREDECESSEUR. Cette fonction procède ainsi : Les différents prédécesseurs P_k possibles pour atteindre P_j aboutissent à des fonctions de coût différentes, dont le minimum est connu et appelé T_j^k. La fonction MEILLEUR_PREDECESSEUR sélectionne le prédécesseur P_k tel que $l_j \in [l_j^{k-1*}, l_j^{k+1*}]$ et T_j^k est minimal, comme il est montré dans la figure 4.21b.

Le lecteur intéressé trouvera la description détaillée de ces fonctions dans l'annexe 2 de cette thèse.

4.5.2 Complexité temporelle dans le pire des cas

Evaluons tout d'abord le temps de calcul nécessaire à une itération i de l'algorithme :

- Ligne 1 (Récupération de la tête P_i du front d'onde) : $O(1)$ (temps constant) si une structure de tas binaire (ou équivalent) est utilisée.
- Ligne 2 (Suppression de P_i du front d'onde) : $O(1)$.
- Ligne 3 (Récupération des voisins P_j de P_i) : $O(\rho_i)$, où ρ_i est le nombre de voisins de P_i.
- Lignes 8 à 11 (Evaluation d'un voisin) : $O(\kappa)$.
- Ligne 12 (Ajout de P_j à la liste des prédécesseurs de P_j) : $O(\rho_i)$
- Ligne 13 (Ajout de P_j dans le front d'onde) : $O(\log n_i)$, pour maintenir la structure de tas, où n_i est la taille du front d'onde à l'itération i.

Ainsi, l'ensemble de l'algorithme requiert un temps en $O\left(\sum_{i=1}^{N} \rho_i \log n_i\right)$.

La quantité ρ_i est bornée. Elle est inférieure à ρ_{max}, le nombre maximal d'échantillons autour d'un échantillon donné. ρ_{max} dépend de la répartition des échantillons. Par exemple, $\rho_{max} = 4$ dans la figure 4.15a et $\rho_{max} = 6$ dans la figure 4.15b.

De plus, quelle que soit l'itération i, on a $n_i < N$, où N est le nombre total de bords des ZEC (autrement dit, le nombre de segments contenus dans le diagramme de Voronoï)

On obtient finalement une complexité temporelle dans le pire des cas en $O(N \log N)$. Ainsi, notre approche possède une complexité équivalente à la propagation d'onde classique.

5 Ajout d'obstacles fixes dans l'environnement

Supposons à présent que l'environnement contient un ensemble d'obstacles fixes polygonaux. Nous allons voir dans cette section les modifications nécessaires sur notre approche pour les éviter.

Une première idée pour éviter ces obstacles est simplement de jouer sur la borne B_i associée à chaque curseur C_i, en interdisant les parties recouvertes par les obstacles (parties rouges sur la figure 4.22). Ce procédé permet d'imposer aux points de passage P_i d'être en dehors des obstacles. Mais, comme nous l'avons vu pour les B-splines, ceci ne suffit pas à garantir un chemin sans collisions.

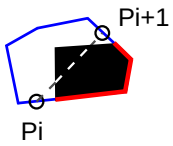

FIG. 4.22 – Problème posé par un obstacle dans une ZEC
P_i et P_{i+1} sont tous deux dans l'espace valide (en dehors des parties rouges) mais ce n'est pas le cas de l'ensemble du segment $[P_i, P_{i+1}]$, qui intersecte l'obstacle fixe (en noir).

Pour éviter ce problème, nous pouvons compléter cette idée par le post-traitement suivant sur les ZEC :

1. Calculer une décomposition cellulaire de l'environnement (fig. 4.23a), relative aux obstacles fixes, en utilisant l'une des techniques décrites page 24. Cette décomposition aboutit à un ensemble Λ de cellules, dont la taille et la position dépendent de la position des sommets des obstacles. Dans le cas de la décomposition triangulaire, par exemple, Λ est un ensemble de triangles, comme illustré dans la figure 4.23a.

2. Si un sommet s appartient à une zone de courant ZEC_i, diviser ZEC_i à l'aide du sous-ensemble de cellules Λ_s de Λ partant de s (fig. 4.23c). Concrètement, on effectue l'intersection de Λ_s avec ZEC_i pour former des sous-zones de courants. Toutes ces sous-zones de courants ont la même valeur de courants, ce qui introduit une petite redondance d'information.

En procédant de la sorte, on obtient des cellules convexes sans obstacle (fig. 4.23d). Ainsi, tout segment reliant deux bords de ces cellules est garanti d'être sans collision. Notons que le reste de l'algorithme reste inchangé.

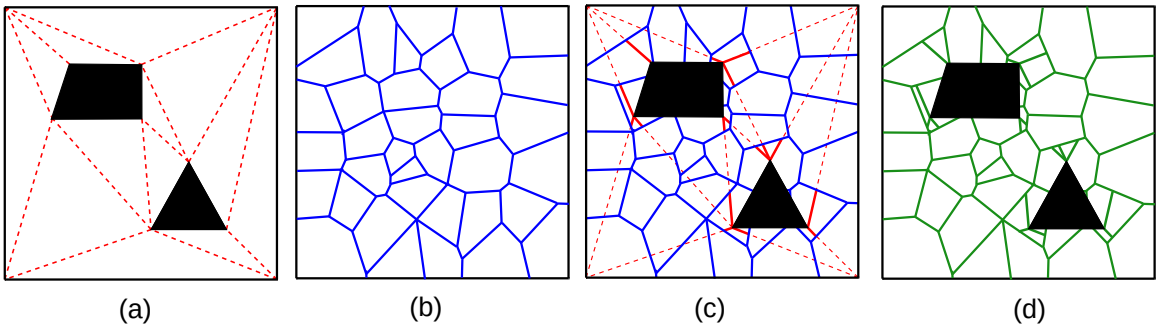

FIG. 4.23 – Post-traitement sur les ZEC en cas d'obstacles fixes
(a) Décomposition cellulaire (triangulaire) relative aux obstacles noirs ; (b) ZECs de la figure 4.15b ; (c) Intersection de (a) et (b) sur les cellules contenant des sommets ; (d) résultat obtenu : ZECs sans obstacle.

6 Résultats expérimentaux

Le but de cette section est d'évaluer le gain de fiabilité (en termes de succès de planification) qu'apporte la propagation d'onde coulissante par rapport à la propagation d'onde classique, avec différentes fonctions de coût. Pour cela, nous avons suivi le protocole expérimental suivant :

1. Nous avons collecté, de façon quotidienne et pendant plus de trois mois successifs, des cartes des vents fournies sur le site de Météo France [MF], afin de constituer une base de 100 environnements réalistes contenant des courants. Chaque carte contient environ 1000 échantillons de courant (dimensions 28×35), comme on peut le voir dans la figure 4.24

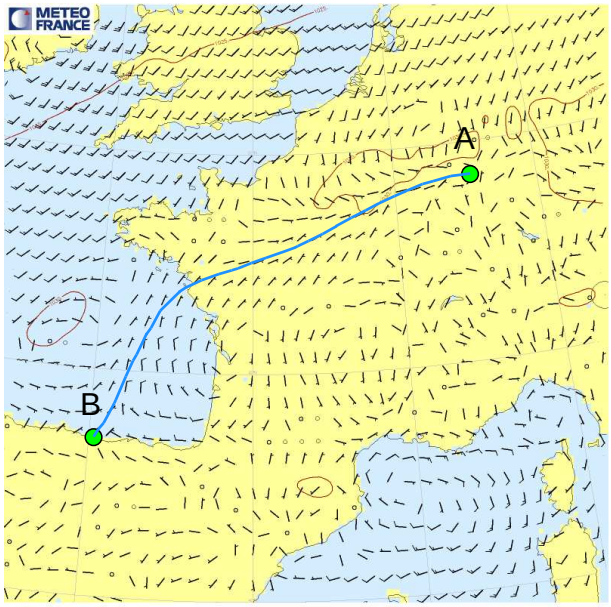

FIG. 4.24 – Un cas de test et le chemin obtenu (en bleu) composé de 36 segments de droite

2. Nous avons défini la notion d'intensité du courant I comme étant :

$$I = \max\{w\}/v^w \tag{4.21}$$

Dans nos tests, nous avons volontairement choisi $I > 1$ pour provoquer les problèmes d'incorrection et d'incomplétude évoqués dans la section 3.

Notons que pour de telles intensités, il existe naturellement des situations où aucune solution n'existe. Par exemple, un environnement où le courant est globalement de droite à gauche, et le mouvement à effectuer pour relier A et B est de gauche à droite. Selon l'équation 4.16, pour $I = 1$, seulement un demi-plan de l'environnement est accessible au robot à chaque mouvement. Ainsi, le robot a globalement une probabilité de 50% d'effectuer un mouvement invalide. Dans ces conditions, environ 50% des cas de test seront impossibles. Ce pourcentage augmente avec I. C'est ce que l'on peut observer dans la figure 4.25.

3. Pour chaque valeur de I, nous avons généré 500 cas de tests. Un cas de test est construit de la façon suivante : (1) tirage aléatoire d'une carte des vents dans la base et (2) placement aléatoire de A et B sur cette carte.

4. Nous avons appliqué les algorithmes suivants à chaque cas de test :
   - (Cla_1) : propagation d'onde classique, fonction de coût τ_1 ;
   - (Cla_2) : propagation d'onde classique, fonction de coût τ_2 ;
   - (Cla_3) : propagation d'onde classique, fonction de coût τ_3 ;
   - (Cou) : propagation d'onde coulissante, fonction de coût τ_3 ($\kappa = 100$).

5. Enfin, nous avons mesuré le taux de succès de planification S, c'est à dire le nombre de planifications réussies sur les 500 cas de tests. Ce taux est représenté dans la figure 4.25.
   Notons qu'une planification a été considérée comme *réussie* si l'algorithme a été capable de fournir un chemin valide pour le cas de test considéré. Par opposition, un échec de planification est caractérisé par la fourniture d'un chemin invalide, ou d'aucun chemin.

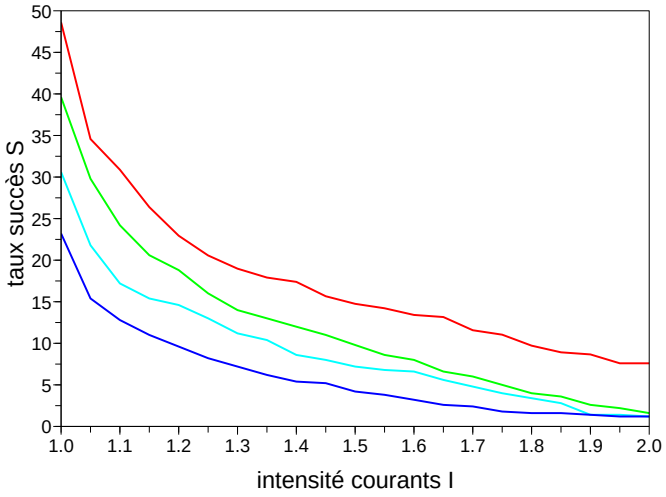

FIG. 4.25 – Taux de succès des algorithmes de propagation d'onde en présence de courant forts
Taux de succès S en fonction de l'intensité I des courants. De bas en haut : Cla_1 (bleu foncé), Cla_2 (bleu clair), Cla_3 (vert) ; Cou (rouge).

Le taux de succès S est représenté dans la figure 4.25 pour les différents algorithmes. L'observation de cette figure amène aux commentaires suivants :

Tout d'abord, les moins bons résultats ont été obtenus par l'algorithme (Cla_1), c'est à dire la propagation d'onde classique associée à la fonction de coût τ_1. Comme attendu, cet algorithme souffre de très importants problèmes d'incorrection : plus de 75% des chemins calculés par (Cla_1) se sont révélés être invalides, même pour les plus faibles intensités de nos tests.

Ensuite, toujours sur la base de la propagation d'onde classique, les résultats s'améliorent avec la qualité de la fonction de coût.

Comme expliqué précédemment, la fonction de coût τ_2 pénalise plus fortement les mouvements impossibles que la fonction de coût τ_1 (un facteur 5 au lieu d'un facteur 2 dans l'exemple de la figure 4.2). Dans ces conditions, il paraît naturel que la proportion de chemins valides fournis par (Cla_2), et donc le taux de succès S de (Cla_2), soit supérieur à celui de (Cla_1). C'est effectivement ce que l'on constate sur la figure. La fonction de coût τ_3 est valide, c'est à dire qu'elle interdit catégoriquement les mouvements physiquement irréalisables (ce qui peut être interprété comme une pénalisation infinie). Ceci permet d'encore améliorer le taux de succès.

Enfin, la propagation d'onde coulissante (Cou) s'appuie sur les bonnes propriétés de τ_3, en y ajoutant la notion de support de propagation continue, basée sur les curseurs. Les curseurs permettent d'accéder à n'importe quel point du secteur d'accessibilité, et ainsi de ne pas perdre de solutions. Dès lors, on obtient le meilleur taux de succès S avec cet algorithme. L'amélioration dans les valeurs de S entre propagation d'onde classique (avec la meilleure fonction de coût) et la propagation d'onde coulissante est de l'ordre de 10%.

Toutefois, le prix à payer est le temps de calcul. Nous avons vu que les complexités temporelles étaient équivalentes, ce qui signifie que les temps de calculs sont les mêmes à un facteur multiplicatif près. Ce facteur dépend principalement de ρ_{max} (la taille du voisinage) et de κ (le nombre maximal d'itérations). Dans nos tests, $\rho_{max} = 4$ (distribution régulière des échantillons) et $\kappa = 100$ ont abouti à un facteur d'environ 5.

Une étude plus approfondie de l'influence de ces deux grandeurs serait nécessaire, en particulier l'influence de κ, car seul κ est réellement un paramètre de l'algorithme (ρ_{max} dépend des données du problème, sur lesquelles nous ne pouvons pas forcement agir). Il serait notamment intéressant d'étudier, de façon théorique, l'impact de κ sur la qualité de la solution.

7 Conclusion

Nous avons introduit dans ce chapitre une nouvelle variante de la propagation d'onde, appelée *propagation d'onde coulissante*, qui conserve tous les avantages de la propagations d'onde (optimalité globale de la solution, temps de calcul en $N \log N$, intégration aisée des obstacles fixes), sans souffrir des problèmes d'incorrection ou d'incomplétude pouvant apparaître quand les courants deviennent forts, c'est à dire plus rapides que le robot.

La propagation d'onde coulissante permet donc un saut substantiel dans la fiabilité de la planification de trajectoire (en termes de succès de planification) en présence de courants forts, tout en conservant un temps de calcul raisonnable. En effet, au cours de nos tests (réalisés dans des

environnements réalistes), le taux de succès obtenu avec la propagation d'onde coulissante est d'environ 10% supérieur à celui obtenu avec les meilleures instances connues de la propagation d'onde classique. Le temps de calcul est lui multiplié par un facteur constant, d'environ 5 sur nos jeux de test.

L'ensemble de ces résultats a été publié dans les actes de la conférence ICRA'08 (International Conference on Robotics and Automation). Le lecteur interessé pourra consulter [STR08a].

Dans la continuité de ces travaux, il serait intéressant d'adapter la propagation d'onde coulissante afin que le robot évite également des obstacles mobiles dans l'environnement. Une piste serait de directement appliquer l'algorithme dans un espace-temps 3D $(\vec{x}, \vec{y}, \vec{t})$.

Dans un tel espace-temps, chaque obstacle mobile génère un volume, défini par la forme et le mouvement de l'obstacle. Par exemple, si l'obstacle effectue un mouvement rectiligne uniforme, alors le volume engendré sera un cylindre, dont la base sera de la forme de cet obstacle, comme illustré dans la figure 4.26.

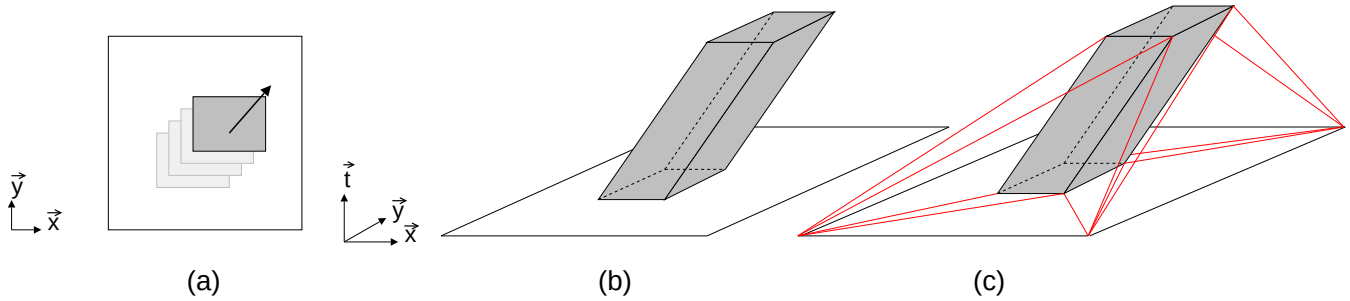

FIG. 4.26 – Représentation d'un obstacle mobile dans un espace-temps 3D
(a) mouvement (rectiligne uniforme) de l'obstacle mobile dans l'espace initial ; (b) volume résultant dans l'espace-temps (parallélépipède) ; (c) décomposition pyramidale de l'espace-temps (en rouge) relative à ce volume.

Les ZEC étant quant à elles immobiles, celles-ci génèrent des colonnes dans l'espace-temps, visibles sur la figure 4.27.

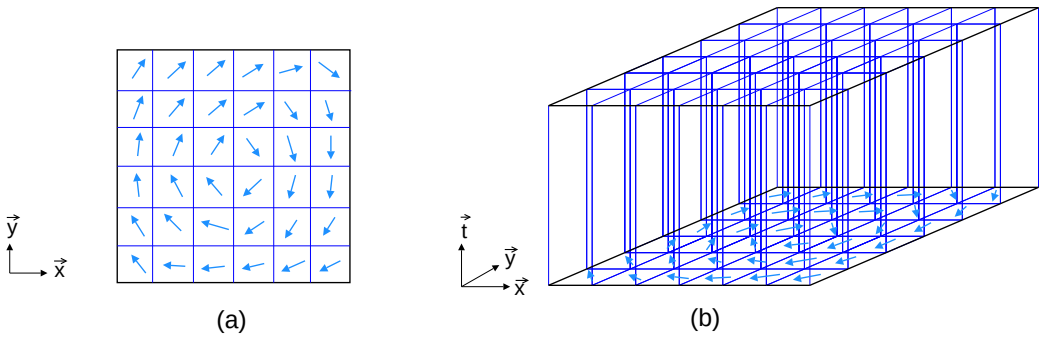

FIG. 4.27 – Représentation des ZEC un espace-temps 3D

Disposant de cette représentation, la démarche pour intégrer les obstacles mobiles est la même que celle utilisée dans la section 5 pour intégrer les obstacles fixes. Elle consiste à :

- Calculer une décomposition cellulaire de l'espace temps, relative aux volumes générés par les obstacles mobiles. On peut par exemple opter pour une décomposition pyramidale (traits rouges dans figure 4.26a), extension de la décomposition triangulaire en 3D.
- Calculer l'intersection de cette décomposition cellulaire avec les ZEC (traits bleus dans figure 4.27b).

Nous disposons alors de ZECs dans l'espace-temps 3D dépourvues d'obstacles. Comme précédemment, nous pouvons associer un curseur à chaque face de ces polyèdres. Ces curseurs auraient deux degrés de libertés (au lieu d'un), puisqu'ils pourraient se déplacer sur des portions de plan (et non plus sur des portions de droite). Le reste de l'algorithme resterait a priori inchangé, mais des études plus approfondies sont nécessaires pour s'en assurer.

Compte tenu de l'intérêt mais aussi du travail que représente cette extension, celle-ci fait partie des perspectives présentées dans la conclusion de cette thèse. Le lecteur intéressé est invité à se reporter à la section 3 page 208.

Chapitre 5

Planification de trajectoire en présence de courants variables dans le temps

Sommaire

1 Introduction

Dans le chapitre précédent, nous avons considéré que les courants étaient constants dans le temps. C'est à dire que, pour un point de l'espace où le courant est connu, sa direction et son intensité sont considérés comme étant invariants sur toute la durée de la mission.

Cette hypothèse est acceptable pour des missions de courte durée (quelques dizaines de minutes), mais n'est plus réaliste pour des missions de longue durée. Le long des côtes, notamment, le courant peut évoluer de façon significative en une seule journée, au point de s'inverser totalement. Ceci est illustré dans la figure 5.1.

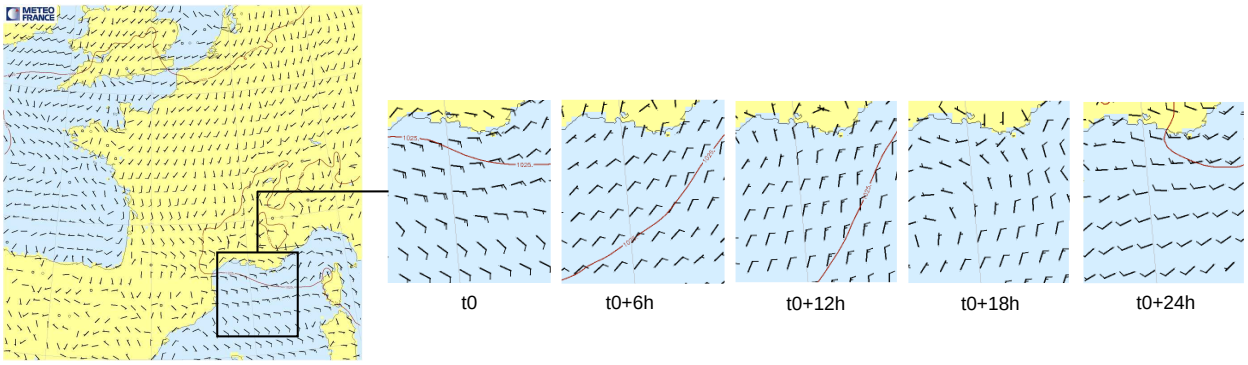

FIG. 5.1 – Phénomène d'inversion des courants
Au voisinage des côtes, les courants peuvent totalement s'inverser totalement une seule journée (Parties de cartes des vents tirées du site de météo France [MF], en Méditerranée).

Dans le cadre des missions longue durée, une nouvelle problématique apparaît : disposant de prévisions (météorologiques) sur l'évolution des courants, il s'agit de déterminer la meilleure date de départ possible, afin de se déplacer dans des courants les plus favorables possibles au cours de la mission.

Or, toutes les méthodes de planification en mesure de traiter les courants (y compris celle introduite dans le chapitre 4) nécessitent de fixer la date de départ du robot pour planifier une trajectoire. En effet, fixer cette date permet de connaître l'état initial des courants, et donc leur évolution au cours de la progression du robot.

Sur la base de ces méthodes, le seul moyen d'approximer la date de départ optimale est procéder par échantillonnage. Il s'agit de répéter la méthode de planification à des dates différentes, et de choisir la date aboutissant au meilleur résultat (par exemple, au temps de parcours le plus faible).

Cette démarche est purement locale : elle n'exploite pas l'intégralité des informations sur les courants. A chaque planification (à une date donnée), on se contente de "subir" les changements de courant, sans les anticiper. Dans ces conditions, un grand nombre de planifications peut s'avérer nécessaire pour approximer la solution optimale avec une qualité convenable.

Pour être plus efficace, une démarche globale semble nécessaire. Elle consisterait à tenir compte, à chaque pas de la planification, de l'évolution future des courants.

Partant de ce constat, nous proposons une nouvelle approche, appelée *la propagation d'onde symbolique*. Il s'agit d'une généralisation de la propagation d'onde, où le coût associé une case C n'est plus un scalaire, mais une fonction, comme illustré dans la figure 5.2. Cette fonction matérialise le temps de parcours pour atteindre C pour n'importe quelle date de départ. Ainsi, à chaque étape de la propagation, nous disposons de l'ensemble des informations nécessaires pour choisir la meilleure date de départ.

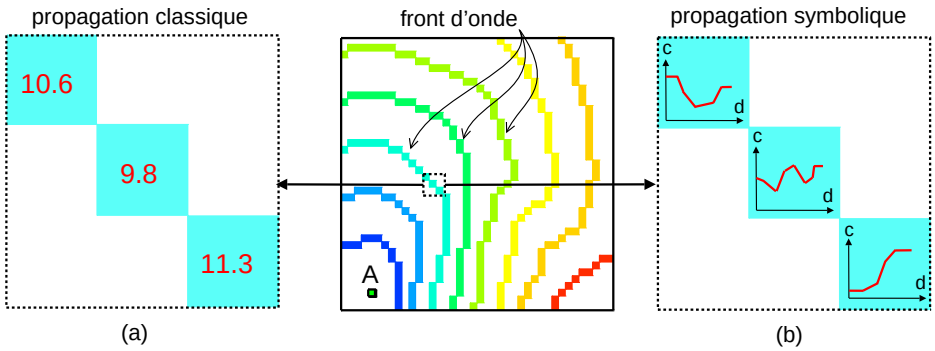

FIG. 5.2 – Principe de la propagation d'onde symbolique
(a) les coûts sont des scalaires ; (b) les coûts sont des fonctions linéaires par morceaux $c = a \cdot d + b$, modélisées par une liste de coefficients de droite $\{(a_i, b_i)\}$, d étant la date de départ en A.

2 Formulation du problème

2.1 Description

Le problème consiste à trouver la date de départ, ainsi que le chemin associé, permettant à un robot de relier deux points A et B en un temps minimal, dans un environnement planaire contenant des courants variables dans le temps.

La date de départ peut varier dans une fenêtre temporelle $W = [0, T]$. La borne T correspond à la date à partir de laquelle il n'est plus pertinent de réaliser la mission (horizon temporel). Il peut correspondre à des limites énergétiques (quantité de carburant, durée d'ensoleillement) ou à des contraintes stratégiques.

L'évolution des courants est connue à travers k cartes des courants, notées $C_1, C_2, ..., C_k$ et correspondant à des prévisions météorologiques.

2.2 Formalisation

L'environnement dans lequel évolue le robot est modélisé par un espace Euclidien $\mathcal{C}$ de dimension 2, de repère orthonormé $\mathcal{R} = (O, \vec{x}, \vec{y})$. Cet environnement est discrétisé au moyen d'une grille régulière G de dimensions $m \times n$. Chaque case X de G possède les attributs suivants :

- $\vec{w_X}$: le vecteur-vitesse courant appliqué dans l'ensemble de la case X,
- c_X : le coût, ici le temps de parcours, nécessaire pour atteindre X à partir de A,
- d_X : la date de départ en A minimisant c_X.

Notons que les points de départ et d'arrivée A et B sont approximés par les cases qui les contiennent, notées $\widetilde{A}$ et $\widetilde{B}$.

Comme évoqué précédemment, les courants présents dans l'environnement sont définis à tout instant par un ensemble de k cartes. La $i$ème carte, notée C_i, définit la valeur de $\vec{w_X}$ appliqué dans chaque case X de G (de façon homogène), du temps $t = t_{i-1}$ au temps $t = t_i$. Par convention on a $t_0 = 0$ et $t_k = T$.

Il est important de noter que la transition entre deux cartes de courants successives est considérée comme instantanée. Cette hypothèse de travail est justifiée par le fait que les cartes des courants

sont des prévisions, et qu'elles contiennent donc, par nature, une certaine incertitude. Modéliser une transition plus douce dans ce contexte n'a pas vraiment de sens. Toutefois, si le grain temporel entre les cartes est vraiment très grossier, il est tout à fait possible d'ajouter artificiellement des cartes de courants intermédiaires en utilisant des techniques d'interpolation, comme celles proposée par Inanc [ZIOM08].

En utilisant ces notations, notre problème consiste à calculer simultanément :

- La date de départ d^*, pour partir de A, dans la fenêtre $W = [0, T]$,
- Le chemin $\mathcal{P}$, c'est à dire la séquence de $l + 2$ cases $\{\widetilde{A}, C_1, C_2, ..., C_l, \widetilde{B}\}$

minimisant le temps de parcours entre $\widetilde{A}$ et $\widetilde{B}$.

Une instance de ce problème est fournie dans la figure 5.3.

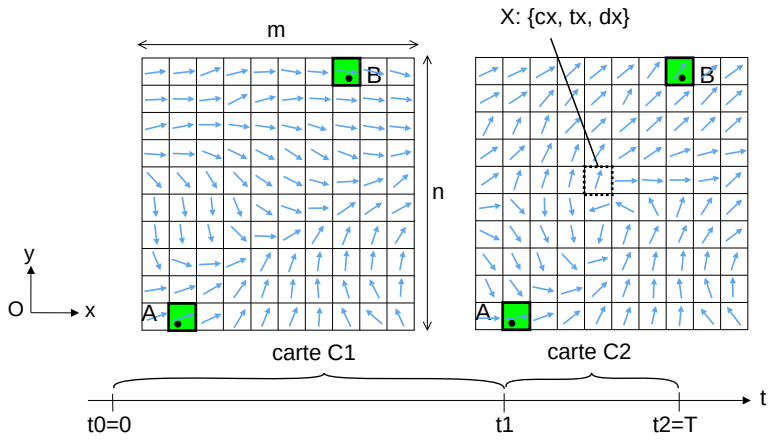

FIG. 5.3 – Un problème de planification de trajectoire en présence de courants forts

La figure présente une instance de problème avec 2 cartes de courants, de taille 10.

3 Limitations des méthodes existantes

Nous avons expliqué dans le chapitre 4 (page 97) qu'en présence de courants, la propagation d'onde était la seule approche (à notre connaissance) permettant d'obtenir une solution globalement optimale en un temps polynomial.

Nous rappelons que cet algorithme, introduit par Dorst et Trovato [DT88], consiste à propager un front d'onde F à partir de la case de départ $\widetilde{A}$, jusqu'à ce que celui-ci ait atteint la case d'arrivée $\widetilde{B}$. A chaque itération de cette propagation, la case H de coût minimal, appelée la *tête* du front d'onde, est extraite de F. Ses voisins sont évalués à l'aide d'une métrique $\mathcal{M}$ puis ajoutés à F si nécessaire. L'ensemble de l'algorithme est fourni dans la figure 1.18 page 36.

Des métriques spécifiques ont été proposées pour tenir compte des courants, par Pêtrès [PPPL05] et Garau [GAO05], et par nous-mêmes [Sou07].

Munis de l'une de ces métriques, une démarche paraît assez naturelle pour approximer la meilleure date de départ : planifier des trajectoires pour q dates différentes, puis choisir la meilleure solution, c'est à dire celle aboutissant au plus faible temps de parcours. Pour ce faire, une possibilité est de découper la fenêtre W en $q-1$ parties égales, et de planifier des trajectoires à 0, $\frac{T}{q-1}$, $\frac{2T}{q-1}$, ..., T.

Cette démarche revient à échantillonner la fonction de coût $c_{\widetilde{B}}$, donnant le temps de parcours pour atteindre $\widetilde{B}$ en fonction de la date de départ d, comme illustré dans la figure 5.4.

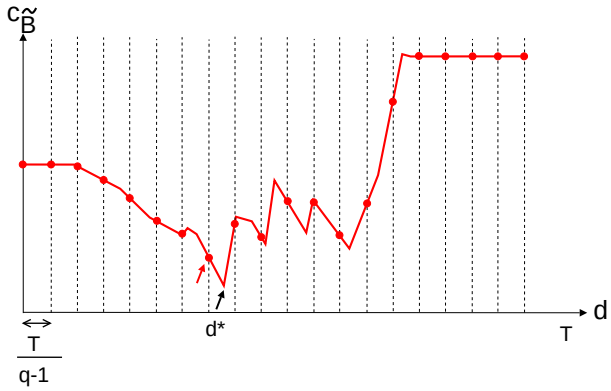

FIG. 5.4 – Approximation de la date de départ optimale par échantillonnage
La figure illustre la fonction c_B (temps de parcours pour atteindre $\widetilde{B}$ en fonction de la date de départ d en $\widetilde{A}$) pour le problème de la figure 5.3, ainsi que son approximation à l'aide de $q = 20$ propagations d'ondes (points rouges). La meilleure solution approximée (indiquée par une flèche rouge) donne un temps de parcours environ 13% plus important que la solution optimale (indiquée par une flèche noire).

Une valeur assez élevée de q peut s'avérer nécessaire pour obtenir une bonne approximation la solution optimale. Par exemple, pour le problème de la figure 5.3, il faut environ 60 répétitions pour être à +10% du coût optimal, alors qu'il s'agit d'une instance très simple (grille très grossière, un seul changement de courant).

De plus, la qualité de la solution ne peut être garantie en fonction de q, de part la "myopie" de la propagation d'onde. En effet, à partir de la date de départ, l'état des courants est progressivement découvert, au fur et à mesure de la progression du front d'onde. L'état des courants à l'itération i de l'algorithme est figé une fois pour toutes, selon la position du front d'onde et l'état des courants à l'itération $i-1$.

Dans ces conditions, on peut être amené à sur-échantillonner des régions où le temps de parcours est invariant (paliers de droite dans la figure 5.4), et sous-échantillonner des régions où le temps de parcours varie rapidement (partie centrale).

Pour effectuer la juste quantité de calcul, nous proposons de calculer directement la fonction $c_{\widetilde{B}}$ plutôt que de l'échantillonner. Pour cela, nous allons proposer une généralisation de la propagation d'onde, appelée *propagation d'onde symbolique*. Dans cette généralisation, les coûts propagés ne sont plus des scalaires mais des fonctions de la date de départ. Ainsi, dans l'algorithme initial, les opérateurs numériques (tels que l'addition de deux nombres) sont remplacés par des opérateurs symboliques (l'addition de deux fonctions), d'où le nom de l'approche.

Nous avons récemment découvert que dans un domaine fort différent (routage de messages dans un réseau de communication dynamique[1]) une approche similaire avait été proposée. L'algorithme correspondant, appelé *SW2* (pour *Source Waiting 2*), a été élaboré en partant du principe que les fonctions de coût manipulées étaient quelconques. Ainsi, contrairement à notre approche, la propagation des informations effectuée dans SW2 n'est pas guidée. A chaque itération i, chaque noeud du réseau dont l'état a changé depuis l'itération $i-1$ est une source de propagation d'information. Autrement dit, il n'y a plus de notion de "tête" de front d'onde (c'est à dire une source unique de propagation), puisque à chaque itération, la propagation se fait simultanément depuis plusieurs sources.

Or, dans notre problématique, les fonctions de coût ne sont pas totalement quelconques, mais reflètent des phénomènes physiques, ici l'évolution de courants dans le temps. Nous pensons que dans ces conditions, garder la notion de tri du front d'onde peut permettre une économie substantielle dans le nombre de noeuds évalués, et donc dans les temps de planification.

Ainsi nous monterons expérimentalement que, même si la propagation d'onde symbolique et SW2 possèdent la même complexité temporelle théorique dans le pire des cas, notre approche se révèle bien plus efficace sur des instances de notre problème.

4 La propagation d'onde symbolique

4.1 Algorithme

Comme expliqué précédemment, le principe de notre approche est de considérer que le coût (i.e. le temps de parcours) c_X associé à une case X n'est plus un scalaire, mais une fonction de la date de départ d. Muni de ce nouveau type de coût, la résolution de notre problème devient très naturelle. Elle consiste à propager le front d'onde jusqu'à la case $\widetilde{B}$, comme précédemment, et de localiser le minimum de la fonction $c_{\widetilde{B}}$. Ce minimum correspond à la date de départ optimale recherchée.

Pour que cela soit possible, l'algorithme original, fourni dans la figure 1.18 page 36, doit être modifié afin de manipuler des fonctions au lieu de scalaires. L'algorithme résultant, appelé propagation d'onde symbolique, est fourni dans la figure 5.5.

Il est important de noter que cet algorithme est une généralisation de la propagation d'onde, dite "classique", rappelée dans la section précédente. En effet, si les courants sont invariants dans le temps, alors les fonctions c_X seront des constantes, et les algorithmes des figures 1.18 et 5.5 donneront exactement le même résultat. Le nombre d'opérations sera globalement le même, aux affectations de l'attribut d_X près (aux lignes 3, 4 et 12), qui deviendront inutiles.

Les différences majeures entre les algorithmes des figures 1.18 (page 36) et 5.5 sont les suivantes :

1. Ligne 8 (Métrique) : La métrique $\mathcal{M}'_{H,V}$, définissant le temps de parcours entre H et V, ne renvoie plus un nombre mais une fonction du temps.
2. Ligne 8 (Opération d'évaluation) : Le changement de métrique ci-dessus a deux conséquences sur l'évaluation de V :
   (a) L'opération de composition "$\circ$" fait son apparition.
   (b) L'opération d'addition "+" n'additionne plus deux nombres, mais deux fonctions.

[1] Cette problématique est très similaire à la notre, dans la mesure où le message est assimilable au drone, et l'état des canaux de communication à celui des courants.

PROPAGATION_ONDE_SYMBOLIQUE(A, B, G, W)

▷ **Entrée :** $\widetilde{A}, \widetilde{B}$: cases de départ et d'arrivée
▷ **Entrée :** G : grille
▷ **Entrée :** W : fenêtre temporelle pour la date de départ en $\widetilde{A}$
▷ **Locale :** F : front d'onde courant
▷ **Locale :** H : tête de F (case de moindre coût)
▷ **Locale :** V : un voisin de H
▷ **Locale :** c_V^{temp} : évualuation temporaire de V

1 **Début**
2 $\quad F \leftarrow \{\widetilde{A}\}$
3 $\quad c_X \leftarrow +\infty, d_X \leftarrow 0, \forall X \in G \setminus \{\widetilde{A}\}$
4 $\quad c_{\widetilde{A}} \leftarrow 0, d_{\widetilde{A}} \leftarrow 0$
5 $\quad$ **faire**
6 $\quad\quad H \leftarrow \arg\min\{c_X(d_X), X \in F\}$
7 $\quad\quad$ **pour chaque** $V \in \mathcal{V}(H)$ **faire**
8 $\quad\quad\quad c_V^{temp} \leftarrow c_H + \mathcal{M}'_{H,V} \circ (c_H + d), d \in W$ //opération d'évaluation
9 $\quad\quad\quad c_V^{temp} \leftarrow \min\{c_V, c_V^{temp}\}$ //opération de comparaison
10 $\quad\quad\quad$ **si** $c_V^{temp} \neq c_V$ **alors**
11 $\quad\quad\quad\quad c_V \leftarrow c_V^{temp}$
12 $\quad\quad\quad\quad d_V \leftarrow \min\{c_V(d), d \in W\}$
13 $\quad\quad\quad\quad F \leftarrow F \cup \{V\}$
14 $\quad\quad F \leftarrow F \setminus \{H\}$
15 $\quad$ **tantque** $H \neq \widetilde{B}$
16 **Fin**

FIG. 5.5 – La propagation d'onde symbolique

3. Ligne 9 (Opération de comparaison) : De façon analogue à l'opération d'évaluation, l'opérateur "min" ne compare plus deux nombres, mais deux fonctions.

4. Ligne 12 (Localisation du minimum) : Cette opération est nouvelle par rapport à la propagation d'onde classique. Il s'agit de localiser le minimum de la fonction c_V, et de le stocker dans l'attribut d_V, pour une utilisation ultérieure (plus précisément à la ligne 6, lors de l'extraction de la tête du front d'onde).
   Nous verrons que, contrairement à ce que l'on peut penser *a priori*, cette recherche de minimum est simple, et ne fait appel à aucun algorithme d'optimisation (telle que la descente du gradient, par exemple).

5. Ligne 13 (Mise à jour du front d'onde) : Dans la propagation d'onde classique, toute case préalablement développée par le front d'onde ne pouvait plus l'être à l'avenir. Dans la propagation d'onde symbolique, ce n'est plus le cas. Ceci est dû au fait que chaque case X du front d'onde est trié selon le minimum des fonctions c_X (ligne 6). Même si ce choix est souvent pertinent, ceci n'est pas toujours le cas, car le minimum de c_X n'est pas représentatif de l'ensemble des points de c_X.

Nous allons décrire chacun de ces changements en détail. Leur principale caractéristique est d'exploiter la linéarité par morceaux des fonctions de coût.

4.1.1 Une nouvelle métrique

Nous rappelons que le courant est défini par k cartes C_1, C_2, ... C_k et que chaque carte C_i est appliquée dans l'environnement du temps $t = t_{i-1}$ au temps $t = t_i$.

Commençons par étudier l'impact du dernier changement de courant (c'est à dire le passage de la carte C_{k-1} à la carte C_k) sur le déplacement entre la tête du front d'onde H et l'un de ses voisins V. Si t représente la date d'arrivée en H alors :

- Si $t \geq t_k = T$ le déplacement $H \rightarrow V$ est entièrement effectué dans la carte C_k. Le temps de parcours entre H and V est donc égal à :

$$\tau_k = \mathcal{M}_{H,V}(t_k)$$

- Si $t \in [t_{k-1}, t_k - \mathcal{M}_{H,V}(t_{k-1})]$ le déplacement $H \rightarrow V$ est entièrement effectué dans la carte C_{k-1}. Le temps de parcours entre H and V est donc égal à :

$$\tau_{k-1} = \mathcal{M}_{H,V}(t_{k-1})$$

- Enfin, si $t \in [t_k - \mathcal{M}_{H,V}(t_{k-1}), t_k]$, un changement de carte des courants intervient pendant le déplacement $H \rightarrow V$, matérialisé par le point M dans la figure 5.6.

  Dans ces conditions, le début du déplacement est réalisé dans la carte C_{k-1}, et la fin dans C_k. Le temps de parcours correspondant est donc une moyenne pondérée entre les deux coûts précédents, donnée par :

$$\tau_{k-1,k} = \left(\frac{\tau_k}{\tau_{k-1}} - 1\right) t + \left[\tau_k - \tau_k \cdot \left(\frac{\tau_k}{\tau_{k-1}} - 1\right)\right] \tag{5.1}$$

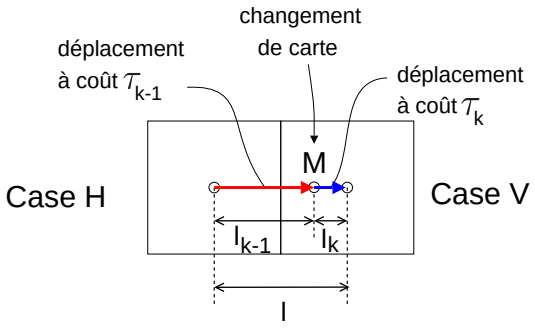

FIG. 5.6 – Changement de carte des courants pendant un déplacement
La figure illustre le passage de la carte des courants C_{k-1} à la carte C_k pendant le déplacement $H \rightarrow V$, de longueur totale l. La première partie de ce déplacement (longueur l_{k-1}) est effectuée dans la carte C_{k-1} et la seconde (longueur l_k) dans C_k.

PREUVE de l'équation 5.1

Considérons la situation de la figure 5.6 : le déplacement entre les points H et V nécessite de parcourir une longueur l. Cependant, le passage de carte de C_{k-1} à C_k, modélisé par le point M, se produit avant que le robot n'ait eu le temps de parcourir la totalité de cette distance. Pour calculer le temps de parcours associé au déplacement $H \rightarrow V$, il faut donc calculer les temps de parcours associés aux déplacements $H \rightarrow M$ puis $M \rightarrow V$, et en faire la somme.

▷ Temps de parcours $H \rightarrow M$

Soit t la date de départ à la case H et t_k la date de passage de la carte C_{k-1} à la carte C_k. Selon ces notations, le temps de parcours nécessaire pour atteindre M à partie de H est :

$$\delta_{k-1} = t_k - t$$

▷ Temps de parcours $M \rightarrow V$

La longueur l_{k-1} entre H et M vérifie :

$$l_{k-1} = \delta_{k-1} \cdot v_{k_1}$$

où v_{k-1} est la vitesse de déplacement du robot (par rapport au sol) dans les courants de la carte C_{k-1}.

Nous savons que si le robot avait parcouru l'ensemble de la distance l dans C_{k-1}, son temps de parcours aurait été égal à τ_{k-1}. On en déduit donc que la vitesse v_{k-1} est égale à l/τ_{k-1}. De plus, nous avons vu précédemment que $\delta_{k-1} = t_k - t$.

On obtient ainsi :

$$l_{k-1} = (t_k - t) \cdot \frac{l}{\tau_{k-1}}$$

Nous en déduisons la longueur l_k parcourue entre M et V :

$$l_k = l - l_{k-1} = \left[1 - \frac{t_k - t}{\tau_{k-1}}\right] \cdot l$$

De plus, nous savons que l_k vérifie :

$$l_k = \delta_k \cdot v_k$$

où δ_k et v_k représentent respectivement le temps de parcours et la vitesse du robot entre M et V.

Comme $v_k = l/\tau_k$, il vient finalement :

$$\delta_k = \left[1 - \frac{t_k - t}{\tau_{k-1}}\right] \cdot \tau_k$$

▷ Temps de parcours $H \rightarrow V$

Le temps de parcours global entre H et V est la somme des deux quantités précédentes :

$$\begin{aligned} \delta_{k-1} + \delta_k &= (t_k - t) + \left[1 - \frac{t_k - t}{\tau_{k-1}}\right] \cdot l\tau_k \\ &= \left(\frac{\tau_k}{\tau_{k-1}} - 1\right) t + \left[\tau_k - \tau_k \cdot \left(\frac{\tau_k}{\tau_{k-1}} - 1\right)\right] \end{aligned}$$

Ce qui correspond à l'expression de $\tau_{k-1,k}$ de l'équation 5.1.□

Notons que $\tau_{k-1,k}$ peut être vu comme une *phase transitoire* entre les coûts τ_{k-1} et τ_k, illustrée en rouge dans la figure 5.7.

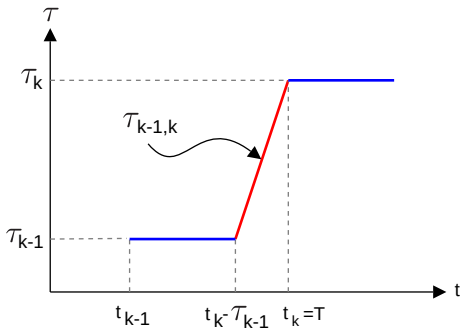

FIG. 5.7 – Evolution du temps de parcours lors du dernier changement de courant

Ces premiers résultats permettent de définir la métrique $\mathcal{M}'$ sur l'intervalle $[t_{k-1}, t_k]$, comme étant :

$$\mathcal{M}'_{H,V}(t) = \mathcal{M}_{H,V}(t_k) \quad \text{si} \quad t \geq t_k$$

Et :

$$\mathcal{M}'_{H,V}(t) = \begin{cases} \tau_{k-1} & \text{si} \quad t \in [t_{k-1}, t_k - \tau_{k-1}] \\ \gamma_k \cdot t + [\tau_k - t_k \cdot \gamma_k] & \text{si} \quad t \in [t_k - \tau_{k-1}, t_k] \end{cases}$$

avec $\gamma_k = \left(\frac{\mathcal{M}'_{H,V}(t_k)}{\mathcal{M}_{H,V}(t_{k-1})} - 1\right)$, $\tau_k = \mathcal{M}_{H,V}(t_k)$ et $\tau_{k-1} = \mathcal{M}_{H,V}(t_{k-1})$.

Si nous reproduisons ce raisonnement pour l'avant-dernier changement de courant, c'est à dire pour le passage de la carte C_{k-2} à C_{k-1}, nous pouvons définir, de manière similaire, la métrique $\mathcal{M}'$ sur l'intervalle $[t_{k-2}, t_{k-1}]$:

$$\mathcal{M}'_{H,V}(t) = \begin{cases} \tau_{k-2} & \text{si} \quad t \in [t_{k-2}, t_{k-1} - \tau_{k-2}] \\ \gamma_{k-1} \cdot t + [\tau_{k-1} - t_{k-1} \cdot \gamma_{k-1}] & \text{si} \quad t \in [t_{k-1} - \tau_{k-2}, t_{k-1}] \end{cases}$$

avec $\gamma_{k-1} = \left(\frac{\mathcal{M}'_{H,V}(t_{k-1})}{\mathcal{M}_{H,V}(t_{k-2})} - 1\right)$, $\tau_{k-1} = \mathcal{M}_{H,V}(t_{k-1})$ et $\tau_{k-2} = \mathcal{M}_{H,V}(t_{k-2})$.

En répétant cette démarche jusqu'au premier changement de courant, c'est à dire jusqu'au passage de C_0 à C_1, nous obtenons la définition complète de $\mathcal{M}'$ sur $[0,T]$:

$$\mathcal{M}'_{H,V}(t) = \mathcal{M}_{H,V}(t_k) \quad \text{si} \quad t \geq t_k$$

Et :

$$\forall i \in [1,k] : \quad \mathcal{M}'_{H,N}(t) = \begin{cases} \tau_{i-1} & \text{si} \quad t \in [t_{i-1}, t_i - \tau_{i-1}] \\ \gamma_i \cdot t + [\tau_i - t_i \cdot \gamma_i] & \text{si} \quad t \in [t_i - \tau_{i-1}, t_i] \end{cases}$$

avec $\gamma_i = \left(\frac{\mathcal{M}'_{H,V}(t_i)}{\mathcal{M}_{H,V}(t_{i-1})} - 1 \right)$ et $\tau_i = \mathcal{M}_{H,V}(t_i)$.

Le graphe de la métrique $\mathcal{M}'$ ainsi définie est illustré dans la figure 5.8. Dans le cas nominal, ce type de graphe est constitué d'une alternance de paliers à coût constant et de phases de transition. Plus précisément, entre t_{i-1} et t_i, il y a généralement un palier suivi d'une phase de transition.

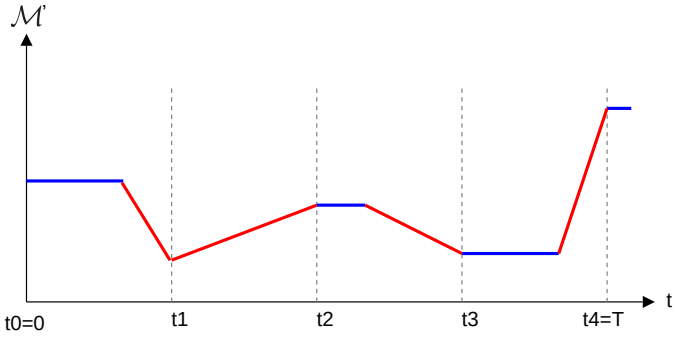

FIG. 5.8 – Illustration de la métrique $\mathcal{M}'$

La figure présente le graphe de $\mathcal{M}'$ pour 4 changements de courants, à intervalles réguliers, dans $[0,T]$.

Toutefois, ce schéma n'est pas respecté dans le cas où le temps de parcours du robot devient plus élevé que l'espacement temporel entre deux cartes de courants. En effet, si la valeur de $\mathcal{M}_{H,V}(t_{i-1})$ est supérieure que la "durée de vie" de la carte C_{i-1} (c'est à dire $t_i - t_{i-1}$), alors il n'y aura pas de palier à coût constant dans l'intervalle $[t_{i-1}, t_i]$. Ce phénomène apparaît dans la figure 5.8 pour $i = 2$: deux phases de transitions se succèdent entre t_1 et t_2.

4.1.2 Une nouvelle opération d'évaluation

L'opération d'évaluation consiste à calculer le coût d'une nouvelle case V à partir d'une case H, déjà évaluée.

Dans la propagation d'onde classique (figure 1.18 page 36), le calcul de c_V découle des étapes suivantes :

1. c_H donne, par définition, le temps de parcours global de $\widetilde{A}$ à H,
2. $c_H + d$ donne la date d'arrivée en H (puisque d représente la date de départ en $\widetilde{A}$)
3. $\mathcal{M}_{H,V}(c_H + d)$ donne, par définition, le temps de parcours local de H vers V,
4. $c_H + \mathcal{M}_{H,V}(c_H + d)$ donne enfin le temps de parcours global de $\widetilde{A}$ à V

L'ensemble de ces étapes est illustré dans la figure 5.9. Il est important de noter que, puisque toutes les grandeurs manipulées (c_H, d et $\mathcal{M}$) sont des scalaires, toutes les opérations d'addition sont numériques.

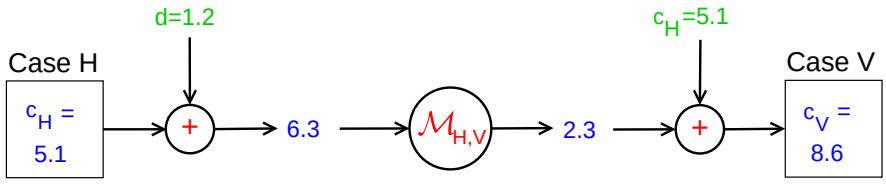

FIG. 5.9 – L'opération d'évaluation dans la propagation d'onde classique

Dans la propagation d'onde symbolique (figure 5.5 page 133), les étapes aboutissant au calcul de c_V sont similaires :

1. c_H donne, par définition, le temps de parcours global de $\widetilde{A}$ à H,
2. $c_H + d$ donne la date d'arrivée en H (puisque d représente la date de départ en $\widetilde{A}$)
3. $\mathcal{M}'_{H,V} \circ (c_H + d)$ donne, par définition, le temps de parcours local de H vers V,
4. $c_H + \mathcal{M}'_{H,V} \circ (c_H + d)$ donne enfin le temps de parcours global de $\widetilde{A}$ à V

Cependant, comme les coûts manipulés ne sont plus des scalaires mais des fonctions, les étapes sont de nature très différente de précédemment.

Dans l'étape 2, tout d'abord, l'addition $c_H + d$ n'est plus numérique mais symbolique : si c_H était définie par :

$$c_H : d \mapsto a \cdot d + b$$

Alors $c_H + d$ sera définie par :

$$c_H + d \mapsto (a + 1) \cdot d + b$$

Ensuite, dans l'étape 3, l'évaluation de $\mathcal{M}$ à la date $c_H + d$ est remplacée par une composition de $\mathcal{M}'$ avec la fonction $c_H + d$.

Enfin, dans l'étape 4, l'addition de c_H et $\mathcal{M}'$ est elle aussi symbolique.

L'ensemble de ces étapes est illustré dans la figure 5.10.
Pour une meilleure compréhension, déroulons en détail l'exemple de la figure 5.10.

1. La fonction c_H est définie par :

$$c_H : d \rightarrow \begin{cases} 1 & \text{si} \quad d \in [0, 5] \\ 3 \cdot d - 14 & \text{si} \quad d \in [5, 6] \\ 4 & \text{si} \quad d \in [6, 10] \end{cases}$$

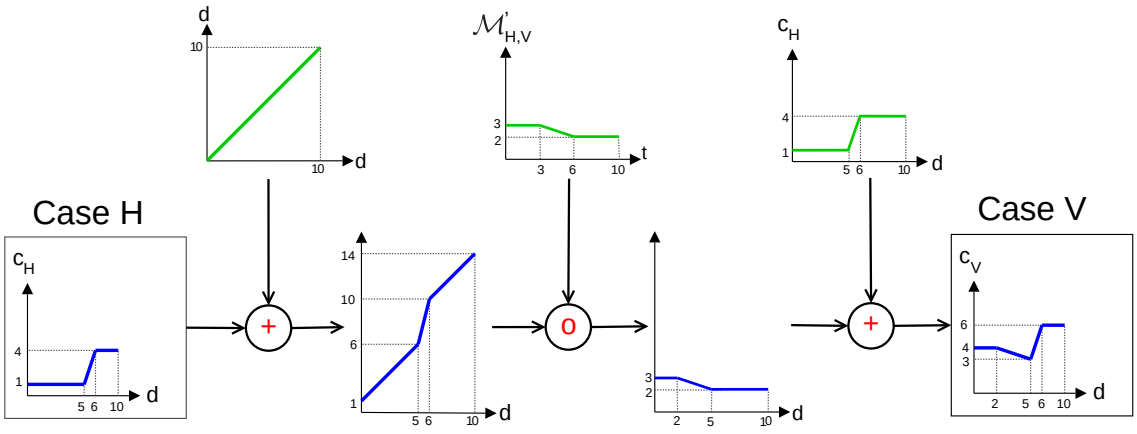

FIG. 5.10 – L'opération d'évaluation dans la propagation d'onde symbolique

2. Nous en déduisons l'expression de la fonction $c_H + d$:

$$c_H + d : d \rightarrow \begin{cases} d+1 & \text{si} \quad d \in [0,5] \\ 4 \cdot d - 14 & \text{si} \quad d \in [5,6] \\ d+4 & \text{si} \quad d \in [6,10] \end{cases}$$

3. En composant $c_H + d$ avec la métrique $\mathcal{M}'_{H,V}$ d'expression :

$$\mathcal{M}'_{H,V} : t \rightarrow \begin{cases} 3 & \text{si} \quad t \in [0,3] \\ -1/3 \cdot t + 4 & \text{si} \quad t \in [3,6] \\ 2 & \text{si} \quad t \in [6,+\infty[\end{cases}$$

Nous obtenons :

$$\mathcal{M}'_{H,V} \circ (c_H + d) : d \rightarrow \begin{cases} 3 & \text{si} \quad d \in [0,2] \\ -1/3 \cdot t + 11/3 & \text{si} \quad d \in [2,5] \\ 2 & \text{si} \quad d \in [5,10] \end{cases}$$

4. Enfin, en additionnant cette dernière fonction avec c_H, nous obtenons l'expression de c_V :

$$c_V : d \rightarrow \begin{cases} 4 & \text{si} \quad d \in [0,2] \\ -1/3 \cdot d + 14/3 & \text{si} \quad d \in [2,5] \\ 3 \cdot d - 12 & \text{si} \quad d \in [5,6] \\ 6 & \text{si} \quad d \in [6,10] \end{cases}$$

Une première idée pour réaliser cette opération de l'évaluation est d'utiliser des opérateurs symboliques standards (pour l'addition et la composition de fonctions), présents dans les logiciels du marché (comme Mathlab ou Mapple, par exemple). Toutefois, l'utilisation de tels opérateurs peut se révéler lourde et coûteuse en ressources.

Pour de meilleures performances, nous proposons d'exploiter les caractéristiques des fonctions manipulées.En effet, comme nous l'avons vu précédemment, la métrique $\mathcal{M}'$, et par conséquent les fonctions de coût c_X sont linéaires par morceaux. En utilisant cette connaissance, la construction de la fonction c_V se trouve grandement simplifiée.

En effet, si sur un intervalle donné, la fonction c_H est de la forme :

$$c_H : d \mapsto a \cdot d + b$$

Et $\mathcal{M}'_{H,V}$ est de la forme :

$$\mathcal{M}'_{H,V} : t \mapsto a' \cdot t + b'$$

Alors c_V est définie par :

$$c_V : d \mapsto [a'(a+1) + a] \cdot d + [b(a'+1) + b'] \tag{5.2}$$

Preuve de l'équation 5.2

Comme expliqué précédemment, $c_H + d : d \mapsto (a+1) \cdot d + b$.
D'où : $\mathcal{M}'_{H,V} \circ (c_H + d) : d \mapsto a'(\mapsto (a+1) \cdot d + b) + b = [a'(a+1)] \cdot d + [a'b + b']$.
Enfin : $c_V = c_H + \mathcal{M}'_{H,V} \circ (c_H + d)$, d'où le résultat. □

En appliquant ce résultat sur tous les morceaux de c_H et $\mathcal{M}'_{H,V}$, nous obtenons un opérateur d'évaluation *ad hoc*. Cet opérateur, utilisé lors de nos tests de performances, est détaillé dans l'annexe 3 de cette thèse.

4.1.3 Une nouvelle opération de comparaison

L'opération de comparaison permet de comparer deux provenances concurrentes H_0 et H_1 pour atteindre une même case V, plus précisément les coûts c_V^0 et c_V^1, associés aux trajets $H_0 \to V$ et $H_1 \to V$.

Dans la propagation d'onde classique, cette opération consiste à comparer les deux nombres c_V^0 et c_V^1. Le trajet $H_i \to V$ aboutissant au coût c_V^i le plus faible sera choisi atteindre V, comme illustré dans la figure 5.11. Notons que l'autre trajet (équivalent ou plus coûteux) sera définitivement abandonné.

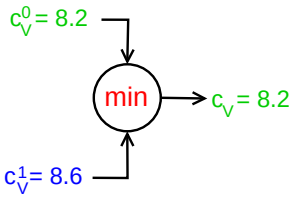

Fig. 5.11 – L'opération de comparaison dans la propagation d'onde classique

Dans la propagation d'onde symbolique, l'opération de comparaison concerne les deux fonctions c_V^0 et c_V^1 de la façon suivante : si le graphe de c_V^i est en dessous du graphe de c_V^{1-i} sur une période donnée, alors le trajet $H_i \rightarrow V$ sera choisi pour atteindre V pendant cette période, comme illustré dans la figure 5.12.

Selon ce principe, le trajet optimal pour atteindre V peut varier selon les périodes (ce qui n'était pas le cas précédemment). En effet, si les graphes des fonctions c_V^0 et c_V^1 se chevauchent, alors il peut y avoir "alternance" dans les trajets optimaux. Dans la figure 5.12, par exemple, le trajet optimal pour atteindre V est $H_0 \rightarrow V$ (partie verte dans le graphe de c_V), puis $H_1 \rightarrow V$ (partie bleue), puis de nouveau $H_0 \rightarrow V$.

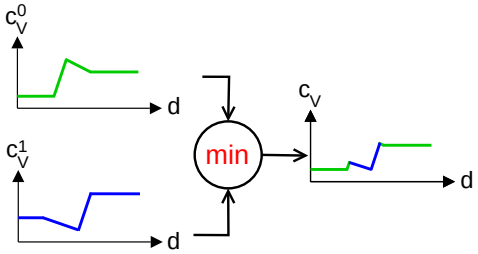

FIG. 5.12 – L'opération de comparaison dans la propagation d'onde symbolique

Si nous détaillons l'exemple de la figure 5.12, nous avons les expressions de fonctions suivantes :

$$c_V^0 : d \mapsto \begin{cases} 4 & \text{si} \quad d \in [0,2] \\ -1/3 \cdot d + 14/3 & \text{si} \quad d \in [2,6] \\ 3 \cdot d - 12 & \text{si} \quad d \in [5,6] \\ 6 & \text{si} \quad d \in [6,10] \end{cases}$$

Et :

$$c_V^1 : d \mapsto \begin{cases} 2 & \text{si} \quad d \in [0,3] \\ 3 \cdot d - 7 & \text{si} \quad d \in [3,4] \\ -1/2 \cdot d + 7 & \text{si} \quad d \in [4,6] \\ 4 & \text{si} \quad d \in [6,10] \end{cases}$$

L'opération de comparaison donne :

$$c_V : d \mapsto \begin{cases} 2 & \text{si} \quad d \in [0,3] \\ 3 \cdot d - 7 & \text{si} \quad d \in [3,7/2] \\ -1/3 \cdot d + 14/3 & \text{si} \quad d \in [7/2,5] \\ 3 \cdot d - 12 & \text{si} \quad d \in [5,38/7] \\ -1/2 \cdot d + 7 & \text{si} \quad d \in [38/7,6] \\ 4 & \text{si} \quad d \in [6,10] \end{cases} \tag{5.3}$$

Tout comme pour la composition de fonctions, l'opération de comparaison peut être effectuée à l'aide d'opérateurs standards, mais aussi à l'aide d'un opérateur spécifique, exploitant les propriétés des fonctions manipulées.

Dans cette optique, nous proposons un opérateur fonctionnant de la façon suivante :

- Initialiser c_V à $t = 0$: $c_V = c_V^i$ si $c_V^i(0) < c_V^{1-i}(0)$; $c_V = c_V^{1-i}$ sinon.
- A chaque intersection à l'abscisse $d = d_I$ entre les graphes de c_V^0 et c_V^1, changer la définition de c_V : si c_V était définie par l'expression c_V^i avant d_I, alors c_V est définie par l'expression c_V^{1-i} après d_I.

Le calcul des intersections entre les graphes est rendu très simple par la linéarité par morceaux des fonctions. Un opérateur suivant ce principe sera utilisé lors de nos tests de performances. Il est détaillé dans l'annexe 3 de cette thèse.

4.2 Localisation du minimum

Comme expliqué précédemment, chaque fonction c_X est composée d'une succession de segments de droite $[M_i, M_{i+1}]$, avec $M_i = (d_i, c_X(d_i))$.

Le minimum de c_X peut donc être facilement calculé en énumérant ses segments, et en localisant la valeur de d_i pour laquelle $c_X(d_i)$ est minimal. Ainsi, aucune méthode d'optimisation n'est nécessaire ici.

4.3 Mise à jour du front d'onde

Comme on peut le voir dans la ligne 13 de l'algorithme, le front d'onde est successivement développé à partir de sa "tête" H, c'est à dire la case de moindre coût.

Dans le cas de coûts scalaires, cette façon de procéder permet de garantir l'optimalité du coût entre la case de départ et H. En effet, toute autre case X du front d'onde ayant déjà un coût supérieur ou égal à H, il n'y a pas de moyen d'améliorer le coût pour atteindre H en faisant un détour par X. Formellement parlant, puisque à la base nous avons $c_H \leq c_X$ et puisque $\mathcal{M}_{H,X} > 0$, alors nécessairement $c_H < c_X + \mathcal{M}_{H,X}$.

Sur ce constat, toute case du front d'onde ayant déjà été développée n'a plus à être ré-évaluée par la suite, puisque toutes les évaluations futures seront forcement plus élevées. Ainsi, dans l'algorithme de la propagation classique (figure 1.18 page 36), les cases ayant été développées par le front d'onde sont stockées dans un ensemble D. Par la suite, les cases appartenant à D sont exclues du processus d'évaluation.

Dans le cas où les coûts sont des fonctions, ce principe n'est plus applicable. Il n'y a plus véritablement de notion d'ordre entre les coûts, puisque dès lors que le graphe de deux fonctions c_X et c_Y se chevauchent, il n'est plus possible d'identifier si c_X est supérieure à c_Y et inversement.

Nous avons partiellement résolu ce problème en ne comparant pas les fonctions entre elles, mais des représentants de ces fonctions. Dans notre algorithme, nous avons choisi le minimum des fonctions, mais d'autres choix auraient été possibles, comme la valeur moyenne par exemple. Cependant, quel que soit le représentant utilisé, celui-ci est une valeur scalaire qui aboutit inévitablement à une perte d'information sur la fonction initiale (domaine de variation, pente, etc.).

Dès lors, nous n'avons plus la garantie, comme précédemment, que le coût de toute case développée par le front d'onde ne peut être amélioré par la suite. C'est notamment le cas si les coûts de déplacement, donné par la métrique $\mathcal{M}'$, évoluent de façon radicalement différente, voir opposée, dans une même région de la grille.

Pour illustrer ce point, prenons l'exemple de la figure 5.13. Les métriques $\mathcal{M}'$, représentées dans les ellipses, illustrent le coût de quelques déplacements entre des cases A, B, C, D et E. Comme on peut le voir, l'évolution du coût de déplacement entre A et D est très différente selon que nous passons par B ou par C. Si nous passons par B, le coût est initialement élevé puis diminue, et si nous passons par C, nous avons exactement l'inverse.

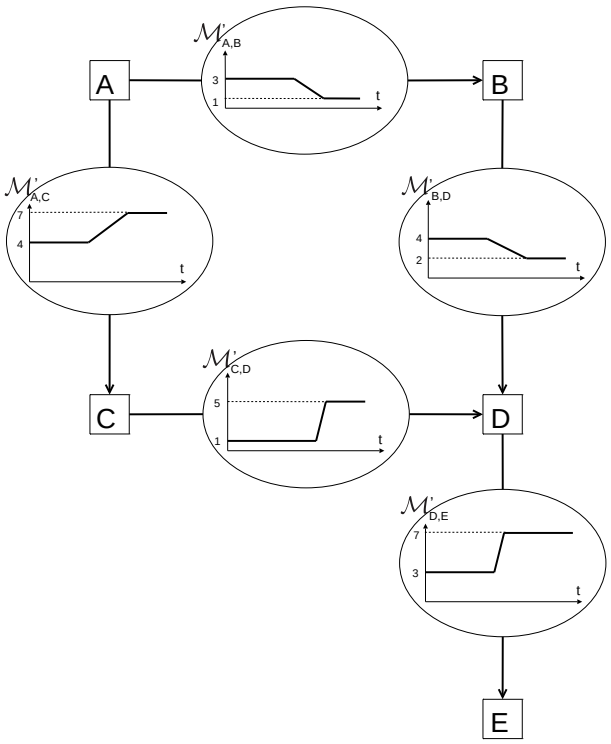

FIG. 5.13 – Exemple de fonctions de coût entraînant des re-developpements de cases
Les carrés représentent des cases de la grille, et les ellipses contiennent les fonctions $\mathcal{M}'$ modélisant l'évolution du coût de certains déplacements entre ces cases.

Nous allons montrer que dans ce contexte, nous ne pouvons plus nous permettre de développer les cases du front d'onde une seule fois, sous peine d'être sous-optimal.

Appliquons la propagation d'onde symbolique sur l'exemple de la figure 5.13. Les développements successifs sont illustrés dans la figure 5.14. Dans (a), la case A évalue ses deux voisins B et C. Ensuite, dans (b) la case de coût minimal est choisie pour être développée, c'est à dire B (car le minimum de la fonction c_B est égal à 1, ce qui est inférieur au minimum de la fonction c_C, égal à 4). Puis, selon le même principe, la case D est développée, et évalue la case E.

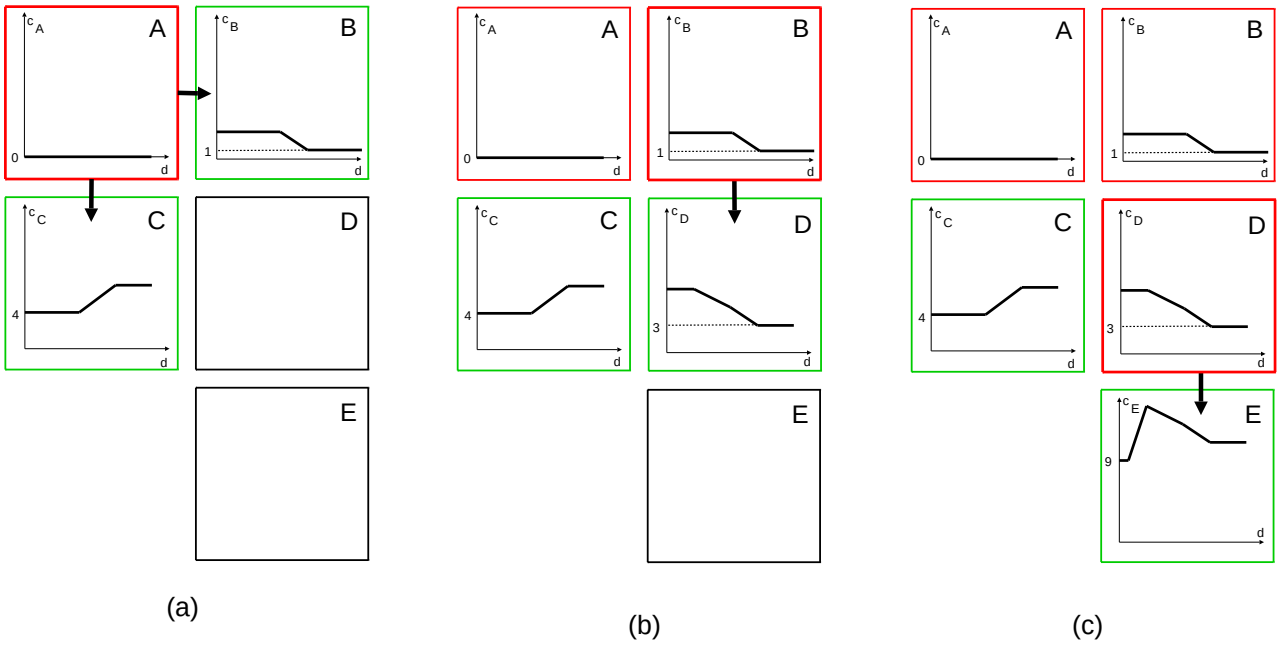

FIG. 5.14 – Propagation d'onde symbolique avec développement unique des cases.
La figure illustre l'exécution de l'algorithme de la propagation d'onde symbolique sur l'exemple de la figure 5.13, avec la contrainte de ne développer chaque case qu'une seule fois. Les cases rouges sont les cases développées, les cases vertes en attente de développement, et les cases noires sont inexplorées. A chaque itération, la tête du front d'onde est mise en valeur en gras.

Si nous gardons le principe de ne développer les cases qu'une seule fois, le coût de la case D ne pourra plus jamais être amélioré, notamment depuis C. Il en va de même pour le coût de E. Nous aboutissons donc à la conclusion que la date optimale pour atteindre E est $d = 0$, le chemin associé étant $A \rightarrow B \rightarrow D \rightarrow E$, de coût 9.

Mais, en réalité, cette solution n'est pas optimale. Pour s'en convaincre, illustrons, dans la figure 5.15, ce que nous aurions obtenu en autorisant une ré-évaluation de D en provenance de C. Nous pouvons constater dans la partie (e) que la partie droite de c_D est améliorée, passant d'un coût 7 à un coût 5. Ensuite, dans la partie (f), cette amélioration est répercutée sur la fonction c_E, ce qui permet de diminuer le coût du trajet de A à E. Cette fois, pour une même date de départ $d = 0$, le coût est de 7 et le chemin associé est $A \rightarrow C \rightarrow D \rightarrow E$.

Ce problème est connu depuis bien longtemps dans le domaine beaucoup plus général de la théorie des graphes. Dans [Eve79] notamment, Even explique que dans le cas de fonctions variables dans le temps, il n'est pas possible d'appliquer des algorithmes de plus court chemin de type Dijkstra [Dij59], dont la propagation d'onde (classique) fait partie.

Partant de ce principe, Orda et Rom ont donc proposé un algorithme de plus court chemin de type Ford [FF58], c'est à dire ne privilégiant pas une case en particulier pour propager les coûts. Dans cet algorithme, appelé SW2 (pour Source Waiting 2), toutes les cases dont le coût a changé entre deux itérations sont utilisées comme source de propagation.

Si cette manière de procéder permet de garantir l'optimalité de la solution, ce n'est pas la seule.

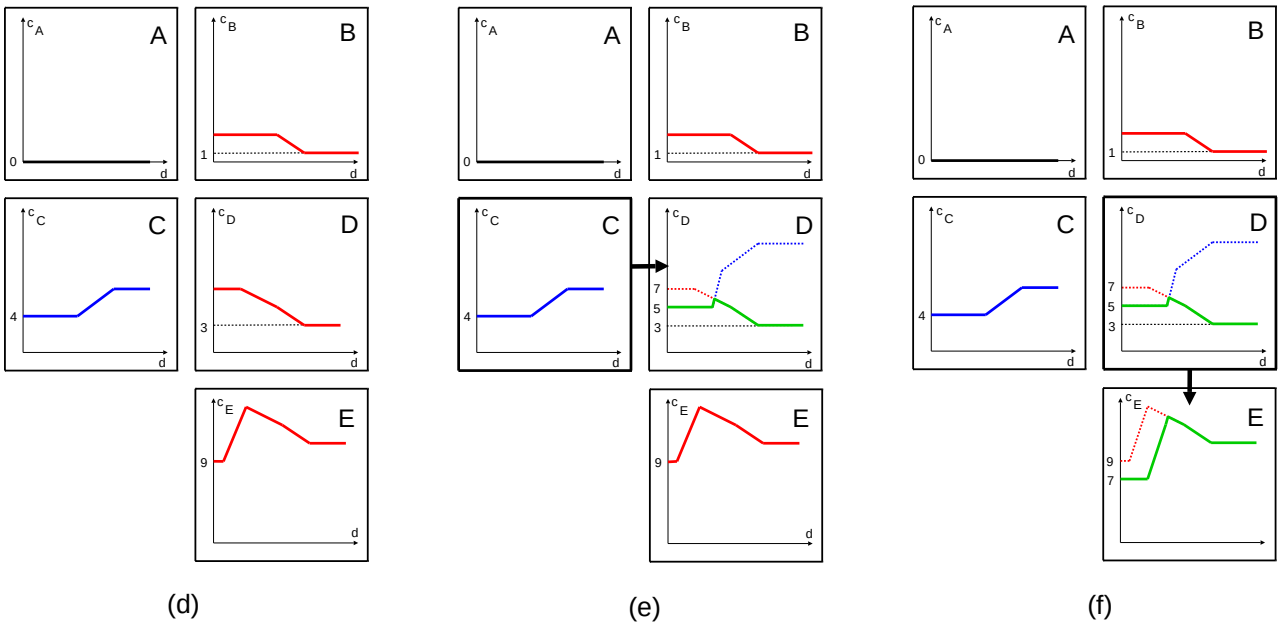

FIG. 5.15 – Propagation d'onde symbolique avec développement multiple des cases.
La figure illustre continue l'exécution de la figure 5.14, en autorisant le re-développement de cases. En (f), la case D est re-développée, permettant d'améliorer le coût d'atteinte de la case E, qui passe de 9 à 7.

Nous pouvons également modifier la propagation d'onde classique en autorisant le re-développement des cases. C'est ce que nous avons fait dans la propagation d'onde symbolique, à la ligne 13. Toute case dont le coût est amélioré peut être ré-injectée dans le front d'onde.

La différence majeure avec l'approche de Orda et Rom est que la notion de tête de front d'onde est conservée, ce qui permet de globalement guider la recherche vers les régions de moindre coût.

Nous allons voir dans la partie expérimentale de ce chapitre que ce guidage est particulièrement efficace pour des instances de notre problème, c'est à dire en présence de courants variables dans le temps. Ceci est dû au fait que l'évolution des courants est généralement progressive dans l'espace et dans le temps, ce qui rend l'apparition d'exemples pathologiques (comme la figure 5.13) plutôt rare et localisée.

4.4 Extraction de la solution optimale

Par définition, la date de départ optimale d^* recherchée, c'est à dire celle minimisant le temps de parcours entre $\widetilde{A}$ et $\widetilde{B}$, est le minimum de la fonction de coût $c_{\widetilde{B}}$, comme illustré dans la figure 5.16a.

Pendant l'exécution de l'algorithme de propagation d'onde, pour chaque case X explorée, le minimum de la fonction de coût c_X est calculé, et stocké dans l'attribut d_X. Ainsi, dès que le front d'onde atteint la case d'arrivée $\widetilde{B}$, la date d^* est connue : il s'agit de la valeur d'attribut $d_{\widetilde{B}}$.

Une fois la date d^* connue, le temps de parcours optimal pour chaque case de la grille est également connu : il est simplement donnée par $c_X(d^*)$. En évaluant ainsi toutes les fonctions de coût c_X à la date d^*, nous revenons dans le cadre "classique" de la propagation d'onde : une grille de

coûts uniquement constituée de grandeurs scalaires. Dans ces conditions, les méthodes classiques de construction de chemin, telles que le hill climbing, sont applicables.

La trajectoire optimale obtenue pour le problème de la figure 5.3 est présentée dans la figure 5.16b. La partie de la trajectoire colorée en bleu correspond à la portion exécutée dans chaque carte des courants. On constate que la date d^* choisie de telle sorte que le robot soit globalement porté par les courants sur chaque portion. En d'autres termes, la valeur de d^* fait en sorte que la trajectoire associée exploite au mieux chacune des deux cartes des courants.

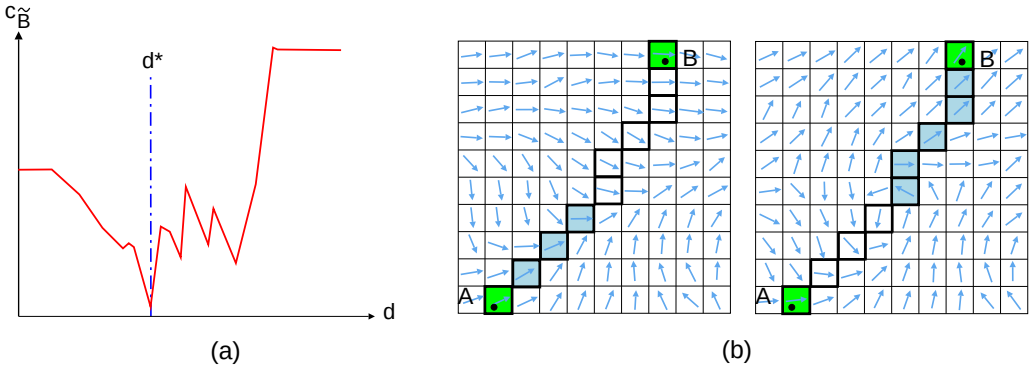

FIG. 5.16 – Solution optimale pour le problème de la figure 5.3
(a) fonction de coût c_B et date de départ optimale ; (b) trajectoire associée. Dans chaque carte des courants, la partie bleue, exécutée par le robot tire au mieux parti des courants pour minimiser le temps de parcours global entre $\widetilde{A}$ et $\widetilde{B}$.

4.5 Complexité temporelle dans le pire des cas

Evaluons tout d'abord le temps de calcul nécessaire à une itération i de l'algorithme :

- Ligne 6 (Récupération de la tête H du front d'onde) : $O(1)$ (temps constant) si une structure de tas binaire (ou équivalent) est utilisée.
- Ligne 8 et 9 (Calcul du coût temporaire c_V^{temp} de V) : $O(k \cdot i)$;
- Ligne 10 (Effectation du coût c_V) : $O(1)$, par référence.
- Ligne 12 (Localisation du minimum de c_V) : $O(k \cdot i)$.
- Ligne 13 (Ajout de P_j dans le front d'onde) : $O(\log(i))$, pour maintenir la structure de tas.
- Ligne 14 (Suppression de H du front d'onde) : $O(1)$.

PREUVE : Nombre d'opérations associées aux lignes 8 et 9 et 12.

Le nombre d'opérations nécessaires tant pour l'évaluation d'une cellule V que pour la localisation de la date optimale est en $O(n_V)$, où n_V est le nombre de morceaux (i.e. de segments de droite) composant la fonction de coût c_V.

Nous savons que c_V est définie par :

$$c_V : d \mapsto \min\{c_V^0, c_V^1\}$$

où c_V^j représente les fonctions de coût associées aux 2 origines concurrentes H_0 et H_1 pour atteindre la même case V.

Nous savons également que chaque fonction c_V^j est donnée par :

$$c_V^j = c_{H_j} + \mathcal{M}'_{H_j,V} \circ (c_{H_j} + d)$$

Comme nous l'avons vu dans la partie 4.1.1, le graphe de la métrique $\mathcal{M}'$ est constitué d'une série de paliers et de phases transitoires. A chaque changement de courant, on trouve (éventuellement) un palier suivi d'une transition. Comme il y a k changements de courants, le graphe de $\mathcal{M}'_{H_j,V}$ est composé d'au plus $2k$ morceaux.

Ainsi, si n_{H_j} était le nombre de morceaux de c_{H_j}, alors c_V^j contient au plus $2k + n_{H_j}$ morceaux.

Il vient donc :

$$n_V \leq 4k + n_{H_0} + n_{H_1}$$

Si nous réitérons ce raisonnement sur les cases H_0 et H_1, nous obtenons :

$$n_H^0 \leq 4k + n_{H_2} + n_{H_3}$$

où H_2 et H_3 sont les 2 origines concurrentes pour H_0.

Et :

$$n_H^1 \leq 4k + n_{H_4} + n_{H_5}$$

où H_4 et H_5 sont les 2 origines concurrentes pour H_1.

D'où :

$$n_V \leq 4k + 4k + 4k + n_{H_2} + n_{H_3} + n_{H_4} + n_{H_5}$$

Et ainsi de suite, sur les cases H_j restantes.

Au bout de la $r$ème application de ce raisonnement, tous les prédécesseurs H_j seront égaux à leur origine commune, c'est à dire la case $\widetilde{A}$. Le nombre de morceaux n_V sera alors borné par :

$$n_V \leq 4k \cdot (2^r + 1)$$

La quantité 2^r représente le nombre de cases couvertes par le front d'onde à l'itération i. A chaque itération, ρ cases sont développées, où ρ désigne la taille du voisinage (8 pour le voisinage de Moore, par exemple). Donc, à la $i$ème itération, le front d'onde a couvert au plus $\rho \cdot i$ cases. Ainsi il vient :

$$2^r \leq \rho \cdot i$$

Et :

$$n_V \leq 4k \cdot (\rho \cdot i + 1)$$

D'où un nombre d'opérations en $O(k \cdot i)$. □

Comme on peut le voir dans la ligne 13 de l'algorithme, à l'itération i, toute case de l'environnement peut être ajoutée au front d'onde si sa fonction de coût a changé, y compris celles qui ont déjà précédemment appartenu au front d'onde (aux itérations $k = [1, i]$). Ainsi, chaque case du front d'onde peut être amenée à être développée i fois.

Soit $N = n \times m$ le nombre de cases dans la grille. Dans le pire des cas, le front d'onde explore l'intégralité des N cases. On en déduit que l'algorithme requiert un temps de calcul en :

$$O\left(\sum_{i=1}^{N} i \cdot (k \cdot i + \log i)\right) = O\left(\sum_{i=1}^{N} k \cdot i^2\right) = O\left(k \cdot \frac{N(N+1)(2N+1)}{6}\right) = O\left(k \cdot N^3\right)$$

A titre comparatif, selon Orda et Rom, l'algorithme SW2 proposé dans [OR90] requiert un temps de calcul en $O(V \cdot E \cdot F)$ où V est le nombre de noeuds dans le réseau, E le nombre d'arcs reliant ces noeuds, et F le nombre d'opérations nécessaires pour effectuer les traitements élémentaires sur les fonctions de coût (composition et comparaison). Dans notre problème, les cases de la grille jouent le rôle des noeuds du réseau. On a donc $V = N$. Ensuite, du fait que les noeuds sont organisés en grille, E est en $O(N)$. Enfin, nous avons vu ci-dessus que F était en $O(k \cdot N)$. Ainsi, avec nos notations, nous obtenons une complexité temporelle dans le pire des cas en $O(k \cdot N^3)$.

Ainsi, en fonction de la taille du problème (c'est à dire les quantités k et N) la propagation d'onde symbolique et l'algorithme SW2 requièrent tous deux un temps de calcul en $O(k \cdot N^3)$ dans le pire des cas[2].

Toutefois, nous allons voir que dans le cadre de notre application (des courants variables dans le temps) la propagation d'onde symbolique évalue beaucoup moins de cases que l'algorithme SW2. Ceci est dû au fait que dans ce contexte spécifique, le guidage du flux calculatoire (par la notion de tête de front d'onde) est particulièrement efficace, et évite des ré-évaluations inutiles. C'est ce que nous montrerons dans la partie expérimentale, section 6.

5 Ajout d'obstacles dans l'environnement

Dans cette partie, nous allons étudier l'impact de l'ajout d'obstacles (fixes et mobiles) dans l'environnement sur l'algorithme et sur la solution obtenue.

[2]Notons que dans ce temps de calcul, l'influence de N (la taille de la grille) est prépondérante, puisque l'ordre de grandeur de N est déjà de 10^4 pour une résolution moyenne, et que cette quantité est élevée au cube. k de l'ordre de 10, et n'agit que comme un facteur multiplicatif.

5.1 Obstacles fixes

Les obstacles fixes sont depuis longtemps gérés dans la propagation d'onde classique, de façon très simple, par un processus d'exclusion des cases faisant partie de ces obstacles. Plus précisément, les cases appartenant aux obstacles fixes sont identifiées par un prétraitement et labellisées. Par la suite, ces cases ne seront pas utilisées pour propager le front d'onde dans l'environnement.

Une possibilité pour mettre en oeuvre ce principe est de fournir l'ensemble O des cases appartenant aux obstacles fixes à l'algorithme de la figure 5.5 (page 133), puis de les éviter lors de l'évaluation du voisinage de la tête du front d'onde. Concrètement, il s'agit de modifier la ligne 7 de l'algorithme :

$$\textbf{pour chaque } V \in \mathcal{V}(H) \textbf{ faire}$$

Par celle-ci :

$$\textbf{pour chaque } V \in (\mathcal{V}(H) \setminus O) \textbf{ faire}$$

De cette manière, les cases de l'ensemble O vont conserver leur coût initial, c'est à dire $+\infty$, et seront naturellement évitées lors de la phase de hill climbing. Bien sûr, d'autres moyens de réalisation sont possibles, comme l'ajout d'un attribut "*interdit*" (booléen) à chaque case, où l'utilisation d'un coût spécial, comme nous l'avons fait dans la collision d'ondes (page 167).

Grâce à cette unique modification, la propagation d'onde symbolique permet à présent de planifier des trajectoires évitant des obstacles fixes. A titre illustratif, nous avons appliqué cette nouvelle version de l'algorithme au problème de la figure 5.3 en présence d'un obstacle fixe rectangulaire, et fourni le résultat dans la figure 5.17.

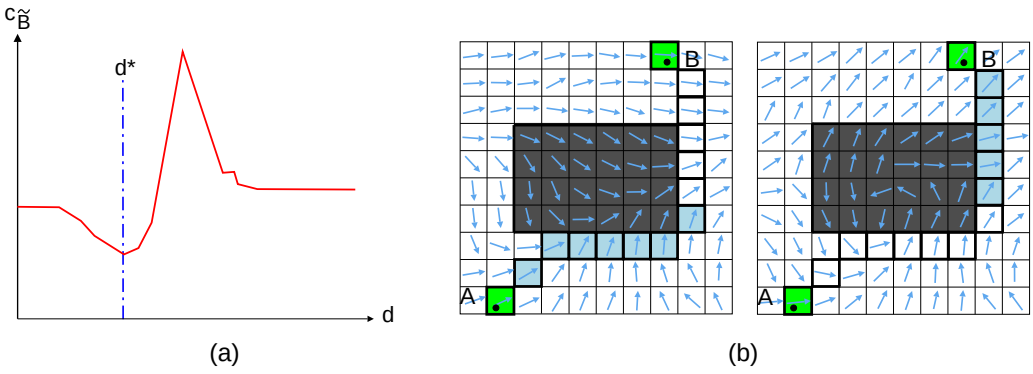

FIG. 5.17 – Impact d'un obstacle fixe sur le problème de la figure 5.3
La figure présente la solution optimale calculée en ajoutant un obstacle fixe rectangulaire (en gris) dans l'environnement de la figure 5.3. (a) fonction de coût c_B (rouge) et date de départ optimale (ligne bleue) ; (b) trajectoire associée. Dans chaque carte des courants, la partie bleue, exécutée par le robot tire au mieux parti des courants pour minimiser le temps de parcours global entre $\widetilde{A}$ et $\widetilde{B}$.

5.2 Obstacles mobiles

Pour garantir l'évitement d'obstacles mobiles (dont nous connaissons la trajectoire) une possibilité est de s'inspirer de la démarche réalisée par Kimmel *et al* dans [KKB98], consistant à intégrer dans les coûts les éventuels ralentissements dus aux obstacles mobiles. Nous allons appliquer ici cette démarche à la propagation d'onde symbolique.

La trajectoire des obstacles étant connue, il est très aisé de calculer pour chaque case X de la grille l'intervalle des *dates interdites*, noté I_X, pendant lesquelles X est occupée par un obstacle. On peut par exemple recourir à la programmation par contraintes, en utilisant les contraintes proposées dans le chapitre 7 (page 186 et suivantes).

L'intervalle I_X est une union de sous-intervalles $I_X^1 \cup I_X^2 \cup ... I_X^o$. Chaque sous-intervalle I_X^i correspond à la $i$ème occupation de X par un obstacle mobile (potentiellement le même). o est le nombre total d'occupations.

Si le robot arrive à la case X pendant un sous-intervalle $I_X^i = [\underline{t_i}, \overline{t_i}]$, alors celui-ci devra ralentir pour n'arriver qu'à la date $\overline{t_i}$, c'est à dire quand X sera libérée. Nous pouvons modéliser ce ralentissement par une fonction r_X définie comme suit :

$$r_X(t) = \begin{cases} \overline{t_i} - t \text{ si} & \exists i \ / \ t \in [\underline{t_i}, \overline{t_i}] \\ 0 & \text{sinon} \end{cases} \tag{5.4}$$

où t représente la date d'arrivée en X.

Le graphe de r_X contient des "pics" dans les intervalles où la case X est occupée par un obstacle, comme illustré dans la figure 5.18. L'interprétation de ces pics est la suivante : le ralentissement est maximal en début d'occupation de la case par l'obstacle (à $t = \underline{t_i}$) et diminue progressivement au cours du temps, jusqu'à s'annuler quand l'obstacle quitte la case (à $t = \overline{t_i}$).

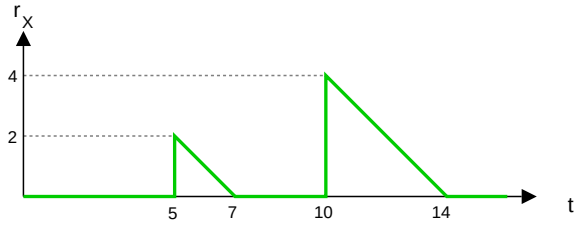

FIG. 5.18 – Fonction de ralentissement r_X
La figure illustre la fonction de ralentissement associée à une case X occupée par un obstacle dans les intervalles $I_X^1 = [5, 7]$ puis $I_X^2 = [10, 14]$.

Grâce à la fonction r_X, nous pouvons simplement refléter l'impact des obstacles mobiles sur les fonctions de coût, et donc sur la trajectoire du robot. Il faut pour cela introduire cette fonction dans l'opération d'évaluation de la propagation d'onde symbolique.

Dans l'algorithme de la figure 5.5 (page 133), le coût d'une nouvelle case V, atteinte à partir d'une case H était évalué comme suit :

$$c_V^{temp} \leftarrow c_H + \mathcal{M}'_{H,V} \circ (c_H + d), d \in W$$

Pour tenir compte des obstacles mobiles, il faut rectifier le coût c_V^{temp} de V, de façon tout à fait analogue, grâce à la fonction de ralentissement r_V :

$$c_V^{temp'} \leftarrow c_V^{temp} + r_V \circ (c_V^{temp} + d), d \in W$$

Cette ligne devra être insérée à la ligne 9 de l'algorithme. Dans la ligne suivante, "$c_V \leftarrow \min\{c_V, c_V^{temp}\}$", le coût c_V^{temp} devra être remplacé par le coût rectifié $c_V^{temp'}$.

La figure 5.19 illustre l'impact de cette modification sur la fonction de coût c_V^{temp} de la figure 5.10 (page 139).

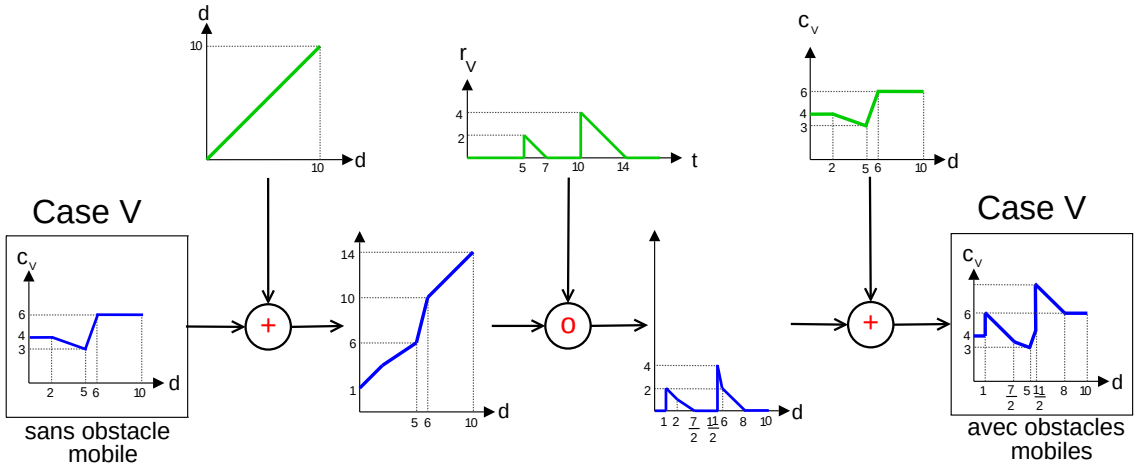

FIG. 5.19 – Impact de la fonction de ralentissement sur la fonction c_V de la figure 5.10

On peut constater que la fonction r_V agit comme un "masque" sur la fonction c_V^{temp} : on retrouve les pics de r_V dans le graphe de $c_V^{temp'}$ partout où le robot doit ralentir. De façon plus formelle, nous avons :

$$c_V^{temp} : d \rightarrow \begin{cases} 4 & \text{si} \quad d \in [0,2] \\ -1/3 \cdot d + 14/3 & \text{si} \quad d \in [2,5] \\ 3 \cdot d - 12 & \text{si} \quad d \in [5,6] \\ 6 & \text{si} \quad d \in [6,10] \end{cases}$$

D'où :

$$c_V^{temp} + d : d \rightarrow \begin{cases} d + 4 & \text{si} \quad d \in [0,2] \\ 2/3 \cdot d + 14/3 & \text{si} \quad d \in [2,5] \\ 4 \cdot d - 12 & \text{si} \quad d \in [5,6] \\ d + 6 & \text{si} \quad d \in [6,10] \end{cases}$$

La fonction de ralentissement r_V de la figure 5.18 a pour expression :

$$r_V : t \rightarrow \begin{cases} 0 & \text{si} \quad t \in [0,5[\\ 7-t & \text{si} \quad t \in [5,7] \\ 0 & \text{si} \quad t \in [7,10[\\ 14-t & \text{si} \quad t \in [10,14] \\ t & \text{si} \quad t \in [14,+\infty[\end{cases}$$

La composition de r_V avec $c_V^{temp} + d$ donne ensuite :

$$r_V \circ (c_V^{temp} + d) : d \rightarrow \begin{cases} 0 & \text{si} \quad d \in [0,1[\\ -d+3 & \text{si} \quad d \in [1,2] \\ -2/3 \cdot d + 7/3 & \text{si} \quad d \in [2,7/2] \\ 0 & \text{si} \quad d \in [7/2,11/2[\\ -4 \cdot d + 26 & \text{si} \quad d \in [11/2,6] \\ -d+8 & \text{si} \quad d \in [6,8] \\ 0 & \text{si} \quad d \in [8,10] \end{cases}$$

Enfin, en additionnant cette dernière fonction avec c_V^{temp}, on obtient finalement $c_V^{temp'}$:

$$c_V^{temp'} : d \rightarrow \begin{cases} 4 & \text{si} \quad d \in [0,1[\\ -d+7 & \text{si} \quad d \in [1,7/2] \\ -1/3 \cdot d + 14/3 & \text{si} \quad d \in [7/2,5] \\ 3 \cdot d - 12 & \text{si} \quad d \in [5,11/2[\\ -d+14 & \text{si} \quad d \in [11/2,8] \\ 6 & \text{si} \quad d \in [8,10] \end{cases}$$

Grâce aux quelques modifications proposées, la propagation d'onde symbolique permet à présent de planifier des trajectoires évitant des obstacles mobiles. A titre illustratif, nous avons appliqué cette nouvelle version de l'algorithme au problème de la figure 5.3 en présence d'un obstacle mobile rectangulaire, et fourni le résultat dans la figure 5.20.

On constate que l'ajout de l'obstacle mobile introduit un "pic" dans la fonction de coût, là où était précédemment localisée la date optimale (sans obstacle mobile). La date de départ optimale est donc différée, de sorte de limiter au plus le surcoût dû au contournement de l'obstacle.

De manière plus générale, la propagation d'onde symbolique ainsi modifiée permet d'éviter les obstacles mobiles en effectuant un compromis entre les éléments suivants :

1. Décalage de la date de départ,
2. Contournement spatial de l'obstacle (détour),
3. Variation de la vitesse du robot (ici, ralentissement uniquement).

Ainsi, la propagation d'onde symbolique semble constituer une alternative séduisante aux techniques de modulation de vitesse, notamment celle que nous proposons dans le chapitre 7. En effet, les techniques de modulation de vitesse n'agissent, par nature, que sur le 3ème levier de la liste ci-dessus, ce qui peut aboutir aux problèmes de sous-optimalité et d'incomplétude mis en avant dans l'état de l'art (page 54).

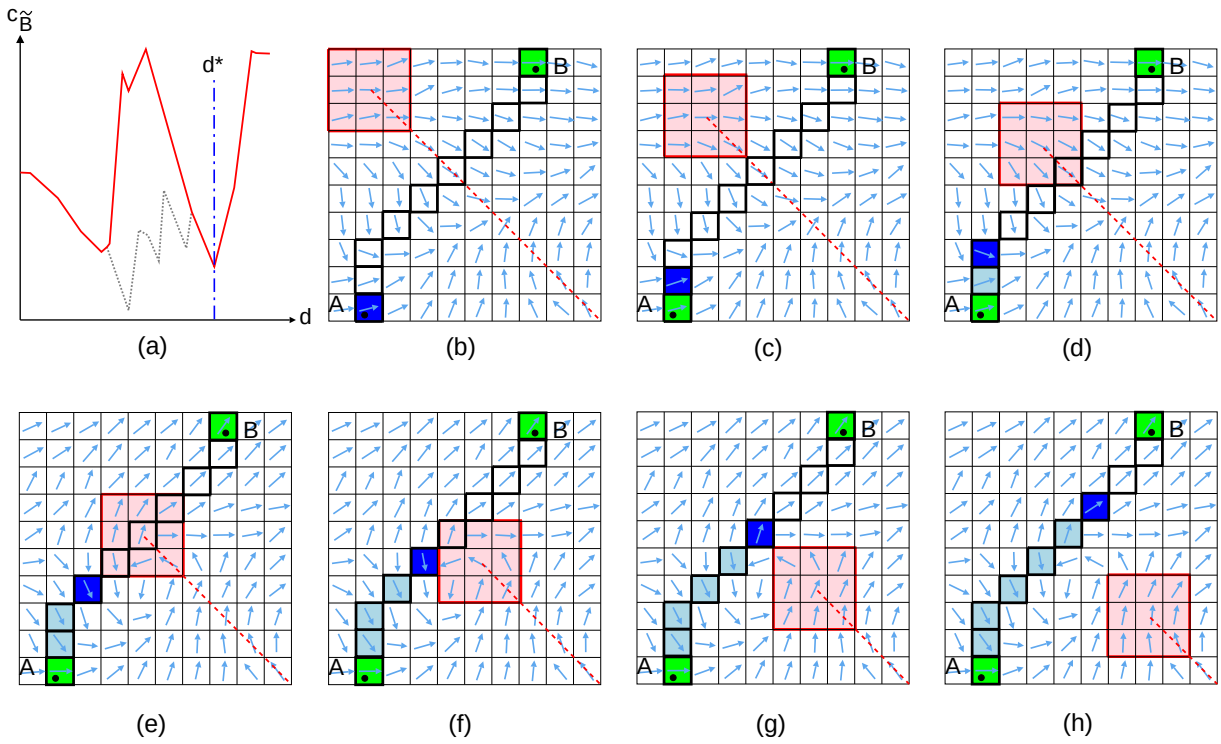

FIG. 5.20 – Impact d'un obstacle mobile sur le problème de la figure 5.3
La figure présente la solution optimale calculée en ajoutant un obstacle mobile rectangulaire (en rouge) dans l'environnement de la figure 5.3. (a) fonction de coût c_B (gris : fonction de coût sans obstacle mobile ; rouge : avec obstacle mobile) et date de départ optimale (ligne bleue) ; de (b) à (f) trajectoire optimale (rouge : obstacle mobile, bleu foncé : position courante du robot ; bleu clair = portion de trajectoire déjà effectuée).

L'ajout des leviers 1 et 2 pourrait permettre de limiter voire supprimer ces problèmes, au prix d'un temps de calcul plus long. Une étude plus approfondie, notamment une comparaison de performances (similaire à celle effectuée ci-dessous) serait intéressante.

6 Résultats expérimentaux

Le but de cette section est d'apprécier les gains de performances apportés par propagation d'onde symbolique par rapport à l'algorithme SW2 d'Orda et Rom. Pour cela, nous avons suivi le protocole expérimental suivant :

- Nous avons utilisé les mêmes cartes des vents que dans le chapitre 4 : une base de 100 cartes collectées sur Metéo France [MF]. Chaque carte a une validité de 6 heures.
- Sur la base de ces cartes, chaque cas de test a été généré de la façon suivante :

  1. Les points de départ et d'arrivée A et B ont été placés aléatoirement dans l'environnement ;
  2. L'indice $p \in [1, 100]$ de la première carte des vents dans la base, ainsi que le nombre de changement de courants $k \in [1, 5]$ ont également été choisis de façon aléatoire[3].

[3] Avec toutefois la contrainte que $p + k \in [1, 100]$, pour conserver la cohérence temporelle des données. Un bouclage sur la carte numéro 1 au dela de 100 n'aurait pas de sens physique.

Les valeurs de p et k permettent de connaître intégralement l'évolution de l'environnement au cours de la mission. Par exemple, si $p = 56$ et $k = 2$ la mission aura une durée $T = 6 \times 3 = 18h$, et l'enchaînement des courants sera le suivant : courants de la carte n°56 de $t = 0$ à $t = 6$, puis carte n°57 de $t = 6$ à $t = 12$, et ainsi de suite. Cette évolution est illustrée dans la figure 5.21.

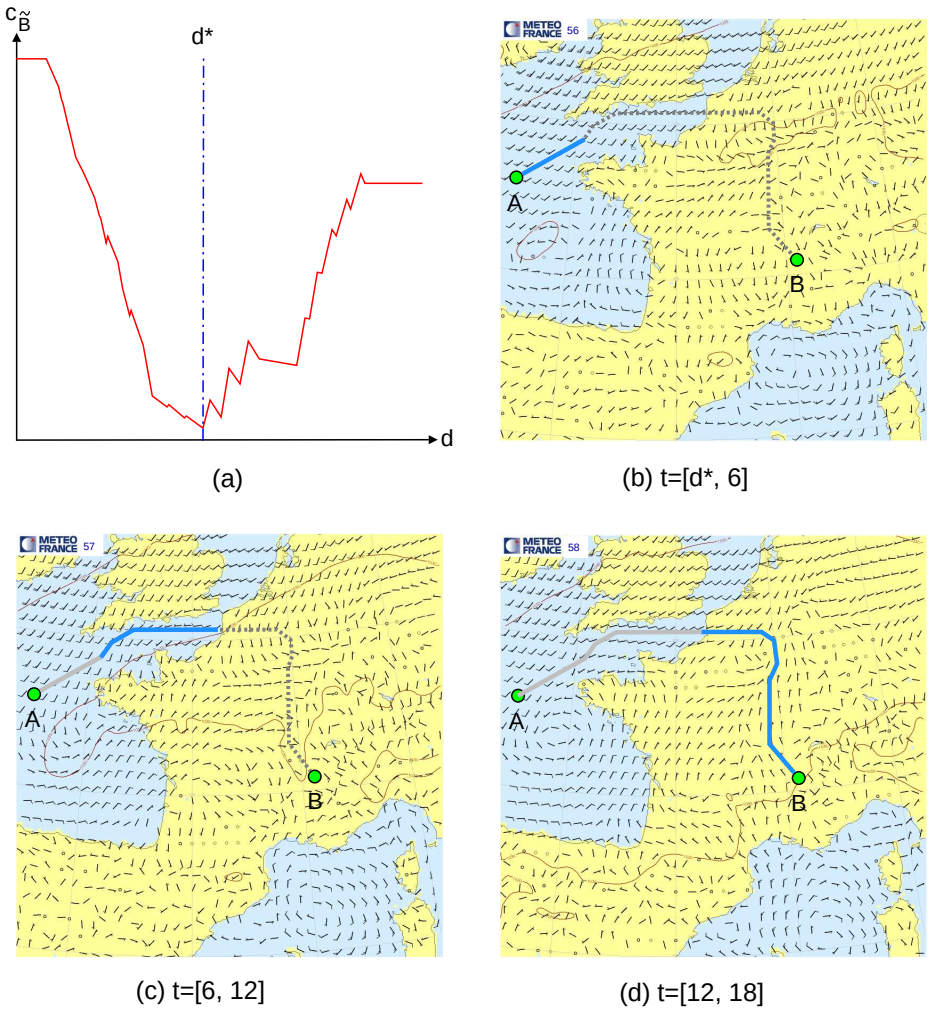

FIG. 5.21 – Un cas de test et la solution obtenue par la propagation d'onde symbolique (a) fonction de coût c_B (rouge) et date de départ optimale (ligne bleue) ; De (b) à (d) : trajectoire associée (gris = partie déjà effectuée, bleu = partie en cours d'exécution, pointillés = partie restant à effectuer).

Dans ce contexte, nous avons comparé le nombre de cases développées par la propagation d'onde symbolique et l'algorithme SW2, pour des grilles de N cases. Nous savons que dans le pire des cas, le nombre de cases développées par chacun des algorithmes est en $O(N^2)$, c'est à dire que chaque case de la grille est développée $O(N)$ fois.

Nous proposons d'évaluer dans quelle mesure le nombre observé de cases développées s'éloigne de ce pire des cas dans le contexte particulier de nos tests, c'est à dire quand l'évolution des coûts dans le temps est directement lié à des cartes de courant réalistes.

Pour ce faire, nous avons appliqué les deux approches 500 fois pour chaque valeur de N, et calculé le nombre moyen de cases développées. Les résultats sont illustrés dans les figures 5.22 et 5.23 ci-après.

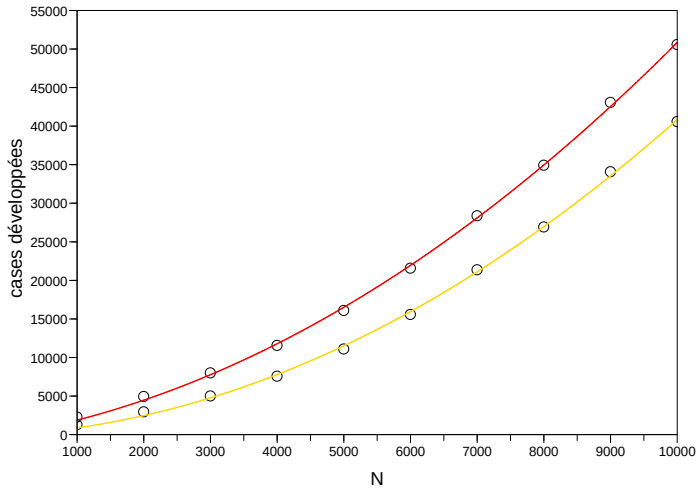

FIG. 5.22 – Nombre moyen de cases développées par l'algorithme SW2

En rouge : tous développements compris ; en jaune : re-développements uniquement.

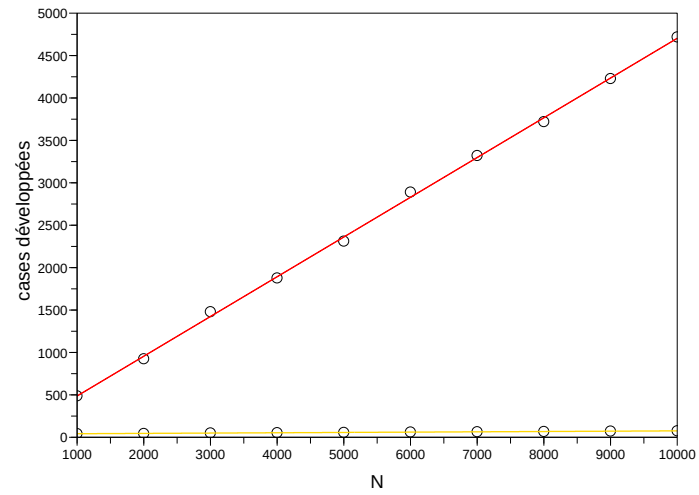

FIG. 5.23 – Nombre moyen de cases développées par la propagation d'onde symbolique

En rouge : tous développements compris ; en jaune : re-développements uniquement.

En effectuant des régressions sur les données, nous avons constaté que le nombre de cases développées était en $O(N)$ pour la propagation d'onde symbolique et en $O(N^2)$ pour l'algorithme SW2 (avec un coefficient de détermination $R^2 > 0.999$). Les performances des deux algorithmes sont donc radicalement différentes sur nos jeux de tests.

Pour donner un ordre de grandeur, pour $N = 10000$ (ce qui correspond à une grille de taille 100×100, résolution raisonnable pour des applications en robotique mobile), l'algorithme SW2 est environ 10 fois plus lent que la propagation symbolique, et il semblerait que cet écart augmente avec N. En effet, sur la base des régressions effectuées, le rapport entre le nombre de cases développées par l'algorithme SW2 et par la propagation d'onde symbolique est en $O(N)$.

Cette différence dans les performances est due à la fois à la nature des données du problème et à la nature des algorithmes.

Tout d'abord, les situations "pathologiques" entraînant des re-développements (comme celui explicité page 143), c'est à dire de brusques changements dans l'expression des fonctions de coûts (une quasi-inversion), sont très peu fréquentes dans nos cartes. En effet, un changement brutal au niveau les coûts suppose un changement brutal dans les courants (dans leur direction et/ou dans leur intensité).

Or, sur l'ensemble de nos cartes, l'évolution des courants est globalement progressive, tant dans l'espace que dans le temps. Nous avons notamment expliqué dans l'introduction de ce chapitre qu'une inversion des courants était possible, mais sur une longue durée, et en passant par une série d'états intermédiaires. Une inversion soudaine de courant, bien que possible, est plutôt rare et localisée.

Ensuite, les deux algorithmes n'ont pas le même comportement vis à vis de ces re-développements. Supposons qu'une cellule X ait déjà été développée. Celle-ci a donné naissance à des "descendants" $X_1, ..., X_j$ (les voisins de X, puis les voisins des voisins, etc.). Si le coût c_X de cette case X est amélioré par la suite, les deux algorithmes ne réagissent pas du tout de la même manière :

- Dans la propagation d'onde symbolique, la case X va être ré-injectée dans le front d'onde pour être de nouveau développée. Par construction, le minimum de la fonction c_X est strictement inférieur au minimum des fonctions $c_{X_1}, ..., c_{X_j}$. Comme les cases du front d'onde sont triées par minima de leurs fonctions de coût, la case X va être re-developpée avant ses descendants $X_1, ..., X_j$. Ainsi, l'impact de l'amélioration du coût c_X sera répercuté sur les fonctions $c_{X_1}, ..., c_{X_j}$, avant que celles-ci n'aient eu le temps de donner naissance à de nouveaux descendants. En d'autres termes, toute une partie du front d'onde (les descendants de X) est "gelée" jusqu'à ce que les nouvelles informations sur c_X l'ait atteinte.

- Dans SW2, toutes les cases ayant changé entre deux itérations de l'algorithme sont sources de propagation d'information. Ainsi, si à l'itération i la case X est modifiée, celle-ci sera re-développée à l'itération $i+1$, mais il en sera également de même pour les derniers descendants de X (évalués à l'itération i). Ainsi, à partir de ces descendants, toute une partie de la grille va être évaluée avec des coûts que nous savons sous-optimaux. L'ensemble des ces cases sera dont inévitablement ré-évalué par la suite.

En résumé, l'impact du re-développement d'une case est confiné à un certain rayon autour de cette case pour la propagation d'onde symbolique, alors qu'il concerne toute une partie de la grille (donc un nombre de cases en $O(N)$) pour l'algorithme SW2. D'après les travaux de Vladimirsky [Vla03], il semble possible d'estimer ce rayon en fonction du rapport entre la vitesse du drone et la vitesse des courants (appelé coefficient "d'anisotropisme").

Il serait intéressant de mener des études complémentaires pour tenter de mieux comprendre dans quelles conditions apparaissent des re-développements (différence de norme ou de direction entre deux vecteurs-vitesses de courant successifs), mais aussi pour quantifier le rayon évoqué précédemment dans notre cadre applicatif.

7 Conclusion

Nous avons introduit dans ce chapitre une nouvelle variante de la propagation d'onde, appelée *propagation d'onde symbolique*, capable de déterminer, étant donné l'évolution des courants dans le temps, la date de départ optimale d^* (et la trajectoire optimale associée) minimisant le temps de parcours entre deux points de l'environnement.

Nous avons montré expérimentalement que la propagation d'onde symbolique pouvait permettre de substantiels gains de performances comparée à l'algorithme SW2, dans le cas particulier de missions standard de longue durée pour drones. Il serait intéressant d'étudier dans quelle mesure ce résultat est généralisable.

Nous avons également montré que cette approche était facilement modifiable, afin de permettre l'évitement d'obstacles dans l'environnement, fixes comme mobiles.

L'ensemble de ces résultats a été publié dans les actes de la conférence planSIG'08 (Workshop of the UK Planning and Scheduling Special Interest Group) ainsi que dans ceux de la conférence ICRA'09 (International Conference on Robotics and Automation). Le lecteur interessé pourra respectivement consulter [STR08b] et [STR09].

Il est important de mentionner que la propagation d'onde symbolique ne permet pas seulement d'obtenir la trajectoire optimale associée à la date d^*, mais toutes les trajectoires optimales possibles, pour n'importe quelle date de départ $\underline{d}$.

Pour cela, il suffit d'appliquer exactement la même démarche que celle présentée dans la partie 4.4 (page 145) : évaluer toutes les fonctions de coût c_X à $d = \underline{d}$ puis appliquer l'algorithme de hill climbing.

Ainsi, sans changement dans les données du problème, la construction de la trajectoire optimale associée à une date quelconque $\underline{d}$ est quasi-immédiate, puisque la propagation d'onde n'a pas à être re-effectuée. Le temps requis par le hill climbing est quant à lui négligeable.

Cette propriété permet à l'utilisateur de rapidement tester des dates de départ alternatives. Si le robot, pour une raison technique par exemple, ne peut partir à $d = d^*$, mais à $d = \underline{d}$, alors l'utilisateur pourra apprécier très rapidement l'impact de ce changement sur le coût mais aussi sur la chemin associé.

Dans la figure 5.24 notamment, on peut constater qu'un départ à $d = \underline{d}$ change le côté de contournement de l'obstacle fixe.

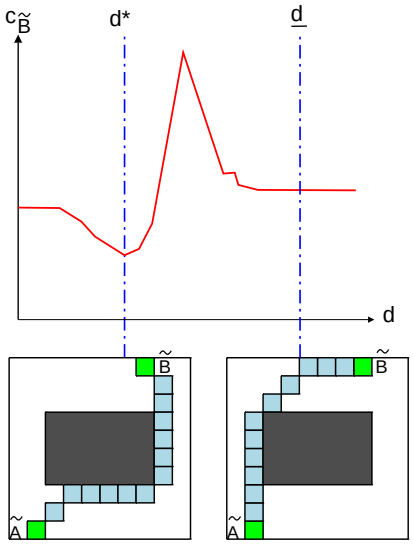

FIG. 5.24 – Impact d'un départ différé sur l'exemple de la figure 5.17
Un départ à $d = \underline{d}$ au lieu de $d = d^*$ peut significativement modifier l'allure de la trajectoire optimale (en bleu). Ici, le côté de contournement de l'obstacle a changé. La propagation d'onde symbolique permet d'apprécier ce changement de façon quasi-instantanée.

A l'issue de ce chapitre, nous disposons de donc deux variantes de la propagation d'onde : la propagation d'onde *coulissante*, capable de traiter les courants forts (introduite dans le chapitre précédent), et la propagation d'onde *symbolique*, capable de traiter les courants variables dans le temps.

La continuité naturelle de ces travaux serait donc de fusionner les résultats de ces deux variantes, pour traiter ces deux problématiques de manière unifiée. Cette fusion peut être réalisée de deux manières différentes :

- Introduire la date de départ d dans la propagation d'onde coulissante,
- Introduire la notion de curseur (cf. page 112) dans la propagation d'onde symbolique

Ces deux modifications reviennent à étendre la propagation d'onde coulissante dans un espace-temps 3D $(\vec{x}, \vec{y}, \vec{t})$. Cette extension est présentée dans la partie "perspectives" de cette thèse, page 208.

Troisième partie

Autres contributions

Introduction

Cette partie présente deux autres contributions un peu plus mineures dans de domaine de la planification de trajectoire.

Le chapitre 6 introduit le concept de *collision d'ondes* [ST07], qui permet la planification de multiples chemins (entre plusieurs points à visiter) de façon efficace. Nous montrons que ce concept peut diviser le temps de calcul jusqu'à 4, comparé à l'utilisation répétée (pour chaque chemin) de la propagation d'onde.

Le chapitre 7 propose d'utiliser la *programmation par contraintes* pour moduler la vitesse du robot [STR07]. La modulation de vitesse consiste à générer un profil de vitesse que doit suivre le drone sur un chemin prédéfini. Ce profil a pour but d'éviter les obstacles mobiles (en accélérant ou en ralentissant). La programmation par contraintes aboutit à une technique de modulation de vitesse robuste aux changements des caractéristiques du problème. En effet les changements dans la formulation du problème peuvent être assez aisément pris en compte par l'ajout, la modification ou la suppression de contraintes dans le modèle.

Chapitre 6

Planification de trajectoire pour missions multi-sites

Sommaire

1 Introduction

Jusqu'à présent, nous avons considéré que les robots mobiles étaient utilisés pour effectuer des missions assez simples, consistant à relier deux sites A et B dans son environnement. Mais en règle générale, les missions à réaliser impliquent bien plus que deux sites.

Notamment, les missions de reconnaissance, de surveillance ou encore de sauvetage peuvent amener à visiter plusieurs dizaines de sites, dans un ordre non connu à l'avance. C'est au cours du processus de planification que cet ordre sera fixé, afin de minimiser un coût c.

A titre illustratif (et par souci de cohérence avec les autres chapitres) nous supposerons que c représente le temps de parcours du robot. Cependant, d'autres coûts sont utilisables selon le domaine d'application : consommation d'énergie, probabilité d'être détecté, etc.

Dans ce contexte multi-sites, la planification de trajectoire se révèle donc beaucoup plus complexe qu'auparavant, puisqu'elle consiste à résoudre deux sous-problèmes de niveau d'abstraction différents :

1. Une planification *microscopique* : calculer, pour chaque couple de sites (S_i, S_j), le tronçon de trajectoire permettant de relier S_i et S_j en minimisant c et en tenant compte de l'environnement.
2. Une planification *macroscopique* : déterminer l'ordre de visite des sites S_i minimisant c. Autrement dit, il s'agit de trouver la succession de tronçons (calculés au niveau microscopique) qui, placés bout à bout, permettent de construire une trajectoire globale passant par tous les sites en un coût c minimal.

Le deuxième sous-problème, que nous avons qualifié de planification macroscopique, est une instance d'un problème très connu en intelligence artificielle, appelé *voyageur de commerce*. Nous ne traiterons pas ce problème ici, qui constitue à lui seul une voie de recherche très active.

Dans ce chapitre, nous nous attachons donc à résoudre le premier sous-problème, appelé planification microscopique. Nous allons proposer dans la section 3 un concept appelé *collision d'ondes*, permettant de planifier *simultanément* tous les tronçons de trajectoire entre les sites. Nous montrerons que ce concept permet de diviser les temps de calcul par près de 3.5, par rapport à une planification séquentielle des tronçons.

2 Formulation du problème

Un robot, dans un environnement planaire, doit visiter un ensemble de n sites $\{S_i\}_{i\in[1,n]}$ en un temps minimal, en présence d'obstacles fixes, tel qu'il est illustré dans la figure 6.1a.

Les résultats de ce chapitre étant antérieurs à ceux des chapitres 4 et 5, nous supposerons que l'environnement ne contient pas de courants. Toutefois, nous verrons dans la discussion de ce chapitre dans quelle mesure les résultats de ce chapitre sont généralisables à la présence de courants.

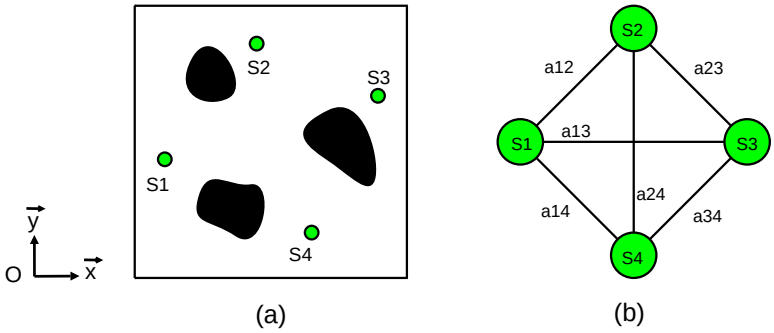

FIG. 6.1 – Un problème de planification de trajectoires muti-sites
(a) un problème contenant 4 sites (points verts) et 3 obstacles fixes (surfaces noires) ; (b) le graphe de connectivité associé.

Le robot est représenté par un objet ponctuel évoluant dans un espace Euclidien $\mathcal{C}$ de dimension 2, de repère orthonormé $\mathcal{R} = (O, \vec{x}, \vec{y})$. Les obstacles fixes sont quant à eux représentés par des surfaces fermées de forme quelconque.

L'ensemble des mouvements possibles entre les sites est modélisé par un graphe, appelé *graphe de connectivité*, représenté dans la figure 6.1b.

Formellement parlant, il s'agit d'un couple $(\mathcal{N}, \mathcal{A})$, avec :

- $\mathcal{N} = \{S_i\}_{i \in [1,n]}$ l'ensemble des noeuds, un noeud étant associé à chaque site à visiter ;
- $\mathcal{A} = \{a_{ij} = (i, j, c_{ij})\}$ l'ensemble des arcs, chaque arc a_{ij} étant associé au tronçon trajectoire $T_{i,j}$ entre S_i et S_j. La quantité c_{ij} représente le coût, en termes de temps de parcours, de ce tronçon.

Ce graphe est dit *complet* car il contient les $n(n-1)/2$ arcs possibles reliant ses n noeuds.

Notre problème consiste à calculer tous les tronçons de trajectoire $T_{i,j}$ possibles entre les sites, c'est à dire un tronçon par arc a_{ij} du graphe de connectivité. Ces tronçons doivent éviter les obstacles fixes et minimiser le temps de parcours du robot.

3 La collision d'ondes

3.1 Principe

Comme nous l'avons vu précédemment, la propagation d'onde, initialement proposé par Dorst et Trovato [DT88], est le seul algorithme garantissant l'absence de minima locaux et un temps de calcul raisonnable (polynomial). Nous allons donc nous baser sur cet algorithme pour planifier les tronçons de trajectoire entre les sites.

Si nous planifions de façon séquentielle les $n(n-1)/2$ tronçons possibles entre les n sites, il faudra au moins $n-1$ propagations d'ondes successives :

- 1 propagation d'onde d'origine S_1, pour les n tronçons $T_{1,2}, T_{1,3}, T_{1,4}, \ldots, T_{1,n}$
- 1 propagation d'onde d'origine S_2, pour les $n-1$ tronçons $T_{2,3}, T_{2,4}, \ldots, T_{2,n}$
- 1 propagation d'onde d'origine S_3, pour les $n-2$ tronçons $T_{3,4}, \ldots, T_{3,n}$
- ...
- 1 propagation d'onde d'origine S_{n-1}, pour le dernier tronçon $T_{n-1,n}$

A la $i$ème propagation d'onde, tous les tronçons atteignant S_i en provenance des sites S_{i-1}, S_{i-2}, ..., S_1 ont déjà été calculés au cours des $i-1$ propagations précédentes. Ainsi, la propagation d'origine S_i ne sert qu'à évaluer les tronçons restants. Ceci est illustré dans la figure 6.2, pour l'exemple de la figure 6.1a.

Cette démarche séquentielle conduit à des calculs inutiles, car elle considère les différentes propagations d'ondes comme indépendantes. Sous cette hypothèse, une même case pourra être évaluée plusieurs fois, par plusieurs fronts d'ondes différents. Suite aux 3 propagations d'ondes de la figure 6.2, par exemple, 28% des cases ont été évaluées 3 fois, et 38% l'ont été 2 fois. Ces régions sont explicitées dans la figure 6.3. Dans le cas général, où l'on a $n-1$ propagations d'ondes, on pourra donc être amené à évaluer $n-1$ fois la même case !

Pour limiter cette redondance d'information, nous proposons d'effectuer les propagations d'ondes simultanément plutôt que séquentiellement. Pour cela, nous nous sommes inspirés d'un concept utilisé dans la branche des Rapid Random Tree (RRT).

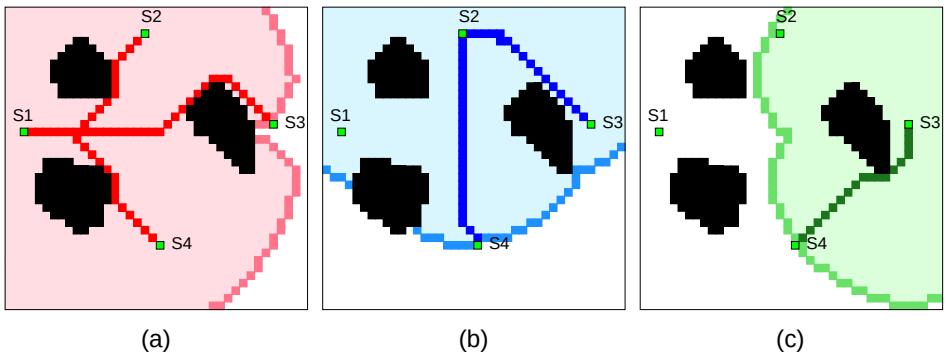

FIG. 6.2 – Planification des tronçons par propagations d'ondes successives
Pour $n = 4$ sites, il y a $n(n-1)/2 = 6$ tronçons possibles. (a) la propagation d'onde d'origine S_1 permet d'en calculer 3 : $T_{1,2}$, $T_{1,3}$ et $T_{1,4}$; (b) la propagation d'onde d'origine S_2 permet d'en calculer 2 : $T_{2,3}$, $T_{2,4}$; (c) Enfin, la propagation d'onde d'origine S_3 permet de calculer le dernier : $T_{3,4}$.

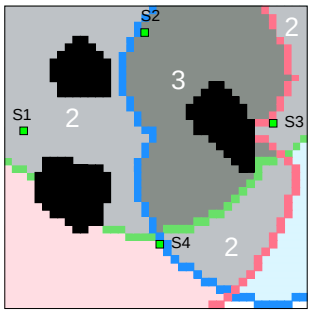

FIG. 6.3 – Redondance de calcul dans le cas de propagations d'ondes successives
Les zones illustrées en gris ont été évaluées 2 voire 3 fois au cours des propagations d'ondes de la figure 6.2, car celles-ci sont réalisées de façon indépendante.

Nous rappelons qu'un RRT [LaV98] est un arbre de recherche qui est progressivement étendu dans l'environnement, à partir du point de départ, en utilisant des échantillons aléatoires. Cette extension s'arrête quand l'arbre est suffisamment proche du point d'arrivée[1].

Pour accélérer ce processus, les auteurs ont proposé le concept de RRT-connect [KL00], illustré dans la figure 6.4a. Il consiste à développer simultanément deux arbres de recherche : l'un à partir du point de départ, l'autre à partir du point d'arrivée, jusqu'à ce qu'ils soient suffisamment proches. Ils sont alors "connectés", puis les deux parties du chemin sont construites à partir de ce point de connexion, en remontant dans les deux arbres.

Nous proposons de transposer cette idée à la propagation d'onde : pour planifier un chemin entre deux sites S_i et S_j, deux fronts d'ondes vont être simultanément développés à partir de ces points, jusqu'à ce qu'une collision soit détectée (c'est à dire jusqu'à ce qu'au moins une case est commune aux deux fronts). Ensuite, les deux parties du chemin sont construites à partir du point de collision, en effectuant deux descentes du gradient (plus précisément, son équivalent discret, le *hill climbing*) dans les deux surfaces recouvertes par les fronts d'ondes. Cette transposition est illustrée dans la figure 6.4.

[1]Pour plus de détails sur l'approche RRT, le lecteur pourra se reporter à la page 31.

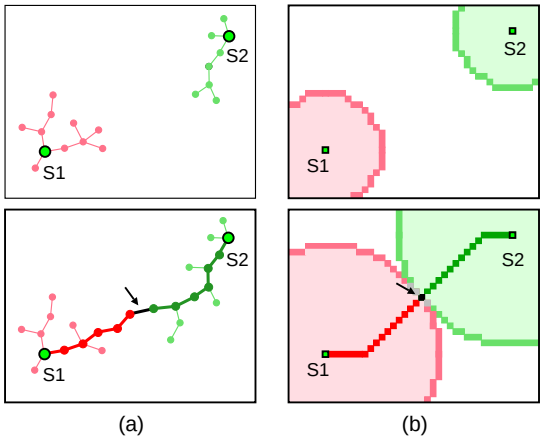

FIG. 6.4 – Analogie entre RRT-connect et la collision d'ondes
(a) RRT-connect et (b) la collision d'ondes avec 2 sites. Dans les deux cas, le processus d'expansion est interrompu dès qu'une collision est détectée (au niveau de la flèche noire). Les deux parties du chemin sont construites à partir du point de collision jusqu'à chaque site.

Mais en plus de cette transposition, nous allons étendre le concept de 2 à n sites. Cette extension est appelée la *collision d'ondes*. Elle consiste à développer simultanément les n fronts d'ondes, et à construire un chemin à chaque collision entre deux fronts d'ondes. Au bout de $n(n-1)/2$ collisions, l'ensemble des tronçons reliant les n sites seront ainsi calculés.

3.2 Algorithme

3.2.1 La phase de discrétisation

Comme nous l'avons expliqué dans le chapitre 1 (page 35), l'algorithme de la propagation d'onde nécessite de discrétiser l'environnement en un ensemble de cellules, généralement des cellules régulières appelées cases. Ces cases forment une grille de dimensions $L \times C$, où L désigne le nombre de lignes, et C le nombre de colonnes.

Dans ce contexte, chaque case contenant un site S_i est notée $\widetilde{S}_i$. Les cases $\widetilde{S}_i$ sont les représentantes des sites S_i dans le monde discrétisé. Notons qu'elles introduisent une incertitude sur la visite des sites, car elles remplacent des sites initialement ponctuels par des surfaces rectangulaires, dont la taille peut être importante si la résolution de la grille est grossière.

Par ailleurs, à chaque cellule X de la grille sont associés n coûts, notés $cout_i(X)$, $i \in [1, n]$. La quantité $cout_i(X)$ représente le coût (ici le temps de parcours) nécessaire pour atteindre $\widetilde{S}_i$ à partir de X, dans l'environnement discrétisé. Elle est initialisée de la façon suivante :

- $cout_i(X) = 0$ si X contient un site (i.e. $\exists i \ / \ X = \widetilde{S}_i$) ;
- $cout_i(X) = +\infty$ si X appartient à un obstacle fixe ;
- $cout_i(X) = ndef$ sinon ($ndef$ pour "non-defini").

3.2.2 La phase de propagation d'onde multiple

L'idée de la propagation d'onde multiple est la suivante : à chaque site S_i est associé un front d'onde F_i. Tous les fronts d'ondes sont développés simultanément, jusqu'à ce qu'ils aient percuté les autres.

Cette idée est motivée par le raisonnement suivant. Comme nous l'avons expliqué précédemment, à chaque fois que F_i percute un autre front d'onde F_j, toute l'information nécessaire à la construction du tronçon $T_{i,j}$ est disponible : il suffira d'effectuer deux *hill climbing* à partir du point de collision, jusqu'aux cases $\widetilde{S_i}$ et $\widetilde{S_j}$. Ainsi, dès que le front d'onde F_i aura percuté tous autres fronts d'ondes, tous les chemins partant ou arrivant au site S_i pourront être construits par hill climbing. Comme il est donc inutile de poursuivre son expansion, ce front d'onde est donc arrêté. Selon ce principe, au bout d'un moment, tous les fronts d'ondes seront à l'arrêt. Ceci est illustré dans la figure 6.5.

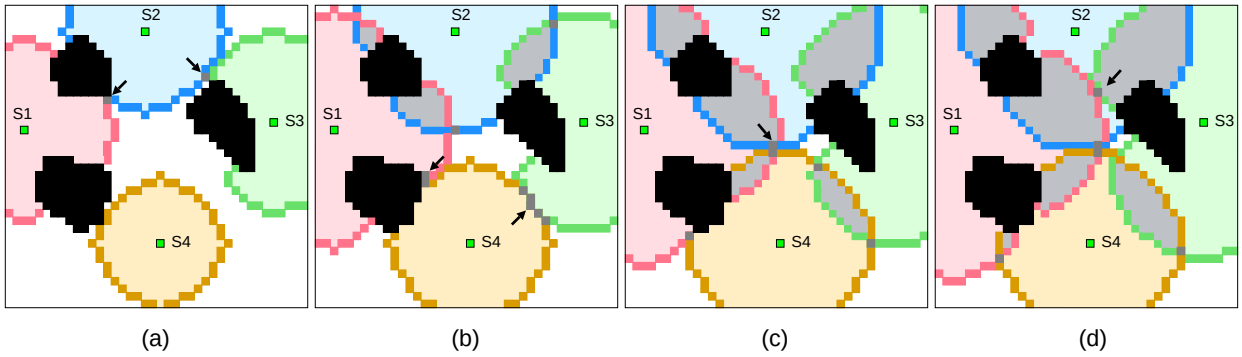

FIG. 6.5 – La propagation d'ondes multiple
Chaque front d'onde F_i est développé en parallèle, jusqu'à ce qu'il ait percuté tous les autres. (a) collisions entre F_1 et F_2 et entre F_2 et F_3 ; (b) collisions entre F_1 et F_4 et entre F_3 et F_4 ; (c) collision entre F_2 et F_4, ces deux fronts d'ondes ont fini leur liste de collisions et sont donc stoppés ; (d) collision entre F_1 et F_3, stoppés à leur tour.

Le développement simultané (et non plus séquentiel) des fronts d'ondes permet de diminuer considérablement les évaluations redondantes. Sur la mission de figure 6.1a, par exemple, la proportion de cases évaluées plusieurs fois (zones grises dans la figure 6.5) n'est que de 18%, au lieu de 66% précédemment. De plus, ces cases n'ont été évaluées que 2 fois au total, mais jamais 3, comme ce fut le cas lors des propagations séquentielles. De manière générale, ce gain en efficacité permet de diminuer le temps de calcul de façon significative, comme nous le verrons dans la section 4.

Pour la réalisation pratique de la propagation d'onde multiple, nous proposons d'associer à chaque front d'onde F_i un attribut col_i, représentant la liste des collisions qu'il doit effectuer avec les autres. La propriété $j \in col_i$ signifie que F_i devra percuter F_j, et implique nécessairement $i \in col_j$.

Au départ, chaque front d'onde est uniquement constitué de sa case d'origine $\widetilde{S_i}$, et doit percuter tous les autres. On a donc $col_i = \{j\}_{j \neq i}$.

Ensuite, tous les fronts d'ondes sont développés de façon pseudo-parallèle, selon un principe analogue à celui de l'algorithme round-robin [Nag87], utilisé pour l'ordonnancement de tâches dans les systèmes d'exploitations. Dans une boucle principale, chaque front d'onde F_i effectue, tour à

tour, une opération d'expansion élémentaire. Cette opération consiste à sélectionner la tête T de F_i (c'est à dire la case de moindre coût), d'évaluer ses voisins[2] V (avec une métrique $\mathcal{M}$), puis de les ajouter à F_i.

Si un voisin V a déjà été évalué par un autre front d'onde F_j, alors cela signifie que F_i et F_j se percutent au niveau de la case V[3]. Les indices i et j sont respectivement supprimés de col_j et de col_i, pour modéliser que la collision entre F_i et F_j n'est plus à faire. Par ailleurs, les informations relatives à cette collision sont stockées sous la forme d'un triplet (i, j, V).

Cette boucle s'arrête quand tous les fronts d'ondes ont effectué leurs collisions, c'est à dire quand $\forall i \in [1, n] : \ col_i = \emptyset$.

L'ensemble de l'algorithme est fourni ci-dessous. Notons que cet algorithme permet également de détecter les objectifs non réalisables dans une mission. En effet, si un site S_i n'est pas atteignable (car entouré d'obstacles, par exemple), alors toutes les collisions avec le front d'onde F_i ne pourront pas être effectuées. Ainsi, si la liste de collisions C renvoyée par l'algorithme contient un nombre d'éléments inférieur à $n(n-1)/2$ (le nombre de tronçons à calculer), alors au moins un site est inaccessible (facilement identifiable en recherchant les indices manquants dans C).

```
PROPAGATION_ONDE_MULTIPLE(S̃, G)
  ▷ Entrée : S̃ : cases contenant les sites
  ▷ Entrée : G : grille
  ▷ Sortie : C : liste de collisions
  ▷ Locale : F : liste de fronts d'ondes
  ▷ Locale : i, n : entiers
1 Début
2     n ← longueur(S̃)
3     pour i ← 1 à n faire
4         F_i ← {S̃_i}
5         col_i ← {1, ..., i − 1, i + 1, ..., n}
6     i ← 1
7     C ← ∅
8     faire
9         C ← C ∪ EXPANSION_ELEMENTAIRE(i, F, G)
10        i ← PROCHAIN_FRONT_ONDE(i, n, F)
11    tantque i > 0
12    retourner C
13 Fin
```

[2]Notons que les voisins appartenant à des obstacles sont exclus, reconnaissables par leur coût infini.

[3]Il est possible que la collision entre F_i et F_j ait lieu sur plusieurs cases (comme la collision entre F_3 et F_4 dans la figure 6.5b, par exemple). Dans ce cas, la case X minimisant la somme des coûts $cout_i(X) + cout_j(X)$ est choisie.

EXPANSION_ELEMENTAIRE(cur, F, G)
▷ **Entrée :** cur : indice du front d'onde courant
▷ **Entrée :** F : liste de fronts d'ondes
▷ **Entrée :** G : grille
▷ **Sortie :** C : liste des nouvelles collisions
▷ **Locale :** T : tête du front d'onde courant F_{cur}
Début
  $C \leftarrow \emptyset$
  $T \leftarrow \arg\min\{cout_{cur}(X), X \in F\}$
  $F_{cur} \leftarrow F_{cur} \setminus T$
  pour chaque $X \in \mathcal{V}(T)$ / $cout_{cur}(X) = ndef$ **faire**
    $cout_{cur}(X) \leftarrow cout_{cur}(T) + \mathcal{M}_{T,X}$
    $F_{cur} \leftarrow F_{cur} \cup X$
    si $\exists i \in [1, n] / cout_i(X) \neq ndef$ **et** $i \in col_{cur}$ **alors**
      $col_{cur} \leftarrow col_{cur} \setminus \{i\}$
      $col_i \leftarrow col_i \setminus \{cur\}$
      $C \leftarrow C \cup \{(cur, i, X), (i, cur, X)\}$
  retourner C
Fin

[La fonction $\mathcal{V}(T)$ renvoie les cases voisines de T (généralement le voisinage de Moore). La métrique $\mathcal{M}_{T,X}$ mesure le coût entre les cases T et X]

PROCHAIN_FRONT_ONDE(cur, max, F)
▷ **Entrée :** cur : indice du front d'onde courant
▷ **Entrée :** max : nombre total de fronts d'ondes
▷ **Entrée :** F : liste des fronts d'ondes
▷ **Sortie :** nxt : indice du prochain front d'onde
Début
  $nxt \leftarrow cur + 1$
  faire
    si $nxt > max$ **alors**
      $nxt \leftarrow 1$
    si $col_{nxt} = \emptyset$ **alors**
      $nxt \leftarrow nxt + 1$
    sinon retourner nxt
  tantque $nxt \neq cur$
  retourner 0
Fin

3.2.3 La phase de construction des tronçons

La phase de propagation d'onde multiple renvoie une liste de collisions sous la forme de triplets (i, j, C_{ij}), où C_{ij} représente la case où les fronts d'ondes F_i et F_j sont entrés en contact pour la première fois.

A partir de ces informations, nous pouvons construire chaque tronçon de chemin $T_{i,j}$ comme suit :

1. Une première application de l'algorithme de hill climbing, dans les valeurs de la fonction $cout_i$ en partant de la case C_{ij}, permet de relier C_{ij} et $\widetilde{S_i}$ à coût optimal ;
2. Une deuxième application de l'algorithme de hill climbing, dans les valeurs de la fonction $cout_j$ en partant de la case C_{ij}, permet de relier C_{ij} et $\widetilde{S_j}$ à coût optimal.
3. La concaténation des deux résultats précédents permet d'obtenir le tronçon $T_{i,j}$ recherché.

Ce procédé est illustré dans la figure 6.6 ci-après.

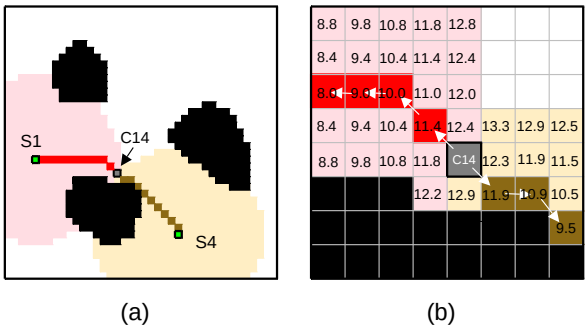

FIG. 6.6 – Construction d'un tronçon
(a) construction des deux parties du tronçon $T_{1,4}$ (en rouge et en brun) à partir du point de collision C_{14} ; (b) zoom autour de C_{14}, explicitant des déplacements effectués lors des deux applications du hill climbing (flèches).

4 Etude de performances

Dans cette partie, nous allons évaluer de façon théorique puis expérimentale, le gain de performances apporté par le concept de la collision d'ondes. Pour cela, nous allons appliquer l'algorithme de la propagation d'ondes sur un même ensemble de missions, avec et sans collisions d'ondes, et comparer le nombre de cases évaluées.

Cette comparaison est effectuée dans les conditions suivantes :

1. **Opérateurs standard.**
   Dans l'algorithme de propagation d'ondes, nous utiliserons les mêmes opérateurs que ceux généralement utilisés dans la littérature : distance euclidienne pour la métrique $\mathcal{M}$ et voisinage de Moore. Ce choix n'est pas négligeable, car il détermine la forme des fronts d'ondes.
   En effet, la conjonction de ces deux opérateurs aboutit à des fronts d'ondes octogonaux (et non à des cercles, comme on pourrait le croire de façon intuitive) : la distance euclidienne implique une propagation isotrope, donc une symétrie centrale dans la forme du front d'onde. Le voisinage de Moore implique quant à lui 8 directions de propagations, donc 8 faces[4].

2. **Environnement carré.**
   Nous supposerons que l'environnement est de dimensions $c \times c$, pour réduire l'influence de la taille de l'environnement à une seule variable. Notons que l'unique différence entre des environnements carrés et rectangulaires est une dilatation sur une dimension. Cette différence n'a

[4]Pour plus d'explications, le lecteur pourra se reporter à la thèse [Pêt07], où une étude très complète sur ce sujet est menée.

pas de conséquences sur les conclusions de notre étude, puisque nous allons raisonner par la suite sur des aires. En effet, les résultats obtenus dans un carré $c \times c$ sont strictement équivalents à ceux obtenus dans un rectangle de dimension $l \times L$ avec $L = c^2/l$ (leur aire est la même).

3. **Absence d'obstacles.**
   Nous supposerons que l'environnement est dépourvu d'obstacles, car ceux-ci ont pour effet de "déformer" les fronts d'ondes. S'il est plus intéressant de contourner un obstacle par le bas, par exemple, alors la partie basse du front d'onde franchira l'obstacle avant la partie haute (c'est le cas du front F_3, en bleu, dans la figure 6.5). Ces déformations sont ignorées, ce qui constitue une des faiblesses de notre étude.

4. **Grand nombre de missions.**
   Nous supposerons enfin que les algorithmes sont appliqués sur un nombre de missions suffisamment important pour que l'on puisse considérer la distribution des sites dans l'environnement comme aléatoire.

4.1 Etude théorique

▷ 1er cas : Sans collision d'ondes.

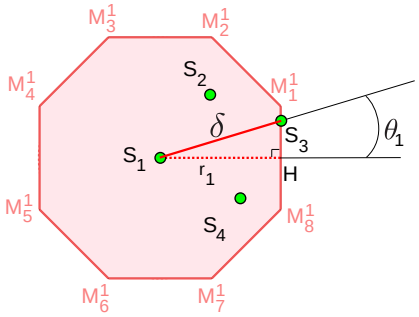

FIG. 6.7 – Aire recouverte sans collision d'ondes
La figure illustre l'aire recouverte par le front d'onde F_1, dans la mission de la figure 6.1a, sans obstacle et sans collision d'ondes.

Comme nous l'avons vu dans la partie 3, en l'absence de collision d'ondes, $n-1$ fronts d'ondes sont développés de façon indépendante. Chaque front F_i s'étend dans l'environnement, jusqu'à ce qu'il ait couvert l'ensemble des sites $E_i = \{S_{i+1}, S_{i+2}, ..., S_n\}$.

Soit $\overline{S_j}$ le site de l'ensemble E_i le plus éloigné de l'origine de F_i, c'est à dire du site S_i. L'expansion du front F_i ne s'arrêtera que lorsqu'il touchera le point $\overline{S_j}$. La distance parcourue par le front d'onde est donc la distance entre S_i et $\overline{S_j}$.

Comme un nombre important de missions est supposé, cette distance correspond à la distance moyenne entre deux points tirés aléatoirement dans l'environnement. Il a été démontré [Tro99] que la distance moyenne était, dans un carré de côté 1 :

$$\delta = \frac{1}{15}\left(2 + \sqrt{2} + 5\ln\left(1+\sqrt{2}\right)\right) \approx 0.521$$

Notre environnement étant de coté c, la distance entre S_i et $\overline{S_j}$ est donc égale à $\Delta = c \cdot \delta$.

De plus, comme nous l'avons vu précédemment, F_i est un octogone. Soient M_k^i ($k \in [1,8]$) ses sommets, et $[\overline{M_k^i}\ \overline{M_l^i}]$ le coté touchant le site $\overline{S_j}$. Le cercle inscrit dans F_i a pour rayon $r = S_iH$, où H est le projeté orthogonal de S_i sur $[\overline{M_k^i}\ \overline{M_l^i}]$.

Par ailleurs, les distances r_i et Δ sont liées par $r_i = \cos\theta_i \cdot \Delta$, où θ_i designe l'angle entre les segments $[S_iH]$ et $[S_i\overline{S_j}]$.

L'aire couverte par le front d'onde F_i est donc :

$$\mathcal{A}_i = 8 \cdot r_i^2 \cdot (\sqrt{2}-1) = 8\cos^2\theta_i \cdot \Delta^2 \cdot (\sqrt{2}-1) \tag{6.1}$$

▷ 2ème cas : Avec collision d'ondes.

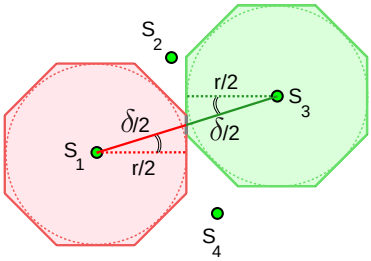

FIG. 6.8 – Aire recouverte avec collision d'ondes
La figure illustre l'aire recouverte par le front d'onde F_1, dans la mission de la figure 6.1a, sans obstacles et avec collision d'ondes.

Si nous introduisons la collision d'ondes, n fronts d'ondes sont cette fois développés en parallèle. Chaque front d'onde F_i sera étendu dans l'environnement jusqu'à effectuer sa dernière collision, avec le front d'onde $\overline{F_j}$. En effet, le site $\overline{S_j}$ étant le plus éloigné de S_i et les propagations d'ondes étant isotropes, alors nous avons la garantie que $\overline{F_j}$ sera le dernier front d'onde à percuter F_i.

Par ailleurs, les fronts d'ondes F_i et $\overline{F_j}$ se développant à la même vitesse, ils auront parcouru, au moment de leur collision, la même distance, c'est à dire $\Delta/2$. Le rayon inscrit dans F_i devient donc $r'_i = \cos\theta_i \cdot \Delta/2$, et l'aire correspondante est :

$$\mathcal{A}'_i = 8 \cdot r_i'^2 \cdot (\sqrt{2}-1) = 8\cos^2\theta_i \cdot \frac{\Delta^2}{4} \cdot (\sqrt{2}-1) \tag{6.2}$$

▷ Comparaison.

Si nous comparons les aires couvertes par l'ensemble des fronts, sans et avec collision d'ondes, nous obtenons le ratio :

$$\gamma = \frac{(n-1)\mathcal{A}_i}{n \cdot \mathcal{A}'_i} = \frac{n-1}{n} \cdot \gamma^\infty \qquad \text{avec} \quad \gamma^\infty = 4 \tag{6.3}$$

Ainsi, en moyenne, une propagation d'onde multiple évalue γ fois moins de cases que $n-1$ propagations d'ondes successives. Le facteur γ vaut au moins 2 (car $n \geq 2$), et tend vers 4 quand le nombre de sites augmente. Ceci est illustré dans la figure 6.9.

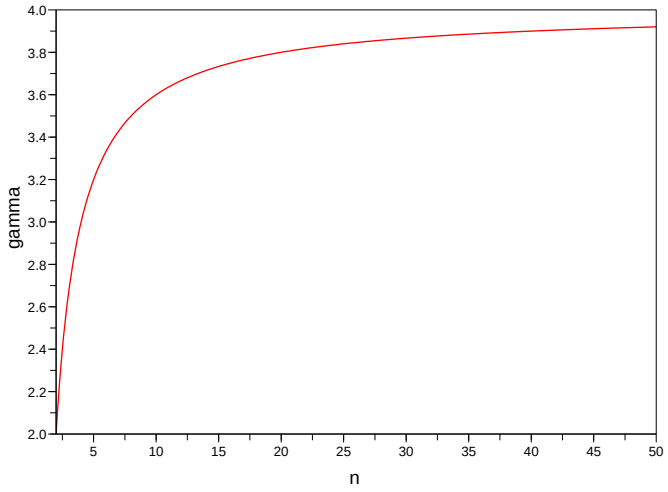

FIG. 6.9 – Graphe du ratio théorique γ, donné par l'équation 6.3.

4.2 Résultats expérimentaux

Afin de vérifier les résultats de notre étude théorique, nous avons mesuré le nombre de cellules développées au cours de la phase de propagation d'onde, sans et avec collision d'ondes. Nous nous sommes placés dans une grille de taille 100×100 afin de limiter les effets de la discrétisation tout en conservant un temps de calcul acceptable (quelques secondes pour chaque mission). Nous avons fait varier le nombre n de sites de 2 à près de 50, et pour chacune de ses valeurs, nous avons appliqué les deux approches sur 500 missions générées aléatoirement.

Les résultats obtenus, fournis dans le tableau 4.2, permettent de déterminer des valeurs expérimentales pour le ratio γ, que nous avons appelé γ_{exp}.

n	2	3	5	7	12	22	32	42
$(Sans)$	10643	29033	74199	124099	264744	564365	834932	1114377
$(Avec)$	5444	11556	25778	41000	83591	171533	249199	331658
γ_{exp}	1.95	2.51	2.87	3.02	3.16	3.29	3.35	3.36

TAB. 6.1 – Nombre de cases évaluées pour n sites. $(Sans)$: sans collision d'ondes ; $(Avec)$: avec collision d'ondes ; γ_{exp} : (1)/(2)

Une identification de paramètres par moindres carrés a permis de déterminer que γ_{exp} était de la forme :

$$\gamma_{exp} \approx 2.90 \cdot \frac{n-1}{n} + 0.53 \tag{6.4}$$

La courbe correspondante, illustrée dans la figure 6.10, tend vers $\gamma_{exp}^{\infty} = 3.53$ quand n augmente.

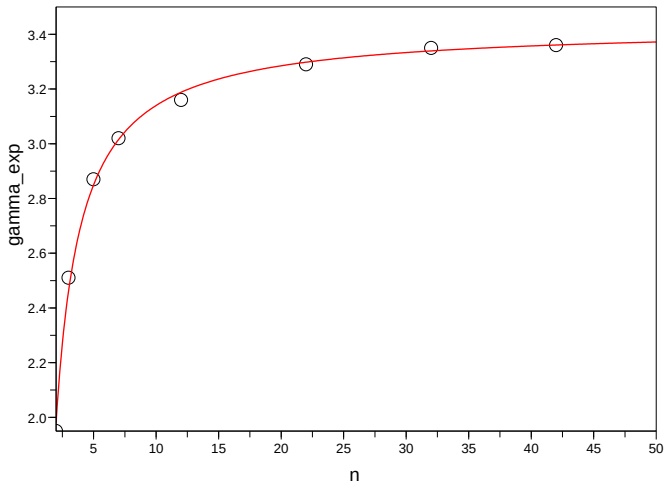

FIG. 6.10 – Graphe du ratio expérimental γ_{exp}, donné par l'équation 6.4.

4.3 Discussion

Même si les facteurs γ (théorique) et γ_{exp} (expérimental) ont une évolution analogue en fonction de n, leur limite à l'infini est sensiblement différente. En effet, la limite estimée de γ_{exp} (3.43) est près de 15% plus faible que celle attendue (4.00). Nous pensons que cette différence est due à la simplicité de notre modèle, particulièrement sur les points suivants :

1. **Influence des bords de l'environnement**
   Nous avons supposé, dans l'étude théorique, que les front d'ondes étaient des polygones entiers. Or, les limites de l'environnement ont un effet analogue à de larges obstacles : ils arrêtent prématurément l'extension des fronts d'ondes, ce qui aboutit à des polygones partiels, des bandes étant éliminées sur les bords. Et plus les polygones sont larges, plus les bandes éliminées le seront. Or, l'approche de propagation d'onde sans collision d'ondes aboutit à des polygones 4 fois plus grands (en terme de surface) qu'avec collision d'ondes. Il semble donc que la première approche soit avantagée par les dimensions finies de l'environnement.

   Si nous reprenons les figures 6.7 et 6.8 en ajoutant les bords de l'environnement, nous obtenons la figure 6.11 ci-dessous. Dans la partie (a) (sans collisions d'ondes) l'aire couverte par le front F_1 est réduite de près de 40% par l'effet des bords. Dans la partie (b) (avec collision d'ondes), cette même aire n'est réduite que de 17%. Ainsi, le rapport des aires couvertes passe de 4 à 2.91 environ.

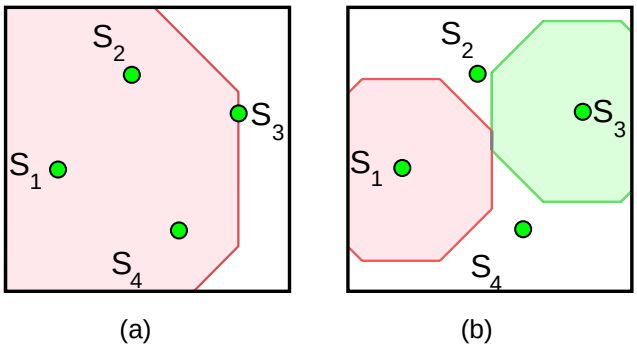

FIG. 6.11 – Effet des bords de l'environnement
Les parties (a) et (b) reprennent respectivement les situations des figures 6.7 et 6.8, en ajoutant les bords de l'environnement (carré noir). Les parties de polygones éliminées par ces bords sont bien supérieures en (a) qu'en (b).

Nous aurions pu effectuer les tests de telle manière que ces effets de bords soient inexistants, en faisant boucler l'environnement sur lui-même (comme un planisphère), où en propageant les fronts d'ondes dans un environnement plus grand que l'environnement initial (par exemple, un carré de $3c \times 3c$ au lieu de $c \times c$). Mais ces tests n'auraient pas été réalistes, puisque dans la pratique, ces effets de bords apparaîtront. Pour réduire l'écart entre théorie et pratique, il faudrait revoir notre étude théorique, en tenant compte de ce phénomène.

2. **Distance parcourue par les fronts d'ondes.**
Dans notre étude théorique, nous avons supposé que la distance entre l'origine S_i du front d'onde F_i et le site le plus éloigné $\overline{S_j}$ était toujours Δ. En réalité, cette distance dépend du nombre de sites à couvrir par le front d'onde F_i présents dans l'ensemble E_i.

Si E_i ne contient qu'un site, c'est à dire si $E_i = \{S_j\}$, la distance moyenne parcourue d_i^1 par le front F_i pour couvrir S_j est donnée par l'espérance que le front F_i atteigne le site S_j :

$$\begin{aligned} d_i^1 &= E[d(S_i, S_j) = d] \\ &= \int_{S_i}\int_{S_j} p(d(S_i, S_j) = d) dS_i dS_j \\ &= \int_{x_i}\int_{x_j}\int_{y_i}\int_{y_j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} dx_i dx_j dy_i dy_j \\ &= \Delta \qquad \text{d'après la définition de } \Delta \text{ dans [Tro99]} \end{aligned}$$

où $E[X]$ désigne l'espérance et $p(X)$ la probabilité de l'événement X.

On a donc effectivement $d_i = \Delta$ dans le cas d'un seul site.

Mais si à présent E_i contient deux sites, c'est à dire si $E_i = \{S_j, S_k\}$, la distance moyenne parcourue d_i' par F_i, pour couvrir S_j et S_k est plus élevée que précédemment. En effet, d_i^2 est cette fois donnée par l'espérance que F_i touche l'un des sites comme précédemment, mais qu'en plus l'autre soit à l'intérieur de F_i. On a alors :

$$
\begin{aligned}
d_i^2 &= E[(d(S_i,S_j)=d \cap d(S_i,S_k)\leq d) \cup (d(S_i,S_k)=d \cap d(S_i,S_j)\leq d)] \\
&= \int_{S_i}\int_{S_j}\int_{S_k} [p(d(S_i,S_j)=d \cap d(S_i,S_k)\leq d)]\, dS_i dS_j dS_k \\
&\quad + \int_{S_i}\int_{S_k}\int_{S_j} [p(d(S_i,S_k)=d \cap d(S_i,S_j)\leq d)]\, dS_i dS_j dS_k \\
&= \int_{S_i}\int_{S_j}\int_{S_k} 2\,[p(d(S_i,S_j)=d \cap d(S_i,S_k)\leq d)]\, dS_i dS_j dS_k \\
&= \int_{S_i}\int_{S_j} 2\left[p(d(S_i,S_j)=d) - \int_{S_k} p(d(S_i,S_k)\geq d) dS_k\right] dS_i dS_j \\
&= \underbrace{\int_{S_i}\int_{S_j} p(d(S_i,S_j)=d) dS_i dS_j}_{\Delta} + \underbrace{\int_{S_i}\int_{S_j}\left[p(d(S_i,S_j)=d) - \int_{S_k} p(d(S_i,S_k)\geq d) dS_k\right] dS_i dS_j}_{\varepsilon_1}
\end{aligned}
$$

On a donc $d_i^1 = \Delta$ et $d_i^2 = \Delta + \varepsilon_1$.

Par un raisonnement par récurrence, on montre que :

$$d_i^k = \Delta + \sum_{j=1}^{k-1} \varepsilon_k$$

L'expression analytique des quantités ε_k n'étant pas aisée à calculer, nous les avons évaluées expérimentalement dans un premier temps. Ces premiers résultats, représentés dans la figure 6.12, suggèrent que ε_k varie en $1/k$.

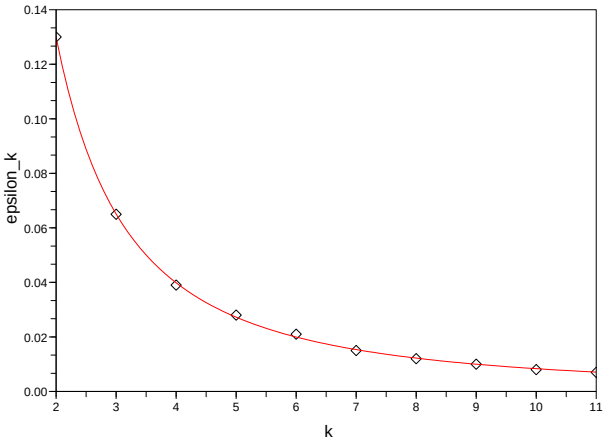

FIG. 6.12 – Valeurs expérimentales de ε_k

La prise en compte des effets de ε_k dans notre modèle est aussi un élément qui pourrait réduire l'écart observé entre notre étude théorique et nos résultats expérimentaux.

Malgré cette différence entre théorie et pratique, le gain de performances mesuré expérimentalement, près de 3.5, permet de réduire significativement les temps de planification.

Enfin, même si cela n'a pas été présenté dans ce chapitre, le concept de collision d'ondes est tout à fait utilisable en présence de courants. Pour cela, il suffit d'utiliser l'une des variantes introduites dans les chapitres 4 ou 5, à la place de la propagation d'onde classique.

La figure 6.13 présente la collision d'ondes obtenue entre F_1 et F_3, en utilisant la propagation d'ondes symbolique du chapitre 5, en présence d'un courant constant de type tourbillon.

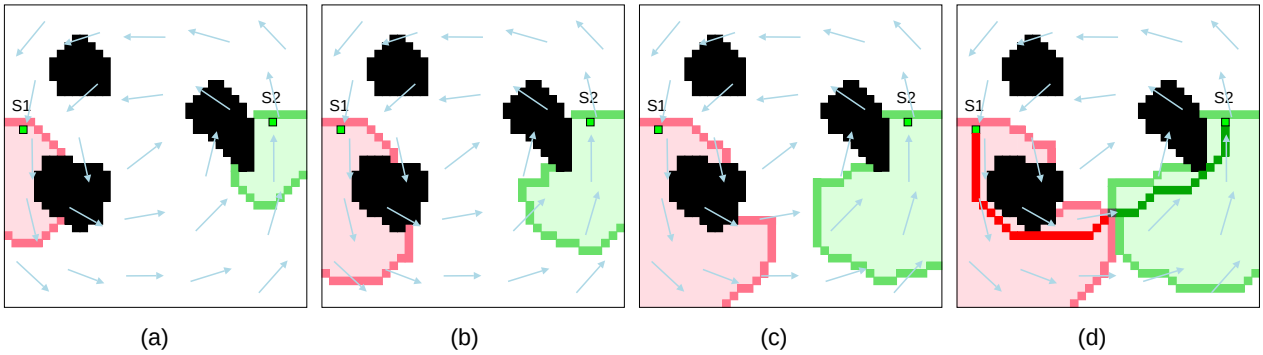

FIG. 6.13 – Collision d'ondes en présence de courant
La figure illustre quelques étapes de la collision d'ondes en présence d'un courant de type tourbillon (flèches bleues). Le chemin obtenu en (d) suit globalement les courants pour maximiser la vitesse du robot.

Il est important de noter que la présence de courants rend la propagation d'onde non isotrope, ce qui ne correspond plus au cadre de notre étude théorique, ni de nos expériences. Dans ce telles circonstances, il est possible que le gain de performances apporté par la collision d'ondes s'écarte significativement de la valeur mentionnée dans ce chapitre (en mieux, comme en pire). Il serait intéressant de mener des tests supplémentaires, pour mieux comprendre la corrélation entre les caractéristiques du courant (direction, intensité, répartition spatiale) et le gain observé.

5 Conclusion

Dans ce chapitre, nous avons introduit le concept de collision d'ondes, permettant de calculer simultanément tous les tronçons de trajectoire entre n sites. Ce concept semble suffisamment générique pour intégrer de nouvelles contraintes, telle que la présence de courants.

L'ensemble de ces résultats a été publié dans les actes de la conférence IROS'07 (International Conference on Intelligent Robots and Systems). Le lecteur interessé pourra consulter [ST07].

La suite logique de ces résultats semble être une amélioration de notre modèle, afin que notre étude théorique soit plus fidèle à la réalité. Comme évoqué précédemment, les deux premières pistes d'approfondissement sont : (1) l'effet des bords de l'environnement et (2) une évaluation plus fine de la distance à parcourir par chaque front d'onde.

En plus de ces deux pistes, il serait intéressant de mieux comprendre l'impact des éléments de l'environnement, comme les obstacles ou la présence de courant, qui ont tous deux l'effet de déformer les fronts d'ondes par rapport aux polygones initiaux.

Chapitre 7

Modulation de vitesse à base de Programmation Par Contraintes

Sommaire

1 Introduction

Comme nous l'avons vu dans le chapitre 2, le problème de planification de trajectoire est un problème NP-difficile [Can88]. La conséquence pratique de ce fait est que, dans le pire des cas, le temps de calcul peut varier exponentiellement en fonction du nombre d'obstacles.

Ainsi, les quelques méthodes qui tentent de résoudre *directement* le problème de planification de trajectoire (Programmation Lineaire Mixte [RH02] ou Programmation Quadratique Séquentielle [WZD08]), sont particulièrement sensibles au nombre d'obstacles. Sur des exemples pathologiques, le temps de planification peut subitement exploser par le simple ajout d'un obstacle, alors qu'il demeurait raisonnable jusqu'alors. Ce phénomène, prévu par la théorie, a récemment été mis en évidence de façon expérimentale dans [YELP08].

Pour l'éviter, des méthodes dites *indirectes* ont été introduites par Kant et Zucker [KZ86] et sont toujours très utilisées aujourd'hui. Elles consistent à décomposer le problème de planification de trajectoire en deux sous-problèmes : une planification de chemin puis une modulation de la vitesse du robot sur le chemin planifié. Cette décomposition permet de résoudre le problème de planification de trajectoire en un temps polynomial[1].

Dans ce chapitre, nous nous intéressons à l'étape de modulation de vitesse. Cette étape consiste à déterminer le profil optimal de vitesse pour le robot, c'est à dire lui permettant d'atteindre le point d'arrivée au plus tôt tout en évitant les obstacles mobiles.

Cette étape est généralement effectuée en construisant un espace-temps à deux dimensions : la longueur l parcourue sur le chemin, et le temps t écoulé. Dans cet espace-temps, chaque obstacle mobile génère une ou plusieurs surfaces interdites (polygones de la même couleur que les obstacles), notées S_j et illustrées dans la figure 7.1b.

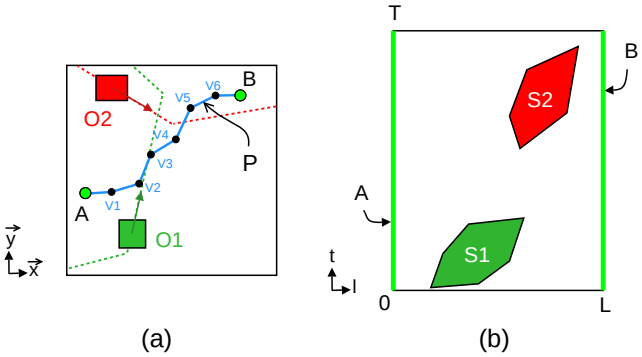

FIG. 7.1 – L'espace-temps utilisé pour la modulation de vitesse
(a) le problème initial : le robot doit ajuster sa vitesse sur le chemin bleu pour éviter les obstacles mobiles (polygones rouge et vert) ; (b) l'espace-temps correspondant.

Le problème de modulation de vitesse est donc reformulé en un problème de planification de chemin dans cet espace-temps.

Ce problème est le plus souvent résolu en utilisant des méthodes génériques, telles que les méthodes de décomposition (fig. 7.2a et 7.2b), légèrement adaptées pour tenir compte des contraintes spécifiques à l'espace-temps. Par exemple, le temps étant strictement croissant, les profils de vitesse sont forcement orientés de bas en haut.

On note également l'utilisation de méthodes spécialement mises au point pour résoudre ce problème, comme l'optimisation de B-splines (fig. 7.2c) ou l'algorithme des *lignes brisées* (fig. 7.2d), que nous avions introduit dans [ST06].

[1] Ceci se fait au prix de la complétude : certaines situations aboutiront à un échec de planification, alors qu'il existait une solution. Pour plus de détails, se référer à la page 54.

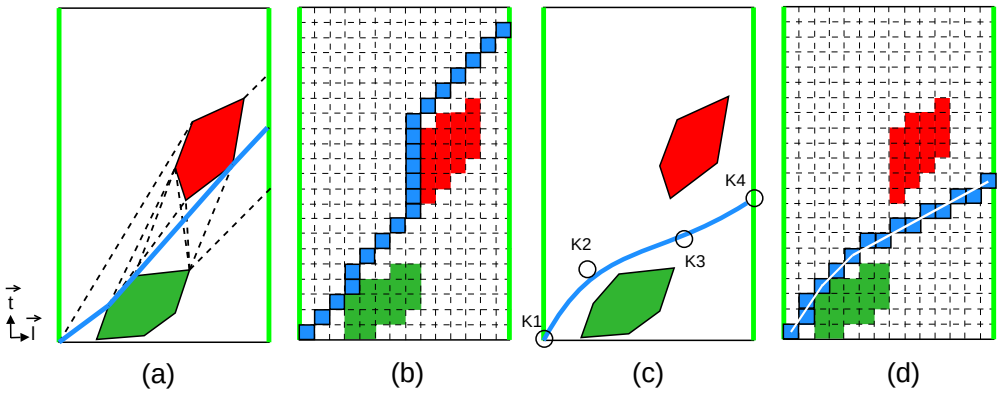

FIG. 7.2 – Principales techniques de modulation de vitesse
Méthodes de décomposition adaptées : (a) graphe de visibilité, (b) cellules régulières ; (c) optimisation de B-splines ; (d) algorithme des lignes brisées.

Il est important de comprendre que l'ensemble de ces techniques sont *ad hoc*, c'est à dire qu'elles exploitent toutes, à leur manière, certaines spécificités du problème. Par exemple, la méthode basée sur le graphe de visibilité (fig. 7.2a) exploite les hypothèses suivantes :

1. Le robot est un objet ponctuel,
2. Le robot n'a pas de contraintes sur son rayon de braquage, ni sur son accélération,
3. Les variations de vitesse du robot sont uniquement dues aux obstacles mobiles,
4. Les obstacles mobiles sont de forme polygonale,
5. Le robot évite un obstacle mobile en passant par l'un de ses sommets.

C'est sous l'ensemble de ces hypothèses que l'on peut démontrer que les surfaces interdites générées dans l'espace-temps sont polygonales, et que le profil de vitesse optimal passe nécessairement par les sommets de ces polygones. C'est pourquoi nous pouvons limiter la recherche aux arcs reliant les sommets des surfaces interdites et respectant certaines contraintes propres à l'espace-temps.

Cette façon de procéder aboutit à des algorithmes très performants (les temps de calcul sont généralement inférieurs à la seconde) mais aussi très peu robustes à des changements dans l'énoncé du problème.

En effet, un changement dans les caractéristiques du problème, même mineur, nécessite souvent d'adapter les algorithmes correspondants, ce qui peut être laborieux, voire impossible.

Par exemple, la méthode basée sur le graphe de visibilité citée précédemment suppose que les variations de vitesse du robot sont uniquement dues aux obstacles mobiles (hypothèse 3 ci avant). Ceci exclut, en particulier, la présence de courants dans l'environnement, qui selon leur orientation, peuvent significativement accélérer ou ralentir le robot.

Comme nous l'avons vu précédemment, une catégorie assez importante de robots mobiles est amenée à évoluer au sein de courants (tous les robots aériens, marins ou sous-marins). Dans ces conditions, l'algorithme devra être considérablement modifié. A présent, les bornes sur la vitesse du robot (par rapport au sol) varie selon le courant en présence. Par exemple, si le robot, en l'absence de courant, pouvait se déplacer dans la direction $\vec{x}$ avec une vitesse $v_c \in [0, 100]$, l'ajout d'un courant

$\vec{w} = 100\vec{x}$ décale cet intervalle à $[100, 200]$. En effet, par principe de composition des vitesses, même si le robot est immobile par rapport au courants (moteur éteint), celui-ci se déplacera à vitesse $100\vec{x}$ par rapport au sol.

La conséquence dans l'espace-temps est la suivante : il n'y a plus forcement un arc entre deux sommets, mais plusieurs, selon les courants traversés par le robot sur son chemin. Ceci est illustré dans la figure 7.3b. Le calcul de ces différents arcs complique donc considérablement l'algorithme initial, ce qui aboutit à sa réécriture.

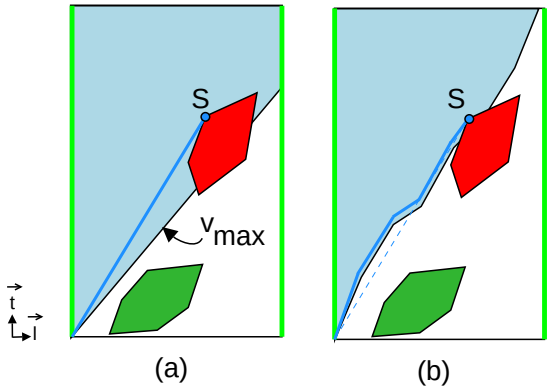

FIG. 7.3 – Impact de courants dans l'espace-temps
(a) sans courant, la vitesse maximale du robot (par rapport au sol), dénotée v_{max}, est constante et délimite le secteur bleu. On peut donc atteindre le sommet s en utilisant un unique segment ; (b) si on ajoute un courant $\vec{w} = -50\vec{x}$ dans l'environnement initial (fig. 7.1a), cette vitesse varie sur chaque portion du chemin. Plusieurs segments sont nécessaires pour délimiter le secteur accessible, et donc pour atteindre s.

Pour éviter ces problèmes, l'approche utilisée pour la modulation de vitesse doit être la plus générique possible. C'est pourquoi nous proposons dans ce chapitre une approche basée sur la Programmation Par Contraintes (PPC). Nous présenterons deux versions de notre approche, sur domaines finis tout d'abord (approche initiale) puis sur domaines continus (amélioration récente). Puis nous montrerons que cette approche permet de changer les caractéristiques du problème par simple modification de contraintes dans le modèle, sans travail de réimplémentation. Nous illustrerons ce point à travers l'exemple qui met en défaut les méthodes existantes : l'ajout de courants dans l'environnement, qui n'étaient pas pris en compte initialement.

2 Formulation du problème

Un robot ponctuel se déplace sur un chemin précalculé $\mathcal{P}$ entre deux points A et B, en présence d'obstacles mobiles, comme illustré dans la figure 7.1a. Le problème consiste à trouver le profil de vitesse optimal pour le robot, c'est à dire lui permettant d'atteindre le point B en un temps minimal, tout en évitant les obstacles mobiles.

D'un point de vue plus formel, le robot est modélisé comme un objet ponctuel évoluant dans un espace Euclidien $\mathcal{C}$ de dimension 2, de repère orthonormé $\mathcal{R} = (O, \vec{x}, \vec{y})$. La vitesse du robot par rapport à $\mathcal{R}$ est notée $\vec{v^r}$.

Le chemin $\mathcal{P}$ est composé d'une liste de n viapoints $V = \{V_i\}_{i\in[1,n]}$, reliés par des lignes droites (en bleu sur la figure 7.1a). Chaque viapoint V_i est situé à l'abscisse curviligne l_i sur $\mathcal{P}$. Par convention, $V_0 = A$ et $V_n = B$.

L'environnement contient m obstacles mobiles. Chaque obstacle est noté O_j et est modélisé par un rectangle[2] de dimensions D_{j_x}, D_{j_y}, effectuant une série de mouvements rectilignes uniformes. Ce rectangle correspond à un objet ponctuel entouré d'une zone de sécurité. Le robot devra éviter l'ensemble de cette zone, afin de ne pas passer trop près de l'obstacle mobile.

En utilisant ces notations, nous pouvons reformuler notre problème comme suit :
Le problème de modulation de vitesse consiste à trouver un profil de vitesse σ :

$$\sigma : M \in \mathcal{P} \mapsto t \in [0, T] \tag{7.1}$$

minimisant la date d'arrivée $t_B = \sigma(B)$, sous les contraintes suivantes :

1. vitesse maximale[3] : $v^r \leq \nu_{max}$,
2. évitement de tous les obstacles mobiles O_j,
3. horizon temporel T, matérialisant la date d'arrivée au plus tard en B.

L'horizon temporel peut correspondre à des limites énergétiques (quantité de carburant, durée d'ensoleillement) ou à des contraintes stratégiques (il n'est plus pertinent de réaliser la mission après T).

3 Principe de la PPC

La Programmation Par Contraintes [MS98] (PPC) consiste à reformuler le problème initial sous la forme d'un Problème de Satisfaction de Contraintes (PSC). Un PSC est un ensemble de relations mathématiques, appelées *contraintes*, entre plusieurs *variables*. Chaque contrainte restreint les valeurs que peuvent prendre simultanément ces variables dans leur domaine. Ces domaines peuvent être finis ou continus.

Pour illustrer la notion de PSC, prenons l'exemple suivant :

$$\begin{cases} x \in [0, 100] \\ y \in [-40, 40] \\ 2 \cdot x + y = 70 \\ x + 3 \cdot y = 20 \end{cases} \tag{7.2}$$

Les variables de ce PSC sont x et y, de domaine initial $[0, 10]$ et $[-4, 4]$. Les domaines initiaux correspondent aux connaissances a priori sur les variables. Par exemple, dans le problème de modulation de vitesse, nous savons que le temps de parcours est strictement positif, mais aussi inférieur à l'horizon temporel T. Le domaine initial pour toutes les variables temporelles sera donc $[0, T]$.

[2] Notons que ce choix est uniquement destiné à simplifier l'expression des contraintes 7.6 page 187. Cette zone rectangulaire est facilement généralisable à un polygone à k côtés, en introduisant les k équations de droite qui le délimitent.

[3] Pour un certain type de robots, il peut également exister une vitesse minimale ν_{min} (les robots aériens, notamment, qui doivent se maintenir en l'air). Dans ces conditions, il suffira de remplacer $v^r \leq \nu_{max}$ par $v^r \in [\nu_{min}, \nu_{max}]$ dans toutes les contraintes.

Ces variables sont liées par deux contraintes linéaires. Ces contraintes modélisent en fait l'intersection entre les droites d'équation $2 \cdot x + y = 70$ et $x + 3 \cdot y = 20$.

Pour résoudre ce PSC, on utilise des algorithmes de filtrage, qui vont élaguer les domaines des variables en supprimant les parties inconsistantes, c'est à dire entrant en contradiction avec les contraintes. Ces algorithmes de filtrage sont généralement combinés à des stratégies de recherche pour sélectionner les solutions qui ont certaines caractéristiques, par exemple la minimisation d'un critère.

Si les domaines sont discrets, alors nous pouvons énumérer les valeurs domaines de chacune des variables, et vérifier s'il est possible de satisfaire les contraintes. Ce test est appelé l'arc-consistance [Mac77].

Pour illustrer ce mécanisme, prenons le problème (7.2) ci-dessus. Initialement, nous savons que $x \in [-40, 40]$ et que $y \in [0, 100]$. Voyons comment la première contrainte $2 \cdot x + y = 70$ nous permet de réduite ces domaines.

Cette contrainte peut se réécrire $x = 35 - y/2$. Parcourons ensuite les valeurs de y. Tout d'abord, $y = -40$ implique nécessairement $y = 55$. Ensuite $y = -39$ est inconsistante car $35 - 39/2$ n'est pas entier, alors que x doit l'être. Puis $y = -38$ donne $x = 54$. Puis $y = -37$ est de nouveau inconsistante. Et ainsi de suite, jusqu'à $y = 40$, qui donne $x = 15$.

Nous en déduisons que toutes les valeurs de x en dehors de l'intervalle $[15, 55]$ sont inconsistantes, mais également toutes les valeurs impaires de y.

Après cette étape, les nouveaux domaines pour x et y sont donc :

$$\begin{array}{rcl} x & \in & [15, 55] \\ y & \in & \{-40, -38, -36, ..., 36, 38, 40\} \end{array}$$

Ce processus est ensuite itéré sur toutes les variables et sur toutes les contraintes, jusqu'à stabilisation des domaines. Dans le problème (7.2), on obtient finalement $x \in \{38\}$ et $y \in \{-6\}$, ce qui correspond à la position de l'intersection entre les droites modélisées par les contraintes.

Si les domaines sont continus, nous ne pouvons plus énumérer les valeurs des variables. Une solution est d'appliquer la consistance d'arc uniquement sur les bornes des domaines. Cette consistance, beaucoup plus faible, est appelée la 2B-consistance [Lho93].

Voyons ce que donne un filtrage par 2B-consistance sur le problème (7.2). Si nous exploitons la première contrainte et $y \in [-40, 40]$, nous obtenons $x \in [15, 55]$, comme précédemment. Puis, en exploitant la deuxième contrainte et qu'à présent $x \in [15, 55]$, il vient $y \in [-11.66666, 6.66666]$, et ainsi de suite :

$$\begin{array}{ll} x \in [34.16666, 40.83333] & y \in [-6.94444, -4.72222] \\ x \in [37.36111, 38.42222] & y \in [-6.14074, -5.78704] \\ x \in [37.89352, 38.07037] & y \in [-5.96451, -6.02346] \\ x \in [37.98226, 38.01173] & y \in [-5.99409, -6.00391] \\ x \in [37.99704, 38.00196] & y \in [-5.99901, -6.00065] \\ ... & \end{array}$$

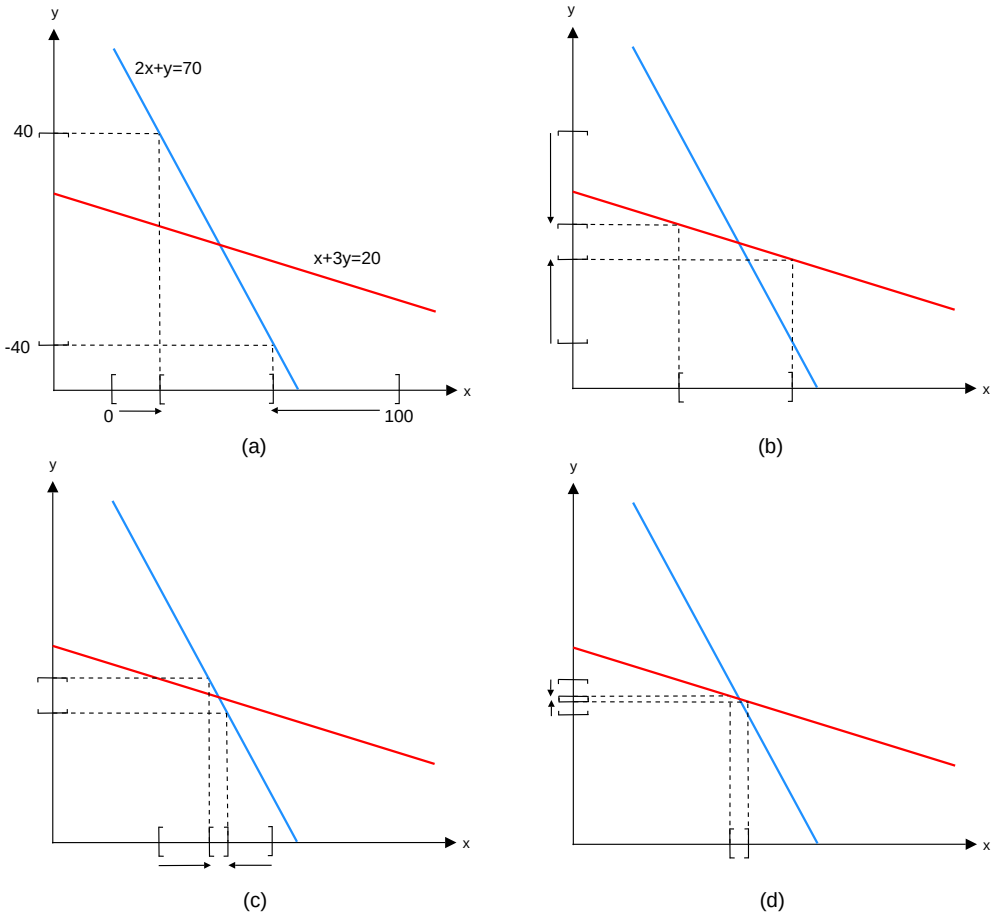

FIG. 7.4 – Filtrage par 2B-consistance
La figure présente les trois premières itérations du filtrage. A chaque itération, un domaine est réduit par projection sur la droite représentant une contrainte.

Graphiquement parlant, ce processus de filtrage peut être interprété comme suit : à chaque itération, les domaines de x et y sont tour à tour projetés sur les droites représentant les contraintes. Ces domaines convergent respectivement vers $\{38\}$ et $\{-6\}$, l'intersection des droites, comme illustré dans la figure 7.4.

Cette convergence pouvant être lente, elle est généralement interrompue (tout comme l'était la descente du gradient dans le chapitre 4). Pour ce faire, on trouve l'utilisation des conditions d'arrêt suivantes :

- Une largeur de borne ω : le processus de filtrage est interrompu si la taille des domaines a été réduite de moins de ω à la dernière itération. On parle alors de $2B(\omega)$-consistance.
- Un contrat de temps T_{max} : le processus de filtrage est interrompu si le temps de calcul dépasse T_{max} ;

Il est important de comprendre que le sens des domaines 2B-consistants est beaucoup moins fort que celui des domaines arc-consistants. En effet, dans le cas de domaines arc-consistants, nous avons la certitude que *toutes* les valeurs de ce domaine sont solutions (puisque nous pouvons les vérifier une par une). Dans le cas de domaines 2B-consistants, nous savons uniquement qu'il existe

potentiellement des solutions dans le domaine. En particulier, le domaine peut contenir des valeurs non solutions.

Ainsi, extraire des solutions à partir de domaines 2B-consistants nécessite de prendre des précautions. C'est pourquoi même si les sections 4 et 5 utilisent toutes deux la PPC pour effectuer la modulation de vitesse, les modélisations sont assez différentes.

4 Modulation de vitesse par PPC sur domaines finis

4.1 Définition du Problème de Satisfaction de Contraintes (PSC)

Le PSC modélisant notre problème de modulation de vitesse se divise en deux catégories de contraintes : (1) les contraintes relatives au suivi du chemin et (2) les contraintes relatives à l'évitement des obstacles mobiles.

4.1.1 Suivi du chemin par le robot

Nous savons que le robot doit suivre un chemin précalculé $\mathcal{P}$, composé de $n+1$ segments de droite : $[V_0, V_1]$, $[V_1, V_2]$, ..., $[V_{n-1}, V_n]$, avec $V_0 = A$ et $V_n = B$.

Considérons le mouvement du robot sur un segment arbitraire $[V_{i-1}, V_i]$. Si l'on note respectivement v_i^r et d_i la vitesse du robot, le temps de parcours et la distance parcourue sur ce segment et t_i la date d'arrivée en V_i, le suivi du chemin $\mathcal{P}$ par le robot peut être modélisé par les contraintes suivantes :

$$\forall i \in [1, n] : \begin{cases} t_i \in [0, T] \\ v_i^r \leq \nu_{max} \\ t_i = t_{i-1} + d_i / v_i^r \end{cases} \tag{7.3}$$

Notons que d_i n'est pas une variable, mais une constante. Elle est donnée par $l_i - l_{i-1}$, où l_i est l'abscisse curviligne de V_i sur $\mathcal{P}$ (connue).

Le domaine des variables temporelles t_i est discrétisé en utilisant un pas constant τ. Ce pas dépend des données du problème, plus particulièrement de l'espacement des viapoints et de la vitesse maximale du robot.

On peut par exemple choisir $\tau = d_{min}/\nu_{max}$, où d_{min} est plus petite distance entre deux viapoints. Dans les missions de drones aériens par exemple, on a $d_{min} \approx 100m$, $\nu_{max} \approx 300km/h$, ce qui aboutit à $\tau \approx 1s$.

4.1.2 Evitement des obstacles mobiles

Comme nous l'avons vu dans la définition du problème, chaque obstacle mobile O_j effectue une série de mouvements rectilignes uniformes, c'est à dire qu'il se déplace successivement sur des segments $[M_{k-1}, M_k]$ à vitesse constante v_k.

Si une collision entre O_j et le robot a lieu sur la portion $[M_{k-1}, M_k]$ cela signifie qu'il existe une date t_c pendant laquelle le robot, de coordonnées (x_r, y_r) se trouve dans la zone rectangulaire entourant O_j, de coordonnées (x_j, y_j). Ceci est illustré dans la figure 7.5.

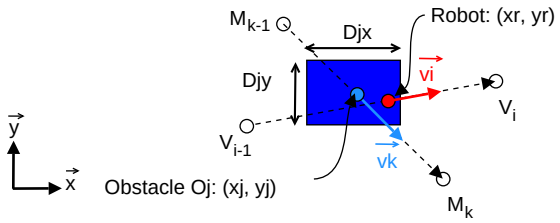

FIG. 7.5 – Collision avec un obstacle mobile
Dans le cas d'une collision entre le robot et un obstacle mobile O_j, le robot (en rouge) appartient nécessairement à la zone de sécurité entourant cet obstacle (rectangle bleu).

Cette situation peut être modélisée par les contraintes suivantes :

$$\begin{cases} a_i = y_{i-1} - y_i \\ b_i = x_i - x_{i-1} \\ c_i = y_i \cdot x_{i-1} - x_i \cdot y_{i-1} \\ a_i \cdot x_r + b_i \cdot y_r + c_i = 0 \\ x_r \geq x_{i-1} \\ x_r \leq x_i \\ y_r \geq y_{i-1} \\ y_r \leq y_i \end{cases} \tag{7.4}$$

[Le robot se déplace sur le segment $[V_{i-1}, V_i]$, d'équation $a_i \cdot x_r + b_i \cdot y_r + c_i = 0$]

$$\begin{cases} d_{kx} = x_k - x_{k-1} \\ d_{ky} = y_k - y_{k-1} \\ d_k = \sqrt{{d_{kx}}^2 + {d_{ky}}^2} \\ v_{kx} = v_k \cdot (d_{kx}/d_k) \\ v_{ky} = v_k \cdot (d_{ky}/d_k) \\ t_c \in D_c \\ x_j = v_{j_x} \cdot (t_c - t_{k-1}) + x_{k-1} \\ y_j = v_{j_y} \cdot (t_c - t_{k-1}) + y_{k-1} \end{cases} \tag{7.5}$$

[L'obstacle se déplace sur le segment $[M_{k-1}, M_k]$ à vitesse v_k, en partant à la date t_{k-1}. d_k représente la distance entre M_{k-1} et M_k, de composantes d_{kx} et d_{ky}]

$$\begin{cases} x_r \geq x_j + D_{j_x}/2 \\ x_r \leq x_j - D_{j_x}/2 \\ y_r \geq y_j + D_{j_y}/2 \\ y_r \leq y_j - D_{j_y}/2 \end{cases} \tag{7.6}$$

[Le robot est à l'intérieur de la zone de sécurité rectangulaire entourant l'obstacle, de dimensions $D_{j_x} \times D_{j_y}$]

Notons que les seules variables du problème sont (x_r, y_r), (x_j, y_j) et t_c, correspondant aux entités mobiles de la scène (le robot, l'obstacle) et l'instant de leur possible collision. Tous les autres éléments sont fixes (points de passage du robot et de l'obstacle) et donc les quantités qui en découlent

$(a_i, b_i, c_i, (x_{i-1}, y_{i-1}), (x_i, y_i)$ dans l'équation 7.4 ; $d_k = (d_{kx}, d_{ky})$, $v_k = (v_{kx}, v_{ky})$ dans l'équation 7.5) sont des constantes.

A la fin du processus de filtrage, le domaine D_c de la variable t_c contiendra toutes les dates de collisions possibles entre l'obstacle et le robot, chacun se déplaçant sur un segment de sa trajectoire. Il est tout à fait possible que ce domaine devienne vide, ce qui traduira l'absence de collision.

A partir de cette information, nous pouvons construire, pour chaque viapoint V_i, l'intervalle des *dates interdites*, noté T_i^{int}. Cet intervalle a la signification suivante : si $t_i \in T_i^{int}$, alors le robot entrera en collision avec un obstacle entre les viapoints V_{i-1} et V_i. Il est construit comme suit :

1. Au départ, $T_i^{int} \leftarrow \emptyset$;
2. Pour chaque obstacle, et pour chacun de ses mouvements : $T_i^{int} \leftarrow T_i^{int} \cup D_c$.

Par construction, chaque intervalle ainsi T_i^{int} est une union de sous-intervalles. L'ensemble de ces sous-intervalles constituent une discrétisation des surfaces interdites, comme on peut le voir dans figure 7.6. La finesse de cette discrétisation dépend de l'espacement des viapoints (résolution sur l'axe $\vec{l}$) et du pas de temps τ (résolution sur l'axe $\vec{t}$).

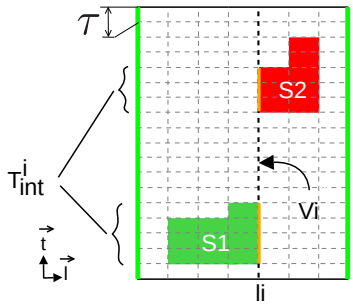

FIG. 7.6 – Intervalles des dates interdites (domaines finis)

L'ensemble des intervalles des dates interdites T_i^{int} constituent une discrétisation des surfaces interdites S_j de la figure 7.1b.

A partir de ces intervalles, une première idée pour modéliser l'évitement des obstacles mobiles consisterait à utiliser les contraintes suivantes :

$$\forall i \in [1, n] : \quad t_i \notin T_i^{int} \tag{7.7}$$

Toutefois, ces contraintes sont absolues et non relatives. En réalité, les variables t_i sont liées de façon implicite. Si une variable t_{i-1} impose, par sa valeur, de contourner une surface interdite par un coté (le bas, par exemple), alors il devra en être de même pour la variable suivante t_i. Si ce n'est pas le cas, bien que les points $P_{i-1} = (l_{i-1}, t_{i-1})$ et $P_i = (l_i, t_i)$ soient tous deux en dehors de la surface interdite (c'est à dire t_{i-1} et t_i respectent bien la contrainte 7.7), il n'en est pas forcément de même pour le segment $[P_{i-1}, P_i]$.

Ceci est illustré dans la figure 7.7 ci-après.

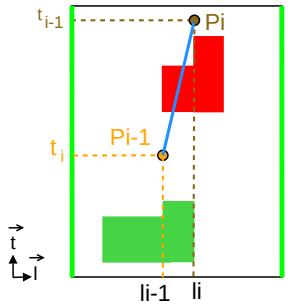

FIG. 7.7 – Faiblesse des contraintes 7.7
Deux points de part et d'autre d'une surface interdite ne peuvent être reliés sans entraîner de collision. Hors cette situation n'est pas interdite par les contraintes 7.7.

Pour éviter ce problème, une solution consiste à introduire des variables binaires b_j dans les contraintes, modélisant le côté de contournement de la surface interdite S_j. L'affectation $b_j = 0$ matérialise un contournement de S_j par le bas, et $b_j = 1$ par le haut.

En utilisant ces variables, les contraintes traduisant l'évitement des obstacles mobiles deviennent :

$$\forall i \in [1,n] : \quad T_i^{int} = [\underline{t_i^1}, \overline{t_i^1}] \cup ... \cup [\underline{t_i^s}, \overline{t_i^s}] \tag{7.8}$$

[L'intervalle des dates interdites T_i^{int} est une union de sous intervalles. Chaque sous-intervalle $[\underline{t_i^j}, \overline{t_i^j}]$ modélise l'ensemble des intersections entre la droite $l = l_i$ et la surface interdite S_j. s est le nombre total des surfaces interdites]

$$\forall j \in [1,s] : \begin{cases} b_j \in \{0,1\} \\ t_i \geq \underline{t_i^j} - T \cdot (1 - b_j) \\ t_i \leq \underline{t_i^j} + T \cdot b_j \end{cases} \tag{7.9}$$

[Le point de contournement $P_i = (l_i, t_i)$ est soit au dessus de la surface S_j, soit en dessous]

Puisque tous les pas de temps t_i partagent les mêmes variables binaires b_j, les points de contournements associés $P_i = (l_i, t_i)$ sont forcés de contourner la surface S_j de la même manière.

Prenons l'exemple d'un contournement par le bas. Le fait que $P_i = (l_i, t_i)$ passe en dessous de la surface S_j est équivalent à $t_i \leq \underline{t_i^j}$. Si cette condition est vérifiée pour un indice $i \in [1,n]$, alors la dernière contrainte du système 7.9 imposera $b_j = 0$, et donc $t_k \leq \underline{t_i^j}$ pour tous les indices $k \in [1,n]$.

Ceci est illustré dans la figure 7.8a ci-après.

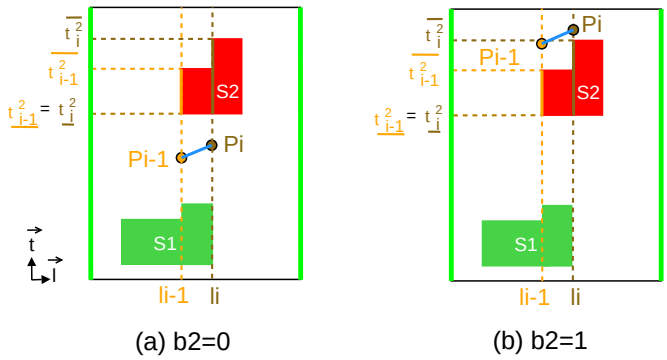

FIG. 7.8 – Illustration des contraintes 7.9
(a) contournement de la surface S_2 par le bas : $t_{i-1} \leq \underline{t^2_{i-1}} \Leftrightarrow b_2 = 0 \Leftrightarrow t_i \leq \underline{t^2_i}$; (b) contournement de la surface S_2 par le haut : $t_{i-1} \geq \underline{t^2_{i-1}} \Leftrightarrow b_2 = 1 \Leftrightarrow t_i \geq \underline{t^2_i}$.

4.1.3 Résumé

Le PSC modélisant le problème modulation de vitesse sur domaines finis est le suivant :

- Variables :
  - t_i ($i \in [1, n]$) : date de passage au viapoint V_i ; n est le nombre de viapoints sur le chemin $\mathcal{P}$.
  - v_i^r ($i \in [1, n]$) : vitesse du robot entre V_{i-1} et V_i.
  - b_j ($j \in [1, s]$) : côté de contournement de la surface interdite S_j dans l'espace-temps. s est le nombre total de surfaces interdites, c'est à dire le nombre d'intersections entre les trajectoires d'obstacles mobiles et le chemin $\mathcal{P}$.

- Contraintes :
  - Equation 7.3 : suivi du chemin $\mathcal{P}$,
  - Equations 7.8 et 7.9 : évitement des obstacles mobiles (contournement des surfaces interdites).

Ce PSC contient $2n + s$ variables et jusqu'à $(1 + 2s) \cdot n$ contraintes, toutes linéaires[4]. Pour le problème de la figure 7.1a, par exemple, on a $n = 8$ (en comptant les points de départ et d'arrivée) et $s = 2$, ce qui aboutit à 18 variables et 40 contraintes maximum.

Notons que les équations 7.4 à 7.6 ne font pas partie du PSC. Elles sont utilisées dans une étape de prétraitement, afin de calculer les intervalles des dates interdites T_i^{int}.

4.2 Extraction de la solution optimale

Si nous fournissons le PSC décrit ci-dessus à un résolveur sur domaines finis, celui-ci va retourner toutes les valeurs solutions du problème. En particulier, nous disposerons, après processus de filtrage, de toutes les dates d'arrivée possibles pour chaque point de passage V_i. Ces dates de passages sont matérialisées par les domaines $[t_i^-, t_i^+]$, associés à chaque variable t_i.

[4] Après le changement de variable $v_i^{r\prime} = 1/v_i^r$.

Pour ensuite obtenir un profil de vitesse permettant d'atteindre le point d'arrivée B au plus tôt, une solution consisterait à ajouter la contrainte $t_A = t_0^-$ dans le modèle (départ de A au plus tôt), résoudre de nouveau le PSC, ajouter la contrainte $t_1 = t_1^-$ (atteinte du viapoint V_1 au plus tôt), résoudre de nouveau le PSC, et ainsi de suite jusqu'au point d'arrivée $B = V_n$.

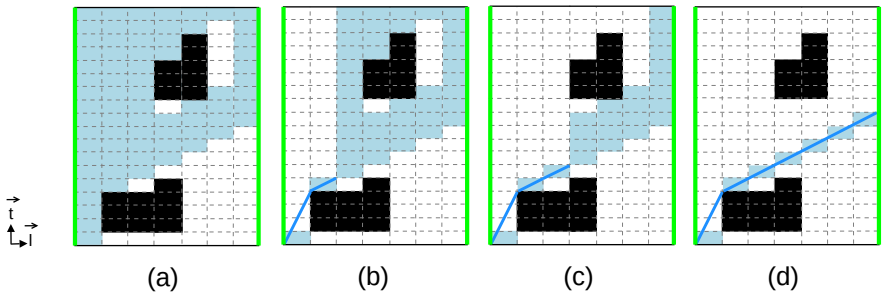

FIG. 7.9 – Extraction de la solution optimale dans les domaines finis
De (a) à (d), les dates de passage t_i associées aux viapoints V_i sont progressivement affectées à la borne inférieure de leur domaine valide (zone bleu clair). Le profil de vitesse optimal (trait bleu foncé) est progressivement déduit.

Cette façon de procéder est bien entendu très inefficace. Nous pouvons obtenir le même résultat en une résolution unique, en énumérant astucieusement les variables puis leur valeurs. Une possibilité est la stratégie d'énumération suivante :

- Ordre d'énumération des variables : t_0, v_0, t_1, v_1, ..., t_n, v_n ; notons qu'il ne sert à rien d'énumérer les variables binaires b_j, car leur valeur est implicitement fixée par les variables t_i et les contraintes 7.9.
- Ordre d'énumération des valeurs :
  - Croissant pour les variables t_i (arrivée au plus tôt)
  - Décroissant pour les variables v_i^r (déplacement au plus vite)

Cette stratégie devra être communiquée au résolveur en même temps que le PSC.

4.3 Problèmes d'incorrection dus à l'utilisation des domaines finis

L'utilisation des domaines finis nécessite de discrétiser toutes les grandeurs du problème, qui sont continues par nature, puisqu'elles modélisent des phénomènes physiques. En particulier, dans notre approche, le temps est échantillonné en utilisant un pas constant τ.

L'interprétation de cet échantillonnage est la suivante : si une propriété est vraie au pas de temps $p_k = k\tau$, alors elle est considérée comme vraie jusqu'au prochain pas $p_{k+1} = (k+1)\tau$. C'est en particulier le cas des collisions avec les obstacles mobiles : si, d'après les contraintes 7.4 à 7.6, aucune collision n'a lieu entre les points de passage V_{i-1} et V_i au pas de temps p_k, alors il est implicitement supposé qu'aucune collision n'aura lieu entre ces points jusqu'au pas de temps p_{k+1}.

Graphiquement parlant, le profil de vitesse calculé sous cette hypothèse contourne une discrétisation des surfaces interdites de l'espace-temps, et non les surfaces interdites initiales. Or, comme

on peut le voir sur la figure 7.10, cette discrétisation n'est que partiellement englobante, c'est à dire que des parties de la surface initiale (polygone rouge) peuvent potentiellement dépasser de la surface discrétisée (cases rouges).

Dans ces conditions, l'évitement des surfaces discrétisées, imposé par les contraintes 7.9, ne garantit l'évitement des surfaces initiales. Ainsi, notre approche, telle qu'elle est décrite dans cette section, est incorrecte : des collisions non détectées pendant l'étape de modulation de vitesse pourront avoir lieu lors de l'exécution réelle de la trajectoire par le robot.

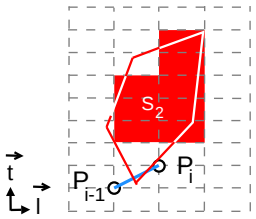

FIG. 7.10 – Incorrection due à l'approximation des surfaces interdites
Le segment $[P_{i-1}, P_i]$ contourne bien la surface S_2 discrétisée (cases rouges) mais pas la surface S_2 initiale (polygone rouge).

Il est important de comprendre que ce problème d'incorrection n'est pas dû à l'utilisation de la PPC, mais à la discrétisation du temps. En effet, toutes les approches de la littérature où le temps est discrétisé sont également incorrectes. C'est notamment le cas des approches à base de décomposition cellulaire, par exemple celles proposées par Fraichard [Fra93] ou van den Berg [vdBO05]. Pour limiter au plus l'apparition de ces problèmes, les auteurs suggèrent d'échantillonner avec un pas de temps τ très faible. En effet, si τ tend vers 0, alors la probabilité de non détection des collisions tend également vers 0.

Toutefois, cette démarche n'est pas satisfaisante, car elle ne fait qu'atténuer le problème sans le résoudre. De plus, la diminution de τ augmente considérablement la taille des domaines liés aux variables t_i (et des autres variables, par effet de bord) et donc le temps de calcul lié à l'opération de filtrage, qui procède à l'énumération de ces domaines.

Si nous reprenons l'exemple des missions de drones aériens, l'horizon temporel (pour une mission de courte durée) est de l'ordre de $T = 1h$, et un pas de temps raisonnable est $\tau = 1s$ (comme vu dans la partie 4.1.1). Les domaines des variables t_i contiennent donc chacun 3600 éléments.

Comme dans le chapitre 4, pour résoudre le problème sans hausse importante du temps de calcul, nous devons le prendre à sa source, c'est à dire éviter la discrétisation de quantités initialement continues, en particulier le temps.

Pour cela, nous allons proposer, dans la section suivante, d'exprimer les contraintes non pas sur domaines finis, mais sur domaines continus, ce qui est plus adapté à la nature du problème.

5 Modulation de vitesse par PPC sur domaines continus

Comme nous l'avons expliqué dans la section 3, les domaines 2B-consistants ont un sens beaucoup plus faible que les domaines arc-consistants. En effet, à la fin du processus de filtrage, nous savons simplement que s'il existe des solutions au problème, alors elles appartiennent à ces domaines. Mais nous n'avons aucune garantie que l'ensemble des valeurs des domaines sont des solutions.

Une autre façon de voir les choses est que ces domaines 2B-consistants encadrent les solutions par excès, de façon plus ou moins large[5]. Par exemple, le domaine $D_x = [-10, 10]$ est 2B-consistant vis à vis de la contrainte $x + x = 0$. Il encadre donc, de façon très large, l'unique solution du problème, c'est à dire $x = 0$.

Nous allons utiliser cette propriété en notre faveur, en modélisant le dual de notre problème initial, décrit dans la section 2. Cette fois, les contraintes vont être utilisées non pas pour modéliser l'espace *valide* (zone bleu clair dans la figure 7.9) mais l'espace *invalide*.

Dans ces conditions, l'espace invalide que nous obtiendrons après processus de filtrage (et en particulier les surfaces interdites) sera éventuellement surestimé mais *jamais* sous-estimé. Ainsi, contrairement à la PPC sur domaines finis, notre approche est ici correcte : tout chemin tracé en dehors de cet espace invalide sera garanti d'être sans collision.

5.1 Définition du PSC

Les zones non accessibles de l'espace-temps se divisent en deux catégories : (1) les zones dues à une vitesse excessive du robot et (2) les zones dues aux collisions avec les obstacles mobiles. L'espace invalide évoqué ci-dessus correspond à la fusion de l'ensemble de ces deux types de zones.

5.1.1 Zones inaccessibles dues à une vitesse excessive

Considérons le mouvement du robot sur un segment arbitraire $[V_{i-1}, V_i]$. Si l'on note respectivement v_i^r, t_i et d_i la vitesse du robot, le temps de parcours et la distance parcourue sur ce segment, nous avons vu dans la partie 4.1.1 que ces quantités étaient liées par :

$$t_i = t_{i-1} + d_i / v_i^r$$

En utilisant cette relation, nous pouvons modéliser une arrivée prématurée au viapoint V_i. En effet, toutes les dates de passage t_i vérifiant cette relation avec $v_i^r > \nu_{max}$ sont par définition inaccessibles, car le drone ne peut aller plus vite que ν_{max}.

Ce raisonnement est applicable sur tous les segments du chemin $\mathcal{P}$. Ainsi, en partant de $A = V_0$ le plus tôt possible (i.e. à 0), une arrivée prématurée du robot sur $\mathcal{P}$ peut être modélisée par les contraintes suivantes :

$$\begin{array}{l} t_0^{prem} = 0 \\ \forall i \in [1, n] : \left\{ \begin{array}{l} t_i^{prem} \in [0, T] \\ v_i^r > \nu_{max} \\ t_i^{prem} = t_{i-1}^{prem} + d_i / v_i^r \end{array} \right. \end{array} \tag{7.10}$$

[5] Cette largeur dépend de la nature des contraintes utilisées : linéarité, monotonie des fonctions, multi-occurence de variables, etc.

De façon similaire, en arrivant en $B = V_n$ le plus tard possible (i.e. à T), une arrivée tardive sur $\mathcal{P}$ peut être modélisée par :

$$\begin{aligned} & t_n^{tard} = T \\ & \forall i \in [1,n] : \begin{cases} t_i^{tard} \in [0,T] \\ v_i^r > \nu_{max} \\ t_i^{tard} = t_{i-1}^{tard} + d_i / v_i^r \end{cases} \end{aligned} \tag{7.11}$$

Ces contraintes génèrent un "plancher" et un "plafond" dans l'espace-temps, comme illustré dans la figure 7.11. Ces zones sont toutes deux délimitées par une droite de pente ν_{max} (en pointillés).

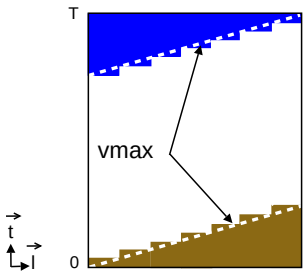

FIG. 7.11 – Zones inaccessibles dues à la vitesse maximale du robot
Ces zones se décomposent en deux parties : un plancher, correspondant à une arrivée prématurée du robot sur le chemin (en brun) et un plafond, correspondant à une arrivée tardive (en bleu).

5.1.2 Zones inaccessibles dues aux collisions avec les obstacles mobiles

Comme dans la partie 4.1.2, pour chaque viapoint V_i, on commence par calculer les intervalles des dates interdites T_i^{int}, en utilisant les contraintes 7.4, 7.5 et 7.6.

Sur domaines continus, l'ensemble des intervalles T_i^{int} discrétisent les surfaces interdites de l'espace-temps par un ensemble de bandes (et non plus de cases), comme on peut le voir dans la figure 7.12.

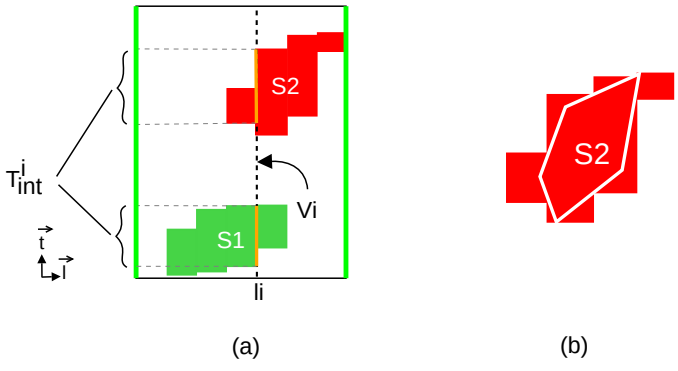

FIG. 7.12 – Intervalles des dates interdites (domaines continus)
(a) l'ensemble des intervalles T_i^{int} correspondent à une discrétisation des surfaces interdites S_j de la figure 7.1b ; (b) zoom autour de la surface S_2 : polygone = surface initiale, bandes rouges = discrétisation.

Ces bandes constituent une approximation par excès des surfaces interdites. Ceci signifie que, contrairement à la figure 7.10, la surface initiale (polygone blanc) est entièrement incluse dans sa version discrétisée (bandes rouges).

5.1.3 Fusion des zones inaccessibles

Les deux types zones inaccessibles décrites ci-dessus ne sont pas indépendantes. En effet, les contraintes de 7.4 à 7.6 induisent un décalage temporel des contraintes 7.10 et 7.11. Graphiquement parlant, les surfaces interdites provoquent une translation du plancher et du plafond de la figure 7.11, comme on peut le voir dans la figure 7.13a.

Prenons pour commencer le cas du plancher. Les contraintes 7.10 modélisent une arrivée prématurée à chaque viapoint V_i, par l'intermédiaire des variables t_i^{prem} : toute date inférieure t_i^{prem} aboutira à une vitesse de déplacement supérieure à la vitesse maximale du robot.

La présence d'obstacles mobiles vient étendre cette notion d'arrivée prématurée, et donc le plancher de l'espace-temps. En effet, si un obstacle O_j occupe V_i pendant la durée $D_j = [\underline{t_i^j}, \overline{t_i^j}]$, alors toute date d'arrivée appartenant à D_j sera également prématurée. En d'autres termes, si le robot atteint V_i dans l'intervalle de temps D_j, alors il devra attendre $\overline{t^j}$ que l'obstacle libère le point.

D'un point de vue plus formel, s'il existe une valeur de t_i^{prem} telle que $t_i^{prem} \in D_j$, alors t_i^{prem} devra être remplacé par $\overline{t_i^j}$ dans les contraintes 7.10. Ce remplacement dynamique est modélisé par les contraintes suivantes :

$$\forall i \in [1,n] : \quad T_i^{int} = [\underline{t_i^1}, \overline{t_i^1}] \cup ... \cup [\underline{t_i^s}, \overline{t_i^s}] \tag{7.12}$$

[Chaque sous-intervalle $[\underline{t_i^j}, \overline{t_i^j}]$ modélise l'occupation du point V_i par un obstacle]

$$\forall j \in [1,s] : \begin{cases} \underline{b_i^j} \in \{0,1\} \\ \overline{t_i^{prem}} \geq \underline{b_i^j} \cdot \underline{t_i^j} \\ \underline{t_i^j} > (1 - \underline{b_i^j}) \cdot t_i^{prem} \end{cases} \tag{7.13}$$

[La variable binaire $\underline{b_i^j}$ modélise le fait que $t_i^{prem} > \underline{t_i^j}$]

$$\forall j \in [1,s] : \begin{cases} \overline{b_i^j} \in \{0,1\} \\ \overline{t_i^j} \geq \overline{b_i^j} \cdot t_i^{prem} \\ t_i^{prem} > (1 - \overline{b_i^j}) \cdot \overline{b_i^j} \end{cases} \tag{7.14}$$

[La variable binaire $\overline{b_i^j}$ modélise le fait que $t_i^{prem} < \overline{t_i^j}$]

$$\forall j \in [1,s] : \begin{cases} b_i^j \in \{0,1\} \\ b_i^j = \underline{b_i^j} \cdot \overline{b_i^j} \end{cases} \tag{7.15}$$

[La variable binaire b_i^j modélise le fait que $t_i^{prem} \in [\underline{t_i^j}, \overline{t_i^j}]$. (conjonction des 2 contraintes précédentes), c'est à dire une intersection du plancher avec la surface interdite S_j à l'abscisse curviligne $l = l_i$]

$$\forall j \in [1,s] : \begin{cases} t_i^{prem'} \in [0,T] \\ t_i^{prem'} \leq b_i^j \cdot \overline{t_i^j} + (1-b_i^j) \cdot (t_{i-1}^{prem'} + d_i/v_i^r) \\ v_i^r > \nu_{max} \end{cases} \tag{7.16}$$

[Contraintes définissant le nouveau plancher, modélisé par les variables $t_i^{prem'}$. Si $b_i^j = 1$, alors la contrainte de type 7.10 est remplacée par la simple contrainte de borne $t_i^{prem} \leq \overline{t_i^j}$]

De façon similaire, les contraintes relatives au nouveau plafond de l'espace-temps sont :

$$\forall j \in [1,s] : \begin{cases} t_i^{tard'} \in [0,T] \\ t_i^{tard'} \geq b_i^j \cdot \underline{t_i^j} + (1-b_i^j) \cdot (t_{i-1}^{tard'} + d_i/v_i^r) \\ v_i^r > \nu_{max} \end{cases} \tag{7.17}$$

Comme on peut le voir dans la figure 7.13a, l'ensemble de ces contraintes génère des "ombres" derrière les surfaces interdites de l'espace-temps, obtenues de la façon suivante : si une bande B appartenant au plancher (resp. le plafond) entre en collision avec une surface interdite S, alors l'ensemble des bandes suivantes est translaté de bas en haut (resp. de haut en bas) jusqu'à ce que B atteigne le haut (resp. de bas) de S. Cette translation est illustrée par les flèches noires.

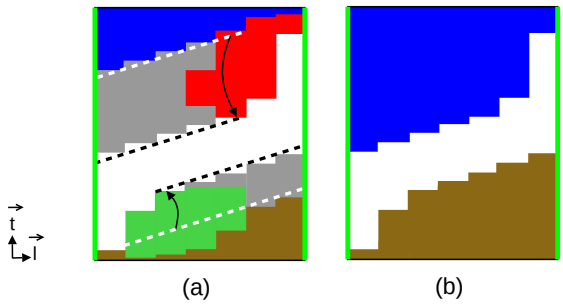

FIG. 7.13 – Effet des contraintes 7.12 à 7.17
(a) les contraintes génèrent des "ombres" (en gris) derrière les surfaces interdites, en translatant du plancher et du plafond de la figure 7.11 ; (b) plancher et plafond étendus ainsi obtenus.

On obtient ainsi de nouveaux plancher et plafond, dits *étendus*, modélisés respectivement par les variables $t_i^{prem'}$ et $t_i^{tard'}$, et illustrés dans la figure 7.13b. Ces zones invalides combinent les interdictions relatives aux capacités de propulsion du robot (vitesse maximale) et aux obstacles mobiles (collisions).

5.1.4 Résumé

Le PSC modélisant le problème modulation de vitesse sur domaines continus est le suivant :

– Variables :

- t_i^{prem}, t_i^{tard} $(i \in [1, n])$: variables modélisant une arrivée prématurée au point V_i, due à une vitesse excessive (supérieure à la vitesse maximale du robot). n est le nombre de viapoints sur le chemin $\mathcal{P}$.
- $t_i^{prem'}, t_i^{tard'}$ $(i \in [1, n])$: variables modélisant une arrivée prématurée au point V_i, due à une vitesse excessive ou à une collision avec un obstacle mobile.
- v_i^r $(i \in [1, n])$: variables modélisant une vitesse excessive du robot entre V_{i-1} et V_i.
- b_i^j $(i \in [1, n], j \in [1, s])$: variable binaire modélisant la collision j entre le robot et un obstacle mobile au point V_i, parmi les s collisions possibles.

- Contraintes :
  - Equations 7.11 et 7.11, matérialisant le plancher et le plafond initiaux (uniquement dus à une vitesse excessive) de l'espace-temps.
  - Equations de 7.12 à 7.17, matérialisant le plancher et le plafond étendus (intégrant les surfaces interdites) de l'espace-temps.

Ce PSC contient jusqu'à $(5 + 3s) \cdot n$ variables et $(2 + 8s) \cdot n$ contraintes. Pour le problème de la figure 7.1a, par exemple, on a $n = 8$ (en comptant les points de départ et d'arrivée) et $s = 2$, ce qui aboutit à 88 variables et 144 contraintes maximum.

On constate une augmentation assez importante de la taille du PSC par rapport aux domaines finis (nous avions au plus 18 variables et 40 contraintes). Cependant, nous verrons dans la section 7 que l'impact sur le temps de calcul n'est pas significatif, puisque la plupart des nouvelles variables sont binaires et n'engendrent au plus que 2 énumérations.

Notons que, comme pour les domaines finis, les équations 7.4 à 7.6 ne font pas partie du PSC. Elles sont utilisées dans une étape de prétraitement, afin pour calculer les intervalles des dates interdites T_i^{int}.

5.2 Extraction de la solution optimale

Il y a deux cas de figure après processus de filtrage par 2B-consistance :

- L'espace-temps est obstrué.

C'est à dire que le plancher touche une ou plusieurs fois le plafond dans l'espace-temps. Dans ces conditions, le problème n'a *a priori* pas de solutions. Cependant, nous ne pouvons pas certifier si c'est effectivement le cas, où s'il s'agit d'un effet de bord dû à l'utilisation de la 2B-consistance (qui peut surestimer l'espace invalide).

Pour limiter ces effets de bords, il est possible, entre autres, de :

- Régler les paramètres du résolveur : augmenter le contrat de temps T_{max} ou diminuer la largeur de borne ω (si utilisés) ;
- Reformuler les contraintes, notamment pour limiter les multi-occurences de variables ;
- Essayer d'autres types de consistances sur domaines continus, comme la Box-consistance [BMvH94].

- Il existe un couloir dans l'espace-temps.

Puisque l'espace invalide est éventuellement surestimé, mais jamais sous-estimé, nous avons la certitude que tout chemin tracé dans ce couloir est entièrement valide. Parmi ces chemins valides, un seul permet d'atteindre le point d'arrivée au plus tôt. Il est obtenu en reliant les parties supérieures du plancher, c'est à dire en choisissant pour chaque viapoint V_i du chemin la première date disponible, comme illustré dans la figure 7.14. Concrètement parlant, ceci consiste à choisir les bornes supérieures des domaines associés aux variables $t_i^{prem'}$ de notre PSC pour dates de passage.

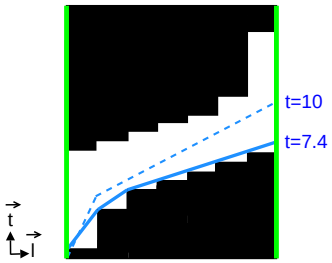

FIG. 7.14 – Extraction de la solution optimale dans les domaines finis
Solution optimale (trait continu bleu) obtenue en reliant les parties supérieures du plancher. Celle-ci permet d'arriver bien plus tôt (à $t = 7.4$) que la solution trouvée dans les domaines finis (trait pointillé bleu, arrivant à $t = 10$).

On peut remarquer qu'en plus de la garantie sur sa validité, la solution obtenue sur domaines continus possède un autre avantage sur celle obtenue sur domaines finis : l'absence d'approximation liée à la discrétisation du temps. En effet, si la première date valide pour visiter V_i est $\overline{t_i}$, il est possible d'approcher $\overline{t_i}$ avec une précision arbitraire sur les domaines continus, alors que dans les domaines finis, nous devrons choisir le premier pas de temps valide après $\overline{t_i}$.

Par exemple, si un pas de temps $\tau = 1min$ est utilisé et si $\overline{t_i} = 1.02min$, alors le premier pas de temps valide immédiatement supérieur à $\overline{t_i}$ est $t = 2min$. Ainsi, un retard de près d'une minute sera pris sur la visite de V_i par rapport à sa visite au plus tôt.

Ce retard s'accumule de viapoint en viapoint, et peut devenir assez important au fur et à mesure que le nombre de viapoints n augmente. Sur la figure 7.14 notamment, on peut constater que la solution sur domaines finis accuse un retard de 35% par rapport à la solution sur domaines continus.

6 Ajout de courants dans l'environnement

Dans cette section, nous allons illustrer que la PPC permet de changer les caractéristiques du problème à travers un exemple précis : l'ajout de courants dans l'environnement. Contrairement aux autres approches de modulation de vitesse, la prise en compte de ces courants ne nécessite que quelques ajustements dans le modèle introduit précédemment.

6.1 Reformulation du problème

Les hypothèses de travail sont que celles décrites dans la section 2, sauf que l'environnement contient à présent des Zones Elémentaires de Courant (ZEC), illustrées dans la figure 7.15.

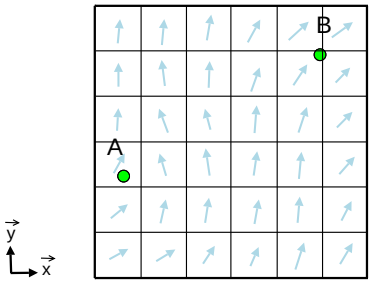

FIG. 7.15 – Environnement avec courants
En plus des éléments décrits dans la section 2, l'environnement contient à présent des Zones Elémentaires de Courant (ZEC) centrées autour d'échantillons (flèches bleues).

Une ZEC est une région polygonale dans laquelle le courant est homogène. La valeur du courant est celle d'un échantillon, c'est-à-dire un vecteur-vitesse issu d'une mesure ou d'une prévision.

Pour plus de détails sur les ZECs, notamment sur leur construction, se reporter au chapitre 4, à la page 111.

6.2 Introduction des viapoints artificiels

Les viapoints définissant le chemin $\mathcal{P}$ matérialisent des changements de direction (dus au contournement d'obstacles fixes, non représentés), mais ils ne tiennent pas compte des changements de courants d'une ZEC à une autre.

Nous allons donc artificiellement introduire de nouveaux viapoints, en procédant comme illustré dans la figure 7.16 : à chaque fois qu'un segment $[V_i, V_{i+1}]$ intersecte le bord d'une ZEC, alors un nouveau viapoint (en rouge sur la figure) est inséré entre V_i et V_{i+1}. Le chemin $\mathcal{P}$ et les bords des ZECs étant tous deux constitués de segments de droite, le calcul de ces intersections est simple à réaliser.

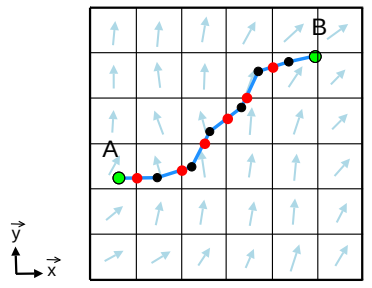

FIG. 7.16 – Viapoints artificiels
Les viapoints artificiels, en rouge, sont de nouveaux viapoints garantissant un courant constant entre deux viapoints successifs.

A la fin de cette étape, nous obtenons une nouvelle liste de viapoints $V' = \{V'_i\}_{i\in[1,n']}$, ayant la propriété suivante : entre deux viapoints successifs V'_i et V'_{i+1}, nous avons la garantie que le courant est constant. Ceci va simplifier l'écriture des contraintes introduites ci-après.

6.3 Nouvelles contraintes pour la vitesse maximale

Considérons le déplacement du robot sur un segment arbitraire $[V'_{i-1}, V'_i]$ du chemin $\mathcal{P}$. Pour ce mouvement, nous pouvons définir les quantités suivantes :

- $\vec{w_i}$ la vitesse du courant (que nous savons constante, par construction),
- $\vec{v_i^r}$ la vitesse du robot par rapport au sol (i.e. au repère $\mathcal{R}$),
- $\vec{v_i^w}$ la vitesse du robot par rapport au courant $\vec{w_i}$.

Nous savons, par principe de composition des vitesses, que les trois vecteurs ci-dessus sont liés par :

$$\vec{v_i^r} = \vec{v_i^w} + \vec{w_i} \tag{7.18}$$

De plus, comme nous voulons imposer au robot de rester sur le chemin $\mathcal{P}$, la vitesse du robot par rapport au sol, $\vec{v_i^r}$, doit être colinéaire au déplacement $\vec{d_i} = \overrightarrow{V'_{i-1}V'_i}$.

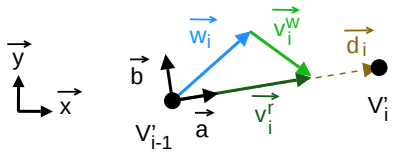

FIG. 7.17 – Suivi de chemin en présence de courants
Pour suivre le chemin $\mathcal{P}$ sur la portion $[V'_{i-1}, V'_i]$ la composition des vitesses $\vec{v_i^w}$ et $\vec{w_i}$ doit être colinéaire à $\overrightarrow{V'_{i-1}V'_i}$.

Dans le repère local $\mathcal{R}_i = (V'_{i-1}, \vec{a}, \vec{b})$, avec $\vec{a} = \vec{d^i}/d^i$ et $\vec{a} \perp \vec{b}$, cette situation, illustrée dans la figure 7.17, peut être modélisée par les contraintes suivantes :

$$\begin{cases} v_i^r = c_a^i + v_i^w{}_a \\ 0 = c_b^i + v_i^w{}_b \end{cases} \tag{7.19}$$

[composition des vitesses exprimée dans le repère local $\mathcal{R}^i$]

$$\begin{cases} w_a = c^i \cdot \cos\alpha \\ w_b = c^i \cdot \sin\alpha \\ \cos\alpha = \vec{w^i} \cdot \vec{d^i}/c^i \cdot d^i \end{cases} \tag{7.20}$$

[coordonnées de $\vec{w^i}$ dans $\mathcal{R}^i$]

Comme nous l'avions vu en introduction, la vitesse maximale du robot (par rapport au sol) dépend à présent du courant $\vec{w_i}$: elle augmente si w_i "pousse" le robot dans le sens du déplacement $\vec{d_i}$, et diminue sinon.

Ce phénomène peut être aisément traduit dans notre PSC, en remplaçant toutes les occurrences de $v_i^r > \nu_{max}$ par les contraintes 7.19 et 7.20 ci-dessus, et 7.21 ci-dessous, qui exprime la limite ν_{max} non plus par rapport au sol, mais par rapport au courant :

$$\begin{cases} v_i^{w2} = {v_i}_a^{w2} + {v_i}_b^{w2} \\ v_i^w > \gamma_{max} \end{cases} \tag{7.21}$$

[vitesse excessive relative au courant $\vec{w^i}$]

Ces substitutions augmentent légèrement la taille de notre PSC, puisqu'elles ajoutent $6n$ contraintes et $5n$ variables. La résolution du PSC obtenu se fait de façon strictement identique que précédemment.

Par ces quelques modifications, nous obtenons une méthode capable d'effectuer la modulation de vitesse du robot en présence de courants, ce qui n'avait jamais était réalisé auparavant, à notre connaissance.

Notons que cette méthode peut être sous-exploitée, pour résoudre des instances très simples de notre problème, comme par exemple, une tâche de suivi de chemin en présence de courants. Il s'agit d'uniquement compenser les perturbations induites par les courants pour maintenir le robot sur $\mathcal{P}$, sans évitement d'obstacles mobiles[6]. Cette instance particulière est très étudiée dans la littérature, et de nombreuses approches ont été proposées, comme la méthode des "champs de vitesse" [NBMB06]. Il serait intéressant d'étudier, à l'avenir, dans quelle mesure la PPC constitue une alternative à ces approches.

7 Résultats expérimentaux

Nous venons de voir qu'utiliser la PPC apportait un gain en flexibilité dans la modulation de vitesse. En effet, l'expressivité importante offerte par les contraintes permet de modéliser assez aisément de nombreuses extensions au problème de base.

A priori, le gain en flexibilité devrait s'accompagner d'une perte en performances. C'est pourquoi nous souhaitons apprécier dans cette partie l'évolution du temps de calcul en fonction de la donnée critique du problème : le nombre d'obstacles mobiles. En effet, il est essentiel que le temps de calcul reste raisonnable quand l'environnement devient encombré, car c'est l'intérêt même de la décomposition planification de chemin/modulation de vitesse (par rapport à des méthodes directes).

Le protocole expérimental a été le suivant :

- Nous avons utilisé les mêmes cartes des vents que dans les chapitres 4 et 5 : une base de 100 cartes collectées sur Metéo France [MF], contenant environs 1000 échantillons de vent (dimensions 28×35).

- Nous avons fait varier le nombre d'obstacles mobiles dans l'environnement, de $m = 1$ à $m = 20$.

- Pour chaque valeur de m, nous avons généré 500 cas de test. Chaque cas de test a été construit de la façon suivante :

[6] Dans ces conditions, les contraintes des sections 5.1.2 et 5.1.3 peuvent être supprimées.

1. Tirage aléatoire d'une carte des vents dans la base.
2. Placement des points A et B de façon aléatoire sur la carte.
3. Planification d'un chemin entre A et B, en utilisant l'algorithme de la propagation d'onde coulissante présenté dans le chapitre 4. Cet algorithme évite la génération de viapoints artificiels, puisque par construction, chaque segment constituant le chemin se trouve dans une zone homogène de courant.
4. Construction des obstacles mobiles. Chaque obstacle mobile est généré de la façon suivante : (a) tirage aléatoire de deux points sur les bords de la carte. Ces deux points définissent la direction de déplacement de l'obstacle. (b) tirage aléatoire des dimensions de l'obstacle (entre 1× et 5× les dimensions des ZEC). (c) tirage aléatoire de la vitesse de déplacement de l'obstacle (entre 0.5× et 2× la vitesse du robot).

Un cas de test est illustré dans la figure 7.18.

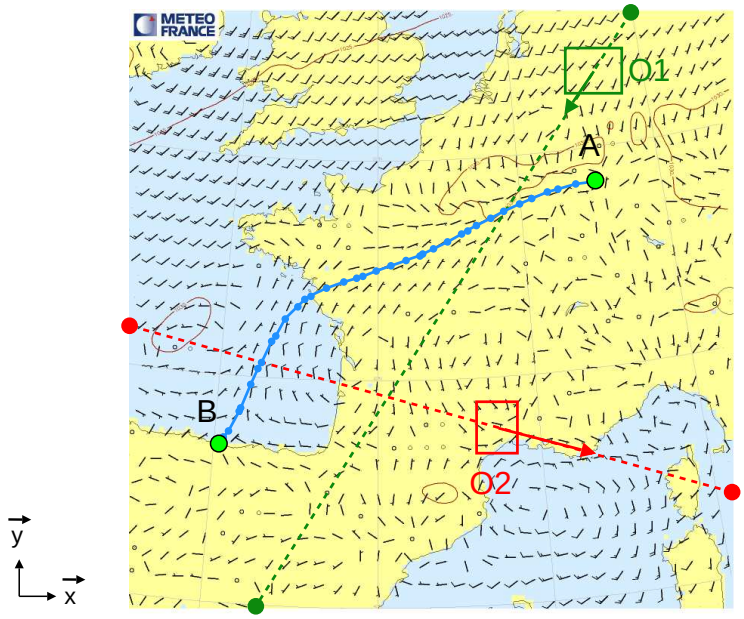

FIG. 7.18 – Un cas de test pour la modulation de vitesse
La figure présente un exemple d'environnement généré pour $m = 2$ obstacles mobiles. Le chemin (en bleu) est constitué de $n = 36$ viapoints, ce qui aboutit à un PSC d'environ 400 variables et 650 contraintes.

– Nous avons mesuré le temps de calcul moyen pour chaque valeur de m sur l'ensemble des cas de tests. Les tests ont été effectués en utilisant le résolveur Interlog [BT93], sur un PC cadencé à $1.7Ghz$ muni de $1Go$ de RAM. Les résultats obtenus ont été représentés dans la figure 7.19. En observant cette figure nous pouvons constater que :

1. Le temps de calcul reste raisonnable, puisqu'il n'atteint que $600ms$ en présence de 20 obstacles mobiles, ce qui correspond à un environnement relativement encombré. Notons que ces performances peuvent être considérablement améliorées en utilisant un résolveur plus performant, tel que RealPaver [GB06].

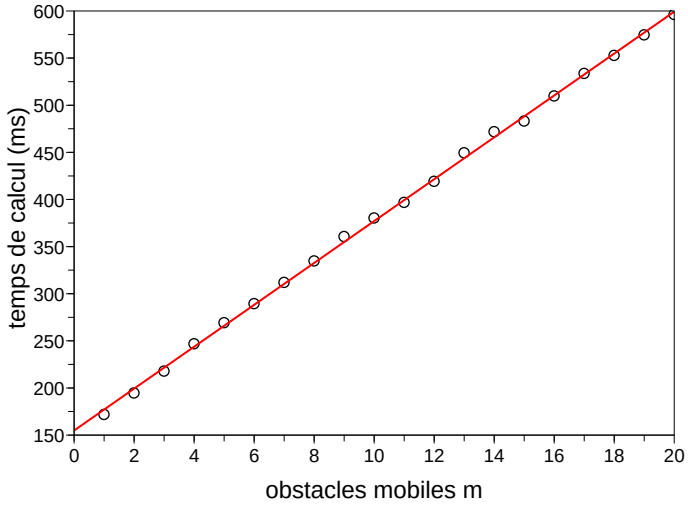

FIG. 7.19 – Performances de la modulation de vitesse par PPC
Temps de calcul (en ms) pour effectuer la modulation de vitesse du robot à base de contraintes, en présence de m obstacles mobiles. Points blancs = temps moyen; Ligne rouge = régression linéaire.

2. Sur la base de la régression illustrée en rouge (avec un coefficient de détermination $R^2 > 0.999$), il semble que le temps de calcul soit linéaire en fonction du nombre d'obstacles mobiles. Ceci paraît a priori logique, car la taille du problème est en $O(\widetilde{n} \cdot m)$, où $\widetilde{n}$ est le nombre moyen de viapoints sur le chemin sur l'ensemble des tests ($\widetilde{n} \approx 20$ dans nos tests). Toutefois, il serait intéressant de confirmer ces résultats expérimentaux par une étude théorique (comme nous l'avons fait pour la collision d'ondes dans le chapitre 6).

8 Conclusion

Dans ce chapitre, nous avons introduit une approche flexible pour effectuer l'étape de modulation de vitesse, basée sur la Programmation Par Contraintes (PPC). La PPC permet de modéliser le problème de modulation de vitesse "basique" (contenant uniquement des obstacles mobiles), puis d'étendre ce modèle à des contraintes plus complexes, qui n'étaient pas forcement prévues au départ.

Nous avons illustré cette flexibilité à travers l'ajout de courants dans l'environnement, initialement ignorés. L'impact de ces courants sur la vitesse du robot a pu être assez aisément pris en compte, par simple substitution de contraintes dans le modèle initial. Nous avons ainsi obtenu la seule méthode de la littérature, à notre connaissance, capable de réaliser une modulation de vitesse en présence de courants.

L'ensemble de ces résultats a été publié dans les actes du workshop PlanSIG'07 (Workshop of the UK Planning and Scheduling Special Interest Group). Le lecteur interessé pourra consulter [STR07].

Le défaut majeur de l'approche est que la sous-optimalité engendrée par la décomposition planification de chemin/modulation de vitesse est considérablement aggravée en présence de courants. En effet, quand l'environnement contient des courants, le chemin planifié dépend de la vitesse de

croisière du robot utilisée pour cette planification. Si cette vitesse est très grande par rapport à celle des courants, alors le chemin optimal entre A est B sera proche de la ligne droite. A mesure que cette vitesse diminue, ce chemin se déforme, pour épouser de plus en plus la direction des courants, afin que le robot se fasse porter le plus possible. Ceci est illustré dans la figure 7.20.

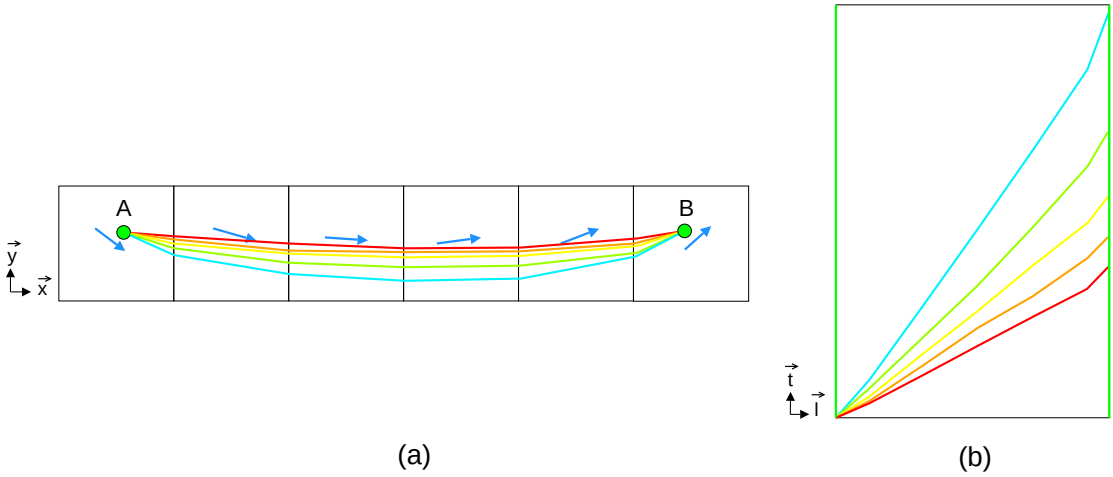

FIG. 7.20 – Déviation du chemin selon la vitesse de croisière du robot
(a) chemins optimaux entre A et B pour différentes vitesses de croisière du robot ; (b) profils de vitesses correspondant dans l'espace-temps. Bleu=$100km/h$ (vitesse du courant), rouge = $300km/h$.

Il serait donc intéressant de modifier les contraintes pour ne pas effectuer un suivi de chemin, mais un suivi de trajectoire. Le but est de maintenir le robot sur le chemin, mais aussi de respecter le profil de vitesse associé à ce chemin. Si une surface interdite écarte le robot du profil optimal, alors celui-ci devra accélérer ou ralentir pour le rejoindre à nouveau, comme illustré dans la figure 7.21. Ceci suppose bien entendu que la vitesse de croisière utilisée pour la planification de chemin laisse suffisamment de marge pour la modulation de vitesse. On pourra par exemple choisir une vitesse de croisière équidistante des vitesses minimales et maximales du robot.

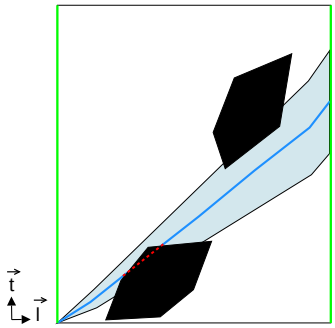

FIG. 7.21 – Rattrapage de profil
En plus d'éviter les surfaces interdites, le profil de vitesse du robot doit s'écarter le moins possible du profil optimal (en bleu foncé) déterminé lors de la planification de chemin. La marge de manoeuvre du robot est illustrée par la zone bleu clair (différences de vitesses admissibles par rapport à la vitesse de croisière).

Conclusions et perspectives

1 Contributions de la thèse

Dans cette thèse, nous avons introduit deux nouvelles techniques de planification prévisionnelle de trajectoire en présence de courants :

- La *propagation d'onde coulissante* [STR08a] permet de planifier des trajectoires de façon fiable même en présence de courants forts. Cette technique constitue une avancée par rapport aux techniques de planification existantes, qui peuvent, dans ce contexte particulier, manquer des solutions (problème d'incomplétude) ou pire, renvoyer des solutions physiquement non réalisables par le drone (problème d'incorrection).

- La *propagation d'onde symbolique* [STR09], permet de planifier des trajectoires en présence de courants variables dans le temps. Etant données des prévisions de courants, cette technique permet de calculer la date de départ minimisant le temps de parcours du drone, ainsi que la trajectoire optimale associée, ce qui, à notre connaissance, n'a pas été proposé à ce jour.

Nous avons également proposé deux autres contributions dans le thème plus large de la planification de trajectoire :

- Le concept de collision d'ondes [ST07], permettant la planification de multiples chemins (entre plusieurs points à visiter) de façon efficace. Nous avons montré que ce concept permettait de diviser le temps de calcul jusqu'à 4, comparé à l'utilisation répétée de la propagation d'ondes classique.

- L'utilisation de la programmation par contraintes pour la modulation de vitesse du drone [STR07]. Nous avons montré que la programmation par contraintes permettait d'intégrer des changements des caractéristiques du problème de façon assez aisée, par l'ajout, la modification ou la suppression de contraintes dans le modèle.

2 Applications

L'ensemble de ces techniques a été mis en oeuvre au sein d'un démonstrateur, *Airplan*. Il permet la préparation de trajectoires pour drones en mission. Ces missions consistent à visiter un ensemble de sites stratégiques en minimisant le temps de parcours total du drone. Les contraintes gérées par le planificateur sont les suivantes :

- Contraintes de précédence immédiate (éventuellement partielles) sur les sites
- Fenêtres temporelles sur les sites (c'est à dire une date au plus tôt et une date au plus tard pour sa visite)
- Zones de vent homogène rectangulaires
- Obstacles fixes rectangulaires
- Obstacles mobiles circulaires, effectuant une série de mouvements rectilignes uniformes.

La première version d'Airplan a été présentée dans d'Airplan [ST06] et dans [Sou07]. Elle était basée sur une décomposition planification de chemin/modulation de vitesse. La planification de chemin était effectuée par application de la propagation d'onde classique (présentée page 35 de l'état de l'art) et la modulation de vitesse par application de l'algorithme des lignes brisées (présentée page 60 de l'état de l'art).

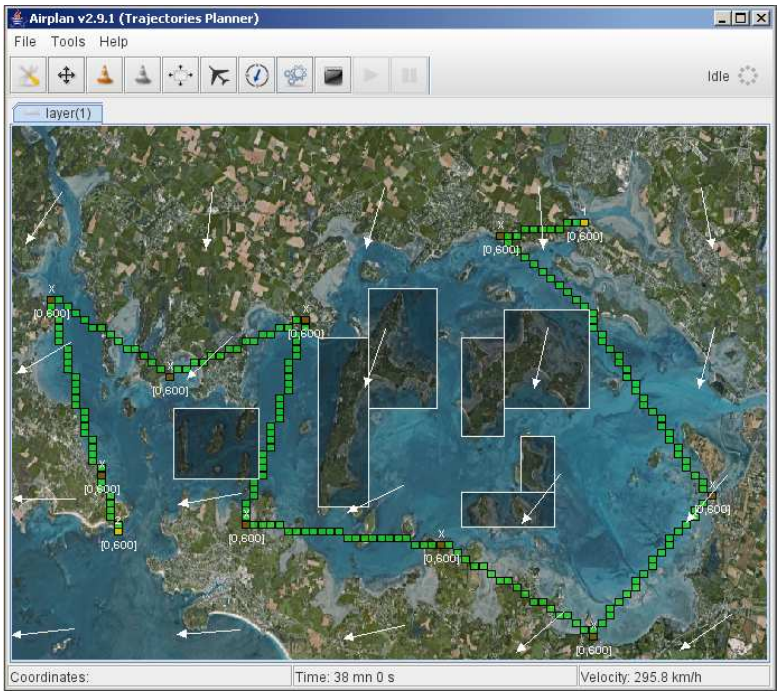

FIG. 7.22 – Exemple de trajectoire planifiée par Airplan
La figure présente en vert la trajectoire planifiée par Airplan pour une mission contenant : 11 points de passage, 7 obstacles fixes (en gris foncé), et 20 zones homogènes de vent (flèches blanches). La planification a duré une seconde environ.

Cette version a été intégrée dans un démonstrateur plus conséquent, *Aerial*, développé par Paul-Edouard Marson. Ce démonstrateur permet la réalisation de missions coordonnées par plusieurs drones, chaque drone étant muni d'un planificateur embarqué (Airplan). Pour plus d'informations, le lecteur pourra consulter [MST07] ou [MST08].

Depuis, plusieurs versions ont vu le jour, par intégrations successives des approches présentées dans cette thèse : la collision d'onde, la propagation d'ondes coulissante, la modulation de vitesse par programmation par contraintes, et, dernièrement, la propagation d'onde symbolique.

Notons que dans la dernière version d'Airplan, l'étape de modulation de vitesse a disparu, puisque l'impact des obstacles mobiles est directement pris en compte lors de la la propagation d'onde symbolique.

3 Perspectives

La thèse propose deux nouvelles approches de planification de trajectoires de façon séparée. D'une part, la propagation d'onde coulissante, qui permet de gérer les courants forts, et d'autre part, la propagation d'onde symbolique, qui permet de gérer les courants variables dans le temps.

Il serait intéressant de proposer une approche unique, permettant de résoudre ces deux problèmes de façon unifiée. Cette unification peut être réalisée de deux manières différentes :

– Introduire la notion de date de départ variable dans la propagation d'onde coulissante,

– Introduire la notion de curseur dans la propagation d'onde symbolique

Ces deux modifications reviennent à étendre la propagation d'onde coulissante dans un espace-temps 3D $(\vec{x}, \vec{y}, \vec{t})$, illustrée dans la figure 7.23.

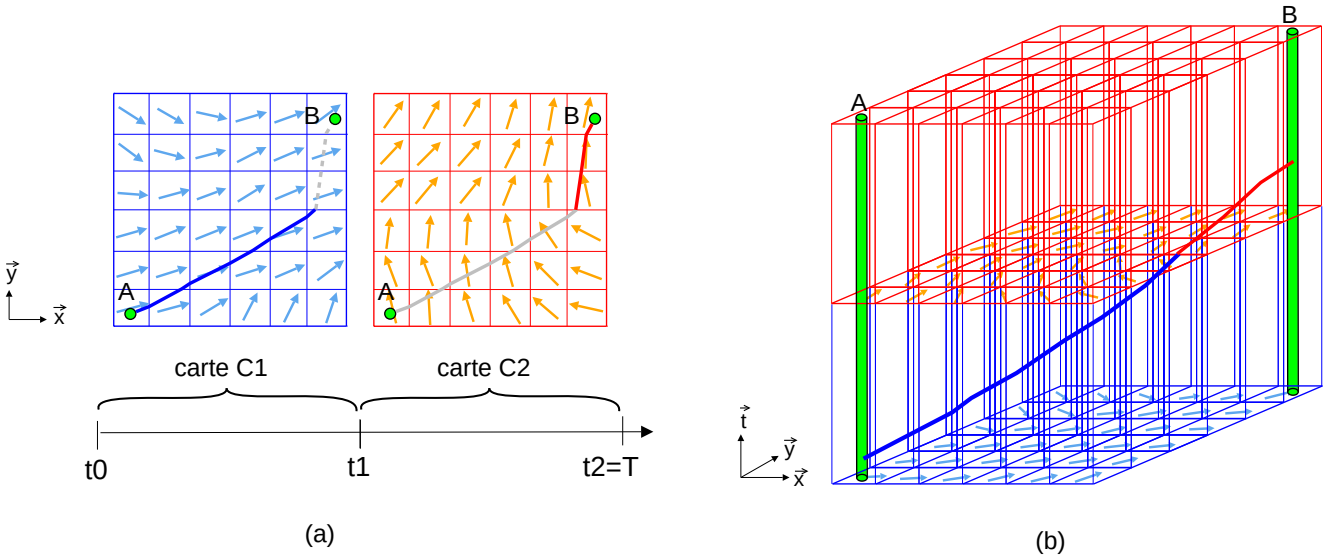

FIG. 7.23 – Extension de la propagation d'ondes coulissante dans un espace-temps 3D
(a) Un problème de planification de trajectoire en présence de courants variables, défini à l'aide de cartes des courants C_1 et C_2 ; (b) L'espace-temps 3D correspondant. La trajectoire que trouverait la propagation d'onde coulissante 3D a été approximée[7] et dessinée en bleu (partie exécutée dans la carte C_1) et rouge (partie exécutée dans la carte C_2).

Dans cet espace-temps, chaque zone de courant est un cylindre parallèle à l'axe $\vec{t}$. Ces cylindres sont découpés par des plans d'équation $t = t_0, t = t_1, \ldots, t = t_k = T$. Chaque date t_i correspond à un changement de carte de courant, c'est à dire au changement simultané de tous les vecteurs-vitesses du courant.

On obtient ainsi des Zones Elémentaires de Courant (ZEC) qui ne sont plus des polygones (comme dans le chapitre 4), mais des polyèdres. Les *curseurs* (points de passage mobiles utilisés pour propager les coûts), peuvent à présent se déplacer sur les faces des ZEC (et non plus sur leurs côtés).

Enfin, le point de départ A peut à présent se déplacer dans l'espace-temps, en "coulissant" selon le vecteur $\vec{t}$, entre 0 et T. L'ordonnée de ce point correspond à la date de départ du robot. Dans la propagation d'onde coulissante présentée dans le chapitre 4, le coût T_i de chaque branche i dépendait uniquement de la position des curseurs sur leur support. Dans l'équation 4.20 page 115, T_i était donné par :

[7] Cette approximation a été calculée de la façon suivante : (1) application de propagation d'onde symbolique pour obtenir la date de départ optimale en A et une estimation "gros grain" (série de cases) du chemin optimal puis (2) application de la propagation d'onde symbolique pour affiner le chemin. Cette décomposition, à vocation illustrative uniquement, est sous-optimale et n'est possible qu'en cas de courants modérés (en cas de vents forts), l'étape 1 échouerait.

$$T_i = \min_{l_j} \tau_3^i = \min_{l_j} \sum_{j=0}^{i-1} \tau_3^{j,j+1} \tag{7.22}$$

Dans l'extension de la propagation d'onde coulissante à un espace-temps 3D, le coût T_i dépend également de la date de départ en A, matérialisée par la variable d_i. L'équation précédente devra être remplacée par :

$$T_i = \min_{d_i, l_j} \tau_3^{i'} = \min_{d_i, l_j} \sum_{j=0}^{i-1} \tau_3^{j,j+1} - d_i \tag{7.23}$$

où d_i est la date de départ associée à la branche i.

Notons que T_i ainsi modifiée ne représente plus une date d'arrivée (comme c'était le cas dans le chapitre 4), mais un temps de parcours (comme dans le chapitre 5).

L'extension de la propagation d'onde coulissante dans un espace-temps 3D serait potentiellement en mesure de permettre l'optimisation de la date de départ en A, tout en se préservant des problèmes d'incorrection ou d'incomplétude dus aux courants forts. Cette extension pourrait faire l'objet de travaux futurs.

Annexes

1 Articles relatifs aux travaux de thèse

[ST06] Michaël Soulignac et Patrick Taillibert,
Fast Trajectory Planning for Multiple Site Surveillance through Moving Obstacles and Wind,
Actes de Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)
pp. 25-33, Nottingham, Royaume-Uni, Décembre 2006.

[MST07] Paul-Edouard Marson, Michaël Soulignac et Patrick Taillibert,
Combining Multi-Agent Systems and Trajectory Planning Techniques for UAV Rendezvous Problems,
Actes de International symposium on COGnitive systems with Interactive Sensors (COGIS)
pp. 26-29, Stanford, Etats-Unis, Juillet 2007.

[Sou07] Michaël Soulignac,
Planification de trajectoires pour engins volants autonomes,
Actes de Rencontre Nationale des Jeunes Chercheurs en Intelligence Artificielle (RJCIA)
pp. 195-212, Grenoble, France, Juillet 2007.

[ST07] Michaël Soulignac et Patrick Taillibert,
Multiple path planning using wavefront collision,
Actes de International Conference on Intelligent Robots and Systems (IROS)
pp. 103-108, San Diego, Etats-Unis, Octobre 2007.

[STR07] Michaël Soulignac, Patrick Taillibert et Michel Rueher,
Velocity Tuning in Currents Using Constraint Logic Programming,
Actes de Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)
pp. 107-113, Prague, République tchèque, Décembre 2007.

[MST08] Paul-Edouard Marson, Michaël Soulignac et Patrick Taillibert,
Aerial : Hypothetical Trajectory Planning for Multi-UAVs Coordination and Control (Demo Paper),
Actes de International Conference on Autonomous Agents and Muliagent Systems (AAMAS)
pp. 1703-1704, Estoril, Portugal, Mai 2008.

[STR08a] Michaël Soulignac, Patrick Taillibert et Michel Rueher,
Adapting the Wavefront Expansion in Presence of Strong Currents,
Actes de International Conference on Robotics and Automation (ICRA),
pp. 1352-1358, Pasadena, Etats-Unis, Mai 2008.

[STR08b] Michaël Soulignac, Patrick Taillibert et Michel Rueher,
Path Planning for UAVs in Time-Varying Winds,
Actes de Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG),
pp. 118-119, Edimbourg, Royaume-Uni, Décembre 2008.

[STR09] Michaël Soulignac, Patrick Taillibert et Michel Rueher,
Time-minimal Path Planning in Dynamic Current Fields,
Actes de International Conference on Robotics and Automation (ICRA),
Kobe, Japon, Mai 2009, A paraître.

2 Détails algorithmiques du chapitre 4

Dans cette partie, nous définissons en détail les fonctions AJOUTER_PREDECESSEUR et MEILLEUR_PREDECESSEUR mentionnés dans les pages et suivantes.

Nous rappelons que dans la propagation d'onde coulissante, chaque point de passage P_i peut coulisser sur son support B_i, et que la variable l_i modélise la position P_i sur B_i.

La valeur de l_i conditionne celle du coût τ_3^i nécessaire pour atteindre P_i à partir du point de départ A. La valeur minimale de τ_3^i est stockée dans l'attribut T_i, et la valeur de l_i correspondante dans l'attribut l_i^*.

Enfin, P_i possède une liste de prédécesseurs potentiels $pred_i$. Chaque prédécesseur est un triplet (P_j, T_i^j, l_i^{j*}). La quantité T_i^j représente le coût minimal pour atteindre P_i à partir de P_j, et l_i^j* la valeur de l_i associée.

AJOUTER_PREDECESSEUR(P_i, P_j)
- ▷ **Entrée :** P_i : point de passage en cours d'évaluation
- ▷ **Entrée :** P_j : nouveau prédécesseur potentiel

1 **Début**
2 $\quad pred_i \leftarrow pred_i \cup \{(P_j, T_j, l_j)\}$
3 **Fin**

MEILLEUR_PREDECESSEUR(P_i)
- ▷ **Entrée :** P_i : point de passage en cours d'évaluation
- ▷ **Sortie :** Prédécesseur optimal de P_i
- ▷ **Locale :** L, E_1, E_2

1 **Début**
2 $\quad L \leftarrow pred_i$
3 $\quad P_2 \leftarrow \emptyset$
4 $\quad$ **faire**
5 $\quad\quad E_1 \leftarrow E_2$
6 $\quad\quad E_2 \leftarrow \arg\min_{l_i^{j*}}(P_j, T_i^j, l_i^{j*}) \in L$
7 $\quad\quad L \leftarrow L \setminus E_2$
8 $\quad$ **tantque** $l_i^{j*} > l_i$ **et** $L \neq \emptyset$
9 $\quad$ **si** $E_1 \neq \emptyset$ **alors**
$\quad\quad$ // $E_1 = (P_1, T_i^1, l_i^{1*})$ et $E_2 = (P_2, T_i^2, l_i^{2*})$
10 $\quad\quad$ **si** $L = \emptyset$ **ou** $T_i^1 < T_i^2$ **alors**
11 $\quad\quad\quad$ **retourner** E_1
12 $\quad$ **retourner** E_2
13 **Fin**

3 Détails algorithmiques du chapitre 5

Dans cette partie, nous définissons en détail les opérateurs d'évaluation et de comparaison mentionnés dans les pages 4.1.2 et suivantes.

Comme nous l'avons expliqué précédemment, le coût c_X associé à une case X est linéaire par morceaux. Le $i$ème segment de droite de c_X est défini par l'équation $d = a_i \cdot d + b_i$. Ce segment est modélisé par un uplet $U_i = (d_i, c_i, a_i, b_i)$, où d_i l'abscisse du début du segment, et c_i le coût associé à d_i (donné par $c_i = a_i \cdot d_i + b_i$).

La métrique $\mathcal{M}'$ est également linéaire par morceaux. Chaque segment de droite de $\mathcal{M}'$ est défini par l'équation $\tau = a'_i \cdot t_i + b'_i$, et stocké sous la forme d'un uplet $U'_i = (t_i, \tau_i, a'_i, b'_i)$.

OPERATEUR_EVALUATION$(c_H, \mathcal{M}'_{H,N})$
- ▷ **Entrée :** $c_H = \{(d_i, c_i, a_i, b_i)\}_{i=1,p}$
- ▷ **Entrée :** $\mathcal{M}'_{H,N} = \{(t_j, \tau_j, a'_j, b'_j)\}_{j=1,q}$
- ▷ **Sortie :** $c_N = c_H + \mathcal{M}'_{H,N} \circ (c_H + d)$
- ▷ **Locale :** $i, j, I, d_{new}, c_{new}, a_{new}, b_{new}$

```
1  Début
2      c_N ← ∅
3      i ← 0
4      pour j ← 1 à q faire
5          tantque i < p faire
6              I ← [t_j, t_{j+1}] ∩ [d_i + c_i, d_{i+1} + c_{i+1}]
7              si I = [t_I^-, t_I^+] ≠ ∅ alors
8                  d_new ← (t_I^- − b_i)/(a_i + 1)
9                  c_new ← a_i · d_new + b_i
10                 a_new ← a'_j(a_i + 1) + a_i
11                 b_new ← b_i(a'_j + 1) + b'_j
12                 c_N ← c_N ∪ {(d_new, c_new, a_new, b_new)}
13                 i ← i + 1
14             sinon rupture
15     retourner c_N
16 Fin
```

```
OPERATEUR_COMPARAISON(c_N^0, c_N^1, c_N)
▷ Entrée : c_N^0 = {(d_i^0, c_i^0, a_i^0, b_i^0)}_{i=1,p}
▷ Entrée : c_N^1 = {(d_j^1, c_j^1, a_j^1, b_j^1)}_{j=1,q}
▷ Sortie : c_N = min(c_N^0, c_N^1)
▷ Locale : i, j, I, d_I^min
1  Début
2    d_- ← 0
3    c_- ← min(c_1^0, c_1^1)
4    c_N ← ∅
5    i ← 0
6    j ← 0
7    tantque i ≤ p and j ≤ q faire
8      si i < q and j < q alors
9        si d_{i+1}^0 < d_{j+1}^1 alors
10         d_+ ← d_{i+1}^0
11         c_+^0 ← c_{i+1}^0
12         c_+^1 ← a_j^1 · d_{i+1}^0 + b_j^1
13       sinon d_+ ← d_{j+1}^1
14         c_+^1 ← c_{j+1}^1
15         c_+^0 ← a_i^0 · d_{j+1}^1 + b_i^0
16     d_I^min ← max(d_i^0, d_j^1)
17     I ← intersection_segments(d_I^min, d_+, a_i^0, b_i^0, a_j^1, b_j^1)
18     si I = [d_I, c_I] ≠ ∅ alors
19       d_- ← d_I
20       c_- ← c_I
21     si c_0^+ < c_1^+ alors
22       c_N ← c_N ∪ {(d_-, c_-, a_i^0, b_i^0)}
23       si d_+ = d_{i+1}^0 alors
24         d_- ← d_i^0
25         c_- ← c_i^0
26         i ← i + 1
27       sinon j ← j + 1
28     sinon c_N ← c_N ∪ {(d_-, c_-, a_j^1, b_j^1)}
29       si d_+ = d_{j+1}^1 alors
30         d_- ← d_j^1
31         c_- ← c_j^1
32         j ← j + 1
33       sinon i ← i + 1
34   retourner c_N
35 Fin
```

INTERSECTION_SEGMENTS($d_{min}, d_{max}, a_0, b_0, a_1, b_1$)

▷ **Entrée :** a_0, b_0 : coeffs of $L_0 : c = a_0 \cdot d + b_0$
▷ **Entrée :** a_1, b_1 : coeffs of $L_1 : c = a_1 \cdot d + b_1$
▷ **Entrée :** d_{min}, d_{max} : bounds for the intersection
▷ **Sortie :** I : intersection between L_0 and L_1
▷ **Locale :** d_I

```
1 Début
2     si a_0 ≠ a_1 alors
3         d_i ← (b_0 − b − 1)/(a_1 − a_0)
4         si d_min ≤ d_i ≤ d_max alors
5             retourner (d_i, a_0 · d_i + b_0)
6     retourner ∅
7 Fin
```

Bibliographie

[ABD+98] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Obprm : an obstacle-based prm for 3d workspaces. In *Proceedings of the workshop on the algorithmic foundations of robotics on Robotics*, pages 155–168, 1998.

[ACO04] A. Alvarez, A. Caiti, and R. Onken. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *Journal of Oceanic Engineering*, 29 :418–429, 2004.

[BBCM00] C. Behring, M. Bracho, M. Castro, and J.A. Moreno. An algorithm for robot path planning with cellular automata. In *International Conference on Cellular Autómata for Research and Industry*, pages 11–18, 2000.

[BL90] J. Barraquand and J-C. Latombe. A monte-carlo algorithm for path planning with many degrees offreedom. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 1712–1717, 1990.

[BL91] J. Barraquand and J-C. Latombe. Robot motion planning : A distributed representation approach. *International Journal of Robotics*, 6 :628–649, 1991.

[BMvH94] F. Benhamou, D. McAllester, and P. van Hentenryck. Clp(intervals) revisited. In *Proceedings of the International Symposium on Logic programming*, pages 124–138, 1994.

[Bor88] J. E. Borrow. Optimal robot path planning using the minimum-time criterion. *Journal of Robotics and Automation*, 4 :443–450, 1988.

[Bor00] S. A. Bortoff. Path planning for uavs. In *Proceedings of the American Control Conference*, 2000.

[BR05] C. Guarino Lo Bianco and M. Romano. Bounded velocity planning for autonomous vehicles. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 685–690, 2005.

[BT93] B. Botella and P. Taillibert. Interlog : constraint logic programming on numeric intervals. In *International Workshop on Software Engineering, Artificial Intelligence and Expert Systems*, 1993.

[BV02] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of the International Conference on Robots and Systems*, volume 3, pages 2383–2388, 2002.

[Can88] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988.

[CBW90] C. I. Connolly, J. B. Burns, and R. Weiss. Path planning using laplace's equation. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 2102–2106, 1990.

[CDM91] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *Proceedings of the European Conference on Artificial Life*, pages 134–142, 1991.

[CDS90] Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. *Advances in neural information processing systems*, 2 :598–605, 1990.

[Cha87] B. Chazelle. Approximation and decomposition of shapes. *Advances in Robotics*, pages 145–185, 1987.

[CM87] P. H. Calamai and J. J More. Projected gradient methods for linearly constrained problems. *Mathematical Programming : Series A and B*, pages 99–116, 1987.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2, 1989.

[Dan60] G. B. Dantzig. On the significance of solving linear programming problems with some integer variables. *Econometrica*, 28 :30–44, 1960.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, pages 269–271, 1959.

[DMC96] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system : optimization by a colony of cooperating agents. *Transactions on Systems, Man, and Cybernetics*, 26 :29–41, 1996.

[DT88] L. Dorst and K. Trovato. Optimal path planning by cost wave propagation in metric configuration space. In *SPIE Proceedings. Advances in Intelligent Robotics Systems : Mobile Robots III*, volume 1007, pages 186–197, 1988.

[DZ94] S.M. Drucker and D. Zeltzer. Intelligent camera control in a virtual environment. In *Proceedings of Graphics Interface*, pages 190–199, 1994.

[ELP86] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. In *Proceedings of the International Conference on Robotics and Automation*, pages 1419–1424, 1986.

[ES98] R. C. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. In *Proceedings of the International Conference on Evolutionary Programming*, pages 611–616, 1998.

[Eve79] S. Even. *Graph algorithms*. Computer Science Press, 1979.

[FF58] L.R. Ford and D.R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6 :419–433, 1958.

[FFP90] D. B. Fogel, L. J. Fogel, and V. W. Porto. Evolving neural networks. *Biological Cybernetics*, 63, 1990.

[FLY+03] X. Fan, X. Luo, S. Yi, S. Yang, and H. Zhang. Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. In *Proceedings of International Conference on Robotics, Intelligent Systems and Signal Processing*, volume 1, pages 131–136, 2003.

[For86] S. Fortune. A sweepline algorithm for voronoi diagrams. In *Proceedings of the second annual symposium on Computational geometry*, pages 313–322, 1986.

[Fra93] T. Fraichard. Dynamic trajectory planning with dynamic constraints : A state-timespace approach. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 2, pages 1393–1400, 1993.

[FS90] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles. In *Transactions on Robotics and Automation*, volume 2, pages 1110–1115, 1990.

[FS93] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. *Proceedings of the International Conference on Robotics and Automation*, 1 :560–565, 1993.

[FS98] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17 :760–772, 1998.

[FS07] D. Ferguson and A. Stentz. Field d* : An interpolation-based path planner and replanner. *Springer Tracts in Advanced Robotics Series*, 28 :239–253, 2007.

[FXGC05] X-W. Fu, X-G.Gao, and D-Q. Chen. Uav path planning in dynamic and uncertain environments. *Transactions on Information Science and Applications*, 2 :1978–1985, 2005.

[GAO05] B. Garau, A. Alvarez, and G. Oliver. Path planning of autonomous underwater vehicles in current fields with complex spatial variability : an a* approach. In *Proceedings of the International Conference on Robotics and Automation*, pages 194–198, 2005.

[GB06] L. Granvilliers and F. Benhamou. An interval solver using constraint satisfaction techniques. In *ACM Trans. on Mathematical Software*, volume 32, pages 138–156, 2006.

[GC00] S.S Ge and Y.J. Cui. New potential functions for mobile robot path planning. *Transactions on Robotics and Automation*, 2000.

[GC02] S.S Ge and Y.J. Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robot*, 13 :207–222, 2002.

[GCFR98] D. Gallardo, O. Colomina, F. Flórez, and R. Rizo. A genetic algorithm for robust motion planning. In *Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 115–121, 1998.

[GFC05] X-G. Gao, X-W. Fu, and D-Q. Chen. A genetic-algorithm-based approach to uav path planning problem. In *Proceedings of the International Conference on Simulation, Modeling and Optimization*, pages 503–507, 2005.

[GKG95] R. Glasius, A. Komoda, and S. Gielen. Neural network dynamics for path planning and obstacle avoidance. *Neural Networks*, 8 :125–133, 1995.

[HCL99] K-S Hwang, H-J Chao, and J-H Lin. Collision-avoidance motion planning amidst multiple moving objects. *Journal of Intelligent and Robotic Systems*, 15 :715–736, 1999.

[HKLR02] D. Hsu, R. Kindel, J-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21 :233–255, 2002.

[HNG94] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. *Evolutionary Computation*, 1 :82–87, 1994.

[HNR68] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science and Cybernetics*, pages 100–107, 1968.

[Hol75] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan press, 1975.

[HSW93] B. Hassibi, D. G. Stork, and G. J. Wolff. Efficient evolution of neural network topologies. In *Proceedings of Conference on Neural Networks*, volume 1, pages 293–299, 1993.

[Hus89] B. Hussien. Robot path planning and obstacle avoidance by means of potential function method. *Thèse de doctorat - Université du Missouri-Columbia, Etats-Unis*, 1989.

[ISM05] T. Inanc, S. C. Shadden, and J. E. Marsden. Optimal trajectory generation in ocean flows. In *Proceedings of the American Control Conference*, pages 674–679, 2005.

[Jan04] D. Janglova. Neural networks in mobile robot motion. *International Journal of Advanced Robotic Systems*, 1, 2004.

[Jar85] R. A. Jarvis. Collision-free trajectory planning using the distance transforms. *Mechanical Engineering Transactions of the Institution of Engineers*, 3 :187–191, 1985.

[JH00] H. Juidette and H.Youlal. Fuzzy dynamic path planning using genetic algorithms. *Electronics Letters*, 36 :374–376, 2000.

[JLH02] M-Y. Ju, J-H. Liu, and K-S. Hwang. Real-time velocity alteration strategy for collision-free trajectory planning of two articulated robots. *Journal of Intelligent and Robotic Systems*, 33 :167–186, 2002.

[JT80] C. L. Jackins and S. L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 1980.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[KF07] H. Kurniawati and T. Fraichard. From path to trajectory deformation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 159–164, 2007.

[Kha80] O. Khatib. Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles. *Thèse de doctorat - Ecole Nationale de l'Aeronautique et de l'Espace, France*, 1980.

[KJ04] M. Karimifar and C. Jeffrey. Ant conversation - an enhancement of aco. In *Proceedings of Multidisciplinary Analysis and Optimization Conference*, 2004.

[KKB98] R. Kimmel, N. Kiryati, and A.M. Bruckstein. Multivalued distance maps for motion planning on surfaces with moving obstacles. *Transactions on Robotics and Automation*, 14 :427–436, 1998.

[KL94] L. Kavraki and J-C. Latombe. Randomized preprocessing of configuration for fast path planning. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 2138–2145, 1994.

[KL00] J. J. Kuffner and S. M. LaValle. Rrt-connect : An efficient approach to single-query path planning. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 995–1001, 2000.

[Kod87] D. Koditschek. Exact robot navigation by means of potential functions : Some topological considerations. In *Proceedings of the International Conference on Robotics and Automation*, volume 4, pages 1–6, 1987.

[KZ86] K. Kant and S. W. Zucker. Toward efficient trajectory planning : the path-velocity decomposition. *The International Journal of Robotics Research*, 5 :72–89, 1986.

[Lat91] J-C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1991.

[LaV98] S. M. LaValle. Rapidly-exploring random trees : A new tool for path planning. *TR 98-11, Computer Science Dept., Iowa State Univ*, 1998.

[Lav06] S. M. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

[LBL04] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *Transactions on Robotics and Automation*, 2004.

[LD60] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28 :497–520, 1960.

[Lho93] O. Lhomme. Consistency techniques for numeric csps. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 232–238, 1993.

[Lin04] F. Lingelbach. Path planning using probabilistic cell decomposition. In *Proceedings of the International Conference on Robotics and Automation*, pages 467–472, 2004.

[LP83] T. Lozona-Perez. Spatial planning : A configuration space approach. *Transactions on Computers*, 32 :108–120, 1983.

[LSR05] D.V. Lebedev, J. J. Steil, and H. J. Ritter. The dynamic wave expansion neural network model for robot motion planning in time-varying environments. *Neural Networks*, 18 :267–285, 2005.

[Mac77] A. Mackworth. Consistency in networks of relations. *Journal of Artificial Intelligence*, 1 :99–118, 1977.

[MF] Météo france - cartes des vents. `http://www.meteofrance.com/FR/mer/carteVents.jsp`.

[Mic98] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1998.

[Mie99] K. M. Miettinen. *Nonlinear Multiobjective Optimization, International Series in Operations Research and Management Science*. Kluwer Academic Publishers, 1999.

[Min83] M. Minoux. *Programmation mathématique. Théorie et Algorithmes*. Dunod Paris, 1983.

[MP69] M. L. Minsky and S. Papert. *Perceptron*. MIT Press, 1969.

[MS98] K. Marriot and P. Stuckey. *Programming with constraints - An introduction*. The MIT Press, 1998.

[MST07] P. E. Marson, M. Soulignac, and P. Taillibert. Combining multi-agent systems and trajectory planning techniques for uav rendezvous problems. In *International symposium on COGnitive systems with Interactive Sensors*, pages 26–29, 2007.

[MST08] P. E. Marson, M. Soulignac, and P. Taillibert. Aerial : Hypothetical trajectory planning for multi-uavs coordination and control (demo paper). In *International Conference on Autonomous Agents and Muliagent Systems*, pages 1703–1704, 2008.

[MTZ05] H. Mei, Y. Tian, and L. Zu. A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. In *Proceedings of International Conference on Intelligent Computing*, pages 78–88, 2005.

[Nag87] J. Nagle. On packet switches with infinite storage. In *IEEE Transactions on Communications*, volume 35, pages 435–438, 1987.

[NBMB06] D. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard. Vector field path following for small unmanned air vehicles. *Proceedings of the American Control Conference*, 2006.

[Nil69] N. J. Nilsson. A mobile automation : An application of artificial intelligence techniques. *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 509–520, 1969.

[OR90] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37 :607–625, 1990.

[PPPL05] C. Pêtrès, Y. Pailhas, Y. Petillot, and D. M. Lane. Underwater path planning using fast marching algorithms. In *Oceans Europe*, volume 2, pages 814–819, 2005.

[Pêt07] C. Pêtrès. Trajectory planning for autonomous underwater vehicles. *Thèse de doctorat - Université de Heriot-Watt, Royaume-Uni*, 2007.

[QK93] S. Quinlan and O. Khatib. Elastic bands : connecting path planning and control. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 802–807, 1993.

[QSLC04] Y. Qin, D-B. Sun, N. Li, and Y-G. Cen. Path planning for mobile robot using the particle swarm optimization with mutation operator. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 4, pages 2473–2478, 2004.

[Rec73] I. Rechenberg. *Evolutionsstrategie : Optimierung technischer System nach Prinzipien der biologischen Evolution.* Fromman-Holzboog, 1973.

[RH02] A. Richards and J.P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the American Control Conference*, 2002.

[RK03] J.C. Rubio and S. Kragelund. The trans-pacific crossing : long range adaptive path planning for uavs through variable wind fields. In *Proceedings of the Digital Avionics Systems Conference*, volume 2, pages 8.B.4.1–12, 2003.

[RKPC02] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi. An evolution based path planning algorithm for autonomous motion of a uav through uncertain environment. In *Proceedings of the Digital Avionics Systems Conference*, volume 2, pages 1–12, 2002.

[Roh86] H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, 3 :71–76, 1986.

[Rum86] D. E. Rumelhart. *Learning internal representations by error propagation.* MIT Press Cambridge, 1986.

[RVR04] J. C. Rubio, J. Vagners, and R. Rysdyk. Adaptive path planning for autonomous uav oceanic search missions. In *Proceedings of the Intelligent Systems Technical Conference*, pages 1–10, 2004.

[RWHK97] U. Roth, M. Walker, A. Hilmann, and H. Klar. Dynamic path planning with spiking neural networks. In *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks*, pages 1355–1363, 1997.

[Sch85] D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms and their Applications*, pages 93–100, 1985.

[Sch08] S. Schoenung. Western states fire mission - advanced uav technology. In *Proceedings International Conference on Robotics and Automation Workshop : 'UAVs : Civilian and Commercial Opportunities'*, 2008.

[SD94] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3) :221–248, 1994.

[Set96] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 93, pages 1591–1595, 1996.

[SM02] K. O. Stanley and R. Miikkulainen. Efficient evolution of neural network topologies. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1757–1762, 2002.

[SN85] M. Sugeno and M. Nishida. Fuzzy control of model car. *Fuzzy Sets and Systems*, 16 :103–113, 1985.

[Sou07] M. Soulignac. Planification de trajectoires pour engins volants autonomes. In *Rencontre Nationale des Jeunes Chercheurs en Intelligence Artificielle*, pages 195–212, 2007.

[SS97] K. Sugihara and J. Smith. Genetic algorithms for adaptive motion planning of an autonomous mobile robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 138–143, 1997.

[ST06] M. Soulignac and P. Taillibert. Fast trajectory planning for multiple site surveillance through moving obstacles and wind. In *Proceedings of the Workshop of the UK Planning and Scheduling Special Interest Group*, pages 25–33, 2006.

[ST07] M. Soulignac and P. Taillibert. Multiple path planning using wavefront collision. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 103–108, 2007.

[Ste94] A. Stentz. Optimal and efficient path planning for partially-knownenvironments. In *Proceedings of the International Conference on Robotics and Automation*, volume 4, pages 3310–3317, 1994.

[STR07] M. Soulignac, P. Taillibert, and M. Rueher. Velocity tuning in currents using constraint logic programming. In *Workshop of the UK Planning and Scheduling Special Interest Group*, pages 107–113, 2007.

[STR08a] M. Soulignac, P. Taillibert, and M. Rueher. Adapting the wavefront expansion in presence of strong currents. In *Proceedings of the International Conference on Robotics and Automation*, pages 1352–1358, 2008.

[STR08b] M. Soulignac, P. Taillibert, and M. Rueher. Path planning for uavs in time-varying winds. In *Workshop of the UK Planning and Scheduling Special Interest Group*, pages 118–119, 2008.

[STR09] M. Soulignac, P. Taillibert, and M. Rueher. Time-minimal path planning in dynamic current fields. In *Proceedings of the International Conference on Robotics and Automation*, 2009. A paraître.

[TCJ03] S. Twigg, A. Calise, and E. Johnson. On-line trajectory optimization for autonomous air vehicles. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 11–14, 2003.

[Tor04] J. C. R. Torroella. Long range evolution-based path planning for uavs through realistic weather environments. *Thèse de doctorat - Université de Washington, Etats-Unis*, 2004.

[Tro99] M. Trott. The area of a random triangle. *The Mathematica Journal*, 7 :89–198, 1999.

[Vaz01] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, 2001.

[vdBO05] J.P. van den Berg and M. H. Overmars. Roadmap-based motion planning in dynamic environments. *Transactions on Robotics and Automation*, 21 :885–897, 2005.

[Vla03] A. Vladimirsky. Ordered upwind methods for static hamilton-jacobi equation : Theory and algorithms. *SIAM Journal of Numerical Analysis*, 41 :325–363, 2003.

[Wel85] E. Welzl. Constructing the visibility graph for n line segments in $o(n^2)$ time. *Information Processing Letters*, 20 :167–171, 1985.

[Wer74] P. Werbos. Beyond regression : new tools for prediction and analysis in the behavioral sciences. *Thèse de doctorat - Université de Harvard, Etats-Unis*, 1974.

[WZD08] S. Wei, M. Zefran, and R.A. DeCarlo. Optimal control of robotic systems with logical constraints : Application to uav path planning. In *From Path to Trajectory Deformation*, pages 176–181, 2008.

[YELP08] N.K. Yilmaz, C. Evangelinos, P.F.J. Lermusiaux, and N. Patrikalakis. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *Ocean Engineering*, page A paraître, 2008.

[YM01] S.X. Yang and M. Meng. Neural network approaches to dynamic collision-free trajectory generation. *Transactions on Systems, Man and Cybernetics*, 31 :302–318, 2001.

[YSSB98] A. Yahja, A. Stentz, S. Singh, and B.L. Brumitt. Framed-quadtree path planning for mobile robots operating in sparse environments. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 650–655, 1998.

[YY99] N. H. C. Yung and C. Ye. An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning. *Transactions on Systems, Man, and Cybernetics*, 29 :314–321, 1999.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8 :338–253, 1965.

[Zal06] S. Zaloga. Trends in the uav market 2006. In *AUVSI Unmanned Systems*, pages 26–29, 2006.

[ZBMD04] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization : A critical survey. *Annals of Operations Research*, 131 :373–395, 2004.

[ZIOM08] W. Zhang, T. Inanc, S. Ober-Bl Obaum, and J. E. Marsden. Optimal trajectory generation for a glider in time-varying 2d ocean flows b-spline model. In *Proceedings of the International Conference on Robotics and Automation*, pages 1083–1088, 2008.

[ZY05] Q. Zhao and S. Yan. Collision-free path planning for mobile robots using chaotic particle swarm optimization. In *Proceedings of the International Conference on Advances in Natural Computation*, pages 632–635, 2005.