MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR

ECOLE NATIONALE POLYTECHNIQUE

DER GÉNIE ÉLECTRIQUE ET INFORMATIQUE

SPÉCIALITÉ: AUTOMATIQUE

Mémoire de fin d'études en vue de l'obtention:
du diplôme d'ingénieur d'état
en automatique.

ETUDE ET SYNTHÈSE D'UN NAVIGATEUR ET PLANIFICATEUR ÉVOLUTIF, A BASE D'ALGORITHMES GÉNÉTIQUES

Proposé et dirigé par:
D^r. M C SOUAMI

Présenté par:
M^lle YASMINA BECIS

PROMOTION 1998

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR

ECOLE NATIONALE POLYTECHNIQUE

DER GÉNIE ÉLECTRIQUE ET INFORMATIQUE

SPÉCIALITÉ: AUTOMATIQUE

Mémoire de fin d'études en vue de l'obtention:
du diplôme d'ingénieur d'état
en automatique.

ETUDE ET SYNTHÈSE D'UN NAVIGATEUR ET PLANIFICATEUR ÉVOLUTIF, A BASE D'ALGORITHMES GÉNÉTIQUES

Proposé et dirigé par:
D^r. M C SOUAMI

Présenté par:
M^lle YASMINA BECIS

PROMOTION 1998

A la mémoire de ma grand mère

Quand deux sages confrontent leurs idées,
ils en produisent de meilleures; le jaune et le
rouge mélangés produisent une autre couleur.
(Proverbe Tibétain)

Mes plus sincères remerciements s'adressent à:

Mes parents qui m'ont offert calme et sérénité.
Mon petit frère adoré qui a fait des efforts pour cesser de m'embêter.
Mon promoteur pour sa présence permanente, sa disponibilité, ses conseils sûrs et son aide précieuse.
Tous mes professeurs (L'INGM et l'école polytechnique) qui durant ces cinq années, ont guidés mes pas.
Mes amis pour leur soutien, leur appui et leurs encouragements si chaleureux.

A tous ceux dont j'ai oublié de citer le nom. Qu'ils trouvent ici l'expression de ma plus profonde gratitude.

Je n'aime pas le travail nulle ne l'aime;
mais j'aime ce qui est dans le travail
l'occasion de se découvrir soi même,
j'entends notre propre réalité, ce que nous
sommes à nos yeux, et non pas en façade.

J.CONRAD, Le coeur des ténèbres

Avant-propos :

« Percevoir le monde environnant et trouver son chemin dans celui ci, sont à la base de la conduite de toute créature mobile intelligente, qu'elle soit artificielle ou biologique [1] ».

Avant d'entreprendre ce travail, (je parle ici à mon nom personnel) je trouvais étonnant et même stupéfiant comment une machine inanimée, pouvait se localiser, prendre connaissance des objets environnants, se retrouver dans un environnement inconnu et par dessus tout se déplacer vers un objectif bien précis sans jamais le perdre de vue, sans heurter aucun objet se présentant sur son chemin et mieux encore, faire tout cela pas n'importe comment mais de la meilleure manière qui soit : oui, étonnant car étant moi-même quasiment dépourvue de sens d'orientation, j'ai beaucoup de difficultés à trouver mon chemin dans un lieu inconnu.

La première idée que j'ai reçu sur la navigation était une vague notion sur la méthode des champs de potentiel artificiels, j'en ai été sous le charme ! n'est il pas ingénieux de penser à plonger l'espace dans lequel évolue le mobile dans un champ attractif généré par le point de destination, aspirant le mobile vers lui, et faire émaner un champ répulsif des obstacles de façon à l'en tenir à distance. Le mobile n'a qu'à se laisser entraîner par les lignes de forces, lui procurant un chemin court et sûr pour atteindre l'objectif, fallait-il y penser.

J'ai été littéralement prise de fascination lorsque j'ai pris connaissance de la méthode de navigation par l'algorithme évolutif présenté par [29]: comment est ce que les gens sont arrivé à faire un rapprochement entre des problèmes tels que la navigation et le phénomène d'évolution des créatures vivantes vers des formes supposées meilleures ? La façon judicieuse et pertinente d'imiter la nature, d'exploiter son savoir faire et de l'adapter au problème à traiter, m'ont poussé à travailler dessus. Il faut avouer aussi que mon enthousiasme n'a pas duré longtemps, j'ai rencontré des déceptions durant l'implémentation, n'ayant pu atteindre que très difficilement un résultat satisfaisant, après maintes modifications et améliorations, résultat qui au bout du compte ne comblait pas toutes les espérances, et ceci pour deux raisons possibles :

- se guidant par le support bibliographique et suivant ses grandes lignes, notre algorithme n'arrivait pas au début à atteindre l'efficacité désirée, les choses n'étaient pas aussi évidentes qu'on l'aurait imaginé.

[1] Vision and navigation ,the Carnegie Mellon Navlab, Charles E. Thope, Carnegie Mellon University

- Il n'est nullement garanti que la technique en elle même présente des performance sûres à toute épreuve.

J'ignore si ce travail aurait servi à quelque chose, néanmoins, le fait de faire trouver son chemin à un véhicule ne me paraît plus aussi sorcier qu'auparavant, du moins par les algorithmes génétiques.

Résumé :

Dans ce mémoire, nous présentons un algorithme de planification et navigation pour un robot mobile autonome, dans un environnement statique structuré, parsemé d'obstacles polygonaux ou assimilés comme tels. L'algorithme de navigation utilise une technique récente : les algorithmes évolutifs.

L'algorithme est en premier lieu mis en œuvre pour un environnement parfaitement connu, puis pour un environnement totalement inconnu a priori, et qui se fait découvrir au fur et à mesure du déplacement du robot sur son chemin le menant de la source au but.

Pour finir, une extension de cet algorithme à un environnement inconnu à obstacles mobiles a été proposée.

Mots clés :
Algorithmes génétiques, navigation, planification, robot mobile.

Sommaire:

CHAPITRE I

INTRODUCTION

La robotique mobile par le développement des véhicules autonomes entre autres pour l'exploration, l'agriculture, les milieux hostiles et autres, a vu une évolution accrue ces dernières années.

L'autonomie de plus en plus souhaitée de ces systèmes a conduit à l'étude et la mise en place d'architectures de commande. Celles-ci font ressortir la nécessité de planifier avant des actions avant de pouvoir les exécuter.

Une action importante est le mouvement. En effet, le robot doit pouvoir se déplacer dans un milieu parsemé obstacles d'où la nécessité de définir une science qui traite ce problème, la navigation.

La navigation, d'après P.J. McKerrow, est l'art de diriger un robot mobile à travers un environnement donné. Conduire ce robot à destination sans qu'il ne se perde ou qu'il ne heurte un obstacle au passage , constitue, à son sens, un désir inhérent à cette tâche[1].

Ceci fait l'objet des algorithmes de planification/navigation. Ces deux derniers diffèrent par leurs modes opératoires : le premier est exécuté en ligne tandis que la seconde fonction hors ligne.

L'apparition de nouvelles techniques de commande et de prise de décision autres que les techniques numériques, et la complexité des problèmes, ont conduit au développement de nouveaux algorithmes, parmi ces algorithmes la navigation évolutive basée sur les algorithmes génétiques.

Dans ce qui suit nous présentons l'architecture générale de commande d'un robot mobile définissant la planification/navigation, les algorithmes associés et les modèles de représentation de l'environnement.

En chapitre III, une brève description du fondement des algorithmes génétiques ainsi que leur structure est présentée.

Dans le chapitre IV, nous présentons un algorithme de base auquel nous apportons des modifications et des améliorations pour finir par généraliser au cas réel de fonctionnement.

Nous concluons notre travail dans le chapitre V, en rappelant les principales difficultés, solutions et recommandations apportées.

[1] McKerrow, P.J., Introduction to robotics, Addison-Wesley, 1991.

CHAPITRE II

TECHNIQUE DE PLANIFICATION ET DE NAVIGATION

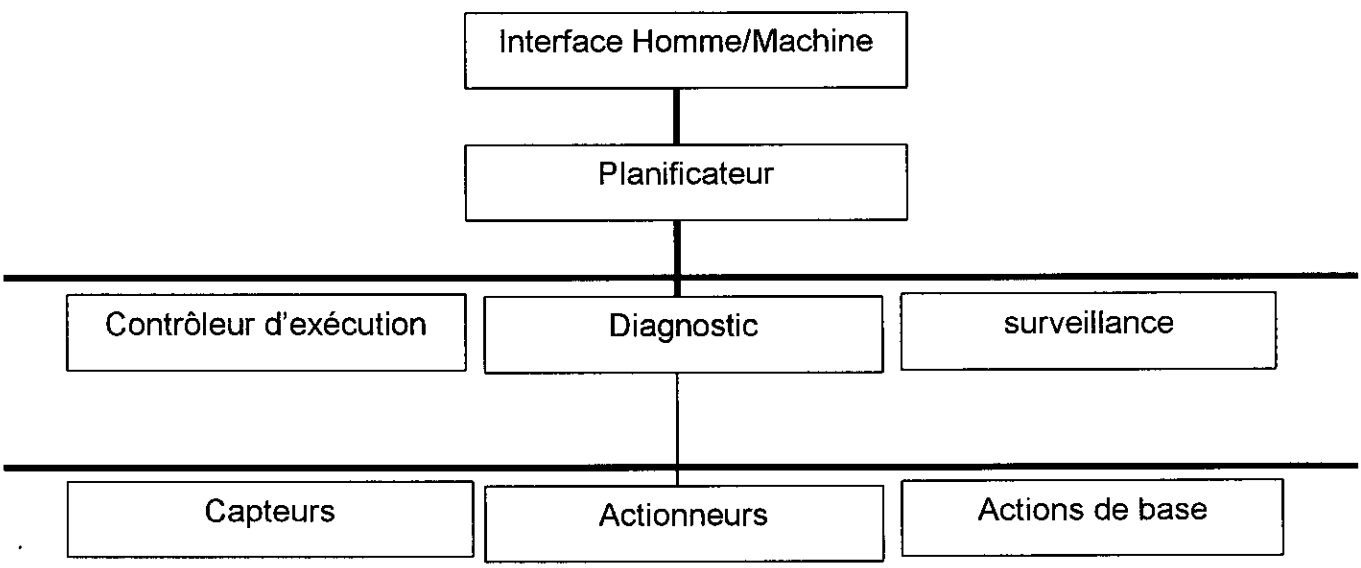
La robotique moderne, une branche de l'automatique a vu le jour entre les années 70 et 80. Les premiers robots de cette génération étaient dotés de la faculté de manipuler des objets, et ont vite trouvé application dans l'industrie. Ces robots avaient une caractéristique commune : celle d'être reliés à un référentiel fixe. En effet, ils ne pouvaient atteindre que des points situés dans des zones bornées de l'espace, et d'éloignement très limité. Ce n'est que plus tard que les progrès technologiques et les besoins de mobilité ont conduit à la réalisation d'un nouveau type de robot : le robot mobile. Ce robot se distingue des premiers par l'absence de toute attache mécanique avec le milieu environnant. Ceci lui procure une liberté de mouvement, qui lui permet d'envisager un nombre illimité d'applications nouvelles, dans divers domaines tel que le transport, l'exploration en environnement hostile, l'agriculture, le génie civile, la lutte contrer les incendies et le terrorisme, l'inspection, le domaine militaire, l'aide aux handicapés. etc... Dans certaines applications, on associe un manipulateur à une base mobile pour obtenir un robot plus opérationnel et de plus grande universalité de point de vue application.

II.1. L'architecture du système de commande :

II.1.1. L'architecture générale [1]:

L'architecture du robot décrit l'organisation des divers modules qui le composent ainsi que leurs interactions. Celle-ci se décompose, en général, en quatre niveaux :

- Niveau interface homme-machine :description de la tâche .
- Niveau planification de tâche : planification des actions.
- Niveau contrôle : exécution.
- Niveau fonctionnel : compte rendu de l'exécution.



Architecture d'un robot mobile

1. Interface homme-machine :

Ce niveau d'architecture gère les communications avec le programmeur ou l'ordonnateur de tâches. La tâche que doit exécuter le robot est décrite selon le mode de communication (écran/ clavier, par exemple) dans un langage plus ou moins évolué. La tâche sera ensuite transmise dans un code adéquat au générateur de plan.

2. Générateur de plan :

A ce niveau se construit le plan de la tâche défini au niveau supérieur. Le planificateur de tâche va déterminer une suite d'actions en utilisant les ressources disponibles et en définissant les modes d'activation.

3. Le contrôleur d'exécution :

Le contrôleur d'exécution organise la mise en œuvre du plan établi au niveau du planificateur. Lors du déplacement, il adapte le comportement du robot aux situations instantanées. Les informations nécessaires à la réactivité, i.e. l'application d'un comportement face à une situation, sont issues d'un module de surveillance de l'environnement, dont le rôle est de fournir des informations sur l'état de l'environnement. Afin de garantir l'intégrité et la fiabilité des informations et l'exécution du comportement qui en est la conséquence, le contrôleur possède un module effectuant le diagnostique du fonctionnement global. Ses informations permettent au robot de réagir aux diverses situations grâce aux potentiels matériels et fonctions disponibles à l'instant considéré.

4. Le niveau fonctionnel :

Ce niveau est le plus bas dans la présente hiérarchie. Il se compose de modules capteurs, effecteurs et actions de base. Les capteurs sont organisés en plusieurs niveaux : l'élément de détection (le capteur physique), l'élément de traitement du signal, et l'élément de traitement évolué (ex : extraction de primitives de type droite). Un module effecteur (actionneur) assure la réalisation du mouvement. Le module des actions de base se compose d'un ensemble de primitives dont l'usage est fréquent (ex : évitement d'obstacle, suivi de mur, mouvement suivant une trajectoire pré-définie telle que cercle, ellipse etc..).

II.1.2. Cas pratique d'architecture :

Une structure hiérarchique à plusieurs niveaux pour un véhicule autonome a été proposée par R. Chavez et A. Meystel dans [2].

S'inspirant de la gestion et distribution des fonctions, et de l'acheminement des tâches dans les couches hiérarchiques d'un "équipage humain", les auteurs ont établi la structure d'un véhicule autonome en trois niveaux :

- Le planificateur
- Le navigateur
- Le pilote

Les différentes étapes sont :

a. Un traitement des informations issues de capteurs ou de caméras de vision puis une reconnaissance des éléments de l'environnement se font à partir des données traitées.

b. les résultat de la reconnaissance sont utilisés pour tracer ou compléter la carte de l'environnement.

c. la carte est affinée (codage de certains objets, simplification d'images, suppression de quelques détails...), mise sous une forme adéquate pour être exploitée par le planificateur, le navigateur et le pilote et filtrée de façon à ne présenter que les informations utiles pour et selon son destinataire.

d. La tâche du planificateur se décompose en deux parties :

   - Conversion de la carte en *Graphes* (voir planification de trajectoire) ou en *espace libre de configurations* (voir modélisation de l'environnement).
   - Recherche, dans ce graphe, d'un meilleur chemin selon des critères établis.

e. Le meilleur chemin est soumis au navigateur sous forme d'une séquence de nœuds intermédiaires (sous-buts).

f. Le mouvement commence à ce niveau. La tâche du navigateur consiste à :

   - tracer la trajectoire, en ligne, reliant les points délivrés par le planificateur.
   - Corriger la trajectoire en cas d'événements imprévus i.e. lorsque la situation le long de la trajectoire originale ne concorderait plus avec ce qui est décrit par la carte.
   - Demander des informations supplémentaires, en cas de besoin, au système de perception.
   - Demander au planificateur de regénérer les nœuds.

g. Des données fournies par le navigateur (telles que la position, l'orientation et la vitesse désirées) sont converties en commandes pour l'actionneur, puis transmises au pilote.

Dans des systèmes plus complexes, comportant plusieurs niveaux de cette structure, le passage d'un niveau à l'autre se fait par un coordinateur figurant à la tête des modules (des planificateurs de ces modules, plus précisément) du niveau inférieur (Il est clair que le pilote ne paraît qu'au dernier niveau).

II.2. La planification de trajectoire :

Pour qu'un robot puisse construire un chemin sûr et en un temps fini, le menant du départ à l'arrivée, il lui est nécessaire de disposer

- dans le cas où il se meut dans un environnement préalablement connu, de la topologie de l'environnement.
- dans le cas contraire, il se sert des informations issues des capteurs lui délivrant des caractéristiques locales de l'environnement.

A partir de ces connaissances sur l'environnement, a priori ou non, il lui est possible de planifier sa trajectoire, ainsi que de maintenir le cap lors de la navigation, en vérifiant continuellement la concordance de sa position avec le chemin planifié.

Planifier, dans la robotique mobile, équivaut à générer une séquence de positions que doit prendre le robot dans son mouvement de la source au but.

II.2.1.Planification globale :

La planification globale consiste à générer une séquence complète de position de la source au but. Elle se décompose en deux types :

- planification dans un environnement connu ;
- planification dans un environnement inconnu ;

II.2.1.1. La planification dans un environnement connu :

Définition [1]:

Soit $A=\{a_0, a_1, a_2, \ldots, a_n\}$, une liste d'actions indépendantes que peut exécuter le robot ;

et $B=\{b_0, b_1, b_2, \ldots b_m\}$, la base de connaissance de l'univers sur laquelle s'appuient les actions de A, les éléments b_i peuvent par exemple, représenter un espace sur lequel les a_i ne peuvent pas s'appliquer.

On définie $Q=T(B)$, un espace sur lequel peut s'appliquer A, et dont les éléments définissent les conditions d'application des éléments de A, T étant une fonction de transformation de B vers Q. Certains éléments de Q sont regroupés en sous-espaces $V=\{v_0, v_1,\ldots,v_l\}$

E est défini comme étant l'ensemble des relations existant entre les éléments v_i de V.

L'entité G=(V,E) est appelée *Graphe* dont V est l'ensemble des *nœuds* et E est l'ensemble des *arcs* reliant ces nœuds.

La planification s'applique sur le graphe G, en faisant intervenir les relations (de l'ensemble E) entre les v_i de V. En fait, elle consiste à rechercher dans V la séquence $\prod=\{v_s, \ldots,v_b\}$, où v_s est le nœud source, et v_b est le nœud but.

Il existe plusieurs méthodes qui fournissent la séquence $\prod$, leur recherche est orientée par un critère précisant l'objectif visé par la recherche qui dépend des exigences de l'application : le chemin optimal peut être le chemin le plus court (de longueur minimale) ou le plus sûr (le plus éloigné des obstacles), ou encore le plus rapide (celui qui est parcouru en un minimum de temps).

Les principales classes d'algorithmes de planification avec des connaissances à priori sont :

– les algorithmes exhaustifs,
– les algorithmes heuristiques
– les algorithmes aléatoires.

Tous ces algorithmes procèdent de la façon suivante : ils créent, à partir du graphe G, une structure de graphe orienté appelée, *arbre de recherche,* qui comporte à l'origine le nœud source. Puis, au fur et à mesure de leur évolution, ils développent les branches de l'arbre en associant à chaque nœud *père* v_i un nœud *fils* v_j.

On appelle nœuds fils, tous les nœuds v_j, qui ont un lien e_k de E (sont reliés par un arc) avec un nœud père v_i,

Remarque :Ce lien existe s'il n y a pas intersection avec un objet de l'environnement. Un nœud est dit *ouvert* si les nœud fils ne lui ont pas tous été associés.

Les algorithmes cités précédemment opèrent de la même façon en différant seulement dans la manière de développer les nœuds.

❖ *Les algorithmes exhaustifs :*

Ces algorithmes sont dits exhaustifs parce qu'au cours de la recherche du chemin, à tous les nœuds ouverts de l'arbre sont reliés des nœuds fils. Le développement de ces nœuds s'effectue *en largeur,* ceci veut dire que le développement se fait sur les nœuds *d'une même profondeur* .

Un exemple de ce type d'algorithmes est celui proposé par Dijkstra [3]. Celui ci associe à chaque arc du graphe un coût positif, tout en limitant le développement des nœuds aux branches de l'arbre représentant les portions de chemins (séquences de nœuds) de coût minimal.

❖ *Les algorithmes heuristiques :*

Cette classe se caractérise par l'utilisation d'heuristique dans le but de limiter le nombre de nœuds. L'heuristique introduite attribue à chaque nœud v une valeur de la fonction d'évaluation f(v) représentant le coût du chemin allant du nœud source au nœud père et passant par le nœud v. Ceci permet de choisir les premiers les nœuds prometteurs. Il est claire que la recherche dans cet algorithme est plus ordonnée et ne se fait pas d'une manière aussi aveugle que dans les algorithmes précédents. (l'algorithme A* est un algorithme heuristique très utilisé [1],[23]).

❖ *Les algorithmes aléatoires :*

Ces algorithmes agissent généralement en combinaison avec des méthodes de recherches locales tels que le gradient de potentiel. Les nœuds sont des sous-buts, et le phénomène aléatoire intervient dans le choix des prochains sous-buts à développer. [4] propose une distribution équilibrée de sous-buts sur l'étendue de l 'espace de travail, en partageant cet espace en cellules uniformes, puis en disposant dans chaque cellule, un nombre de nœuds proportionnel aux dimensions de cette cellule.

[5] proposent une combinaison entre la méthode de gradient de potentiel et une recherche aléatoire pour éviter le blocage du processus de recherche par les minimums locaux propres à la méthode du gradient. Le déplacement aléatoire se fait selon un mouvement Brownien centré au niveau des minimums locaux.

L'application aux problèmes complexes des méthodes aléatoires, est plus intéressant de point de vue rapidité que les méthodes classiques. Par contre, il se peut qu'elles n'aboutissent pas à une solution à moins que leur nombre d'essai soit illimité. Il n'est pas garanti non plus que les solutions qu'elles délivrent soient optimales.

II.2.1.2. Planification dans un environnement inconnu :

La planification sans connaissances a priori s'effectue dynamiquement, durant l'évolution du robot dans la recherche de son chemin, en ne découvrant l'obstacle qu'une fois l'ayant abordé.

Les premières méthodes présentées dans ce sens sont les algorithmes BUG1 et BUG2 dans [1].

Les auteurs formulent les hypothèses suivantes :

- le mobile es un point matériel se mouvant dans un plan.
- Les obstacles sont indépendants ne présentant aucun point de contact entre eux et leurs contours sont des courbes fermées.
- Le mobile dispose des informations suivantes :
  - ses coordonnées instantanées ;
  - un état binaire indiquant s'il est ou non en contact avec un obstacle ;
  - la distance et la direction au but ;
- Le mobile peut effectuer trois actions :
  - se déplacer suivant une ligne droite le reliant au but ;
  - suivre le contour d'un obstacle ;
  - s'arrêter.

❖ *Algorithme BUG1* :

Le principe de l'algorithme BUG1 est le suivant :
A partir de sa position de départ S,

- Le mobile se déplace suivant une ligne droite reliant sa position actuelle au point d'arrivée B, ,jusqu'à rencontrer un obstacle ;
- A partir de son point de contacte avec l'obstacle H, il fait un tour complet de celui-ci dans une direction fixée, en cherchant un point de son contour Q, dont la distance au point but serait minimale. Il repère ce point;
- Une fois de retour au point H (où il a rencontré l'obstacle pour la première fois), il se déplace vers Q, suivant toujours le contour de l'obstacle, et selon la direction qui lui garantie le plus court chemin jusqu'à ce point ;
- Au point Q, il quitte l'obstacle en se dirigeant suivant une droite reliant sa position actuelle au but, jusqu'à rencontrer un autre

obstacle, et ainsi de suite..., ces opérations se répètent jusqu'à ce qu'il atteigne le but.

BUG1 présente les propriétés suivantes :

- le mobile ne peut rencontrer que les obstacles situés à l'intérieur d'un cercle de rayon R centré sur le point but.
- Si p_i est le périmètre d'un tel obstacle, la longueur de la trajectoire parcourue entre le départ et la destination ne peut dépasser $R+1.5\sum p_i$ et ne peut être inférieure à $\sum p_i$.

❖ *Algorithme BUG2*:

L'algorithme BUG2 propose de suivre les étapes suivantes :
A partir de la position de départ,

- Le mobile suit une ligne droite reliant le point source au point but jusqu'à rencontrer un obstacle en point H ;
- Il suit le contour de l'obstacle jusqu'à croiser la droite reliant la source au but, en un point L tel que :
  1. la distance de ce point au point but soit inférieure à celle du point H (où il a rencontré l'obstacle) au point but.
  2. la droite reliant le point L au but n'intersecte pas l'obstacle en contact.
- il quitte l'obstacle au point L et se dirige vers le but suivant la droite le reliant à celui-ci, jusqu'à rencontrer un nouvel obstacle...ou le but.

L'algorithme BUG2 régissant le mouvement d'un mobile, lui garantie que :

- les obstacles rencontrés sur sa route ne peuvent être autres que ceux qui intersectent la droite reliant la source au but.
- Dans le cas où la configuration d'obstacles ne présente pas de cyclage, la longueur du chemin ne peut être supérieure à $R+\sum p_i$.

Un troisième algorithme dans la même série, qui combine les deux algorithmes BUG1 et BUG2 en rassemblant leur avantage, est proposé par les mêmes auteurs.

Toutes les méthodes qui viennent d'être mentionnées sont des méthodes primitives de base et qui n'exigent aucune connaissance de l'environnement si ce n'est le contact avec l'objet, d'où aucune nécessité de capteurs évolués. Un simple mécanisme tout ou rien, s'enclenchant au contact, est en fait utilisé (boutons poussoirs, fin de course...).

Les principaux inconvénients de ces méthodes sont qu'elles tendent à faire errer le robot ou à le faire tourner en rond. Et s'apparente

à un être aveugle n'utilisant que son sens du toucher pour se déplacer en tâtonnant le long des objets qu'il trouve sur son passage.

II.2.2. La planification locale :

La planification de la trajectoire est locale lorsque les connaissances sur l'environnement ne concernent qu'un voisinage de la tâche courante du robot, ou une certaine région de l'espace globale. C'est généralement un sous-problème de la planification de la trajectoire globale, où la source et le but ne sont que sous-source et sous-but, points de passage de cette trajectoire.

Deux orientations différentes existent pour la résolution de ce type de problème:

– Milieu inconnu :

N'ayant pas de connaissances préalables sur le milieu, le robot construit sa carte par régions en utilisant les informations issues des capteurs, et son comportement est géré par l'influence des obstacles. Une telle planification peut se faire par la méthode des champs de potentiels artificiels. (elle sera illustrée dans § modélisation).

– Milieu connu :

S'appuyant sur une base de données dont il dispose, sur son environnement, et considérant une géométrie locale, il adapte à la configuration rencontrée des actions spécifique à entreprendre. Ces actions forment un algorithme qui sert d'expert local. La traversée d'un couloir en L [6] et le passage à travers une porte [7], sont des illustrations de cette méthode .

II.3. La modélisation de l'environnement :

Le mouvement d'un objet quelconque est assujetti par la présence d'autres objets. Cette contrainte est étroitement liée à la forme de ces objets et leurs emplacement . C'est pourquoi, il est indispensable au robot, de disposer de certaines connaissances sur le milieu dans lequel se fait son acheminement vers le but.

Une carte de l'environnement est construite par le "cartographe" à partir de données issues :
- soit d'un module de vision,
- soit de capteurs fournissant une représentation locale de l'environnement,

- soit encore d'une base de donnée CAO décrivant les objets par leur géométrie et leur position dans l'espace.

Les différentes données sous leur forme primaire comme décrite à l'instant, ne peuvent être intégrées directement dans la planification, et pour ce faire l'étape de *modélisation sert à adapter ces données.*

La modélisation de l'environnement est une représentation de cette carte de façon à ce que le module chargé de *la recherche du chemin* puisse la comprendre et l'exploiter.

Dans l'architecture de R. Chavez [2], la modélisation de l'environnement est la première tâche du planificateur (voir § 2.1. d.), et la recherche de chemin dans l'environnement modélisé en est la deuxième.

II.3.1. La modélisation statique de l'environnement :

II.3.1.1. Définitions :

Définition 1 [1] :
La modélisation est dite *statique, si* les éléments de l'environnement à modéliser sont connus aussi bien par leurs géométrie, par leurs dimensions, que par leurs localisations et ne subissent aucun changement de formes ni de positions.

Définition 2 [1] :
Un environnement structuré est un espace E représentés sous forme de polygones :

$$E = \{P_1, P_2, \ldots, P_n\}$$

avec $P_i = \{p_1, p_2, \ldots, p_m\}$ un polygone formé par un ensemble de segments de droites dont les extrémités sont les sommets p_i du polygone et dont les paramètres sont parfaitement connus.

Définition 3 [1]:
Le modèle topologique est élaboré à partir du modèle géométrique en liant un ensemble de zones dans lesquelles le mobile peut se mouvoir par des fonctions de connexités qui tiennent compte des caractéristiques géométriques des zones et du mobile, de leur localisation relative, du temps et des caractéristiques cinématiques du mobile.

Qu'est ce qu'est l'espace libre :

Définition 1 [1] :
On appelle zones occupées, les régions occupées par des obstacles matériels ou des sites et dont l'accès est interdit.

Définition 2 [1] :
Les zones libres sont les régions complémentaires aux zones occupées sur l'ensemble de l'espace de travail.

Définition 3 [1] :
La modélisation de l'espace libre est une représentation binaire de l'ensemble de l'espace qui consiste à marquer les zones libres ou, dans certains cas un sous-espace inclus dans ces zones.

II.3.1.2. Type de modèles :

On distingue trois types de modélisations de l'espace libre:
- modèle par points ;
- modélisation par discrétisation de l'espace ;
- modélisation par rétraction de l'espace .

II.3.1.2.1. Le modèle par points :

Deux grandes méthodes constitue cette approche:
- Graphes de Visibilité ;
- Graphes des Tangentes.

❖ *Graphes de Visibilité :V-Graph* [1]

Considérons un espace E parsemé d'obstacles polygonaux (ou assimilés comme tels) : $E = \{P_1, P_2, \ldots, P_n\}$, où P_i est un polygone défini par ses sommets $P_i = \{p^i_1, p^i_2, \ldots, p^i_m\}$.

Construire le graphe de visibilité revient à réduire l'espace libre en un ensemble de points de passage reliés par des segments de droites. Les points de passages ne sont autres que les sommets p^i_k des obstacles. Ceci peut être formulé comme suit :

Soit N l'ensemble de tous les sommets des obstacles de E :

$$N=\cup p^i_k ;$$

Soit L l'ensemble des droites l_{ij} reliant les sommets de N de façon à ne pas intersecter les obstacles :

$$L=\cup l_{ij} \quad \text{avec} \quad l_{ij}\cap P_k=\varnothing ;$$

On définit le graphe de visibilité sur l'espace E par un graphe VG (voir définition de graphe, dans 2.1.) :

$$VG=(N,L) ;$$

La recherche de chemin s'effectuera sur le graphe :

$$G=(N',L') ;$$

Où N' est un ensemble contenant les sommets de tous les obstacles, le point source S et le point but B ;

$$N'=N\cup\{S,B\} ;$$

Et L' l'ensemble des droites l'_{ij} reliant les sommets de N de façon à ne pas intersecter les obstacles :

$$L'=\cup l'_{ij} \quad \text{avec} \quad l'_{ij}\cap P_k=\varnothing ;$$

Le graphe de visibilité est conçu en vue de l'obtention d'un chemin optimal de point de vue distance parcourue.

❖ *B- Graphe des tangentes :T-Graph* [1] (Fig.2)

La modélisation par graphe de visibilité ne s'applique qu'aux obstacles polygonaux, d'où la nécessité d'approximer les autres obstacles à des polygones.

Un modèle topologique est proposé en extension au précédent, il définit l'espace libre comme étant une séquence de segments de droites tangentes à l'enveloppe convexe des obstacles qu'ils soient polygonaux convexes ou polygonaux concaves, ou courbes, et de segments curvilignes correspondant à des parties des limites convexes des obstacles courbes.

Cette représentation garantie aussi l'optimalité au sens de la distance parcourue et nécessite un espace mémoire moindre que V-Graph.

L'optimalité du chemin, dans les deux modèles, n'est en vérité garanti qu'avec l'hypothèse que le robot soit un point matériel et ne possède aucune dimension.

En réalité, les dimensions du robot complique la navigation, d'une part par le fait que le passage entre deux obstacles trop rapprochés peut s'avérer trop étroit pour un robot réel avec une certaine grosseur, cas qui ne poserait pas de problème pour un point matériel.

D'autre part, par le fait que le chemin passant par les sommets des obstacles (dans le cas de V-Graph) ou tangent à ceux-ci (dans le cas de T-Graph), fait approcher de trop près le robot des obstacles. C'est pourquoi il a été prévu de transférer les dimensions du robot sur l'environnement en grossissant les obstacles selon une orientation donnée du robot.

Aspect bien peu accommodant puisque à chaque orientation du robot correspond un modèle différent de l'environnement. Il est aussi possible bien sûr d'agrandir les obstacles de tous les côtés par la même

grandeur qui serait la dimension maximale du robot, garantissant ainsi un chemin sûr quelque soit l'orientation du robot, seulement cela présente un risque de débordement sur un passage serré et le barrer, dans un cas extrême.

Il est aussi possible d'agrandir l'ensemble des obstacles de l'environnement d'un coefficient de sécurité, sans prendre en compte l'orientation du robot. Cette solution est plus favorable dans une configuration d'obstacles plus ou moins espacée.

Le problème est moindre pour un robot circulaire. Il suffit dans ce cas d'accroître les obstacles en déplaçant leurs sommets d'une distance de sécurité τ égale au moins au rayon du robot. On remarquera alors que les sommets des obstacles grossis deviendront des arcs de cercle de rayon τ.

Comme il est plus aisé de travailler dans un espace polygonal, les arcs de cercles seront tronqués introduisant des erreurs de modélisation et provoquant un risque de collision dans le cas d'une troncature par défaut, et un risque d'éliminer une solution dans le cas de troncature par excès.

II.3.1.2.2. modélisation par la discrétisation de l'espace :

La discrétisation de l'espace consiste à subdiviser soit l'environnement entier, soit l'espace libre, en un ensemble de cellules. De nombreux algorithmes sont proposés dans la littérature, ils diffèrent par :

- la procédure de la partition,
- la forme et la taille des cellules,
- et dans leur représentation informatique.

On peut distinguer deux grandes classes de discrétisation :

- un partage de l'environnement suivant ses limites naturelles : en prolongeant les arrêtes des obstacles par exemple ou en utilisant les arcs du graphe de visibilité...
- une discrétisation choisissant une forme particulière pour les cellules, s'adaptant au mieux à l'espace libre.

❖ Exemple de cas pratique :

Prenons l'exemple de [8], où les auteurs proposent un algorithme de recherche de chemin pour un robot rigide polygonal, à trois degrés de liberté – deux translations suivant les axes x, y et une rotation θ – en mouvement dans un environnement plan parsemé d'obstacles polygonaux. L'algorithme est basé sur une représentation de l'espace de configurations (la configuration est l'ensemble des paramètres

indépendants caractérisant la position de chaque point de l'objet, et l'espace des configurations est l'espace de toutes les configurations possibles que peut prendre le mobile.

L'espace des configurations est un espace à trois dimensions où la configuration de l'objet mobile est réduite à un point dit point de configuration (*configuration point*), dont les deux premières coordonnées sont les coordonnées (x_0, y_0) de l'origine d'un repère associé à cet objet dit point de référence, par rapport au référentiel de base dans lequel se meut le mobile, et la troisième coordonnée définit l'orientation θ_0 de ce repère toujours par rapport au repère de base.

L'espace des configurations contient des régions interdites (*configuration obstacles*) dans lesquelles les coordonnées du point de configuration serait telles qu'elles induiraient des collisions du mobile avec les obstacles, une telle région, dans le cas d'obstacles polygonaux convexes, peut être représentée par une superposition de formes polygonales planes parallèles et normales à l'axe des rotations, chaque polygone représente un obstacle grossi par le mobile quand il prend une orientation θ fixe, et variant d'une tranche à une autre (on y reviendra plus tard).

Le problème de recherche de configurations sans collision dans un tel espace équivaut au problème de positionner le point de configuration dans l'espace des configurations, en dehors des zones interdites. L'espace ainsi défini est subdivisé, en cellules réctangloïdes, aux arêtes parallèles aux trois axes de l'espace des configurations, ces cellules sont notées *pleines* lorsque aucun point de ces cellules n'est en dehors des régions interdites, *vides* si au contraire elles se situent complètement hors de ces zones et *mixtes* si une partie seulement est à l'intérieur de l'obstacle.

La recherche d'un chemin se fait dans les réseaux de cellules vides connectées entre elles et contenant les point–configurations but et source, si un tel réseau n'existe pas, le chemin est autorisé à passer par des cellules mixtes, qui sont par la suite, coupées en deux par un plan normal à l'un des axes des coordonnées, l'endroit de la coupure sera choisi de sorte à remplir trois objectifs :

- au moins une des deux cellules résultant de la division est ou vide ou pleine.
- La cellule pleine ou vide est de volume aussi grand que possible.
- La division est aussi rapide que possible.

La recherche de chemin est refaite sur l'ensemble des cellules vides, dans le cas d'absence de solutions, d'autres cellules mixtes sont

divisées...et ainsi de suite, jusqu'à atteindre une résolution maximale, au-delà de laquelle le problème est classé insoluble.

❖ Exemple d'utilisation de cônes .

Brooks dans [9], décrit l'espace libre sous forme d'un réseau de cônes généralisés. Les cotés du cône sont définis par deux arêtes de deux obstacles qui " se regardent" , et par deux arcs de cercles à ses extrémités ou des lignes droites les approximant. Le mouvement du robot dans ce type de représentation se limite à des translations en lignes droites à direction fixe et des changements d'orientation à position fixe. Le mobile se meut le long des axes des cônes. A chaque cône est associé un intervalle de direction qu'il peut prendre en gardant son centre sur l'axe.

Pour passer entre deux cônes communiquant, le mobile doit ajuster sa direction à ce dernier. Pour cela, il faut que les deux intervalles d'orientations de deux cônes s'intersectent. Le mouvement se résume alors à des translations le long des axes des cônes alternés avec des mouvements de rotation à l'intersection de ces axes. La trajectoire est recherchée dans l'ensemble des cônes reliant la position initiale et la position finale, et entre lesquels l'accès est autorisé.

II.3.1.2.1. modélisation par la rétraction de l'espace :

La modélisation de l'espace d'évolution consiste à rétracter l'espace libre des configurations sans collision L sur un sous-espace R de dimension inférieure.

Cela revient à définir à l'intérieur de l'espace L, des zones privilégiées qui constitueront l'espace rétracté R et qui assureront la minimisation des informations sur l'espace R.

La dimension du problème est ainsi réduite et la recherche d'un chemin optimal dans un espace libre plan est ramenée à un problème de recherche dans un graphe. [10]

On distingue deux types de rétractions :
- la rétraction sur les diagrammes de Voronoï (au plus loin des obstacles).
- la rétraction sur le bord de l'espace libre des configurations (au plus près des obstacles).

❖ *La rétraction au plus loin des obstacles : Le diagramme de Voronoï :*

Définition du diagramme de Voronoï : [1],[10]

Soit un ensemble $M=\{m_1, m_2, \ldots, m_n\}$ de n points de R^d, le diagramme de Voronoï est un « pavage » de cellules ou de polyèdres V_i, i =1,n.

On définit un polyèdre de Voronoï associé au point $m_i \in M$, comme suit :

$$V_i=\{x\in R^d : \delta(x, m_i) \leq \delta(x, m_j) \ \forall\ j\leq n, j\neq i\}$$

Où δ désigne la distance euclidienne.

Propriété 1 :
Les polyèdres de Voronoï sont convexes et n'ont pas plus de n-1 sommets et arrêtes ;

Propriété 2 :

- $\bigcup_{i=1}^{n} V_i = R^d$;
- l'intersection des intérieurs des deux cellules V_i et Vj est vide ;
- deux cellules voisines sont adjacentes sur une distances d'<d, si leurs bords s'intersectent.

Propriété 3 :
Chaque point appartenant aux arrêtes des polyèdres de Voronoï est exactement à égale distance de deux points de M. Chaque sommet est à égale distance de trois points.

Propriété 4 :
Le diagramme de Voronoï contient exactement n régions.

Diagramme de Voronoi généralisé

Afin d'appliquer le diagramme de Voronoï où l'environnement est jonchés d'obstacles généralement polygonaux et non ponctuels on a défini le *diagramme de Voronoï généralisé.*

Définition du diagramme de Voronoï généralisé [11]*:*

Le diagramme de Voronoï d'un ensemble d'obstacles convexes est une décomposition de l'espace libre situé entre les obstacles en Régions R_i dites de Voronoï, de façon à ce que chaque point appartenant à cette région soit le plus proche d'un et d'un seul obstacle. Chaque région de Voronoï est bornée par les côtés de Voronoï de l'extérieur, et par les arrêtes de l'obstacle qu'elle entoure, de l'intérieur.

Propriété 1 :
Chacune des régions de Voronoï est fermée (car l'espace de travail est borné), et leurs bords sont communicant.

Propriété 2 :
Les interactions entre les sommets et les arrêtes des obstacles provoquent la formation de trois types d'arêtes de Voronoï, qui sont :

a) un segment de droite bissecteur entre deux arêtes d'obstacles différents et qui se regardent : chaque point de ce segment est équidistant et le plus proche de ces deux arêtes. Ce type d'arête divise l'espace libre en deux régions, chacune étant la plus proche d'une arête d'obstacle.
b) un segment de droite médiatrice de deux sommets, d'obstacles différents et se regardant ; Ce type d'arête partage l'espace libre en deux régions, chacune étant la plus proche d'un des sommets.
c) Un arc de parabole représentant le lieu géométrique des points à égale et minimale distance entre une arête et un sommet.

Propriété 3 :
L'intersection de chaque paire d'arêtes de Voronoï de types différents donnent naissance à 6 types de régions .

L'ensemble des arêtes de Voronoï forment un réseau de chemins assurant de tenir le mobile aussi loin des obstacles que possible. Un mobile en forme de disque navigue sur ces chemins de manière à ce que le centre du disque glisse le long des arêtes de Voronoï. Si le robot est polygonal alors on le circonscrit dans un cercle.

Avant de présenter la rétraction sur le bord de l'espace libre des configurations, définissons l'espace libre des configurations.

❖ *Espace libre des configurations :*

Lors de la modélisation de l'espace libre, les dimensions et orientation du robot, n'ont pas été prise en compte. Or c'est un aspect important qui pose souvent problème dans le cas où les obstacles ne sont pas assez espacés et la possibilité de franchir un passage étroit par exemple, dépend de la largeur du robot, de sa longueur ainsi que de son orientation. La navigation dans des zones délicates exige la spécification exacte de la séquence de configurations à prendre, sans quoi la traversée ne serait pas possible. Il a été précisé que la question de modéliser l'espace libre des configurations se posait seulement pour la recherche de chemin dans des environnements encombrés (*clustered)* en

obstacles, car dans le cas contraire, cette recherche pourrait simplement s'effectuer dans l'espace libre décrit précédemment en lui retranchant une épaisseur de sécurité.

Dans la réalité, la recherche de traversée optimale met en jeu toutes les variables liées aux degrés de liberté du robot.

Définition :

Considérons un robot quelconque à n degrés de liberté.
Soit $Q=\{q_1, q_2, \ldots, q_n\}$ le vecteur des coordonnées des articulations,
O l'ensembles des obstacles de l'environnement,
et u est un nombre fini de configurations que peut prendre le robot, alors l'espace des configurations est définit comme suit :

$$Cs = \bigcup_{i=1}^{u} Q_i \left| (Q_i \cap O) = \varnothing \right.$$

l'espace de configurations représente l'ensemble des états que peuvent prendre les différentes coordonnées du robot lui garantissant un mouvement sûr, en dehors des obstacles. Pour un robot mobile ces coordonnées se résument à une translation suivant l'axe x, une autre suivant y et une rotation θ autour de l'axe z, soit deux translation et une rotation dans le plan.

Sommes de Minkowski :

On définit l'espace de travail *E* comme un ensemble de points de R^d ; *d* étant la dimension du problème (*d*=2, pour les problèmes plans et *d*=3, pour les problèmes dans l'espace).

Considérons un obstacle *O* de l'environnement comme étant un ensemble borné de points de *E* fixes. Et *M*, un ensemble borné de points mobiles de *E*, constituant le robot.

Soient deux ensembles *A* et *B* définis dans un repère d'origine *o* comme suit :

$$A = \left\{ a / a = \vec{om} : m \in O \right\} \quad \text{et} \quad B = \left\{ b / b = \vec{om} : m \in M \right\} ;$$

Définition : la différence de Minkowski est définie par l'ensemble suivant :

$$A - B = \{ a - b / a \in A, b \in B \};$$

où $a - b$ désigne la différence vectorielle.

Il a été remarqué [10] que, pour porter le mobile *B* à entrer en collision avec l'obstacle *A*, il lui faut appliquer une translation de vecteur appartenant à l'ensemble *A–B*, car le vecteur $a-b$ n'est autre que le vecteur reliant un point de *B* à un point de *A*, l'amenant ainsi de *B* vers *A*.

En se basant sur ce qui vient d'être dit, on peut affirmer que le robot est autorisé à occuper exclusivement l'espace complémentaire à *A–B* dans *E*.

Lozano-Pérez a proposé dans [12] une approche de planification de trajectoire utilisant une représentation de l'espace des configurations qui consiste à définir, en fonction de l'orientation θ du robot, l'espace accessible à un point choisi sur ce robot (un sommet de référence). Ceci est décrit en découpant dans l'espace de la tâche, des polygones qui sont les obstacles augmenté de la dimension du robot pour l'orientation θ en question.

Il a montré aussi que cet accroissement des obstacles correspondait en fait à la différence de Minkowski entre l'obstacle et le robot dans sa configuration initiale. Son idée se résume en bref dans ce qui suit :

Soit R un polyèdre (polygone dans le cas plan) convexe qui délimite l'espace E de la tâche contenant le corps mobile A (qui est l'union de polyèdres (polygones) convexes $A = \cup A_i$), et un ensemble B d'obstacles polyédriques (polygonaux) convexes ($B = \cup B_i$).

L'auteur pose les deux problèmes fondamentaux de la planification, et les formule de la manière suivante :

- *la recherche de l'espace de configurations sûres, pour le mobile A, dans R* : elle revient à rechercher les configurations pour lesquelles on a pour tout i et pour tout j :$A_i \cap B_j = \varnothing$;
- *la recherche d'un chemin sûr pour A* : cela revient à rechercher un chemin à partir de la configuration *source* jusqu'à la configuration *but*, de manière à ce que A reste toujours dans R et toutes ses configurations sur ce chemin soient *sûres*.

Si le solide A est à n degrés de liberté, sa position et son orientation sont totalement spécifiées par un vecteur de dimension n qui définit sa configuration.

L'entier espace de configurations de dimension n du solide A est noté $Cspace_A$.

On définit $CO_A(B)$ comme étant l'ensemble de configurations de $Cspace_A$ pour lesquelles A recouvre B: $A \cap B \neq \varnothing$; de même on définit $CI_A(B)$, comme étant l'ensemble de configurations pour lesquelles A est à l'intérieur de B : $A \subseteq B$.

Ces deux notions comportent les informations nécessaires pour la résolution des deux problèmes formulés ci-dessus.

Considérons à présent un problème à deux dimensions et A un polygone convexe. Si l'orientation de A est fixe. Il suffirait de connaître les coordonnées x et y d'un certain sommet de référence rv_A de A pour déterminer complètement la configuration de A. $Cspace_A$ se réduit alors au plan xy. La présence d'un obstacle polygonal B contraint rv_A à rester en dehors de $CO_A(B)$ qui est un polygone contenant l'obstacle B et dont les côtes seraient décrits par le sommet rv_A lorsque le polygone A glisse sur les côtés du polygone B sans intersecter son intérieur, et en gardant toujours la même orientation. De ce fait, rechercher les configurations sûres pour le polygone A revient à placer le sommet rv_A en dehors de $CO_A(B)$ et à l'intérieur de $CI_A(R)$.

Il a été montré par l'auteur que :

$$CO_A(B) = B - (A)_0,$$

où $(A)_0$ est le polygone A, et –, est l'opérateur de différence de Minkowski.

Lorsque l'orientation θ du polygone varie durant le mouvement, une troisième dimension s'ajoute à $CO_A(B)$ et $CI_A(B)$, celle de l'angle de rotation θ, et ils deviennent des volumes non convexes à surface courbes, néanmoins ils gardent la propriété énoncée par le théorème suivant [12] :

Théorème :

Si le mobile A et l'obstacle B sont des polygone convexes, alors les sections suivant θ de $CO_A(B)$ ainsi que celles de $CI_A(B)$ sont toujours bornées, polygonales et convexes.

Nous allons nous en tenir au cas plan car c'est tous ce qui nous intéresse dans le présent travail. Nous n'allons pas aborder non plus le second problème formulé par Lozano-Pérez : la recherche d'un chemin sûr, car ce n'est pas le sujet du présent chapitre.

❖ *La rétraction sur le bord de l'espace libre des configurations :*

Deux exemples d'une telle rétraction figurent dans [8],[13], ils consistent dans la résolution du problème de déplacement en translation et rotation d'un objet dans le plan. Ces mouvements se font sur le bord ∂L de l'espace libre des configurations L, ∂L correspond aux configurations limites du robot avant d'entrer en collision avec l'obstacle, pour lesquelles le robot est en contact avec l'obstacle. Ces configurations s'obtiennent lorsqu'un sommet du robot appartient à un côté de l'obstacle ou vice versa. L'espace L est tridimensionnel, les deux premières dimensions

étant celles du problème (x et y), et la troisième, représente l'orientation θ du robot. Le bord de L est hélicoïdal, et lorsqu'il est coupé par un plan perpendiculaire à l'axe des rotations, présente des segments de droites, par conséquent la projection de l'espace complémentaire à l'espace libre des configurations L, sur un tel plan, est polygonale, et représente l'espace des configurations interdites pour une rotation fixée du robot.

La recherche du chemin se fera donc sur le bord ∂L enveloppant les obstacles, et lorsque la connexion entre ces enveloppes (relatives aux différents obstacles) n'est pas assurée, ou si l'une des configurations but ou source (ou même les deux) n'est pas connectée au reste, il faudrait dans ces cas prévoir une méthode auxiliaire assurant ces connexions.

II.3.2. La modélisation dynamique de l'environnement :

La modélisation de l'environnement est dynamique, lorsque ce dernier est mal connu ou lorsqu'il n'est pas connu du tout ou alors dans le cas où il n'est pas statique, mais amené à évoluer dans le temps. L'appréhension de l'environnement comme le nom l'indique se fait dynamiquement, lors du mouvement du robot, en temps réel. La planification de la trajectoire dans un tel milieu ne peut être que locale, car les connaissances dont dispose le robot à un moment donné ne concernent qu'une partie plus ou moins avoisinante de l'espace de travail. La construction de la carte de l'environnement se fait sur la base des informations télémétriques fournies par les capteurs, ces données sont reportées ensuite sur la carte globale. Pour cela, la localisation des capteurs, et donc du robot, par rapport au référentiel de base est nécessaire.

Ceci est réalisé grâce à des balises, soit artificielles implantés dans l'environnement dans ce but précis, soit naturelles faisant partie de celui-ci.

Il faut prendre en considération le fait que la précision et la finesse du modèle dépend essentiellement des capteurs, qui délivrent une mesure entachée d'erreurs qu'il est essentiel de modéliser afin de pouvoir les incorporer lors de la construction de la carte. On peut distinguer trois types d'erreurs :

- *les erreurs géométriques* : elles surgissent lors de la localisation du référentiel robot relativement au référentiel globale. Cette localisation se fait à l'aide de accéléromètres (comme son nom l'indique), speedomètre (compteur de vitesse), gyromètres (appareils mesurant les variations de direction) et odomètres (appareils de mesure de la distance parcourue en terme de rotation des roues). L'information est déduite de ces capteurs par intégrations successives, avec tous les

inconvénients de l'intégration numérique principalement l'accumulation d'erreurs.

- *les erreurs transitoires* : elles résultent du caractère non statique de l'environnement. La prise de conscience d'événements, leur intégration, la prise de décision et l'action sont des opérations qui ne se font pas instantanément, dès lors, entre le moment où le robot reçoit une information concernant un événement évoluant dans le temps et l'instant où il entreprend d'agir en conséquence, il est plus que probable que l'événement en question ait changé. Le comportement du robot, de ce fait, risque d'être incohérent : la prédiction, dans ces cas-là, peut s'avérer très utile et apporte souvent la solution souhaitée au problème.

- *Les erreurs de mesures* : elles sont caractéristiques des systèmes de mesure et consistent en les incertitudes sur les grandeurs issues des capteurs. On parle généralement de ce type d'erreur lorsqu'il s'agit de capteurs télémétriques qui mesurent la distance aux objets présent dans l'environnement. Prenons par exemple, les capteurs acoustiques (à ultrason), les plus utilisés dans ce domaine pour leur coût peu élevé relativement aux capteurs optiques (Laser ou infrarouge), mais présentant une précision nettement moins élevée. Ces capteurs émettent des faisceaux coniques d'ondes sonores qui sont réfléchies quand elles rencontrent un milieu différent du milieu de propagation. L'imprécision est due d'une part au fait que la vitesse de propagation (avec laquelle la distance aux obstacles est calculée) est dépendante de la température du milieu, et d'autre part, la fiabilité des mesures est fonction de la position du point détecté (appartenant à un obstacle) dans le cône formé par le faisceau sonore : la probabilité qu'un point P fasse partie d'une région vide est maximale à la source S de l'émission (au sommet du cône) et va en décroissant jusqu'à s'annuler à la distance R entre les points S et P, retournée par le capteur. La probabilité que ce point appartienne à une région occupée prend une valeur maximale à la distance R de la source. Les deux probabilités que le point soit vide ou occupé sont maximales sur l'axe du cône et décroissent jusqu'à s'annuler sur ces génératrices.

II.3.2.1. Les grilles d'incertitude [1] :

Le modèle de l'environnement peut être décrit sous forme d'une grille, à chaque cellule de laquelle est associé la probabilité de son occupation par un obstacle. Les différentes méthodes utilisant ce modèle, varient d'après la taille et le nombre des cellules, la loi de probabilité et de leur mise à jour.

Le modèle bayesien est le plus utilisé dans la littérature. D'après le théorème de Bayes, la probabilité conditionnelle renseignant sur l'état s_i d'une cellule en prenant en compte la mesure M donnée par les capteurs s'écrit :

$$P(s_i \setminus M) = \frac{P(M \setminus s_i)P(s_i)}{\sum_{j=1}^{2} P(M \setminus s_j)P(s_j)} \qquad \text{(II.1)}$$

où s_i est l'état vide ou occupé d'une cellule.*(i=1 ou 2)* ;

$P(M \setminus s_i)$ est la probabilité que la mesure M soit prise sachant que la cellule soit à l'état s_i, elle est issue du modèle probabiliste du capteur qui aurait dû être défini en amont ;

$P(s_i)$ est la probabilité à priori que la cellule soit à l'état s_i, elles sont basées sur les connaissances dont on dispose avant d'avoir la mesure M, ces connaissances peuvent provenir des mesures antérieures.

N'ayant que deux états : $s_1=O$ (occupé) et $s_2= V$ (vide), et en faisant intervenir la mise à jour séquentielle, la probabilité que la cellule considérée soit occupée en tenant compte de la mesure M_{k+1}:

$$P(O \setminus M_{k+1}) = \frac{P(M_{k+1} \setminus O)P(O \setminus M_k)}{P(M_{k+1} \setminus O)P(O \setminus M_k) + P(M_{k+1} \setminus V)P(V \setminus M_k)} \qquad \text{(II.2)}$$

comme la cellule ne peut être que vide ou occupée les probabilités conditionnelles associées à ces deux événements sont exclusifs, et la probabilité que la cellule soit vide en tenant compte de la mesure M_{k+1} est donnée par :

$$P(V \setminus M_{k+1}) = 1 - P(O \setminus M_{k+1}) \qquad \text{(II.3)}$$

les probabilités $P(M_0 \setminus V)$ et $P(M_0 \setminus O)$, lorsque aucune mesure n'a été encore prise sont égales à 0.5 (qui correspond à l'état inconnu).

L'emploi de la grille d'incertitude permet d'avoir une distribution spatiale de probabilités de présence d'obstacle, après une exploration de l'environnement. Il est surtout avantageux d'y recourir lorsque l'environnement est dynamique.

Une application intéressante de cette méthode a été proposée dans [14] qui consiste en une approche d'évitement d'obstacle en temps réel, en intégrant la représentation de l'environnement, par la grille d'incertitude avec navigation, par champ de potentiel. Une telle alliance permet de palier aux imprécisions des capteurs ainsi qu'au problème de minima locaux propre à la recherche de chemin par le champ de

potentiel, et dès lors, de naviguer dans un milieu inconnu, à une assez grande vitesse évitant les obstacles sans pour cela avoir à s'arrêter.

II.3.2.2. Le modèle géométrique :

La méthode précédente a l'inconvénient de ne pas prendre en considération la forme géométrique et le lien entre les objets. La modélisation géométrique de l'environnement consiste à le représenter par des formes géométriques constituées fondamentalement de segments de droites (des polygones, par exemple).

J.L. Crowley [15] propose une méthode pour extraire des droites de l'environnement à partir des informations issues d'un capteur à ultrason rotatif. Le principe est le suivant : le capteur durant sa rotation fournit des mesures à une fréquence donnée ; chaque intervalle de temps donné (ou intervalle d'angle balayé), une série de n mesures décrivant la distance à un ensemble de n points d'un bord d'obstacle, sont obtenues. A partir de ces n points de mesure :

1. les points extrêmes sont reliés pour former une droite,
2. les paramètres de l'équation de cette droite sont calculés,
3. la distance entre chacun des n-2 points restants et cette droite est calculée,
4. si la distance au point le plus éloigné est inférieure à une certaine tolérance, alors, la droite dont les paramètres ont été calculés en 2. est représentative de tous les n points.
5. sinon, l'ensemble des n points est partagé en deux, au niveau du point le plus éloigné du segment de droite, et le dit point est relié aux deux extrémités de ce segment pour former deux nouvelles droites sur lesquelles sont appliquées les 5 étapes. Le processus est renouvelé jusqu'à ce que la distance au point le plus éloigné soit inférieure à la tolérance.

Cette méthode est néanmoins lourde et mal adaptée à des scènes changeantes rapidement.

II.4. Evitement d'obstacle :

Lozano-Pérez a regroupé, dans [16], l'ensemble des algorithmes constituant les différentes procédures d'évitement d'obstacles, en trois classes :

– Hypothèses et tests : *hypothesize-and-test* ou *generate-and-test* ;
– Fonction de pénalité ;
– Explicitation de l'espace libre.

II.4.1. La méthode Hypothèse et test :

Cette méthode est l'une des premières proposées pour l'évitement d'obstacles, elle se résume en trois étapes :

1. *Générer* un chemin candidat reliant les configurations initiales et finales du robot ;
2. Sélectionner quelques configurations intermédiaires et pour chacune de cette configuration, *tester* la possibilité de collision avec un(des) obstacle(s).
3. Si la possibilité de collision existe, *proposer un mouvement évitant cette collision,* en examinant le(les) obstacle(s) l'ayant provoqué.

Ce processus est répété pour le chemin modifié. La méthode est intéressante de par sa simplicité. Une opération de base consiste dans la détection de collisions potentielles qui se traduisent par des intersections non vides entre la configuration du robot et de l'obstacle. L'autre opération s'avère plus délicate : la modification du chemin le rendant en dehors des obstacles peut être difficile à réaliser. Des méthodes à grandes précautions ont été proposées à cet effet, formulant des hypothèses comme enfermer les obstacles dans des sphères.

A ce propos, les auteurs de [11] ont proposé un schéma de modélisation d'objets (robot mobile et obstacles) polyédriques, nommé *SSA (successive spherical approximation)*. Ce modèle consiste à représenter ces objets par un ensemble de secteurs sphériques concentriques de façon à ce que chaque face du polyèdre soit bornée par deux secteurs sphériques, l'un de l'intérieur de rayon égal à la plus petite distance du centre à la face, l'autre de l'extérieur, de rayon égal à la plus grande distance du centre à la face. Ceci leur a permis, en suivant la méthode « hypothesize-and-test » mentionnée ci-dessus, de concevoir une stratégie de navigation offrant une détection efficace d'obstacles et une planification rapide de chemin dans un espace tridimensionnel, parsemé d'obstacles.

Autre exemple concernant la présente méthode :J.G. de Lamadrid présente dans [18] une méthode de navigation d'un robot en présence d'obstacles mobiles. Les obstacles sont approximés par des cercles, et leur équation de mouvement est quadratique en fonction du temps. Le robot est un point ayant un mouvement rectiligne uniforme. Il se déplace sur la droite reliant la source au but, déviant de sa trajectoire à chaque détection de collision sur celle-ci. Cette collision est détectée quand la distance du mobile au centre de l'obstacle, à un moment donné, est inférieure au rayon de l'obstacle.

Deux méthodes ont été utilisées pour la replanification :

- la première étant en définissant un sous-but sur la tangente à l'obstacle en point de collision, et à une certaine distance de ce point.
- La seconde étant en plongeant l'espace dans un champ de potentiel (de Khatib) répulsif autour des obstacles, et attractif autour du but et du segment de droite source-but.

Arrivé au sous-but, le mobile continue d'avancer en ligne droite se dirigeant vers le but jusqu'à détection d'une nouvelle collision pour laquelle il définira un nouveau sous-but...et ainsi jusqu'à arriver au but.

II.4.2. Méthode utilisant une fonction de pénalité :

Dans cette méthode, une fonction de pénalité est définie et est appliquée aux configurations du robot, elle informe sur la présence des obstacles, assure une évaluation, en tout point de l'environnement, de la proximité et de l'influence de ceux-ci, et empêche le robot de s'en approcher de trop près. Cette fonction est defini de sorte à prendre des valeurs croissantes plus on s'approche des obstacles allant généralement jusqu'à l'infini pour les configurations provoquant des collisions. La valeur totale de la pénalité est la somme des pénalités pour chaque obstacle et éventuellement celle infligée à la déviation du plus court chemin. La recherche du chemin se fait en calculant les dérivées partielles de cette fonction par rapport aux paramètres de la configuration (position, orientation), puis, une séquence de configurations à prendre est déterminée suivant les minima locaux de la fonction de pénalité qui sont le compromis entre la sûreté et la longueur du chemin.

Cette méthode paraît intéressante par le fait qu'elle apporte un moyen simple de gérer plusieurs contraintes à la fois (celles de plusieurs obstacles et de la longueur du chemin). Seulement, ne se basant que sur des informations locales, elle peut mener le robot dans une situation d'impasse. Lozano-Pérez remarqua dans [16], qu'il serait avantageux d'utiliser cette méthode comme solution locale dans un problème de recherche de chemin global, en la combinant avec la méthode *generate-and-test* par exemple, car il est plus opportun de l'utiliser là où le chemin n'a besoin que de petits changements.

❖ *Méthode des champs de potentiel :*

C'est une méthode de modélisation de l'environnement qui consiste à implanter un champs de potentiel répulsif U_{rep} autour de chaque obstacle dont le contour ou les parois en seraient, par une vue de l'esprit, les générateurs. Un champs attractif U_{att} est également généré par le point but. L'environnement plongé dans un tel champs est caractérisé par la présence d'une part, de barrières de potentiel à proximité des obstacles dont l'amplitude croît en inverse avec la distance à ceux-ci, et d'autre part d'un puits de potentiel, un minimum global situé au but. Le robot se déplaçant dans un tel environnement serait soumis à la somme vectorielle des forces dérivant des deux potentiels :

$$\vec{F}(p) = \vec{F}_{rep}(p) + \vec{F}_{att}(p) \quad \text{(II.4)}$$

où $\vec{F}_{rep}(p) = -\vec{grad}(U_{rep})$; et $\vec{F}_{att}(p) = -\vec{grad}(U_{att})$; (II.5)

On peut imaginer l'environnement comme un terrain comportant des montagnes à l'emplacement des obstacles se transformant graduellement en vallées puis en plaines plus on s'en éloigne, et présentant une cavité à l'endroit du but. Le robot n'a qu'à dévaler la pente à partir de sa source jusqu'à arriver à la cavité[1], si toutefois, il ne trouve pas de creux sur son chemins duquel il ne pourra sortir : c'est le problème de minima locaux, inhérent à cette méthode.

Le champs répulsif doit être défini de sorte à ce que son influence se limite à une certaine distance de l'obstacle pour ne pas qu'il crée des perturbations au loin, là où l'obstacle ne présente aucune gène au passage du mobile. Le potentiel attractif doit par contre être assez puissant et étendre son action sur tout l'espace, pour éviter au robot de se perdre au chemin, ce qui arriverait si la valeur de la force l'appelant au but venait à s'affaiblir et devenir négligeable.[19]

II.4.3. Méthode utilisant la modélisation de l'espace libre :

Cette méthode consiste à construire une représentation explicite de l'espace libre, et la recherche de chemin évitant les obstacles se fait sur cette partie de l'espace en essayant de relier la configuration du départ à la configuration de l'arrivée.

V. Nguyen [20] présente un algorithme heuristique de planification de trajectoire sans collisions, extension du travail de Brooks[9], utilisant la même représentation de l'environnement en essayant d'embrasser un plus grand espace libre de configurations. Il propose une description de quatre régions locales qui représentent les différentes liaisons entre les cônes communicants, en leur adaptant des experts qui déterminent le mouvement du robot dans ces régions :

- cônes qui se recouvrent : ce sont des cônes limités par le même côté d'un obstacle. Cette sorte de liaison suggère une translation le long du côté commun.
- Région en forme de V : c'est une région où deux cônes partagent le même sommet d'un obstacle : elle inspire une rotation du robot autour du sommet commun.
- Région libre convexe : c'est l'espace local borné par plusieurs côtés d'obstacles différents ; le mouvement du robot dans cette région consiste dans un alignement suivi d'une translation à l'intérieur de la dite région.

[1] Référence à un document incomplet, dont le titre est ignoré.

- Intersection en forme d'étoile : c'est une région formée par plusieurs sommets d'obstacles ; le chemin à l'intérieur d'une telle région passe entre les deux sommets les plus proches puis se translate à l'intérieur de l'intersection.

L'algorithme utilise les quatre experts pour déterminer le chemin entre les cônes reliés. Quant au chemin reliant deux configurations arbitraires, il est trouvé par l'algorithme A* [23].

Toujours pour illustrer la planification de trajectoire dans l'espace libre, on propose l'exemple donné par [21] dans lequel est présentée une structure particulière de la carte locale d'environnement, construite à partir d'informations issues de capteurs acoustiques, en vue de la résolution d'un problème de recherche de chemin sans collision.

Le robot étant rectangulaire, cette structure consiste à accroître les obstacles de la carte locale d'une distance perpendiculaire égale à la moitié de la largeur du véhicule, ce grossissement est en fait décrit par le centre de ce véhicule lorsqu'il tourne autour de l'obstacle en maintenant en contact permanent.

Quant au mobile, sa largeur est réduite à zéro (par le grossissement des obstacles) il devient un segment de droite passant par son centre le long de sa longueur.

Le problème de la navigation consiste à faire passer ce segment à travers un couloir d'obstacles grossis. Et ceci en le faisant pivoter d'abord jusqu'à le diriger suivant la droite le reliant à l'angle visible le plus proche du couloir, une fois que son centre ait coïncidé avec l'angle, il pivote par rapport à cet angle (et autour de son centre) pour se diriger le mieux qu'il peut vers la direction centrale du couloir, il part tout droit jusqu'à rencontrer un obstacle par le devant, puis il recommence les mêmes étapes jusqu'à ce que son centre coïncide avec le point but.

II.5. Exécution de trajectoire :

La commande d'un robot, qu'il soit manipulateur ou mobile peut être partagée en deux parties : la planification de trajectoire et la commande de mouvement.

La planification de trajectoire, d'après M. Brady [22], est la conversion d'une description d'un mouvement (ex : move P_1) en une trajectoire qui définie une séquence temporelle de configurations intermédiaires d'un robot entre la source P_0 et la destination P_1.

Qui dit configuration d'un corps indéformable, dit :

- coordonnées d'un point convenablement choisi sur ce corps (origine du repère lié à ce corps) par rapport à un référentiel extérieur à ce corps.
- orientation d'un axe lié à ce corps (un des axes de son repère),dans un référentiel extérieur.

Donc, pour définir une trajectoire, il est nécessaire de définir :

- La courbe décrite par l'effecteur (dans le cas d'un robot manipulateur), ou l'origine du repère lié au robot (pour généraliser aux robots mobiles). Cette courbe est appelée *« the path » –le chemin* – de la trajectoire.
- La courbe de rotation, qui est une séquence temporelle d'orientations de l'effecteur, ou du robot mobile.

Remarques:

- Pour un robot manipulateur, le chemin et la courbe d'orientation sont fonction d'un paramètre s, dans un espace tridimensionnel. En ce qui concerne un robot mobile ce déplaçant sur un plan, il est clair que le chemin devient à deux dimensions quant à l'orientation, elle se réduit à une seule.
- La trajectoire étant une fonction du temps, les vitesses et accélérations, de translation et angulaires, se déduisent de cette trajectoire par différentiation.

La planification de trajectoire dans le paragraphe suivant, consiste en une planification de bas niveau qui se fait par le navigateur,

Cette planification de trajectoire peut se faire selon deux approches[22],[23]:

- La première est vue comme une satisfaction de certaines contraintes, spécifiées explicitement par le niveau supérieur, comme les positions, orientations, vitesses et accélérations initiales et finales et un ensemble de configurations intermédiaires (appelées « nœuds »), auxquelles la continuité en vitesse et en accélération doivent être vérifiées. Le planificateur dans ce cas, choisit une fonction pour la trajectoire parmi une classe de fonctions, souvent polynomiale d'un degrés adéquat, qui remplie toutes les conditions, et fournie par interpolation aux nœuds, une trajectoire lisse et continue.

- Dans la deuxième approche, le planificateur définit explicitement le chemin par une fonction analytique : en segments de droites reliant les nœuds, par exemple. Le navigateur donne une trajectoire lisse et continue (en vitesse, accélération,...),par approximation du chemin désiré.

Dans la première approche, seules des configurations en certains nœuds sont imposées, le chemin est libre de prendre n'importe quelle forme, pourvu qu'il passe par ces nœuds et qu'il y satisfasse certaines contraintes. Alors que dans la seconde approche la forme même du chemin est imposée, et l'exécutant doit approximer au mieux ce chemin

par un autre qui serait plus réalisable. Il faut remarquer que cette dernière approche est plus fiable.

En effet, en liant les points intermédiaires par des courbes de la première approche, ces courbes risquent de déborder sur l'espace interdit, car aucune contrainte n'est spécifiée en dehors des points de passage

Dans ce qui suit, nous ne présenterons que la première approche, la seconde telle qu'elle est décrite dans le livre [22] cité plus haut, est beaucoup plus utilisée pour les robots manipulateurs. Le lecteur intéressé trouvera le résumé des premiers fondements de cette dernière approche dans [22].

II.5.1. Tracé de la trajectoire passant par des nœuds imposés :

M. Brady [22] l'explique de la façon suivante :

- Le mouvement le plus simple peut être défini par une configuration de départ P_0 et une configuration finale P_1. La trajectoire définie par une fonction de temps normalisée (i.e. pour t=0, P=P_0 et pour t=1, P=P_1), peut avoir la forme (proposée parmi une multitude de formes possibles) suivante :

$$P(t)=t\,P_1+(1-t)P_0 \qquad \text{(II.6)}$$

  Cette trajectoire est seulement contrainte par ses positions initiales et finales rapportées au temps.
  Dans cet exemple, elle a été prise comme une combinaison linéaire de P_0 et P_1, et représente une trajectoire à vitesse constante.
- Si en plus des contraintes précédentes, on veut que la trajectoire obéisse aux contraintes de vitesse aux nœuds initial et final, le polynome suivant est utilisé:

$$P(t) = (1-t)^2\left\{P_0 + t(2P_0 + \dot{P}_0)\right\} + t^2\left\{P_1 + (1-t)(2P_1 - \dot{P}_1)\right\} \qquad \text{(II.7)}$$

ou :

$\dot{P}_0$ et $\dot{P}_0$ sont respectivement les vitesses aux instants t=0 et t=1 respectivement.

Cette trajectoire est un polynôme d'ordre 3 en t.

- Par ailleurs si des contraintes d'accélération sont a satisfaire, le polynome suivant est utilisé :,

$$P(t) = (1-t^3)\left\{P_0 + (3P_0 + \dot{P}_0)t + (\ddot{P}_0 + 6\dot{P}_0 + 12P_0)\frac{t^2}{2}\right\} \qquad \text{(II.8)}$$
$$+\left\{P_1 + (3P_1 - \dot{P}_1)(1-t) + (\ddot{P}_1 - 6\dot{P}_1 + 12P_1)\frac{(1-t)^2}{2}\right\}$$

Cette fois-ci, la trajectoire est un polynôme d'ordre 5 en t.

Remarque :

– La trajectoire polynomiale d'ordre *n* garantie la continuité pour toutes ses dérivées car de classe n.

D'autre type de trajectoires peuvent être proposées, :
- la trajectoire en cosinus, qui comme la trajectoire polynomiale de (II.7), satisfait aux contraintes des positions et vitesses, initiales et finales.

$$P(t) = \cos^2\left(\frac{\pi}{2}(1-t)\right)\left\{P_1 - 2\frac{\dot{P}_1}{\pi}\cos\frac{\pi t}{2}\right\} + \cos^2\frac{\pi t}{2}\left\{P_0 + 2\frac{\dot{P}_0}{\pi}\cos\left(\frac{\pi}{2}(1-t)\right)\right\} \qquad \text{(II.9)}$$

- la trajectoire en sinus augmenté d'une rampe (trajectoire linéaire comme en (II.6)) ;
- la trajectoire en somme d'exponentielles amorties ;
- et la trajectoire bang-bang est utilisée, où est appliquée une force (moment) maximale alternativement accélératrice et décélératrice, il en résulte une trajectoire de la forme :

$$P(t) = \begin{cases} P_0 + \dfrac{\alpha}{2}t^2, & 0 \le t \le \dfrac{1}{2}; \\ P_1 - \dfrac{\alpha}{2}(1-t^2), & \dfrac{1}{2} \le t \le 1. \end{cases} \qquad \text{(II.10)}$$

La trajectoire ainsi définie ne satisfait qu'aux contraintes des positions initiales et finales.
Mujtaba [24] dans sa comparaison de ces différentes trajectoires, a trouvé que la trajectoire polynomiale d'ordre 5, celle en cosinus et celle en sinus additionné d'une trajectoire linéaire sont de 10 à 20% plus lentes que la trajectoire bang-bang définie en (II.10), et que celle en somme d'exponentielles amorties est 3 fois plus lente que les autres.

En réalité, la connaissance des configurations initiales et finales en position, vitesse et accélération ne suffit pas pour définir une trajectoire,

car il y a aussi des obstacles à éviter, d'où la nécessité d'introduire des points intermédiaires (*via-points*), qui représentent des contraintes supplémentaires au problème.

Il est possible de développer des trajectoires en polynômes d'ordre supérieur, comme pour (II.6-8), de façon à les faire passer par les points "d'interpolation" et réaliser la continuité en vitesse, accélération,...Néanmoins, le fait de traiter des polynômes d'ordres élevés présente certains inconvénients, parmi lesquels :

- La difficulté de tester la conformité de la trajectoire durant le mouvement, c'est à dire, vérifier la collision avec les obstacles, la transgression des limites physiques du robots, ou de celles de l'environnement. Cette difficulté croit avec l'ordre *n* du polynôme, puisque ce test conduit à la recherche de *n* racines (lors de la détection d'intersections avec des droites, par exemple) ou *n*-1 racines (lors de la recherches des extremums).
- Le polynôme considéré ayant *n* sommets, le chemin aura *n* "virages" et aura tendance à vaciller et s'écarter d'un chemin hypothétiquement droit.
- La précision avec laquelle est calculée le polynôme, est inversement proportionnelle à son degré.

De plus, l'introduction d'un point intermédiaire supplémentaire impose de recalculer tous les coefficients du polynôme.

Une solution à ce problème consiste à utiliser les *splines,* qui sont des polynômes de petit 'ordre, reliés entre eux aux points intermédiaires de façon à respecter la continuité de la trajectoire et celle de quelques unes de ces dérivées, en ces nœuds.

Pour chaque paire de nœuds successifs, un polynôme différent est calculé, répondant à des contraintes locales, relatives aux deux nœuds considérés.

Deux types de splines sont utilisés. Celles pour reliés les points intermédiaires et celles pour relier les points source et but.

Pour les premiers, les contraintes a satisfaire sont :
- Deux contraintes en positions aux extrémités de la portion à interpoler.
- Une, pour la continuité en vitesse.
- Une, pour la continuité en accélération.

Ces splines-là seront alors des polynômes cubiques (i.e. d'ordre 3), qui seront utilisés comme *primitives*.

Pour les secondes, il faut répondre à cinq contraintes :
- Trois à ces deux nœuds.
- Deux, aux nœuds voisins.

Dans ce cas les polynômes seront d'ordre 4.

Pour avoir des *splines* cubiques le long de toute la trajectoire, deux nœuds intermédiaires virtuels sont insérés dans les premier et dernier segments, ajoutant, évidemment deux portions polynomiales et introduisant trois contraintes chacun, qui sont la continuité en :

- position ;
- vitesse ;
- accélération.

Ainsi, les cinq contraintes aux extrémités sont équilibrés par l'apport des nœuds virtuels. Et les splines sont cubiques sur toute la trajectoire.

Dans le cas où le nombre de contraintes est égal au nombre de paramètres à déterminer, la solution est unique.

Dans le cas plus général, n'ayant pas de spécifications pour toutes les contraintes, le problème de déterminer les paramètres de la trajectoire, présente plusieurs solutions. Une approche pour déterminer les bons paramètres de la trajectoire $f(t)$, consiste à minimiser l'intégrale :

$$I(f) = \int \Phi(f, \frac{df}{dt}, \frac{d^2 f}{dt^2}, ...)dt \qquad \text{(II.11)}$$

où Φ est une fonction nommée index de performance qui est une mesure de l'énergie consommée durant la trajectoire. Φ peut aussi être égal à 1, dans ce cas, c'est le temps de la traversée qui sera minimisé.

II.5.2. Lissage de trajectoire en robotique mobile :

En robotique mobile, la planification de trajectoire consiste à décrire une séquence de points de passage ou de configurations intermédiaires permettant au robot d'aller de la source au but. Ces points de passage sont généralement reliés par des segments de droites.

L'exécution de trajectoire ne peut se faire à partir d'un chemin constitué de segments de droites pour les raisons suivantes :

- Cette forme de trajectoire présente des discontinuités qui ne sont pas souhaitables du fait qu'elles engendrent un glissement des roues du robot. Ceci provoque des erreurs dans la localisation par odomètrie. En effet, les odomètres sont présents sur l'ensemble des robots à roues car cette technique est peu coûteuse et consiste à fixer sur les roues des codeurs qui délivrent une impulsion toutes les fractions de tour de roue. L'intégration de ces valeurs permet de déduire la position

et l'orientation du mobile à chaque instant, par rapport à l'endroit d'initialisation du compteur d'impulsions qui se retrouve en erreurs de localisation.

- Le chemin formé par les segments présente des variations brusques de direction qui obligeraient un mobile à roue de s'arrêter afin de pouvoir réaliser un tel changement de direction. Ce fait augmente considérablement la durée de traversée et la consommation d'énergie que les changements de vitesse induiraient.

Pour éviter l'arrêt du véhicule, un raccordement de trajectoire est nécessaire : Ceci est réalise par des arcs de cercle, des courbes à courbure continue dont les clothoïdes.

II.5.2.1. Les arcs de cercles [1] :

Les segments de droites sont raccordés au niveau de leurs intersections par des arcs de cercles de rayon de giration minimal. Ce rayon ne doit pas être trop grand afin de ne pas rendre le chemin non faisable.

En effet, le chemin délivré par le planificateur, sous forme de segments, est certainement faisable. Or le fait de remplacer les angles par des arcs déplace en quelque sorte les points de la trajectoire situés sur l'arc vers l'intérieur de l'angle, de sorte que plus le rayon est grand, plus loin sont déplacés les points de la portion de trajectoire (qui, elle aussi est plus longue).

D'autre part, un rayon de giration trop petit, risquerait de faire déraper le véhicule sous l'action de la force centrifuge qui est inversement proportionnelle au rayon de courbure de la trajectoire et proportionnelle au carré de la vitesse tangentielle : Il faut donc penser aussi à diminuer celle-ci dans les virages.

Le raccordement par arcs de cercles à l'inconvénient principal de présenter des discontinuités de courbure de trajectoire au niveau des jonctions entre les segments et les arcs, qui se manifestent par des brusques variations de vitesse provoquant un glissement des roues.

II.5.2.2. Les courbes à courbure continue:

Pour avoir une trajectoire de courbure continue, il suffirait de choisir une portion de courbe dont la courbure vérifierait les conditions de continuité d'accélération et de vitesse, ainsi qu'aux exigences de sécurité.

Pour ce faire, commençons par définir ce qu'est l'abscisse curviligne et la courbure.

Définition :

Soit un plan rapporté à un repère cartésien $\Re(o, \vec{i}, \vec{j})$

Toute courbe ($\wp$), peut être définie, dans ce plan, en chacun de ses points $p \in (\wp)$ par ses coordonnées x et y, vérifiant la relation : $y=\gamma(x)$.

Tout point p de la courbe est défini par son abscisse curviligne s par rapport à l'origine p_0 (x_0, y_0) comme étant la longueur de l'arc de courbe limité par p et p_0.

On définit l'angle θ comme étant la pente de la tangente en un point $p(x,y)$ de la courbe :

$$\theta = arctg\left(\frac{dy}{dx}\right); \qquad \text{(II.12)}$$

avec la pente aux point origine $\theta_0 = arctg\left(\frac{dy}{dx}\right)\Big|_{x_0,y_0}$

On définit la courbure comme étant la variation de la pente de la tangente par rapport à l'abscisse curviligne :

$$k = \frac{d\theta}{ds}; \qquad \text{(II.13)}$$

On aura alors :

$$\theta(s) = \theta_0 + \int_0^s k(v)dv$$

$$x(s) = x_0 + \int_0^s \cos\theta(v)dv; \qquad : \qquad \text{(II.13)}$$

$$y(s) = y_0 + \int_0^s \sin\theta(v)dv;$$

La portion de courbe aura deux points de jonction sur la trajectoire, avec les deux segment de droite, de part et d'autre de l'angle à éliminer.

Fixons l'origine p_0 de la courbe de raccordement au premier point de jonction, on aura donc à p_0 $s=0$. Soit p_1 le second point de jonction, où $s=s_1$.

La courbure est définie par une fonction *k(s)*:

- continue sur toute la longueur de l'arc p_0p_1 ;

- particulièrement continue aux limites, i.e. en points de jonction p_0 et p_1, ce qui implique que : $k(0)=0$ et $k(s_1)=0$ car la courbure d'une droite est nulle.
- en vertu du point précédent et d'après le théorème de Rolle, la fonction $k(s)$ doit avoir au moins un extremum entre p_0 et p_1 puisqu'elle ne peut être nulle sur toute la longueur de l'arc .

- Les coordonnées de la courbe en fonction de l'abscisse curviligne sont données par l'expression suivante à partir des équations précédentes:

$$x(s) = x_0 + \int_0^s \cos(\theta_0 + \int_0^u k(v)dv)du;$$
$$y(s) = y_0 + \int_0^s \sin(\theta_0 + \int_0^u k(v)dv)du; \qquad \text{(II.14)}$$

Une multitude de fonction peuvent répondre :

- La fonction triangulaire : $k(s)= k_{max}.tri(s/s_1 - 0.5)$;
- La fonction polynomiale, par exemple d'ordre 2 : $k(s)=s(s- s_1)$;
- La fonction sinusoïdale : $k(s)= k_{max}.sin(\pi s/s_1)$;
- en cosinus : $k(s)= k_{max}.(1-cos(\pi s/s_1))$... etc

Remarque : pour pouvoir définir entièrement les portions de trajectoire raccordant et lissant les segments, la courbure doit être une fonction du temps.

Cas particulier : Les clothoïdes :[18]

Les clothoïdes ou les spirales de Cornu sont des courbes très utiles pour le lissage de trajectoires. Ils sont caractérisés par une courbure linéaire par rapport à l'abscisse curviligne :

$$k = k_c s + k_0 \qquad \text{(II.15)}$$

où k_c est une constante caractérisant la forme du clothoïde

et k_0 est sa courbure initiale.

Ces courbes permettent de relier des courbes à courbure nulle (rayon de courbure infini : des droites) à des courbes de rayon de courbure non nul.

On obtient une telle trajectoire en gardant la vitesse tangentielle de déplacement le long de la trajectoire constante :

$$v(t) = v_0 \tag{II.16}$$

et la vitesse angulaire linéaire dans le temps :

$$\omega = at + \omega_0 \tag{II.17}$$

d'où la longueur de l'arc s'écrit :

$$s(t) = v_0 t \tag{II.18}$$

rappelons la relation :

$$\omega(t) = kv(t) = \frac{v(t)}{\rho} \tag{II.19}$$

où $\rho = \frac{1}{k}$ est le rayon de courbure.

Donc

$$k = \frac{\omega(t)}{v(t)} = \frac{a}{v_0}t + \frac{\omega_0}{v_0} = \frac{a}{{v_0}^2}s(t) + \frac{\omega_0}{v_0} \tag{II.20}$$

en appliquant les relations du paragraphe précèdent, et en supposant que $\theta_0 = \omega_0 = 0$, on obtient:

$$x(s) = x_0 + \int_0^s \cos(\int_0^\alpha \frac{a}{{v_0}^2} u du) d\alpha = x_0 + \int_0^s \cos(\frac{a}{2{v_0}^2}\alpha^2) d\alpha \tag{II.21}$$

$$y(s) = y_0 + \int_0^s \sin(\int_0^\alpha \frac{a}{{v_0}^2} u du) d\alpha = y_0 + \int_0^s \sin(\frac{a}{2{v_0}^2}\alpha^2) d\alpha \tag{II.22}$$

par le changement de variables : $u = \sqrt{\frac{a}{\pi {v_0}^2}}\alpha$

On écrit :

$$x(t) = x_0 + \sqrt{\frac{\pi {v_0}^2}{a}} \int_0^{\sqrt{\frac{a}{\pi}t}} \cos(\frac{\pi}{2}u^2) du \tag{II.23}$$

$$y(t) = y_0 + \sqrt{\frac{\pi {v_0}^2}{a}} \int_0^{\sqrt{\frac{a}{\pi}t}} \sin(\frac{\pi}{2}u^2) du \tag{II.24}$$

l'équation du mouvement sur une clothoïde est donnée par les intégrales de Fresnel suivantes:

$$x(t) = x_0 + \sqrt{\frac{\pi v_0^{\,2}}{a}} CF(\sqrt{\frac{a}{\pi}} t) \qquad \text{(II.25)}$$

$$y(t) = y_0 + \sqrt{\frac{\pi v_0^{\,2}}{a}} SF(\sqrt{\frac{a}{\pi}} t) \qquad \text{(II.26)}$$

II.6. Conclusion :

Dans ce chapitre, nous avons passé en revue le type d'architecture de commande d'un robot mobile. Celle-ci nous a permis d'identifier et de situer le rôle de la planification. Cette dernière s'effectue suivant différentes approches que nous avons brièvement présenté pour les milieux connus et inconnus.

Nous avons exposé les différents modèles de l'environnement utilisés, ainsi que quelques exemples typiques de méthodes de planification les utilisant.

Dans ce chapitre, nous concluons notre exposé par une revue de la génération de trajectoire par les méthodes les plus établies. Celles-ci feront l'objet d'utilisation dans le travail qui suit.

CHAPITRE III

ALGORITHMES GÉNÉTIQUES: GÉNÉRALITÉS

III.1. Source d'inspiration, l'idée d'évolution dans la nature : [26]

« Plusieurs siècles avant Jésus-Christ, des philosophes grecs avaient déjà perçu que le monde vivant était soumis à des transformations.

Lamarck, en 1801, a le premier lancé l'idée d'une évolution et a collecté durant sa vie des arguments à l'appui de sa thèse. Celle-ci fut confortée par les travaux de Darwin, en 1859 sur la sélection naturelle des espèces est lancée.

En effet, l'évolution vers des formes de plus en plus élaborées et complexes fut grande, ainsi que les disparitions. Néanmoins il existe des formes qui depuis des centaines de millions d'années n'ont pas changé, car ayant atteint le summum de leur évolution. Tout cela montre la complexité de l'évolution, car il faut expliquer les progressions aussi bien que les régressions ou les disparitions d'espèces.

Lamarck avait montré la « constance relative » des espèces qui « ne sont invariables que temporairement ».En effet, les conditions du milieu dans lequel ces espèces vivent, venaient à changer, celles ci changeraient. Cependant l'action du milieu, dans certaines thèses, a été exagérée et l'automaticité de l'hérédité n'est pas défendable. Le changement n'est pas définitif dans l'histoire de l'espèce car l'hérédité des caractères acquis n'est pas automatique. Ceci fut prouvé par les paléontologistes qui apportèrent les preuves qui avait manqué à Lamarck sur les variations morphologiques en fonction de modification du milieu.

Quant à Darwin, il a cru pouvoir expliquer par la sélection naturelle, en s'exprimant en termes suivants : « Etant donné que plus d'individus sont produits qu'il ne peut en survivre, il doit exister une lutte pour l'existence .(...) Les individus possédant un avantage quelconque, sur les autres, auraient une meilleure chance de survivre et de procréer leur propre type. Inversement toute variation délétère serait impitoyablement éliminée ». L'aspect scientifique de la démarche de Darwin apparaît fragile en dépit d'une somme imposante de données d'observation, car elles n'ont pas de grandes portée en ce qui concerne l'évolution elle-même.

L'idée maîtresse du Néo-Darwinisme est l'intégration des acquisitions de la génétique dans le système, **la sélection naturelle n'intervenant non plus en favorisant la survie du mieux doué, mais en termes de probabilités, par un processus statistique qui confère au plus doué plus de chances de transmettre ses caractères. Ainsi**

donc la sélection naturelle est l'agent d'une transmission préférentielle des caractères inscrits dans les gènes... ».
C'est de là que s'inspirent et puisent leur idée les algorithmes génétiques.

En effet, le Néo-Darwinisme proclame que l'histoire de la plus grande partie de la vie est régie seulement par quelques processus statistiques agissant sur les populations et les espèces. Ces processus sont reproduction, mutation, compétition, et sélection.

- La reproduction étant une propriété évidente de toute vie,
- la mutation (i.e. le changement génétique) devenant aussi évidente pour tout système se reproduisant dans un univers d'entropie positive, (i.e. évoluant d'un certain état de désordre vers un état de désordre accru),
- la sélection et la compétition deviennent inévitables lors de l'expansion de la population dans une arène limitée.
- l'évolution est donc la conséquence de l'interaction de ses processus stochastiques agissant sur les populations génération après génération.

Cette conception s'étend toutefois en dehors de l'étude de la vie. L'évolution est un processus d'optimisation qui peut être simulé est utilisé à de bonnes fins d'engineering. Pour l'appliquer à la résolution d'un problème pratique, on commence par définir une population de solutions quelconques choisi aléatoirement. Puis de nouvelles solutions sont créées en altérant les solutions existantes. Une mesure objective de performances est utilisée pour évaluer l'adéquation (*fitness*) ou l'écart de chaque solution. Par la suite, un mécanisme de sélection détermine les solutions qui devraient être retenues comme parents pour les générations futures. Les différences entre les procédures résident dans :

- les types d'altérations imposées aux solutions pour créer les descendants,
- les méthodes employées pour sélectionner les nouveaux parents
- et la structure de données représentant les solutions.

III.2. L'optimisation et l'algorithme évolutif :

La recherche de solution dans un espace complexe implique souvent un compromis entre deux objectifs : l'exploitation des meilleures solutions disponibles à un moment donné et une exploration robuste de l'espace des solutions possibles. Les Algorithmes Génétiques sont une classe de stratégies de recherche réalisant un compromis équilibré et raisonnable entre l'exploitation et l'exploration. En effet, des analyses théoriques ont montré que les Algorithmes Génétiques géraient ce

compromis d'une façon presque optimale [27]. Leur fonctionnement repose sur une heuristique très simple : les meilleures solutions seront trouvées dans des régions de l'espace de recherche contenant des proportions relativement élevées de bonnes solutions. De plus, ces régions peuvent être identifiées par un échantillonnage robuste et judicieux de l'espace des solutions.

Le problème de l'optimisation en général [28], consiste à trouver un ensemble de vecteurs $\vec{x} \in M$ de paramètres indépendants du système, qui repondent à un critère de qualité : $f : M \to \Re$, (appelé souvent fonction objective) qu'on doit maximiser (ou minimiser).

$$f(\vec{x}) \to \max \qquad \text{(III.1)}$$

Pour résoudre le problème d'optimisation globale (III.1), il faut trouver le vecteur $\vec{x}^*$ tel que $\forall \vec{x} \in M : f(\vec{x}) \le f(\vec{x}^*) = f^*$.
Différentes caractéristiques peuvent rendre le problème d'optimisation difficile pour ne pas dire insoluble dans certains cas.

Une des plus importante est la multi-modalité qui traduit l'existence de plusieurs maximas locaux $\vec{x}'$ tels que :

$$\exists \varepsilon > 0 : \forall \vec{x} \in M : \rho(\vec{x}, \vec{x}') < \varepsilon \Rightarrow f(\vec{x}) \le f(\vec{x}') \qquad \text{(III.2)}$$

Où ρ désigne la distance dans M.

Optimisation sous contraintes :

Lorsque des restrictions sur l'ensemble M existent les solutions valables ne représentent qu'un sous ensemble de l'ensemble des solutions.

$$F = \left\{ \vec{x} \in M \middle| g_j(\vec{x}) \ge 0 \forall j \right\} \qquad \text{(III.3)}$$

ou $g_j : M \to \Re$ est une fonction représentant les contraintes

On peut aussi citer d'autres facteurs comme la grande dimension, ou les fortes non-linéarités, la non-différentiabilité, une fonction objective bruitée ou variant dans le temps. Ces facteurs peuvent rendre le problème extrêmement complexe. Néanmoins, le simple fait de détecter une quelconque amélioration par rapport à une solution réputée la meilleure à un instant donné, peut s'avérer un pas considérable et un grand succès dans des problèmes pratiques. Dans de nombreux cas, les algorithmes évolutifs apportent une méthode efficace pour réaliser ceci.

Dans la réalité, les formes analytiques des fonctions objectives ainsi que celles des contraintes ne sont pas toujours bien définies. Pour traiter ce genre de problème est de développer un modèle approchant le comportement des fonctions d'origine. L'ennui est que l'application des méthodes mathématiques exige des simplifications considérables par rapport au problème réel. Il est vrai que cette approche a fait preuve de grande efficacité dans beaucoup d'applications, mais elle présente aussi des lacunes qui motivent la recherche, la dirigeant vers d'autres directions. Et l'une des voies les plus promettantes est le calcul évolutif.

La particularité de l'approche évolutive apparaît dans le fait qu'il faut l'adapter au cas à traiter car il n'y a pas de recette toute prête à suivre, ou de formule à appliquer, seulement un concept général, un principe qui doit être taillé et affiné suivant l'application courante. Une fois la structure de l'algorithme évolutif construite, il peut s'adapter par incréments au problème considéré, aux changements des exigences du projet, aux modifications du modèle, ainsi qu'au changement des ressources.

III.3. La structure d'un algorithme évolutif :

Les algorithmes évolutifs imitent, comme leur nom indique, le processus de l'évolution naturelle. Quant à l'évolution, on peut dire que c'est le résultat de l'interaction entre la création de nouvelles informations génétiques, leur évaluation et leur sélection. L'individu étant affecté par les autre individus de la population, et par l'environnement, ces chance de survie et de procréation sont d'autant plus grande s'il progresse.

Ses descendants hériteront à leur tour une distribution d'informations génétiques de leurs parents. Cela fait acquérir à toute la population, à travers les générations, les caractéristiques des « meilleurs » éléments. Le caractère non déterministe de la reproduction et de la mutation conduit à un perpétuel changement des informations génétiques (c'est un processus dynamique).

D'après, le modèle néo-Darwinien, on peut décrire le processus d'évolution par l'algorithme suivant :

t:=0 ;
initialiser P(t) ;
évaluer P(t) ;

évaluer P(t) ;
tant que *les conditions d'arrêt ne sont pas vérifiées,*
P'(t) :=variation[P(t)] ;
évaluer P'(t) ;
P(t+1) :=sélection[P'(t)∪Q] ;
t:=t+1 ;
fin

ou :

P(t) est une population contenant μ individus à l'instant *t*.

Q est l'ensemble d'individus de la population *P(t),* qui sont susceptibles d'être sélectionnés (il est possible que *Q=P(t),* ainsi que *Q=∅),* une population de descendants *P'(t)* de la taille λ est produite par des opérateurs de mutation et/ou de recombinaison à partir de la population d'origine *P(t).*

Les individu-fils sont évalués en calculant la valeur de la fonction objective $f(\vec{x}_k)$ pour chaque solution $\vec{x}_k$ puis une sélection basée sur la valeur d'adéquation (*fitness value*) se fait pour conduire le processus vers de meilleures solutions. Il est à noter le cas où l'opérateur de variation ne s'applique que sur un seul individu (λ=1), choisi parmi tous les individus de la population *P(t)* (*Q=P(t)),*on parlera alors d'un schéma de sélection de régime stationnaire. Il est aussi possible de trouver $\lambda > \mu$, c'est le surplus de reproduction, un cas normal dans la nature. Dans les applications génétiques, le plus souvent on prend $1 \leq \lambda \leq \mu$.

III.4. La conception de l'algorithme évolutif :[28]

Dans les applications réelles, on définie les différents paramètres du système étudié qui seront sujets à l'optimisation. Ces paramètres constituent *l'espace phénotype*. Les opérateurs de la transformation, d'autre part, agissent sur des objets mathématiques abstraits qui constituent *l'espace génotype*. Il est évidemment nécessaire de prévoir une fonction de codage entre *l'espace phénotype* et *l'espace génotype.*

Pour ce faire, on peut suivre, en général deux approches :

- La première consiste à choisir un algorithme standard, puis de proposer une fonction de codage en accord avec les exigences de l'algorithme.
- La seconde suggère une représentation aussi proche que possible des caractéristiques de l'espace phénotype se passant presque, d'une fonction de codage. Beaucoup de résultats théoriques et empiriques ont été développés en ce qui concerne les algorithmes standards ce qui avantage la première

approche. D'autre part, des fonctions de codage trop complexes peuvent introduire des non-linéarités supplémentaires ainsi que d'autres difficultés mathématiques, ce qui est une entrave à la recherche des solutions

Un algorithme génétique est un algorithme itératif de recherche globale dont l'objectif est d'optimiser une fonction d'adéquation (*fitness function*) définie par l'utilisateur.

Pour atteindre cet objectif, l'algorithme travaille en parallèle sur une population de points candidats, appelés individus ou chromosomes, distribués dans l'intégralité de l'espace de recherche. Chaque individu ou chromosome est constitué d'un ensemble d'éléments appelés caractéristiques ou gènes, pouvant prendre plusieurs valeurs dans un alphabet non nécessairement numérique. Le but est donc de rechercher une combinaison optimale de ces éléments, qui donne lieu au maximum d'adéquation.

A chaque itération, appelée génération, est créée une nouvelle population avec le même nombre d'individus. cette nouvelle génération consiste généralement en des individus mieux « adaptés » à l'environnement tel qu'il est représenté par la fonction d'adéquation. Au fur et à mesure des générations, les individus vont en général tendre vers l'optimum de la fonction d'adéquation. la génération d'une nouvelle population à partir de la précédente s'effectue en trois étapes :

a. *évaluation* :l'algorithme évalue la fonction d'adéquation de chaque individu de l'ancienne population.

b. *Sélection* : l'algorithme sélectionne les individus sur la base de leur fonction d'adéquation. les individus sélectionnés constituent une population intermédiaire.

c. *Reproduction avec croisement et mutation* :l'algorithme recombine les individus sélectionnés au moyen d'opérateurs génétiques tels que la mutation et le croisement, qui peuvent être considérés comme des mécanismes servant à changer localement les solutions représentées par les parents (mutation) ou à les recombiner (croisement). La mutation agit en modifiant aléatoirement un ou plusieurs gènes d'un chromosome. Tandis que le croisement échange certains gènes d'un parent avec ceux de l'autre. Ces opérateurs sont appliqués avec certaines probabilités, sur chacun des individus sélectionnés, dont on choisira au hasard deux parents et un site de croisement. L'opérateur de croisement effectue la recherche globale de

l'espace par combinaison de « blocs de construction », quant à l'opérateur de mutation, il est là pour veiller à garder une certaine dispersion des « espèces » et éviter une perte irréparable de la diversité, qui conduirait à une limitation de l'espace de recherche à une région donnée.

III.5. Conclusion :

Dans ce chapitre nous avons présenté les fondements des algorithmes génétiques, ainsi que leur structure.

Une démarche type de conception est développée, les principaux caractères des algorithmes génétiques cernés. Cette démarche servira de base aux algorithmes développés par la suite de ce mémoire.

CHAPITRE IV

LA NAVIGATION PAR ALGORITHME ÉVOLUTIF

IV.1.Présentation de la navigation évolutive:

L'ensemble des techniques de navigation présentées jusqu'à présent sont destinées à travailler soit dans un environnement parfaitement connu, faisant une planification globale, ou alors dans un environnement inconnu ou partiellement connu effectuant une recherche locale de trajectoire. La méthode évolutive de navigation présentée ci-dessous, rassemble les deux techniques en un seul algorithme.

L'environnement ici est considéré comme étant à deux dimensions contenant des obstacles polygonaux qui sont grossis par la dimension du robot (que nous prenons circulaire : un robot de la famille Hilare, par exemple).

Le robot quant à lui, sera réduit à un seul point. Son mouvement consistant en une séquence de translation.

La représentation de l'espace libre consiste en une liste ordonnée de sommets d'obstacles connu et grossis, voici une modélisation bien simple qui a l'avantage de prendre un espace mémoire minimum (à la différence des autres modélisations).

IV.1.1.Description de l'algorithme du navigateur évolutif de référence:

Un chemin conduisant le robot du point source au point but, qu'il soit ***faisable*** ou non (voir la définition de la faisabilité d'un chemin plus loin), est représenté par un chromosome dans une population P(t) de génération t.

En un premier temps, chaque chromosome de la population est ***évalué.*** Puis une boucle d'évolution est exécuté comme suit :

a. Un ***opérateur*** est sélectionné selon une certaine ***distribution de probabilité,*** les opérateurs sont de deux types :

   - des transformations unaires c'est a dire de premier ordre, intervenant sur un seul élément, à un seul ensemble de départ, et qui engendrent un descendant en effectuant un petit changement sur un seul individu de la population.

   - des transformations d'ordre supérieur qui créent un descendant en combinant des parties de différents chromosomes.

b. L'individu le moins performant de la population est ensuite remplacé par le nouvel individu résultat de la transformation. De ce fait, les populations P(t),de la génération t et P(t+1), de la génération t+1, ne diffèrent que d'un seul individu .

L'algorithme proposé dans [29] est le suivant :

Algorithme (1).
Début
t :=0 ;
Si *known_path* ***alors,***
Introduire P(t) ;
Sinon
Initialiser P(t) ;
fin
Evaluer P(t) ;
Tant que *les conditions de fin de parcours ne sont pas réalisées,* ***faire***
t:=t+1 ;
Choisir un opérateur op_j avec une probabilité p_j ;
Choisir un(des) parent(s)
Produire un descendant en appliquant l'opérateur p_j au(x) parent(s) choisi(s) ;
Evaluer le nouveau descendant ;
Remplacer le « pire » membre de P(t) par le descendant obtenu ;
Choisir le « meilleur » membre p de la population P(t) ;
Si *online* ***&*** *p faisable* ***&*** *t[n]≡0* ***alors,***
Avancer d'un pas k_{max} du chemin déterminé p tout en reconnaissant l'environnement ;
Modifier les valeurs de tous les chromosomes en leur affectant *une nouvelle valeur de départ ;*
Si *quelques changements sont détectés dans l'environnement* ***alors***
Corriger (actualiser) la carte de l'environnement ;
fin
Evaluer P(t) ;
fin
fin
fin

Le nombre de boucles d'évolution est fixé par l'utilisateur ou par le programme lui même.

A la fin, le meilleur chromosome représente le chemin sous-optimal.

Deux variables booléennes apparaissent dans l'algorithme : *known_path* et *online.*

- *known_path* vrai, montre qu'il y a des connaissances a priori sur le chemin recherché et les fait exploiter par l'algorithme. Lorsqu'il prend la valeur faux, il contraint le programme à ***initialiser*** la population P(0).

- *online* est faux si toute la carte de l'environnement nécessaire à la tâche est disponible. Le chemin sous - optimal est dans ce cas calculé hors ligne (*offline).* Dans le cas contraire, le processus est effectué en temps réel en se basant sur les informations disponibles sur l'environnement ainsi que sur celles acquises au fur et à mesure que le mobile avance (informations captées).

Il se produit deux phénomènes parallèles dans la navigation en temps réel :

- Le premier étant le déroulement continuel du processus d'évolution recherchant une solution toujours meilleure.

- Le second consiste dans la progression du mobile dans son environnement suivant le meilleur chemin produit par le processus d'évolution chaque n périodes et après évaluation de sa faisabilité.

IV.1.2. Les chromosomes, les gènes :

En appliquant l'algorithme génétique à la navigation, on représente l'ensemble des chemins par une population de chromosomes.

Le chemin étant une séquence de nœuds reliés par des segments de droites, les gènes de chaque chromosome ne sont autres que les coordonnées de ces nœuds .

Toute séquence commence par un nœud–source (point de départ du mobile), et se termine par un nœud–but (point d'arrivée du mobile). Ces deux nœuds étant fixes, connus, et les mêmes pour tous les chromosomes. La séquence comporte aussi un nombre variable de nœuds intermédiaires (points de passage du mobile).

IV.1.3. L'initialisation de la population P(t) :

Dans [29], l'initialisation des chromosomes de la population P(0) est décrite comme une génération aléatoire d'un ensemble de chemins ayant tous le même nœud source et le même nœud but et comportant chacun un nombre aléatoire de nœuds aléatoirement générés.

IV.1.4. Définition de la faisabilité d'un chemin :

Définition 1 :

On dit qu'un chemin est faisable s'il ne provoque aucune collision du mobile avec les obstacles.

Définition 2 :

Un chemin ou un chromosome est faisable si et seulement tous ses segments sont faisables.

Définition 3 :*Faisabilité d'un segment*

Un segment est d'abord défini par deux nœuds consécutifs d'un chromosome et qui sont ses extrémités. Un segment est dit faisable si :

- son premier nœud (par lequel il est connecté au segment précédent, le plus proche de la source) est à l'extérieur de tous les obstacles.

- le segment lui-même n'intersecte aucun obstacle.

IV.1.5. Evaluation d'un chemin :

L'évaluation d'un chemin consiste à apprécier sa qualité par rapport à des critères établis. En effet, la fonction d'évaluation mesure le coût du chromosome considéré et est utilisé en vue de réaliser des objectifs d'optimisation.

Hormis quelques différences, la fonction d'évaluation joue le rôle de la fonction d'adéquation (*fitness fuction*) évoquée dans le chapitre précédent. La différence réside dans le fait que la fonction d'adéquation ne peut prendre que des valeurs positives alors que la fonction d'évaluation peut très bien avoir des valeurs négatives.

Aussi, plus grande est la valeur de la fonction d'adéquation, « meilleur » est le chromosome. Par contre dans le cas présent, pour trouver le meilleur chemin, on cherchera à minimiser la fonction d'évaluation.

Deux fonctions d'évaluation respectivement pour les chemins faisables et les chemins non faisables [29] son définies :

1. *Evaluation des chemins faisables :*

Pour les chemins faisables on se pose pour objectif, de :

- Minimiser la distance parcourue, pour cela on définit un terme (à minimiser) mesurant cette distance

$$dist = \sum_{j=1}^{l-1} d(n_j, n_{j+1}) \qquad \text{(IV.1)}$$

n_j étant le j^{ème} nœuds du chromosome évalué ;
l, étant sa longueur (son nombre total de nœuds), l=Nn+2, Nn étant le nombre de nœuds intermédiaires ;
d, étant la longueur du segment j, distance entre les deux nœuds voisins j et j+1.

- Maintenir le chemin assez loin des obstacles, pour ce faire, on définit un terme qui quantifie la proximité de l'obstacle au chemin évalué :

$$clear = \max_{j=1}^{l-1} c(n_j) \qquad \text{(IV.2)}$$

où $c(n_j)$ est donné par :

$$c(n_j) = \begin{cases} g_j - \tau, & si \quad g_j \geq \tau \\ e^{a(\tau - g_j)} - 1, & ailleurs \end{cases} \qquad \text{(IV.3)}$$

g_j étant la plus petite distance du segment $\overline{n_j n_{j+1}}$ à tous les obstacles détectés.
τ est un paramètre définissant la distance de sécurité, a est un coefficient.

Si un chemin s'approche trop prés d'un obstacle, transgressant la distance de sécurité, cette violation lui coûtera une pénalité croissant exponentiellement plus il s'approche de l'obstacle. La sévérité de cette pénalité est quantifiée par le coefficient a.

- Avoir un chemin aussi lisse que possible, et ceci en minimisant un terme :

$$smooth = \max_{j=2}^{l-1} s(n_j) \qquad \text{(IV.4)}$$

n_j étant toujours le $j^{ème}$ nœuds du chromosome évalué ;
l, étant toujours sa longueur (son nombre total de nœuds) ;
s, défini comme suit :

$$s(n_j) = \frac{\theta_j}{\min\{d(n_{j-1}, n_j), d(n_j, n_{j+1})\}} \qquad \text{(IV.5)}$$

θ_j, étant l'angle formé au nœud j par les prolongements des deux segments j-1 et j, ayant respectivement pour extrémités les pairs de nœuds (n_{j-1}, n_j) et (n_j, n_{j+1}), $\theta_j \in [0, \pi]$.

Les trois aspects *dist*, *smooth* et *clear* sont rassemblés en un critère pondéré, leur combinaison linéaire, qui est *à minimiser :*

$$eval_{feas} = w_d dist + w_s smooth + w_c clear \qquad \text{(IV.6)}$$

w_d, w_s *et* w_c étant des poids positifs.

2. Evaluation de chemins non faisables :

Les chemins non faisables sont évalués en se basant sur la mesure des aspects suivants (pour plus de détails se référer à [30]) :

- Le nombre d'intersections du chemin avec les obstacles.
- Le rapport du nombre des segments faisables au nombre des segments non faisables.
- La profondeur de l'intersection.
- La distance totale du chemin.

IV.1.6. Description des opérateurs :

Il a été proposé dans [29], Huit (08) types d'opérateurs susceptibles de générer des chemins de formes arbitraires, afin de faire évoluer la population, éventuellement vers de meilleures formes.

Le choix d'un opérateur à appliquer est aléatoire. Généralement, dans les algorithmes génétiques, la sélection d'un opérateur est ou laissée au pur hasard, ou régie par une distribution de probabilités préalablement fixée pour toute la durée de l'évolution.

Une idée nouvelle et très intéressante a été proposée par [29] qui se résume dans l'adaptation des probabilités à la situation courante, de

façon à rendre l'évolution plus efficace et plus rapide. Car pour tout opérateur, il existe des situations propices pour agir. Il est ainsi possible qu'il améliore un chemin ou qu'il le détériore. Se basant sur ce fait on peut renforcer l'action des opérateurs qui ont fait preuve de performance durant une certaine « époque » et inhiber les autres.

De plus, il est à noter que la sélection n'est pas guidée seulement par la probabilité qui lui est attribuée, mais est aussi contrainte par la nature du chromosome auquel l'opérateur sera appliqué. Car certains opérateurs ne peuvent être appliqués à n'importe quel chemins. Certains s'appliquent qu'aux chemins faisables et d'autres aux chemins non faisables uniquement.

Le processus de sélection sera décrit plus en détails dans le paragraphe suivant. Pour l'instant, décrivons les différents opérateurs utilisés dans le présent travail.

1. *Crossover, opérateur de croisement :*

Cet opérateur choisit au hasard deux chromosomes dans la population, puis dans chaque chromosome il choisit aussi au hasard un nœud intermédiaire (différent du but et de la source) qui divisera le chromosome en deux parties. On combinera la première partie du premier chromosome avec la seconde du second chromosome et la seconde du premier avec la première du second, bref, on interchangera les deux parties des deux parents.

2. Mutate_1, Premier opérateur de mutation :

Cet opérateur n'agit que sur des chemins faisables : il choisit un chemin au hasard puis sélectionne quelques nœuds sur lesquels il agit en changeant leurs coordonnées en vue de créer un chemin plus « dégagé ».

3. Mutate_2, Second opérateur de mutation:

Cet opérateur s'applique aux chemins faisables et non faisables. En prenant aléatoirement un chemin de la population, cet opérateur choisit un nœud au hasard et impose un changement d'assez grande amplitude à ses coordonnées de la façon suivante:

$$x'_j = x_j + s_1 \Delta x$$
$$y'_j = y_j + s_2 \Delta y \qquad \text{(IV.7)}$$

où

x_j, y_j: coordonnées du nœud à muter avant le changement.

x'_j, y'_j: coordonnées du nœud muté après le changement.

Δx, Δy: valeurs absolues de la variation en coordonnées, nombres aléatoires uniformément distribués prenant des valeurs entre 0 et Δx_{max}, Δy_{max}, de façon à ce que les nouvelles coordonnées restent à l'intérieur des limites de l'environnement.

s_1, s_2: deux nombres aléatoires pouvant prendre la valeur -1 avec une probabilité 0.5, ou la valeur +1 avec la même probabilité. Il servent d'affectation de signe à la variation.

4. Insert_Delete: Opérateur d'insertion et suppression:

Cet opérateur agit sur les chemins non faisables, en insérant des nœuds dans des segments non faisables et en supprimant les nœuds qui sont à l'intérieur des obstacles.

5. Delete: opérateur de suppression:

Cet opérateur peut s'appliquer à tous les chemins. Un chemin est pris au hasard, puis quelques uns de ces nœuds, eux mêmes choisis aléatoirement, sont éliminés.

6. Swap: opérateur d'échange:

Il agit sur tout type de chemin, faisable ou non, en interchangeant deux nœuds voisins qu'il choisit avec une probabilité qui dépend du caractère aigu des deux angles adjacents ayant les deux nœuds en question pour sommets.

6. Smooth, opérateur de lissage:

Cet opérateur est très important, il intervient sur les chemins faisables afin d'aplanir des angles aigus. Ceci est fait en choisissant un nœud selon une loi de probabilité qui favorise les nœuds qui sont sommets des angles les plus aigus. Une fois le nœud choisi, deux nœuds sont insérés sur les deux segments connectés d'une part et d'autre à ce nœud. Les deux nœuds sont reliés entre eux par un nouveau segment. Quant au nœud en question il est supprimé.

8. Repair : opérateur réparateur :

Cet opérateur sélectionne des segments passant à travers les obstacles, dans un chemin non faisable évidement, et leur fait contourner ces obstacles en plaçant des nœuds autour de ceux-ci.

IV.1.7. Processus d'adaptation des probabilités:

Nous venons d'expliquer comment et dans quel but agissent les différents opérateurs, de quelle façon sont sélectionnés les chemins et les sites dans ses chromosomes pour subir l'effet de ces opérateurs.
Dans ce qui suit le processus de choix d'un opérateur, à chaque génération t, pour assurer la transformation d'un individu, conduisant ainsi la population de la génération t à la génération suivante t+1 est explicité.

Le choix de l'opérateur est, comme le reste des phénomènes dans l'algorithme évolutif, soumis à une loi probabiliste.

Un calcul judicieux des probabilités et une méthodes systématique de leur adaptation ont été proposés par les auteurs de [29]. Ils se sont basés sur la constatation que, la probabilité qui régit chaque opérateur gère la contribution de celui-ci à tout le processus d'évolution, affectant par chacune de ses valeurs les performances du navigateur et planificateur évolutif.

Pour calculer l'index de performance de chaque opérateur, les aspects suivants sont pris en considération :

- Son efficacité à améliorer un chemin
- Le temps d'opération (durée de calcul).
- L'effet qu'aurait l'opérateur sur les générations futures.

Le rôle de chaque opérateur varie durant les différents stades de l'évolution. Pour cela, chacun des trois aspects est calculé en fonction de l'intervalle $[T_1, T_2]$ où T_1 et T_2 sont la génération de départ et la génération terminale de la période durant laquelle les opérateurs ont été appliqués.

1. L'efficacité d'un opérateur op_i à améliorer un chemin se calcule par un rapport $e_i(T_1, T_2)$ entre le nombre de fois où l'opérateur op_i améliore un chemin N_perf_i, et le nombre total de fois où il est appliqué durant l'intervalle $[T_1,T_2]$, N_tot_i:

$$e_i(T_1, T_2)=\frac{N\_perf_i(T_1,T_2)}{N\_tot_i(T_1,T_2)} \qquad \text{(IV.8)}$$

   Cet aspect est le plus important au regard de l'évolution elle-même, il exprime le rendement de l'opérateur, ou sa capacité à obtenir de meilleurs résultats en un minimum de temps (en terme de générations) et d'efforts (en terme de calcul que suscite chaque transformation).

   Les deux autres aspects n'ont aucune influence sur la qualité de l'évolution, mais ils sont particulièrement utiles lorsque les ressources ont de sévères contraintes (i.e. le temps de calcul requis et l'espace mémoire disponible sont limités).

2. Le temps d'opération $tm_i(T_1,T_2)$ est obtenu en calculant la moyenne sur l'intervalle $[T_1, T_2]$, des durées t_j pendant lesquelles l'opérateur op_i agit.

$$tm_i(T_1,T_2)= \sum_{j=1}^{N\_tot_i} \frac{t_j}{N\_tot_i(T_1,T_2)} \quad \text{(IV.9)}$$

$tm_i(T_1,T_2)$ exprime le temps d'action moyen de l'opérateur op_i.

3. Ce dernier aspect est évoqué du fait que la majorité des 8 opérateurs (delete, insert-delete, crossover, smoothet repair) tendent à changer le nombre de nœuds des chromosomes sur lesquels ils agissent. Il est clair que le nombre de nœuds influe sur l'espace mémoire nécessaire aux différentes opérations effectuées sur les chromosomes et aussi sur le temps d'exécution de l'algorithme. L'effet qu'aurait l'opérateur op_i sur les générations futures est mesuré par $s_i(T_1,T_2)$ qui a pour expression:

$$s_i(T_1,T_2)= \frac{\delta n_i(T_1,T_2).tn}{\overline{Nn}} \quad \text{(IV.10)}$$

où:

- δn_i est la variation moyenne en nombre de nœuds provoquée par l'opérateur op_i durant la période $[T_1,T_2]$. Celle-ci est négative si ce nombre diminue et est positive dans le cas contraire. Elle est calculée en accumulant les différences de nombre de nœuds d'un chemin avant et après l'application de l'opérateur op_i, et ceci à chaque fois qu'il est appliqué sur la période $[T_1,T_2]$.

$$\delta n_i(T_1,T_2)= \sum_{j=1}^{N\_tot_i} \frac{\Delta Nn_j}{N\_tot(T_1,T_2)} \quad \text{(IV.11)}$$

avec ΔNn_j la différence en nombre de nœuds qu'a causé l'opérateur à un chromosome, après son application pour la j^{eme} fois durant la période $[T_1,T_2]$.

- $\overline{Nn}$ est le nombre moyen de nœud d'un chromosome à travers toutes les générations de l'intervalle $[T_1,T_2]$.

$$\overline{Nn}= \frac{\sum_{i=T_1}^{T_2} (\sum_{j=1}^{Nc} Nn_{ij})}{Nc.(T_2-T_1)} \quad \text{(IV.12)}$$

Nn_{ij} étant nombre de nœuds du $j^{ème}$ chromosome de la population $P(i)$ (de la génération i)
Nc étant le nombre de chromosomes dans chaque population (invariable à travers les générations).

- tn est la moyenne pondérée de la durée d'opération (sur un chromosome moyen) de tous les opérateurs durant $[T_1,T_2]$.

$$tn = \sum_{i=1}^{8} \frac{N\_tot_i(T_1,T_2)}{T_2 - T_1} tm_i(T_1,T_2) \qquad \text{(IV.13)}$$

L'indice de performance rassemble les trois aspects dans l'expression suivante:

$$I_i(T_1, T_2) = \frac{e_i(T_1,T_2) + c}{tm_i(T_1,T_2) + s_i(T_1,T_2)} \qquad \text{(IV.14)}$$

Plus grande est la valeur de cet index, meilleure est la performance de l'opérateur op_i.

Le calcul de cet index ne présente aucune difficulté car il utilise des données accumulées par l'algorithme durant son exécution. Et I_i peut être utilisé à son tour, pour le calcule la probabilité p_i de l'opérateur op_i, définie par:

$$p_i = \frac{I_i}{\sum_{j=1}^{8} I_j}, \text{ pour } i=1,8 \qquad \text{(IV.15)}$$

Pour appliquer ce qui vient d'être développé. On procède de la manière suivante :

La durée entière de l'évolution est d'abord partagée en un certain nombre d'intervalles ou d'époques comportant T générations chacune.

Pendant les premières générations $[0,T]$, tous les opérateurs sont mis à l'épreuve avec la même probabilité de valeur égale à $\frac{1}{8}$ (distribution uniforme), car à chaque génération un et un seul opérateur agit sur la population (événements exclusifs). Ceci implique que la somme des probabilités doit forcement être égale à 1. Aussi les 8 opérateurs doivent avoir les même chances, au départ pour faire leurs preuves, d'où la valeur 1/8.

A la fin de la première période (après les T premières générations), les indices de performance I_i et les probabilités p_i sont calculés. Ces nouvelles valeurs de probabilités vont régir le choix des opérateurs pendant l'époque $[T,2T]$ à la fin de laquelle les probabilités sont

recalculées pour servir à la prochaine époque $[2T,3T]$ et ainsi de suite jusqu'à la fin du processus d'évolution.

VI.1.8. Navigation/Planification on-line :

Dans la navigation en temps réel, le robot ne dispose pas de la carte entière de l'environnement. En fait, il n'en possède qu'une partie ou alors il n'en a aucune information.

Il découvre les obstacles au fur et à mesure qu'il avance sur son chemin, à l'aide des capteurs dont il est doté.

Le rôle du capteur – la perception – n'est pas pris en considération ici, il est simplement considéré que seuls sont visibles donc connus. Les obstacles ou quelques uns de leurs sommets qui sont situés à l'intérieur du cercle *champ de vision,* ayant pour centre la position courante du robot et pour rayon R la distance limite à laquelle le capteur peut détecter les objets appelé *étendue de la vue du robot.* Ces obstacles-là ou leurs parties visibles, sont reportés sur la carte du robot, carte qui est remise à jour (*updated*) à chaque fois que de nouveaux éléments sont détectés.

Comme décrit dans l'algorithme de base [29], l'évolution se fait de même que pour la navigation off-line. Un test se fait toutes les n générations pour vérifier la faisabilité du meilleur chemin issu du processus d'évolution. Il faut prendre en compte qu'en on-line, la position initiale (le nœud source) de l'ensemble des chemins est constamment remise à jour. Ce nœud devient source locale et s'identifie à la position courante du robot. Dans le cas où le test de faisabilité est vérifié, le robot avance au nœud prochain du meilleur chemin proposé. Si la longueur du premier segment de ce chemin ne dépasse pas le pas maximal k_{max} que peut effectuer le robot. Dans le cas contraire, il avance le long de ce segment d'une longueur égale au pas maximal et le processus d'évolution continue en affectant à tous les chemins le nouveau nœud source qui correspond à sa nouvelle position, et ainsi de suite jusqu'à arriver au nœud but.

Lors du déplacement du robot dans l'environnement, de nouveaux obstacles qui ne figuraient pas sur sa carte sont ajoutés. Ceci a pour effet de modifier les valeurs de la fonction d'évaluation pour l'ensemble des chemins de la population. Ces valeurs augmentent et des chemins supposés faisables jusqu'alors peuvent devenir non faisables impliquant une nouvelle recherche.

Nous venons de décrire le mécanisme de l'algorithme de la navigation on-line telle qu'il a été présentée dans [29]. Afin d'améliorer les performances de cet algorithme, nous avons apporté certaines modifications qui vont apparaître dans le développement de l'algorithme

que nous allons exposer par la suite et ceux pour des raisons de performance.

Nous avons appliqué la présente méthode de navigation à un environnement complètement inconnu, prenant ainsi le cas le plus défavorable. En effet, si le robot sait naviguer dans un tel environnement, il saura parfaitement le faire dans un milieu partiellement connu.

VI.2. Mise en œuvre et modifications :

VI.2.1. L'initialisation de la population P(t) :

Etant données les coordonnées des nœuds source et but (X_s,Y_s), (X_g,Y_g) respectivement, commençons par fixer la taille 'N_C' de la population (i.e. le nombre de chromosomes à une certaine génération t), qui restera la même –comme il a été dit précédemment– au cour de toute l'évolution (pour toutes les génération futures).

Pour fixer la longueur de chaque chromosome, attribuons lui un nombre aléatoire Nn_i. Ce nombre représente le nombre de nœuds intermédiaires que comporte un chromosome entre les deux nœuds source et but. C'est un nombre entier pris au hasard entre 0 (pour les chemins droits reliant le point de départ directement au point d'arrivé) et un certain nombre Nn_{max} , préalablement fixé, souvent pris égal à la somme des sommets de tout les obstacles de l'environnement [30], afin de prendre en considération le cas le plus défavorable (plutôt exagéré et qui n'est jamais atteint).

Il est à noter que la longueur du chromosome étant en fait Nn_i+2 (où Nn_i est le nombre de nœuds intermédiaires), est une caractéristique non fixe dans le temps car elle varie au cours de l'évolution sous l'effet des opérateurs exposés au paragraphe (IV.1.7).

Le nombre de nœuds étant fixé, il nous reste à affecter des valeurs à leurs coordonnées. Ces valeurs étant des nombres réels aléatoires variants à l'intérieur des limites qui sont :

– soit les limites physiques de l'environnement.

– soit des limites fixé par l'utilisateur .

En plaçant les nœuds sur la carte de l'environnement (il s'agit la d'une carte dépourvue d'obstacles, car on n'en tient pas compte à cette étape d'initialisation), nous les disposerons selon une distribution choisie (dont nous parlerons plus tard), de façon à les disperser dans toute

l'étendue de la surface, pour que les chemins puissent couvrir toutes les régions de l'environnement. Ceci permet de satisfaire l'objectif de faire explorer à l'algorithme la totalité de l'espace des solutions (1er objectif de la recherche optimale).

Il est, à notre avis, judicieux d'exploiter certaines connaissances (heuristiques) sur l'environnement qui montrent qu'il serait apparemment plus avantageux de passer par certaines zones que par d'autres.

Pour cela la probabilité de disposition des nœuds dans ces zones devrait être augmentée. Ceci permettra l'exploitation des meilleures solutions disponibles à l'instant (2^{e} objectif de la recherche).

Nous avons commencé par générer les points intermédiaires des chemins en affectant à leur coordonnés des valeurs uniformément distribuées dans l'espace de travail. Durant la visualisation de ces chemins, nous nous sommes rendus compte à plusieurs reprises que :

- l'ensemble des nœuds sont concentrés dans certaines zones plutôt que dans d'autres (violation du premier objectif).
- Presque tout les chemins présentent une ou même plusieurs boucles, du fait que les points sont générés dans l'ordre suivi par le robot dans son passage par ces points. Ceci a pour effet de ralentir le processus de recherche de l'algorithme car encombré de segments de chemin superflus qu'il a parfois du mal à éliminer tout seul.

Pour remédier au premier problème, nous avons introduit la distribution suivante de points de passage :

- les points source et but sont reliés par une droite d'équation y =D(x) ;
- une variation maximale Δx_{max} pour l'abscisse x, de chaque nœud est fixée (en prenant en compte le nombre maximale de nœuds Nn_{max}, car plus il y a de nœuds, plus ils ont tendance à être rapprochés). Prenons par exemple, la longueur de la projection du segment source–but sur l'axe des x et divisons-la par la moitié ou le tiers de Nn_{max} :

$$\Delta x_{\max} = \frac{\alpha \left| X_b - X_s \right|}{Nn_{\max}}, \quad \alpha > 1 \text{ quelconque} \qquad \text{(IV.16)}$$

- Pour chacun des Nn_i nœuds intermédiaires, pour i =1, N_c, on calcule son abscisse :

$$x_j = x_{j-1} \pm r_1 \Delta x_{\max} \qquad \text{(IV.17)}$$

avec j=2, Nn_i+1 et r_1 un nombre aléatoire uniformément distribué sur [0,1] ;

on calcule son ordonnée :

$$y_j = s\{D(x_j) + r_2(Y_{\max} - D(x_j))\} + \bar{s}\{D(x_j) - r_2(D(x_j) - Y_{\min})$$

(IV.18)

avec j=2, Nn_i+1
et r_2 un nombre aléatoire uniformément distribué sur [0,1] ,
Y_{max} et Y_{min} représentent les limites de l'environnement,
s est une variable booléenne qui a autant de chances d'être à 1 qu'à 0.

Quant au problème du bouclage, nous avons jugé utile et judicieux de conditionner la population initiale en détectant des boucles ou des aller-retour dans les chemins, en les supprimant à l'intersection. Plus précisément , ceci est fait par l'application de l'algorithme suivant :

- tester l'intersection de chaque segment du chemin avec tous les segments précédents ;

- si une intersection est détectée, éliminer tous les nœuds de la boucle, i.e. les nœuds contenus dans la portion du chemin limitée par les deux segments qui s'intersectent (y compris le deuxième nœud du premier segment et le premier nœud du deuxième segment intersecté), après avoir éliminé la boucle, un nouveau nœud est créé au point d'intersection, pour ne pas déformer le chemin original.

IV.2.2. Procédure de vérification de la faisabilité d'un chemin :

La faisabilité d'un chemin est déterminée comme suit :

Nous procédons d'abord à la vérification de la faisabilité de chaque segment en commençant par le premier puis passant de proche en proche jusqu'au dernier. Ceci est fait en testant l'intersection du segment en question avec tous les segments de droites représentant les côtés des obstacles (ces derniers ayant été supposés polygonaux). Une variable booléenne b initialement égale à 0 est mise à 1 à une intersection indiquant que le segment du chemin est entré à l'intérieur, et est réinitialisée à la prochaine intersection, pour montrer que le segment en est sorti. Cette variable sert d'indicateur, lorsqu'elle est égale à un et aucune intersection n'est détectée, que le segment se trouve entièrement à l'intérieur d'un obstacle, et de ce fait est non faisable.

Algorithme (2)
Début

Initialisation : $b = 0$;
Pour chaque segment seg_j, $j=1,Nn_i+1$
pour chaque obstacle obs_k $(k=1,No)$
pour chacun de ses côtés $edge_l$ $(l=1,Ns_k)$
tester l'intersection entre seg_j *et* $edge_l$
si intersection alors,
mémoriser le point d'intersection, seg_j, $edge_l$ *et* obs_k *dans une matrice P_int*
si seg_j *entre ou sort de* obs_k *alors,*
$b = \overline{b}$;
fin de si
fin de si
si intersection ou b=1 alors,
$feas\_seg_j=0$;
sinon
$feas\_seg_j=1$;
fin de si
fin de boucle
fin de boucle
fin de boucle
si pour tout $j=1,Nn_i+1$, $feas\_seg_j=1$ *alors,*
$feas\_path_i=1$;
sinon
$feas\_path_i=0$;
fin de si
fin

No : nombre d'obstacles présents sur la carte du robot.
Ns_k : nombre de sommets (ou de côtés) de l'obstacle k.

Il y a toutefois certains types d'intersection où le segment ne rentre pas dans l'obstacle (ou n'en sort pas) :

- lorsque l'une des deux extrémités du segment appartiennent à un côté d'un obstacle ;
- lorsqu'une partie du segment est confondue avec une partie du côté d'un obstacle ;

Dans ces deux cas, assez rares mais ne devant pas être négligés pour autant, la variable booléenne n'est pas affectée.

Si l'intersection du segment avec un (ou plusieurs) côté(s) d'obstacle est détectée, ou si la variable b est à 1, alors le segment considéré est non faisable. L'indicateur de faisabilité de ce segment $feas\_seg_j$ est mis à 0, et dès lors le chemin est non faisable aussi, (car il suffit d'un seul segment non faisable pour que le chemin le soit). Dans le cas contraire le segment est faisable, et après avoir effectué le test sur l'ensemble des segments, si tous sont faisables alors le chemin l'est, et son indicateur de faisabilité $feas\_path_i$ est mis à 1.

Toutes ses étapes sont résumées dans l'algorithme (2) que nous proposons pour tester la faisabilité d'un chemin *i*

IV.2.3. Correction de la fonction d'évaluation d'un chemin :

1. Pour les chemins faisables :

Apres analyse et tests, nous sommes arrivés aux conclusion suivantes :

1. Minimiser la distance à travers la grandeur $dist$ donnée par (IV.1), est une approche satisfaisante.
2. Par contre la mesure de la sécurité du chemin défini par (IV.2) par utilisation de sanction (pour les chemins trop proches des obstacles) est intéressante, mais présente l'inconvénient suivant :

   si tous les chemins ne s'approchent pas des obstacles, en aucun point plus prés qu'il ne faudrait, i.e. $\forall j = 1, l : g_j > \tau$, alors la minimisation de la grandeur $clear$ reviendrait à minimiser la plus grande distance du chemin aux obstacles, ce qui n'a aucun sens.

   Pour cela nous avons modifié l'expression de $c(n_j)$ de la façon suivante:

$$c(n_j) = e^{a(\tau - g_j)}, \quad \forall\ g_j \qquad \text{(IV.19)}$$

ainsi on garde le bénéfice de la pénalisation, et en même temps on essaye d'avoir un chemin aussi éloigné que possible de tous les obstacle. Car plus g_j s'éloigne de τ, et plus $c(n_j)$ est de valeur petite. Cette expression de $c(n_j)$ apportera sa contribution effective lors de l'évaluation des chemins dans la navigation en temps réel .

3. Une assez faible valeur du terme *« smooth »* évite d'avoir de fréquents changements de direction, assez sensibles, le long d'une petite portion de chemin.

La faisabilité physique du chemin impose de lisser la trajectoire. En effet, si le mobile devait s'arrêter à chaque intersection de segments puis redémarrer, chose réalisable, cela induirait une grande consommation d'énergie, et une perte de temps notable. De plus, changer de direction en traversant de petites distances demanderait une commande assez fine en vitesse et en accélération pour pouvoir démarrer et s'arrêter à temps. Pour éviter ces inconvénients, on procède au lissage de la trajectoire en raccordant les angles de façon à avoir une courbe continue. Cette courbure doit être assez petite (grand rayon de courbure) pour éviter le dérapage du mobile. D'où la nécessité d'avoir d'assez long segments là où les angles sont plus aigus.

Dans l'article [29] est proposée une expression d'évaluation qui permet d'apprécier à quel point la trajectoire est lisse.

Malheureusement celui-ci n'a pas pris en compte une autre contrainte de faisabilité technique qui est l'angle de braquage du véhicule.

En effet, les angles formés par les segments d'un chemin ne doivent pas être inférieurs à une certaine limite. Il est vrai que dans l'expression (IV.5) on cherche à avoir de grands angles, mais relativement aux distance des deux nœuds voisins (il est permis d'avoir un angle assez aigu si les longueurs des segments sont assez grandes).

Dans ce qui suit, nous proposons d'utiliser une autre grandeur dont nous démontrons l'efficacité. Cette grandeur sera ajoutée dans l'expression (IV.5). On pose alors :

$$smooth = Sm + \lambda . Sm' \qquad \text{(IV.20)}$$

avec

$$Sm = \max_{j=2}^{l-1} s(n_j) \tag{IV.21}$$

et
$$Sm' = \begin{cases} \max_{j=2}^{l-1} s'(n_j), & \max_{j=2}^{l-1} s'(n_j) < S_{\max} \\ e^{b(\max_{j=2}^{l-1} s'(n_j))}, & ailleurs \end{cases} \tag{IV.22}$$

où
$$s'(n_j) = \pi - \theta_j \tag{IV.23}$$

et $s(n_j)$ est défini précédemment dans (IV.5)

On introduit l'exponentielle pour sanctionner les chemins qui présentent des angles inférieurs à $\pi - S_{\max}$, b est un coefficient exprimant la sévérité de la sanction λ est un coefficient pour uniformiser les unités.

2. *Evaluation de chemins non faisables :*

En ce qui concerne l'évaluation des chemins non faisables, on lui accorde une importance moindre parce que ces chromosomes-là sont voués à la disparition au bout de quelques générations laissant la place aux chromosomes utiles.

On les évalue quand même pour accélérer l'évolution et la diriger dans le bon sens. Car certains chemins non faisables peuvent être transformés plus facilement (nécessitant moins de changements) et plus rapidement (en moins de générations) en chemins faisables que d'autres.

Il nous a été donné d'observer qu'un chemin non faisable possédant plus de caractéristiques désavantageuses qu'un autre, n'est pas le plus mauvais pour autant. Car si en effet, il demande plus de transformations et plus de temps pour devenir un chemin faisable, une fois la chose faite, il peut s'avérer meilleur que celui qui, étant non faisable, présentait à première vue moins de mauvaises propriétés. Cette notion est très importante car elle traduit le potentiel de bonnes solutions que porte en lui un chromosome non faisable, un potentiel qu'un chromosome faisable ne possède pas. Expliquons ce concept:

Les opérateurs de transformation proposés pour assurer l'évolution de la population et sa conduite vers des solutions meilleures, sont généralement et particulièrement dans la présente application, rarement destructeurs. Par opérateur destructeur on entend, un opérateur qui transforme un chemin faisable en un chemin non faisable; phénomène à première vue fâcheux mais qui s'avère être utile de temps en temps et

parfois même nécessaire. En effet, lorsqu'un chemin est faisable, il est souvent condamné à changer seulement à l'intérieur d'un couloir d'une configuration d'obstacles (c'est le problème du minimum local dans la recherche des meilleurs solutions), il est souvent difficile de le faire passer à travers un autre couloir (pour l'améliorer éventuellement) à moins de le rendre non faisable en le faisant passer par des obstacles (par des opérateurs destructeurs) pour retrouver après un chemin faisable différent. C'est en vertu de ce qui vient d'être dit qu'il serait préférable d'accorder un plus grand rôle au hasard dans l'évaluation des chemins non faisables. C'est pourquoi nous avons établi une fonction d'évaluation dès le début en suivant les brèves spécifications de [29], une fois pour toute sans y revenir, et sans y apporter aucune révision.

Revenons à présent sur les différents facteurs qu'il faut prendre en compte pour évaluer un chemin non faisable, et voyons comment les calculer :[28]

1. Le nombre d'intersections du chemin avec les obstacles *N_inters*, qu'il faut minimiser. Cette donnée est disponible après le test de la faisabilité par l'algorithme (2).

2. Le rapport *ratio,* entre le nombre de segments faisables N_feas et le nombre de segments non faisables $N\_\inf$ (on peut également obtenir ces données d'après le test de faisabilité) :

$$ratio = \frac{N\_feas}{N\_\inf} \qquad \text{(IV.24)}$$

3. La longueur totale des segments faisables et non faisables, *length*, qui est à minimiser (exactement le même que *dist* pour les chemins faisables).

4. La profondeur de l'intersection *depth*, à minimiser également : nous interprétons cette grandeur comme une somme qui s'effectue sur les segments non faisables du chemin :

$$depth = \sum_j dp_j \qquad \text{(IV.25)}$$

où

$$dp_j = \min_{l=1}^{Ns_k} d(som_l, seg_j)$$

où :

som_l est le $l^{ème}$ sommet de l'obstacle ayant Ns_k sommets.

Algorithme (3)
Evaluation
Début
Tester la faisabilité du chemin par l'algorithme (2)
Si le chemin i est faisable : feas_path$_i$ =1 alors,
Pour chaque segment seg$_j$, j=1,Nn$_i$+1
Calculer les distances d(n$_{j-1}$,n$_j$) et d(n$_j$,n$_{j+1}$) entre les nœuds n$_{j-1}$, n$_j$ et n$_j$, n$_{j+1}$
Calculer l'angle θ_j formé par seg$_j$ et seg$_{j+1}$
Calculer s(n$_j$) donné par (IV.5)
Calculer s'(n$_j$), le complément de θ_j (IV.23)
Pour tout obstacle obs$_k$, k=1,No
Pour tout côté edge$_l$, l=1,Ns
Projeter le sommet ver$_l$ sur le segment seg$_j$, si possible
Calculer la distance entre ver$_l$ et seg$_j$, si la projection existe
Projeter le nœud n$_j$ sur le côté edge$_l$, si possible
Calculer la distance entre n$_j$ et edge$_l$, si la projection existe
Fin de boucle
Fin de boucle
Trouver la distance minimale entre tous les sommets d'obstacle et seg$_j$
Mémoriser cette distance et le point de projection correspondant
Trouver la distance minimale entre tous les côtés d'obstacles et le nœud n$_j$
Mémoriser cette distance et le point de projection correspondant
Prendre g$_j$ comme la plus petite des deux distances
Calculer c(n$_j$) =$e^{a(\tau-g)}$, donné par (IV.19)
Fin de boucle
Prendre Sm comme le maximum des s(n$_j$), j=1,Nn$_i$+1
Prendre Sm' comme le maximum des s'(n$_j$), j=1,Nn$_i$+1
Si Sm'>S_{max} alors,
Sm' =$e^{b.Sm}$;
Fin de si
Calculer dist en faisant la somme des d(n$_j$, n$_{j+1}$), j=1,Nn$_i$+1
Calculer smooth =Sm+λSm', donné par (IV.20)
Prendre clear comme le maximum des c$_j$, j=1,Nn$_i$+1
Calculer eval$_{feas}$, donné par (IV.6)
Sinon
Charger la matrice P_int (voir algorithme (2))
Pour chaque portion de chemin intersectant un obstacle,
Pour chaque sommet de l'obstacle intersecté,
Calculer la distance du sommet à la droite reliant les deux points d'intersection de la portion du chemin avec l'obstacle(quand il rentre dans l'obstacle et quand il en sort).
Fin de boucle
Prendre la distance minimale (c'est elle qui caractérise la profondeur de l'intersection).
Fin de boucle
Calcul de depth en sommant la profondeur d'intersection de toutes les portions de chemin non faisables
Calcule de la somme des longueurs des segments non faisables
Calcule de la somme des longueurs des segments faisables
Calcule de ratio, en divisant la première somme par la deuxième
Calcule de length, en additionnant les deux sommes
Calcule de eval$_{inf}$ donné par (IV.26)
Fin de si
Fin

et d dénote la distance,

dp_j étant la distance du segment seg_j traversant un obstacle k (l'intersectant ou étant à l'intérieur de celui-ci), au sommet le plus proche de cet obstacle . Lorsque cette profondeur est nulle le segment ne traverse pas un obstacle, mais l'effleure seulement en un sommet.

On rassemble, les quatre grandeurs mentionnées ci-dessus dans un critère pondéré (comme pour les chemins faisables) en attribuant des pondération positives aux termes à minimiser, et négatives pour ceux à maximiser (leur combinaison étant évidement, à minimiser)

$$eval_{\inf} = v_N\, N\_inters + v_d\, depth - v_r\, ratio + v_l\, length \qquad \text{(IV.26)}$$

v_N, v_d, v_r, v_l étant des poids positifs.

3. Algorithme d'évaluation de chemin:

En appliquant ce qui a été dit jusqu'à présent, nous pouvons écrire l'algorithme d'évaluation de chemin (voir algorithme 3) .

Les points d'intersection du chemin avec les obstacles, ainsi que les segments faisables et les segments non faisables, toutes ces données sont disponibles après le test de faisabilité.

IV.2.4. Les opérateurs :

1. *Crossover, opérateur de croisement :*

Il a été dit précédemment que cet opérateur effectue un croisement entre deux chromosomes différents en interchangeant leurs parties. Il est clair que d'un tel échange résultent deux descendants. Comme il s'agit ici d'une évolution stationnaire qui produit une population intermédiaire à un seul individu (comme il a été expliqué précédemment), des deux enfants on ne doit retenir qu'un. Nous avons introduit le choix heuristique le plus logique d'en prendre le meilleur d'après l'évaluation.

Il faut noter que cet opérateur est très répandu dans les algorithmes génétiques. Il a été question ici, d'un croisement à un seul site ou à point unique. Il est possible d'avoir des croisements multi-sites où l'on divise aléatoirement les deux chromosomes parents en un même nombre de portions puis on échange en alternant ces portions entre les deux parents.

Algorithme (4)
Crossover
Début

Sélectionner deux chromosomes C_1 et C_2 de l'ensemble de la population (de N_c chromosomes), de longueurs $l1$ et $l2$ respectivement.

Sélectionner un nœud dans chacun des chromosomes sélectionnés, soient $n_{1,p}$ et $n_{2,q}$ ces nœuds.

Créer deux nouveaux chromosomes C_1' et C_2', tels que :

- *(C_1') :{$n_{1,1}$, $n_{1,2}$,... , $n_{1,p-1}$, $n_{2,q}$,$n_{2,q+1}$,... , $n_{2,l2}$ } ;*
- *(C_2') :{$n_{2,1}$, $n_{2,2}$,... , $n_{2,q-1}$, $n_{1,p}$, $n_{1,p+1}$,... , $n_{1,l1}$} ;*

Evaluer les deux chromosome-fils par l'algorithme (3).

Considérer le meilleur chromosome comme résultat de l'évolution.

Fin

Le résultat qu'on espère d'une telle alliance entre deux individus est d'en réunir les meilleures caractéristiques et d'en rejeter les mauvaises; espoir bien naïf mais qui se réalise parfois.

Cet opérateur peut être destructeur aussi, car en combinant deux chemins faisables, il peut donner un chemin non faisable.

La procédure de cet opérateur est simple et est donnée par l'algorithme (4).

2. Mutate_1, opérateur de mutation :

Le but de cet opérateur est d'éloigner le chemin des obstacles les plus proches. On choisit donc de viser ce but en procédant comme suit :

1- Premièrement, nous avons été confronté au problème de sélection des nœuds à « muter ». En laissant cela au pur hasard (conférant à tous les nœuds du chemin les mêmes probabilités d'être sélectionnés, i.e. appliquant une distribution uniforme sur le chemin) nous avons remarqué que l'opérateur choisissait souvent des nœuds situés assez loin des obstacles en laissant ceux qui en étaient proches. Il a fallut l'appliquer une multitude de fois pour avoir la chance de le faire agir sur les nœuds « dangereux ». Afin d'améliorer son efficacité, nous avons donc pensé à classer tous les indices j des nœuds n_j dans un vecteur l_g par ordre croissant de la grandeur g_j (qui est la plus petite distance du segment n_j n_{j+1} à l'obstacle le plus proche – calculée dans l'évaluation –). Puis nous avons choisi une loi de probabilité – pour sélectionner des nœuds dans le chromosome – de façon à avoir une grande probabilité d'obtenir les premiers éléments du vecteurs l_g (les indices des nœuds les plus proches des obstacles). Cette probabilité est choisi fortement décroissante quand on avance dans les éléments du vecteur l_g.

Le numéro du nœud à "muter" dans le chromosome est choisi égal au $k^{ème}$ élément du vecteur l_g .
avec:

$$k = E(Nn.(1 - r_1^{(r_2^3)})) + \qquad \text{(IV.27)}$$

E: désignant la partie entière.
Nn : nombre de nœuds intermédiaires du chromosome considéré (sans compter le but et la source)
r_1 et r_2 : deux nombres aléatoires uniformément distribué sur l'intervalle [0.1].

Remarque : Quand on parle de l'indice ou du numéro d'un nœud, on veut dire son rang dans la séquence ordonnée qui constitue le chromosome, le nœud-source étant le premier et le nœud-but le dernier.

2- *Deuxièmement,* ayant trouver une façon de sélectionner chaque nœud à muter, il reste à trouver comment fixer le nombre de nœuds à muter.

Nous avons essayé au début de prendre un nombre quelconque allant de 1 à Nn. Nous avons alors constaté qu'en agissant sur plusieurs nœuds à la fois, l'opérateur améliorait le chemin par endroits et le détériorait en d'autres. Pour dissocier ces deux phénomènes, nous avons décidé de le faire appliquer sur un seul nœud. Ainsi le chemin est soit amélioré, soit détérioré. Cela permet de mieux le classer et d'apprécier l'effet de l'opérateur.

3- *Troisièmement,* Comment éloigner un nœud n_k d'un obstacle?

Le noeud est déplacé suivant une droite reliant ce point à sa projection sur le côté le plus proche de l'obstacle, on le déplace - bien entendu - dans le sens opposé de l'emplacement de l'obstacle, d'une distance assez petite qu'on a pris égale à

$$\Delta = \alpha_{sec} r(\tau - g_k) \qquad \text{(IV.28)}$$

où α_{sec} est un coefficient supérieur à 1,

g_k, la distance entre le nœud et le côté,

τ, la distance de sécurité

et r, un nombre aléatoire appartenant à [0,1].

on a pris Δ aléatoire pour éviter d'avoir le même résultat, si par hasard, l'opérateur serait appliqué plusieurs fois au même chromosome et sur le même nœud.

Nous avons aussi étendu l'idée de déplacer les nœuds pour les éloigner de l'obstacle et au point d'insérer de nouveaux nœuds si le chemin s'approche de trop prés d'un obstacle dans une zone située entre deux nœuds.

Pour être plus précis, deux points appartenants à deux segments (un segment du chemin et un côté de l'obstacle) sont le plus proches possible si l'un est une extrémité d'un des segments et l'autre en est la projection sur le l'autre segment. A ce moment-là, la distance minimale peut se trouver :

- Soit au niveau d'un nœud du chemin et sa projection sur le côté de l'obstacle (dans ce cas le nœud doit être éloigné, comme décrit plus haut).

- Soit au niveau d'un sommet et sa projection sur le segment. Dans ce dernier cas, on crée un nœud sur le prolongement de la droite reliant ce sommet à sa projection sur le segment du chemin, à une distance donnée par (IV.28) de ce point de projection et dans le sens inverse de l'emplacement de l'obstacle. Puis on l'insère entre les deux nœuds du segment.

N.B :Lorsqu'on parle de distance entre deux segment, en entend bien la distance perpendiculaires car il est possible d'avoir deux segments très proches par leurs extrémités n'ayant aucunes projection l'un sur l'autre, ce cas n'est pas considéré ici.

4- Quatrièmement, Pour choisir un chemin auquel s'appliquera l'opérateur, nous avons accordé, au début, les mêmes chances à l'ensemble des chemins faisables d'être choisi. Le problème est le suivant :

Sachant que les chemins de chaque génération sont classés du meilleur au pire. Le chemin-solution, sortant d'une évolution complète est le résultat d'une succession de transformations effectuées sur l'ensemble de la population. Vers la fin du processus, au cours des dernières générations, ce chemin-solution a le plus de chances d'être produit par des transformations opérées sur un groupe limité comportant les meilleurs chemins ; lequel groupe se resserre à mesure que l'évolution avance.

Avec une distribution uniforme de probabilité dans la partie faisable de la population, pour le choix du chemin –siège de cette mutation, nous avons constaté qu'aux dernières générations, l'opérateur agit encore ici et là sur l'ensemble des chemins faisables, n'apportant aucun changement radical, car n'étant (ainsi que *smooth*) qu'un opérateur de forme, dont le rôle est d'affiner le chemin en le rendant plus sûr. L'action de cet opérateur est souhaitable pour les chemins qui ont le plus besoins de finitions, d'ou la conclusion qu'il vaudrait donc mieux concentrer son activité sur ces chemins-là. Ainsi, sa contribution au résultat final de l'évolution serait plus grande que s'il agissait sur les autres membres de la populations, auxquels il ne serait pas d'une grande utilité pour les rendre meilleurs.

L'indice du chemin choisi pour la mutation serait donc tel que lors des premières générations l'ensemble des chemins faisables aient la même chance, puis à force d'avancer dans les générations, plus les

chemins sont meilleurs (par rapport aux autres, et non par rapport aux critères) plus ils ont de chances d'être sollicités. Soit q l'indice du chemin faisable choisi pour la mutation à la t-ième génération du processus à T générations en total, les chemins sont classés comme dit précédemment, du meilleur au pire, i.e., q=1 est l'indice du meilleur chemin, et q=N_{feas} est celui du pire faisable (N_{feas} est le nombre des chemins faisables à la génération t), on donne q comme suit :

$$q = E\left\{(N_{feas} - 1)(1 - r^{(1-\frac{t}{T})})\right\} + 1 \; ; \qquad \text{(IV.29)}$$

où r est un nombre aléatoire uniformément distribué sur [0,1].

Pour *mutate_1,* nous proposons l'algorithme (5).

3. Mutate_2, second opérateur de mutation:

Cet opérateur est là pour apporter de la variété dans la population.

Mutate_2 est autant capable de transformer un chemin non faisable en un chemin faisable qu'inversement (mais cet aspect rentre aussi dans son utilité).

L'algorithme (6) explique sa façon d'agir.

4. Insert_Delete: Opérateur d'insertion et suppression:

Nous avons tenté de réaliser l'effet produit par cet opérateur, en sélectionnant un obstacle que le chemin intersecte, puis en supprimant les nœuds qui sont à l'intérieur, ou en essayant d'insérer des nœuds en dehors des obstacles dans le cas où il n'y aurait pas de nœuds à l'intérieur de l'obstacle.

Le nœud inséré dans le segment $\overline{n_j n_{j+1}}$ a les coordonnées suivantes :

$$x'_j = r_1 (x_j + x_{j+1}) + s_1 \Delta x$$
$$y'_j = r_2 (y_j + y_{j+1}) + s_2 \Delta y \qquad \text{(IV.30)}$$

où

r_1, r_2 deux nombres aléatoires uniformément distribués sur l'intervalle [0,1]. x_j,y_j: coordonnées du nœud avant le changement.

x'_j, y'_j: coordonnées du nouveau nœud après le changement

Δx, Δy: valeurs absolues de la variation en coordonnées, nombres aléatoires uniformément distribués prenant des valeurs entre 0 et Δx_{max}, Δy_{max}, et de façon à ce que les nouvelles coordonnées restent à l'intérieur des limites de l'environnement.

Algorithme (5)
Mutate_1
Début
N_1=0 ; N_2=0 ;
Sélectionner un chemin faisable dans la population suivant (IV.28).
Classer les indices j des nœuds du chemin sélectionné par ordre croissant des valeurs g_j, dans un vecteur I_g.
Un nœud n_k du chemin choisi, est désigné pour être muté, avec k défini dans (IV.27)
Si la distance minimale, du segment représenté par le nœud désigné, au côté le plus proche de l'obstacle le plus proche, est inférieure à la distance de sécurité τ^* et si le nombre d'essais N_1 pour désigner le nœud à muter est inférieur à une certaine limite Ne_1, alors :
Tant que le nombre d'essais N_2 pour éloigner le segment, ne dépasse pas une limite Ne_2, faire :
Si la distance minimale entre le segment et le côté est au niveau du nœud n_k et sa projection sur le côté, alors
Cas=1 ;
Créer un nœud potentiel quelque part sur le prolongement de la droite reliant ces deux points.
Sinon (elle est au niveau du sommet et sa projection sur le segment),
Cas=2 ;
Créer un nœud potentiel quelque part sur le prolongement de la droite reliant ces deux points
Fin de si
Si le nouveau nœud n'est pas en dehors des limites de l'environnement et s'il ne causerait pas d'intersection du chemin avec un obstacle (dans le cas où il aurait été intégré dans le chemin), alors :
Si cas=1, alors :
Remplacer le nœud n_k par le nœud potentiel ;
Sinon
Insérer le nœud potentiel dans le chemin entre les nœuds n_k et n_{k+1} ;
Fin de si
Sinon
Incrémenter le nombre d'essais $N_2 = N_2+1$;
Fin de si
Fin de boucle
Sinon
Incrémenter le nombre d'essais $N_1 = N_1+1$;
Remettre le nombre d'essais N_2 à zéro ;
Fin de si
Fin de boucle
Fin

Algorithme (6)
Mutate_2
Début

1. *Sélectionner un chemin quelconque dans la population ;*
2. *Choisir un nœud quelconque dans ce chemin ;*
3. *Générer un nœud de coordonnées données par (IV.7) ;*
4. *Tant que le nœud généré est en dehors des limites de l'environnement, et le nombre d 'essais N n'atteint pas un nombre limite Ne, faire :*
   Générer un autre nœud ;
   Incrémenter le nombre d'essais : N=N+1 ;
   Fin de boucle
5. *Si le nœud généré est à l'intérieur des limites de l'environnement, alors*
   Remplacer le nœud sélectionné en 2. par le nouveau généré en 3-4 ;
   Sinon (cas très rare)
   Choisir un autre nœud et refaire le processus ;
   Fin de si

Fin

s_1, s_2: deux nombres aléatoires pouvant prendre la valeur -1 avec une probabilité 0.5, ou la valeur +1 avec la même probabilité, il servent d'affectation de signe à la variation.

L'insertion se faisant d'une façon aléatoire, le nouveau nœud ne se trouve pas nécessairement en dehors des obstacles et les deux segments qu'il relie ne sont pas forcement faisables. Pour augmenter les chances d'obtenir la configuration souhaitée, l'opération est menée plusieurs fois. Dans le cas où elle échoue à chaque application, le processus est arrêté au bout d'un certain nombre et la dernière version est retenue.

Nous proposons l'algorithme (7) pour traduire le comportement d'un tel opérateur .

5. Delete: opérateur de suppression:

Nous avons décidé (pour les mêmes raisons que dans mutate_1) de ne faire agir cet opérateur que sur un seul nœud du chromosome. C'est le cas où il est le plus performant.

Parfois, en enlevant un nœud d'un chemin faisable , nous avons constaté que l'opérateur rendait le chemin non faisable. Afin de remédier à cela, nous avons introduit le test suivant :

Si le segment reliant le nœud situé avant le nœud à supprimer au nœud situé après, intersecte avec quelqu'obstacle, un autre nœud à supprimer est choisi puis le test est refait ... et ainsi de suite, on refait cela un certain nombre de fois et si à chaque fois le chemin est détérioré, la suppression ne se fait pas.

L'utilité de cet opérateur est d'éviter certains détours inutiles et d'enlever certains nœuds gênants d'un chemin car il arrive même qu'en enlevant un nœud, une portion non faisable d'un chemin devienne faisable.

L'algorithme (8) résume notre approche.

6. Swap: opérateur d'échange:

Cet opérateur permet souvent d'élargir les angles sur la portion du chemin où il opère. Il permet aussi, avec beaucoup de chances, de faire passer un chemin allant au travers d'un obstacle, en dehors de celui-ci, mais il a tendance de faire l'inverse plus facilement.

Algorithme (7)
Insert_delete
Début

$Ne_1 = Ne_2 = Ne_3 = 0$;

1. *Sélectionner un chemin parmi les chemins non faisables ;*
2. *Désigner un site sur lequel l'opérateur agira : une portion $\{n_k, \ldots n_l\}$ du chemin choisi traversant un même obstacle obs_q, de façon à ce que le premier segment de cette portion, $\overline{n_k n_{k+1}}$ entre dans l'obstacle et le dernier, $\overline{n_{l-1} n_l}$ en sort ($l\text{-}1 \geq k$).*
3. *Insérer un nouveau nœud dans le segment $\overline{n_k n_{k+1}}$, dont les coordonnées sont données par la relation (IV.30), dans laquelle on remplace j par ; le nouveau nœud devient n_{k+1}, et tous ceux qui viennent après se décalent d'une position.*
4. *Tant que le segment $\overline{n_k n_{k+1}}$ n'est pas faisable (selon la définition de faisabilité d'un segment) et le nombre d'essais N_1 n'atteint pas une certaine limite Ne_1,*
   Remplacer le nœud inséré n_{k+1}, en générant un autre, toujours par (IV.30) où $j=k$;
   Incrémenter le nombre d'essais : $N_1 = N_1 + 1$;
   Fin de boucle
5. *Pour chaque nœud n_j situé à l'intérieur de l'obstacle obs_q : $j = k+2, l$ (la portion désignée en 2. comporte un nœud de plus, inséré en 3-4.) :*
   Tant que le segment $\overline{n_j n_{j+1}}$ n'est pas faisable et le nombre d'essais N_2 n'atteint pas une certaine limite Ne_2, faire :
   Remplacer le nœud n_j par un nouveau, généré en (IV.30)
   Incrémenter le nombre d'essais : $N_2 = N_2 + 1$;
   Fin de boucle
   Fin de boucle
6. *Insérer un nouveau nœud dans le segment $\overline{n_l n_{l+1}}$, dont les coordonnées sont données par (IV.30), où $j = l+1$; le nouveau nœud devient n_{l+1}, et tous ceux qui suivent sont décalés d'une position.*
7. *Tant que les deux segments reliés par le nœud inséré n_{l+1}, $\overline{n_l n_{l+1}}$ et $\overline{n_{l+1} n_{l+2}}$ ne sont pas tous les deux faisables en même temps, et le nombre d'essais N_3 n'atteint pas une certaine limite Ne_3,*
   Remplacer le nœud inséré n_{l+1}, par un autre, toujours donné par (IV.30) avec $j=l$;
   Incrémenter le nombre d'essais : $N_3 = N_3 + 1$;
   Fin de boucle

fin

Algorithme(8)
Delete
Début
Choisir un chemin dans la population
Choisir un nœud à supprimer soit n_k
Tant que le segment reliant le nœud n_{k-1} *et le nœud* n_{k+1} *coupe un obstacle et que le nombre d'essais* $N<Ne$, *faire :*
Choisir un autre nœud n_k ;
Incrémenter le nombre d'essais : $N=N+1$;
Fin de boucle
Si le segment $\overline{n_{k-1}n_{k+1}}$ *ne coupe aucun obstacle, alors*
Supprimer le nœud n_k ;
Fin de si
fin

En ce qui concerne le choix de la loi de probabilité qui détermine les nœuds à interchanger, on l'a prise exponentielle en fonction des angles θ_j formé par les deux segments $\overline{n_{i-1}n_i}$ et $\overline{n_i n_{i+1}}$ et θ_{j+1} formé par $\overline{n_i n_{i+1}}$ et $\overline{n_{i+1} n_{i+2}}$

L'algorithme que nous proposons pour décrire l'activité de cet opérateur est très simple (voir algorithme (9)).

Où λ est un coefficient qui, en grandissant rend l'applicabilité de l'opérateur, de plus en plus difficile.

Nous l'avons pris assez faible pour inhiber en quelque sorte cet opérateur qui n'a pas fait preuve de bonne performances lorsque nous lui avons laissé une certaine liberté pour agir (en affectant une assez grande valeur à λ). Une activité intense avait plutôt un effet négatif. Toutefois nous avons pu constater qu'en réduisant cette activité au minimum, il lui arrivait de produire de bons et même étonnants résultats sur le chemin auquel il s'appliquait.

6. Smooth, opérateur de lissage

Au début, pour sélectionner un nœud (qui est sommet de l'angle à couper) dans le chemin candidat, nous avons introduit une loi exponentielle de probabilité en fonction de l'angle à ce nœud (comme pour l'opérateur *swap)*. Après test, nous avons constaté le même problème que pour mutate_1.

Souvent, l'opérateur s'attaquait aux angles pas trop mauvais en laissant les angles vifs et parfois irréalisables, tels quels. Pour remédier à cela nous avons procédé comme pour mutate_1 en formant un vecteur I_θ classant les indices des nœuds par valeurs croissantes des angles dont ils sont les sommets, puis comme pour Mutate_1, nous avons écrit une loi de probabilité favorisant le choix des premiers éléments du vecteur I_θ (i.e. les nœuds qui présentent les angles les plus aigus du chemin).

Une fois l'angle à aplanir sélectionné, il faut evaluer si l'opérateur doit agir ou non sur cet angle. Pour cela, on a gardé la même loi de probabilité exponentielle que pour *swap*, tout en le forçant à agir pour tout angle inférieur à une certaine limite qui elle, doit être supérieure à l'angle de braquage.

Etant donné que le présent opérateur est un opérateur de forme comme *Mutate_1*, le choix de chemin sur lequel il sera appliqué sera régi par la même loi et pour les mêmes raisons que pour *Mutate_1*.

Algorithme(9)
Swap
Début
Sélectionner un chemin de la population
Pour chaque nœud n_k du chemin, k=2,l-2
Prendre un nombre r au hasard dans un intervalle [0,1] ;
Si $r < e^{\frac{\lambda(\theta_k+\theta_{k+1})}{2\pi}} - e^{-\lambda}$, *alors :*
(remplacer les coordonnées de n_k par ceux de n_{k+1} et inversement) :
$n_k \rightarrow n$;
$n_{k+1} \rightarrow n_k$
$n \rightarrow n_{k+1}$;
Sortir de la boucle ;
fin de si
fin de boucle
fin

Nous proposons l'algorithme (10) pour traduire le comportement de l'opérateur smooth.

Algorithme (10)
Smooth
Début

Choisir un chemin de la partie faisable de la population selon la loi (IV.29);
Construire le vecteur I_θ dans lequel les nœuds du chemin choisi sont classé selon les angles dont ils sont les sommets du plus au moins aigus.
Choisir un nœud n_k donné par la relation : $k=E(Nn.(1-r_1^{(r_2^3)}))+1$
Piocher un nombre aléatoire dans l'intervalle [0,1] ;

Si $r < e^{\frac{\lambda\theta_k}{\pi}} - e^{-\lambda}$ ***ou*** $\theta_k < \theta_{sec}$, *alors :*

Créer un nœud hypothétique n'importe où sur chacun des segments $\overline{n_{k-1}n_k}$ *et* $\overline{n_k n_{k+1}}$, *soient* n_1 *et* n_2 *ces nœuds.*
Tant que le segment $\overline{n_1 n_2}$ *coupe un obstacle quelconque et le nombre d'essais N ne, dépasse pas un nombre limite d'essais N<Ne, faire ;*
Créer à nouveau les nœuds n_1 *et* n_2 *;*
Incrémenter le nombre d'essais : N=N+1 ;
Fin de boucle
Si le segment $\overline{n_1 n_2}$ *ne coupe aucun obstacle, alors :*
Insérer les nœuds dans les segments $\overline{n_{k-1}n_k}$ *et* $\overline{n_k n_{k+1}}$ *;*
Fin de si
Fin de si
Fin de l'algorithme.

8. Repair : opérateur de réparation:

La méthode utilisée pour forcer le chemin à contourner les obstacles, est clairement illustrée dans l'algorithme (11).

Algorithme (11)
Repair
Début

1. *Un chemin de la population non faisable est pris au hasard.*
2. *Un nombre est "pioché ", pour déterminer combien il y a de "réparation" à faire. Par réparation on entend, un segment ou un groupe de segments consécutifs, intersectant ou étant à l'intérieur d'un même obstacle, sur lesquels l'opérateur agira. Ce nombre est un nombre aléatoire entier uniformément distribué entre 1 et le nombre d'obstacles que le chemin coupe sans en sortir (il peut arriver qu'un même obstacle soit traversé deux fois (ou plus) par le chemin, lorsqu'entre les deux traversées, il y a un segment entier ou plus, à l'extérieur de l'obstacle considéré).*
3. *Pour chaque réparation à faire:*

- *Un grossissement de l'obstacle intersecté est fait en formant de nouvelles arrêtées parallèlement à celles de l'obstacle en question, à une distance τ_{rep} de l'extérieur de cet obstacle.*
- *Deux nœuds sont créés aux deux points d'intersection du contour de l'obstacle grossi d'une part, avec le premier segment de la "séquence à réparer" - le segment entrant – d'autre part, ceci en ce qui concerne le premier nœud. Le second est formé par l'intersection de ce même contour avec le dernier segment de cette même séquence – segment sortant.*
- *Ces deux nœuds sont insères dans la séquence du chromosome, selon leurs positions respectives dans le chemin (entre les deux nœuds –extrémités du premier et dernier segments de la "séquence en réparation").*
- *Tous les points situés entre ces deux derniers nœuds sont ensuite supprimés.*
- *Les longueurs des deux parties du contour de l'obstacle grossi (le contour étant partagé en deux, par les deux nœuds de l'intersection du chemin original avec ce contour) sont calculées.*
- *Les sommets de l'obstacle grossi situés sur la partie du contour de moindre longueur, sont pris comme nœuds, et sont insérés dans le chemin entre les deux nœuds de l'intersection dans l'ordre de leur apparition sur le chemin de contournement.*

Fin de boucle
Fin

IV.2.5. Modification des index de performances des opérateurs *(Problèmes rencontrés et solution proposée) :*

Après avoir appliqué la méthode d'adaptation des probabilités proposée par [29], nous avons eu quelques problèmes au niveau des valeurs des index I_i .

Malgré la garantie de la positivité de l'index que les auteurs ont rapporté dans leurs article [29] toutefois sans preuves. Il nous est arrivé souvent lors de l'exécution du programme de constater que la quantité $s_i(T_1,T_2)$ était négative et supérieure en valeur absolue à la valeur tm_i (qui est toujours positive) ; d'où un I_i négatif.

Ce dernier produit une probabilité p_i négative. Ce qui à pour effet d'inhiber complètement un opérateur pendant toute une période de T générations ; chose ne devant se produire en aucun cas.

En effet dans le pire des cas, quand l'opérateur op_i ne présente aucune performance durant toute une période, son efficacité e_i est nulle mais son index de performance n'est pas nul, à cause de la petite constante c dans l'expression de I_i qui lui procure une petite probabilité, d'où une petite chance d'agir malgré tout, durant l'intervalle suivant .

Or l'application de l'index I_i en pratique montre qu'il arrivait même qu'un des huit index soit tellement négatif qu'il l'emportait sur les sept autres, rendant la somme négative. Le résultat était catastrophique puisque sept sur huit probabilités acquerraient des valeurs négatives, et il ne restait qu'un seul opérateur pour agir pendant la période suivante.

Ceci nous a poussé à réécrire l'expression de l'index de performance. Cette fois ci, nous l'avons écrit sous forme d'une combinaison linéaire de certaines grandeurs :

$$I_i = e_i(T_1,T_2) - u_i(T_1,T_2) + \max_{j=1}^{8} u_j(T_1,T_2) + c \qquad \text{(IV.31)}$$

où $e_i(T_1,T_2)$ est l'efficacité de l'opérateur op_i donnée par (IV.8)
et $u_i=(T_1,T_2)$ est donné par la relation suivante :

$$u_i(T_1,T_2) = tmr_i(T_1,T_2) + tr_i(T_1,T_2) + \delta nr_i(T_1,T_2) \qquad \text{(IV.32)}$$

où

$$tmr_i(T_1,T_2) = \frac{tm_i(T_1,T_2)}{\max_{j=1}^{8} tm_j(T_1,T_2)} \qquad \text{(IV.33)}$$

$tm_i(T_1,T_2)$ étant le temps moyen d'action de l'opérateur op_i défini dans (VI.9).

$$tr_i(T_1,T_2) = \frac{\sum_{j=1}^{N\_tot_i} t_{ij}}{\sum_{k=1}^{8} \sum_{l=1}^{N\_tot_k} t_{kl}} \qquad \text{(IV.34)}$$

t_{ij} est la durée d'action de l'opérateur op_i à une certaine génération j où il est appliqué.
N_tot_i étant le nombre total de générations où cet opérateur est appliqué.

$tr_i(T_1,T_2)$ est en fait une fraction de la durée totale d'une période d'évolution, de $(T2-T1)$ générations, occupée par l'action de l'opérateur op_i.

$\delta nr_i(T_1,T_2)$ est défini par :

$$\delta nr_i(T_1,T_2) = \frac{\sum_{j=1}^{N\_tot_i} \Delta Nn_{ij}}{\sum_{k=1}^{8} \left| \sum_{l=1}^{N\_tot_k} \Delta Nn_{kl} \right|} \qquad \text{(IV.35)}$$

ΔNn_{ij} est la différence de nombre de nœuds entre les chemins père et fils provoquée par l'opérateur op_i lorsqu'il est appliqué à une certaine génération j.

Les résultats obtenus grâce à ce nouvel index sont probants comme le montreront les figures et tableau.

IV.2.6. Algorithme de la Planification/Navigation off-line:

Les étapes de l'algorithme off-line sont décrites dans l'algorithme (12).

Algorithme (12)
Offline
Début

1. *Introduction des données suivantes:*
   - *La taille de la population Nc.*
   - *La longueur maximale d'un chromosome $Nn_{max}+2$*
   - *Les limites de l'espace dans lequel se fera la tâche.*
   - *Les coordonnées des points de départ et d'arrivée dans cet espace.*
2. *Initialisation de la population: P(0).*
3. *Introduction des différents paramètres nécessaire à l'évaluation (comme les pondérations des fonctions d'évaluation).*
4. *Chargement de la carte des obstacles.*
5. *Evaluation de toute la population initiale.*
6. *Initialisation des index de performance et des probabilités d'application des opérateurs durant la première période d'évolution (p_i =.1/8).*
7. *Introduction de :*
   - *la fréquence d'adaptation fa des probabilités qui consiste en le nombre de fois où se refera la remise à jour des probabilités durant toute l'évolution (i.e. en combien de périodes a-t-on partagé la totale durée du processus)*
   - *la durée T de chaque période (i.e. combien de générations comporte-t-elle)*
8. *Pour chaque période, de 1 à fa,*
   Pour chaque génération de la période, de 1 à T,
   - *8.1. classement des individus de la population du meilleur au pire selon la valeur croissante de sa fonction d'évaluation, en prenant en compte que le pire (dernier) chemin faisable est meilleur que le premier (meilleur) chemin non faisable, donc tous les chemins faisables seront classés en premier lieu et tous les chemins non faisables viendront ensuite.*
   - *8.2. Application d'un opérateur op_i choisi avec la probabilité p_i sur un certain chromosome de la population. La valeur de p_i varie d'une période à l'autre (elle est calculée à la fin de la période précédente si elle existe ou égale à $\frac{1}{8}$ sinon). Le choix et l'application d'un opérateur peut se faire en un if-bloc comme suit:*

     Soit r un nombre aléatoire uniformément distribué sur l'intervalle [0,1],
     Pour i = 1,8

     $$Si \sum_{j=1}^{i-1} p_j \le r < \sum_{j=1}^{i} p_j \text{ alors}$$

     application de l'opérateur op_i
     fin de si
     fin de boucle
   - *8.3. Evaluation de la population intermédiaire, i.e. du nouveau chromosome fils obtenu par la transformation provoquée par l'opérateur op_k qui vient d'être appliqué à un (des) chromosome(s) père(parents).*
   - *8.4. Remplacement du pire chromosome (le dernier d'après le classement de 8.1) par le chromosome fils de cette génération[1].*
   - *8.5. Pour l'opérateur op_k (qui est appliqué en 8.2)*
     - *Incrémentation du nombre de fois où op_k est appliqué durant la période courante, N_tot_k.*
     - *Calcul de la différence en nombre de nœuds ΔNn_k entre le chromosome-père et le chromosome-fils (provoquée par op_k), et accumulation de cette différence à celles des générations précédente de la même période.*
     - *Si le chemin représenté par le chromosome-père étant non faisable est devenu faisable par op_k, ou si les chemins représentés par les deux chromosomes père et fils étant tous les deux faisables ou tous les deux non faisables, ont leurs valeurs de la fonction d'évaluation telles que celle du fils soit inférieure à celle du père, alors :*
       Incrémentation de $N\text{-}perf_k$, le nombre de fois où op_k améliore le chemin.
       Fin de si

   Fin de boucle
   (A la fin de chaque période) calcul des index de performance puis des probabilités p_i i =1.8, en utilisant les relations IV.15 et IV.31.
   Remise à zéro des éléments calculés dans le point 8.5.
   Fin de boucle
9. *(A la fin du processus) le chemin classé premier est pris comme solution au problème.*

[1] *Il semble préférable à première vue, de tester si le nouveau chemin obtenu soit meilleur que le pire des chemins de la population avant de remplacer ce dernier mais comme nous l'avons dit précédemment, un mauvais chemin a toutes ses chances pour devenir meilleur plus tard, de plus, le fait de remplacer à chaque génération le pire chemin sans test marque un point positif pour la préservation de la diversité des espèces.*

IV.2.7. Amélioration de la Navigation/Planification on-line :

1. Nous voudrions, en premier lieu, attirer l'attention du lecteur sur la manière presque aveugle de l'avancement du robot sur le meilleur chemin issu du processus d'évolution. Toutes les n (ou multiple ne n) générations, avec la seule condition que ce chemin-là soit faisable. Or, le fait qu'il soit faisable et même qu'il soit le meilleur faisable parmi les chemins disponibles à cette génération, ne garanti pas que ce chemin soit sûr et physiquement réalisable. Avant d'engager l'exécution de ce chemin, nous avons donc estimé bon d'ajouter les vérifications suivantes:

   - La distance perpendiculaire g_j, (voir évaluation des chemins faisables), entre le prochain segment de ce chemin (segment qu'il s'apprête à suivre) et le plus proche côté d'obstacle g_j est supérieure à la distance de sécurité τ.

$$g_j > \tau \qquad \text{(IV.36)}$$

   - L'angle θ_j, (voir évaluation des chemins faisables), entre le dernier segment traversé par le mobile, l'ayant amené à sa position actuelle, et le prochain segment dans lequel il s'engage, est supérieur à l'angle de braquage, disons θ_{braq}.

$$\theta_j > \theta_{braq} \qquad \text{(IV.37)}$$

   - Le rapport α_j, entre le complément de cet angle et la longueur du segment qu'il s'apprête à suivre, reliant la position courante au nœud prochain (grandeur exprimant la même chose que $s(n_j)$ dans $smooth$, voir évaluation des chemins faisables), est inférieur à une grandeur limite $\alpha_{\lim}$.

$$\alpha_j < \alpha_{\lim} \qquad \text{(IV.38)}$$

   Si toutes les conditions précédentes sont vérifies ,alors seulement le mobile est autorisé à avancer. Dans le cas contraire l'évolution continue afin de trouver un chemin meilleur.

2. le processus d'évolution est un phénomène convergent. Il commence par un ensemble de chemins aléatoirement dispersés dans l'espace de la tâche (résultat de l'initialisation). Après un nombre suffisant de

générations, les chemins deviennent voisins du meilleur chemin. En fait, plus les générations avancent plus ils s'en approche (le meilleur chemin évolue lui aussi, évidemment). Ce phénomène serait apprécié dans le cas où le chemin leader de la population tendrait à coup sûr vers le chemin optimal, malheureusement l'optimalité de ce chemin n'est pas garantie. Il arrive donc, lorsque les conditions citées dans le point précédent ne sont pas vérifiés, et malgré la répétition de l'évolution un nombre de fois suffisant, le chemin bloque dans une certaine plage de variations et n'arrive jamais à remplir ces conditions. Ce problème ne surgirait pas bien sûr si l'on se contentait d'un chemin faisable pour avancer. Dans le cas où la recherche part dans le mauvais sens, il est indispensable de réexplorer l'espace des solutions. Pour cela, nous réinitialisons la population après un certain nombre de fois ou le résultat du test de l'acceptabilité de la solution proposée, est négatif.

3. Dans l'algorithme de base (1), à chaque pas du robot, les auteurs proposent de relier l'ensemble des chemins de la population disponibles de la génération courante, à la position actuelle du robot, remettant ainsi à jour son nœud-source local. Sachant qu'en avançant, le robot découvre à chaque fois de nouveaux obstacles, il est souvent confronté à des situations où tous les chemins dont il dispose exigeraient des changements radicaux. Dans la nouvelles carte de l'environnement (remise à jour), ces chemins paraîtraient presque aussi absurdes que s'il venaient d'être initialisés sauf qu'à ce stade d'évolution ils seraient plutôt concentrés dans un certain couloir (problème mentionné dans le point précédent) et l'évolution dans ce cas repartirait à zéro avec les chemins issus de l'évolution précédente en tant que population initiale.

   Afin de remédier à l'inconvénient, de chercher un moyen de rattacher tous les chemins au nouveau nœud-source, et de perdre du temps à faire repartir l'évolution dans un autre sens et à partir de mauvaises bases (mauvais conditionnement de la population initiale de cette étape), cet auteur a trouvé simple et plus judicieux de réinitialiser la population à chaque pas, d'autant plus que l'étape d'initialisation consomme très peu de temps, et ne prend pas de mémoire supplémentaire au reste de l'algorithme. De plus, le meilleur chemin est sauvegardé, et fait partie de la population au cours du prochain cycle d'évolution.

   Il nous à été donné d'observer, à partir des simulations que l'initialisation de la population est une étape cruciale dans la recherche. L'ensemble des chemins initiaux doit embrasser l'étendu de l'espace des solutions.

4. il a été suggéré dans [29], que toutes les n générations d'évolution, le robot ne peut avancer que d'un seul pas, au plus. Cela veut dire, qu'il ne peut dépasser le premier segment du meilleur chemin courant (reliant sa position courante au prochain nœud) même si ce nœud est très proche.

   Pour accélérer l'exécution de l'algorithme, nous avons autorisé l'accès à tous les nœuds de la portion du chemin entièrement contenue à l'intérieur du cercle centré sur la position actuelle (le nœud-source local) et de rayon k_{max} (pas maximal d'avancement du robot). Le robot passe d'un de ces nœuds à l'autre en vérifiant les trois conditions du point 1. et ceci sur le chemin de la même génération. Si à un moment donné, les conditions ne sont plus vérifiés, alors, l'évolution est relancé à partir du dernier nœud répondant à ces conditions.

5. Il arrive que l'algorithme produit un chemin avec des nœuds très rapprochés et superflus, l'exécuteur de trajectoire doit arrondir toutes les intersections entre segments en les transformant en des courbes lisses et continues.

   Dans le souci de reduire le nombre de point utilisé par la génération de trajectoire, cet auteur a donc jugé préférable de supprimer les nœuds très rapprochés, à condition que cela ne détériore pas le chemin. Pour ce faire, avant que le robot ne prenne la décision d'avancer au nœud prochain, et avant même de tester la conformité du segment à suivre (décrite dans 1.), la distance à ce nœud est mesurée. Dans le cas où celle-ci est inférieure à une distance minimale d_{min}, on vérifie si le segment reliant le nœud courant au nœud qui vient juste après le nœud proche présente les qualités requises (i.e. remplie les trois conditions de 1.). C'est alors et si toutes les conditions précédentes sont vérifiées que l'on supprime le nœud proche.

6. Dans le cas de navigation en temps réel et dans un environnement complètement inconnu, nous avons opéré des modifications radicales dans l'algorithme d'évaluation, en ce qui concerne les chemins faisables.

   En effet, dans la navigation off-line, le chemin est recherché en bloc, et doit répondre aux critères établis en chacun de ses points, par conséquent, la fonction d'évaluation prend en considération tous les segments du chemin. La recherche se fait d'une façon différente dans la navigation on-line. Elle se fait localement ou "*par morceaux*", avec

l'avancement du robot. Car, même si elle venait à prendre en considération tous les segments du chemin, l'évaluation des chemins situés dans les zones inexplorées serait fausse. Seuls les portions du chemin contenues dans les régions connues de l'environnement, sont réellement évaluées.

De toute façon cela n'influe en rien la qualité du chemin car le robot ne peut avancer que d'une longueur limitée par cycle d'évolution. Les nœuds (ou les segments) inatteignables, qu'ils soient "bons" ou "mauvais" ne lui seront d'aucune utilité et n'auront que peu pour ne pas dire aucun effet sur son chemin futur, d'autant plus que dès qu'il aurait avancé, l'ensemble des chemins sera (sauf le meilleur) réinitialisé.

Vue que le temps de l'évaluation est assez important (par rapport aux autres opérations de l'algorithme global), et qu'il dépend directement du nombre de nœuds intermédiaires du chemin à évaluer (voir la boucle :« pour chaque segment du chemin », dans l'algorithme (3)), et vue l'espace mémoire nécessaire pour stocker les informations relatives à chaque nœud (segment) issues de l'évaluation, nous avons réduit ces consommations en bornant le champ de travail de l'évaluation à la portion de chemin contenue dans le champs de vision du robot.

Les termes $dist$, $smooth$ et $clear$ du critère $eval_{feas}$ de (IV.6) ont été modifiés eux aussi, comme suit:

$$dist = \sum_{j=1}^{Nu} d(n_j, n_{j+1}) + 0.5d(n_{Nu+1}, n_l) \qquad \text{(IV.39)}$$

où *Nu* est le nombre de nœuds utiles ;ceux qui sont à l'intérieur du cercle de rayon R, le nœud-source y compris.
n_l : le nœud-but (*l* étant la longueur du chromosome).

Le terme $dist$ est égal à la longueur de la portion du chemin contenue dans le champ de vision augmenté de la distance $d(n_{Nu+1}, n_l)$ du dernier nœud contenu dans le champ de vision au nœud-but, multipliée par un coefficient de pondération pris égal à 0.5. Cette distance est ajoutée afin de faire diriger le chemin vers le but et éviter que le robot n'aille se perdre dans la nature.

Les nouvelles expressions de $smooth$ et $clear$ sont données par :

$$clear = \sum_{j=1}^{Nu-1} \omega_j c(n_j) \qquad \text{(IV.40)}$$

$$smooth = \sum_{j=2}^{Nu-1} \omega_j (s(n_j) + \lambda s'(n_j)) \qquad \text{(IV.41)}$$

où $c(n_j)$ est donné par (IV.19).

$s(n_j)$ est donné par (IV. 5)

et $s'(n_j)$ par (IV.23).

ω_j est un coefficient de pondération, conférant aux $c(n_j)$ et aux $s(n_j)+\lambda s'(n_j)$, un poids d'autant plus petits que la profondeur du nœud n_j dans le chemin est grande (i.e. grand indice *j*).

Dans nos simulations, nous avons pris ω_j comme suit :

$$\omega_j = \frac{(Nu - j + 1)^3}{Nu^3} \qquad \text{(IV.42)}$$

Au lieu de prendre le maximum des $c(n_j)$ ou des $s(n_j)+\lambda s'(n_j)$ comme dans l'expression de $clear$ donnée par (IV .2) ou celle de $smooth$ donnée par (VI.20), respectivement, nous prenons la somme pondérée, donnant aux premiers nœuds le plus d'importance. Car dans la navigation on-line, l'accès aux différents nœuds du chemin est ordonné, et est conditionné par l'accès aux nœuds qui les précèdent . Il est donc inutile que les troisième et quatrième segments du chemin proposé soient assez éloignés des obstacles, alors que le premier en est très proche, ou alors que le chemin présente des angles réalisables dans l'ensemble alors que le premier est très aigu : le robot ne pourrait s'engager sur de tels chemins.

Le fait de juger un chemin d'après ses segments les plus défavorables, comme c'est le cas en (IV.2) et en (IV.3), garantie certes, que l'ensemble des segments ne pourraient être pire que ceux-là, et particulièrement les premiers. Mais nous ne voulons pas condamner un chemin pour une certaine portion qui ne serait pas assez sûre ou irréalisable, pourvu qu'elle se situe le plus loin possible

du début du chemin. Nous voulons en fait, donner au robot une chance d'avancer et vite sur la bonne partie du chemin jusqu'à rencontrer la mauvaise (si elle existe et est accessible), et l'évolution se charge, seulement là, d'améliorer le chemin restant.

Il faut noter qu'une telle transformation de *smooth* et *clear* est une conséquence de l'imposition de conditions dans le point 1. et de l'autorisation à avancer plus d'un pas par cycle d'évolution dans le point 4.

L'algorithme de navigation en temps réel est donné par l'algorithme (13).

Algorithme (13)
Online
Début
N_fail=0 ;
1. *Chargement de toutes les données comme dans l'algorithme offline (sauf la carte des obstacle).*
2. *Construction de la carte de l'environnement à partir des informations captées dès la position-départ.*
3. *Tant que la position courante du robot n'est pas au point-but,*
   - 3.1. *Si le nombre d'échecs N_fail>N ou N_fail=0, alors :*
     - a. *Initialisation de la population (le premier chemin n'est initialisé que dans le cas où N_fail>N).*
     - b. *Remise à jour de la carte de l'environnement du robot.*
     - c. *Classement des chemins du meilleur au pire, ainsi que toutes les données qui leur sont relatives (issues de l'étape d'évaluation, voir algorithme(2)).*
     - d. *Initialisation des index de performance et des probabilités : p_i=1/8.*

     Fin si
   - 3.2. *Pour chaque période d'évolution,*
     - 3.2.1. *Pour chaque génération de la période,*
       - a. *Choisir un opérateur op_i avec une probabilité p_i.*
       - b. *Tant que op_i n'est pas applicable, refaire le choix.*
       - c. *Appliquer l'opérateur :*
         - *Un chemin dans la population est choisi.*
         - *Un nouveau chemin est généré.*
         - *Le nouveau chemin est évalué.*
       - d. *Le pire chemin est remplacé par le nouveau, ainsi que les données qui leurs sont relatives.*
       - e. *Calcul des paramètres qui rentrent dans l'adaptation des probabilités op_i.*

       Fin de boucle.
     - 3.2.2. *Adaptation des probabilités op_i.*

     Fin de boucle
   - 3.3. *Si le meilleur chemin est faisable, alors*
     - 3.3.1. *Mettre le pointeur sur le nœud source locale (position courante du robot) : j=1.*
     - 3.3.2. *Tant que le nœud pointé n_j est à l'intérieur du cercle de vision,*
       - a. *Si la distance entre le nœud pointé et le nœud suivant est inférieure à d_{min} et $g_j^* > \tau$ et $\theta_j^* > \theta_{braq}$ et $\alpha_j^* < \alpha_{\lim}$, alors :*

         Supprimer le nœud n_{j+1} (donc tous les nœuds n_{j+k+1} deviennent n_{j+k} pour k=1,l-1 avec l : longueur du chromosome).

         Sinon

         Si $g_j > \tau$ et $\theta_j > \theta_{braq}$ et $\alpha_j < \alpha_{\lim}$, alors :

         Se brancher à b.

         Sinon

         Incrémenter le nombre d'échecs : N_fail=N_fail+1.

         Sortir de la boucle 3.3.2. et se brancher à 3.1.

         Fin de si

         Fin de si
       - b. *Si la distance entre n_1 et n_{j+1} est inférieure à k_{max}, alors :*

         Incrémenter le pointeur : j=j+1.

         Sinon

         Créer un nouveau nœud sur l'intersection du segment $\overline{n_j n_{j+1}}$ avec le cercle de centre n_j et de rayon k_{max}.

         Insérer ce nouveau nœud entre n_j et n_{j+1}.

         Remettre le nombre d'échecs à zéro : N_fail=0 ;

         Sortir de la boucle 3.3.2. et se brancher à 3.1.

         Fin de si

       Fin de boucle (tant que 3.3.2.)

     Sinon

     Se brancher à 3.2.

     Fin de si

Fin de boucle (tant que 3.)
fin

IV.3. Planification/Navigation en temps réel dans un environnement dynamique :

Afin d'adapter l'algorithme à un environnement dynamique, dans lequel le robot aura à rencontrer des obstacles en mouvement, nous avons apporté des modifications à l'algorithme de planification/navigation développé précédemment pour un environnement statique,

En effet, le problème a été traité jusqu'ici dans un cadre strictement *géométrique* en terme d'équations de segments de droites et leurs interactions dans un plan xy. Il faudrait donc donner une dimension temporelle au problème, c'est à dire, pouvoir déterminer à chaque instant t l'emplacement du robot (sur son chemin solution) ainsi que celui des obstacles et inversement.

Pour cela, nous considérons deux modèles :

- Celui associés aux obstacles mobiles, a cette catégorie nous affectons une vitesse de translation et de rotation et une configuration initiale.
- Quant au robot, la chose n'est pas aussi simple.

En effet, on pourrait croire au premier abord qu'ayant défini un chemin solution le menant de la source au but, on pourrait connaître la position du robot à tout instant t par simple définition de la vitesse d'évolution de celui ci sur ce chemin.

Rappelons que le chemin fourni par l'algorithme évolutif est une séquence de points de passage reliés par des segments de droites, évitant les obstacles et répondant aux critères *géométriques :* longueur, dégagement par rapport aux obstacles et ouverture des angles formés par les segments. Ce serait effectivement le cas si on laissait le robot se déplacer sur un tel chemin en segments en exécutant la séquence :

- accélération
- translation uniforme (éventuellement, pour économiser de l'énergie)
- décélération
- arrêt
- rotation, (changement de direction passant d'un segment à l'autre).
- arrêt
- accélération etc...

Pour éviter une telle séquence qui est aussi désavantageuse en temps qu'en consommation d'énergie, il existe des techniques de lissage

de trajectoire présenté précédemment. Nous en avons choisi celle que nous avons trouvée intéressante par le fait qu'elle permet de garder une vitesse tangentielle constante et la même tout au long de la trajectoire aussi bien le long des portions droites que sur les courbes joignant ces portions. Ce caractère apporte une économie en temps, surtout si l'on prend cette vitesse égale à la vitesse maximale de déplacement du robot. Une vitesse angulaire nulle le long des portions droites apparaît sur les courbes joignantes, croissant et décroissant d'une façon continue jusqu'à redevenir nulle sur la prochaine portion droite. Créant ainsi une courbure continue le long de toute la trajectoire, chose très recherchée pour avoir une exécution de trajectoire aussi précise que possible.

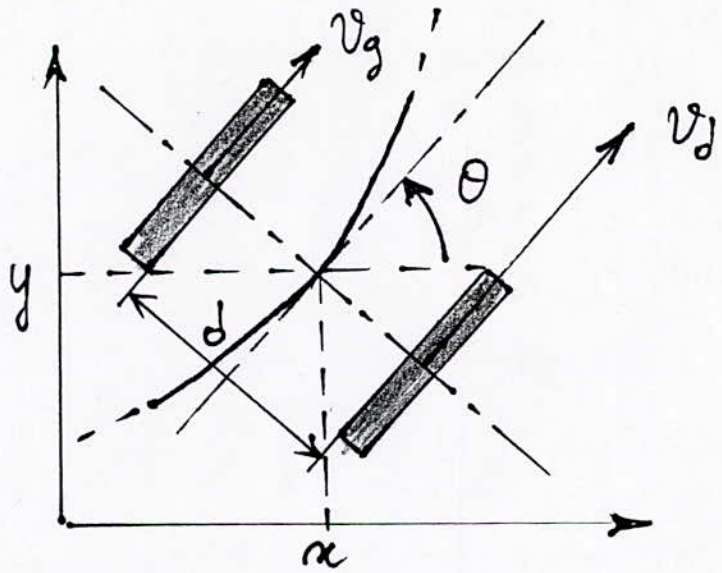
Une fois une telle trajectoire fonction du temps définie, nous pouvons déterminer la position du robot à chaque instant et la comparer aux emplacements des obstacles au même moment, afin de tester et éviter les collisions.

V.3.1. Modélisation du robot mobile :

Considérons un robot de la famille Hilare [34] dont le système de locomotion est composé d'une paire de roues moto-directrices (*driving wheels*) parallèles.

- La longueur de l'axe maintenant ces deux roues est noté d (distance entre ces dernières).
- Le point de référence du robot est pris au milieu de cet axe dont les coordonnées par rapport à un référentiel de base sont notées (x, y).
- La direction du robot définie par l'angle formé entre la médiatrice de l'axe du robot et l'axe des abscisses du référentiel de base, est notée θ c'est aussi la direction de la tangente à la trajectoire au point de référence du robot (puisque la direction des roues est toujours tangente à la trajectoire. La configuration du robot à trois degrés de libertés est alors définie par $(x, y, \theta) \in \Re^2 \times [-\pi, \pi]$.
- Les vitesses de la roue droite et de la roue gauche sont notées respectivement par v_d, v_g.
- Les accélérations de ces deux roues (les entrées du système) sont notées respectivement u_d, u_g. Ces accélérations sont bornées : $|u_d| \leq a, |u_g| \leq a$

On peut traduire les accélérations u_d, u_g en termes d'accélérations tangentielles et angulaire $\dot{v}, \dot{\omega}$ du point de référence du mobile, s'exprimant d'après la loi de Newton comme suit :



ce qui donne :

$$\begin{aligned} \dot{v} &= \gamma = \frac{u_d + u_g}{2} \\ \dot{\omega} &= \ddot{\theta} = \frac{u_d - u_g}{d} \end{aligned} \qquad \text{(VI.44)}$$

Le modèle du robot est alors donné par :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_d \\ \dot{v}_g \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(v_d + v_g)\cos\theta \\ \frac{1}{2}(v_d + v_g)\sin\theta \\ \frac{1}{d}(v_d - v_g) \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} u_d + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_g \qquad \text{(IV.45)}$$

IV.3.2. La commande :

Le Principe du Maximum donné par la théorie de la commande optimale [35], impose d'appliquer au processus une commande de valeur maximale, provoquant l'accélération maximale des roues : $|u_d| = |u_g| = a$. L'imposition de l'une des deux configurations d'accélération suivantes : $u_d = -u_g$ et $u_d = u_g$ fait décrire au point de référence du robot deux sortes de courbes : clothoïdes et anticlothoïdes respectivement. Nous ne nous intéresserons ici qu'aux clothoïdes

(brièvement décrites dans le chapitre II : exécution de trajectoire) qui sont générées lors de l'application de commandes constantes vérifiant : $u_d(t) = -u_g(t) = \pm a = Cte$.

IV.3.3. Description cinématique de la trajectoire ;

L'accélération tangentielle le long du mouvement produit par la commande décrite ci-dessus est nulle : $\dot{v}(t) = \frac{1}{2}(u_d(t) + u_g(t)) = 0$,

ce qui donne, par intégration, une vitesse tangentielle constante : $v(t) = v_0$,

qui intégré à son tour donne l'expression de la longueur de l'arc (l'abscisse curviligne) : $s(t) = v_0 t$.

Il résulte aussi d'une telle commande que l'accélération angulaire est constante : $\dot{\omega} = \frac{1}{d}(u_d(t) - u_g(t)) = \pm\frac{2}{d}a$, ce qui donne une vitesse angulaire linéaire dans le temps : $\omega = \pm\frac{2}{d}at + \omega_0$. L'orientation est alors obtenue en continuant l'intégration :

$$\theta(t) = \pm\tfrac{1}{d}at^2 + \omega_0 t + \theta_0. \qquad \text{(IV.46)}$$

La courbure $k(t)$ qui est le rapport de la vitesse angulaire à la vitesse tangentielle s'écrit alors , en fonction du temps :

$$k(t) = \pm\frac{2a}{dv_0}t + \frac{\omega_0}{v_0} \qquad \text{(IV.47)}$$

puis en fonction de la longueur de l'arc :

$$k(t) = \pm\frac{2a}{dv_0^{\,2}}s(t) + \frac{\omega_0}{v_0} \qquad \text{(IV.48)}$$

On retrouve bien, par cette dernière équation, la caractéristique de linéarité de la courbure par rapport à la longueur de l'arc de la clothoïde (voir § exécution de trajectoire), et dont la constante caractéristique est :

$$k_c = \pm\frac{2a}{dv_0^{\,2}}. \qquad \text{(IV.49)}$$

Il a été remarqué qu'une telle expression de la courbure permet à la clothoïde d'unir d'une façon lisse et continue une droite (de courbure nulle) à une courbe de courbure non nulle et inversement, en prenant $u_d(t) = -u_g(t) = a$ (*1ère configuration.*), puis $u_d(t) = -u_g(t) = -a$ (*2ème configuration*), ou inversement.

Pour unir deux droites de directions différentes, on peut joindre deux clothoïdes de deux manières différentes :

- *Première méthode* : On peut passer d'une courbure nulle $k=0$ à une courbure non nulle $k=k_{max}$ en appliquant la première configuration d'abord, et ce dans le but d'avoir la pente de la courbure (qui est la constante caractéristique) positive, ce qui donne une courbure croissante en fonction de la longueur de l'arc.

  Puis il faut repasser de cette même courbure k_{max} à la courbure nulle qui est celle de la prochaine portion droite, en appliquant la contrainte de la seconde configuration, de sorte à ce que la constante caractéristique soit négative dans l'expression de la courbure qui est décroissante cette fois-ci, en fonction de l'abscisse curviligne.

  De cette façon, la courbure décrit une fonction triangulaire le long des deux clothoïdes liées d'une façon continue et de constantes caractéristiques égales et opposées.

  On obtient ainsi une primitive pour le lissage de trajectoire qui consiste en un virage classique arrondissant l'angle vers l'intérieur. Le mobile suit cette portion de courbe en tournant dans le sens de rotation défini par l'angle que forme le prochain segment du chemin par rapport au précédent.

- *Deuxième méthode :* On peut aussi commencer par appliquer d'abord la seconde configuration qui fera passer la courbure de 0 à $-k_{max}$, puis appliquant la première configuration pour passer de $-k_{max}$ à 0.
  De cette façon, pour effectuer cette portion de courbe, le mobile tournera dans le sens inverse défini par l'angle formé par les deux segments en décrivant une boucle vers l'extérieur de l'angle.

En supposant que l'orientation et la vitesse angulaire initiales sont nulles : $\omega_0 = \theta_0 = 0$, les coordonnées du point de référence du robot décrivant une clothoïde de constante caractéristique k_c définie dans (IV.49), sont données par (voir § exécution de trajectoire, toujours) :

$$x(t) = x_0 + \sqrt{\frac{\pi}{k_c}} CF\left(\sqrt{\frac{2a}{d\pi}} t\right) \qquad \text{(IV.50)}$$

$$y(t) = y_0 + \sqrt{\frac{\pi}{k_c}} SF\left(\sqrt{\frac{2a}{d\pi}} t\right) \qquad \text{(IV.51)}$$

et la direction du robot est donnée par :

- l'instant t_j de passage du robot par ce point-là de la trajectoire lissée, que nous considérerons comme le point correspondant au nœud j en question.

Expliquons ces deux points : l'algorithme évolutif, tel qu'il a été décrit, ne peut manipuler que des chemins définis par des nœuds, des segments reliant ces nœuds et les angles formés par ces segments, il lui est impossible de travailler avec des courbes qui sont en fait des suites continues de nœuds infiniment proches.

Or ces courbes définissent la vraie trajectoire du mobile donnant sa position dans le temps ainsi que dans l'espace. Nous avons donc besoin de faire une correspondance entre la courbe et le chemin défini par la séquence de nœuds avec laquelle travaille l'algorithme. Les deux sont confondues le long des segments et ne se séparent qu'aux sommets des angles, et dont le maximum est le nœud; le point de la courbe le plus éloigné correspond à son intersection avec la bissectrice, axe de symétrie de cette portion de courbe.

Nous avons décidé d'imposer un seul sens de rotation durant toute la trajectoire, disons dans le sens contraire aux aiguilles d'une montre. Ainsi, lorsque l'angle formé par deux segments est positif, le mobile tournera dans le sens de l'angle décrivant un arrondi vers l'intérieur de l'angle, et dans le cas où l'angle est négatif, le mobile tournera toujours dans le sens positif (contraire au sens de l'angle cette fois-ci) en décrivant une boucle vers l'extérieur de l'angle. Ces deux virages sont obtenus en combinant deux clothoïdes de constantes caractéristiques égales et opposées comme décrit plus haut.

Calculons à présent le temps nécessaire pour passer d'un nœud du chemin à un autre :

Sachant que la variation de la direction du robot le long de la courbe joignant deux segments (au nœud j) est exactement l'angle α_j formé par ces deux segments avec un signe adéquat, il est évident qu'au point c, le milieu de cette courbe, l'orientation du robot sera égale à $\frac{\pi - \alpha_j}{2}$.

La moitié de la durée tc_j, nécessaire, pour traverser la portion de courbe arrondissant le sommet au nœud j, est obtenu en remplacant $\theta(t)$ par $\frac{\pi - \alpha_j}{2}$ dans (IV.52).

$$tc_j = \sqrt{\frac{d(\pi - \alpha_j)}{2a}}$$

En remplaçant l'expression de tc_j dans (IV.51) et (IV.52) on obtient les coordonnés (x_c, y_c) du point c dans un repère local lié à la courbe, dont l'axe x est le prolongement du premier des deux segments (formant l'angle à lisser) et l'origine est le point de jonction de la portion de courbe avec ce même segment.

La distance entre le nœud n_j et l'origine du repère local de la courbe, sera notée Δd_j. On peut facilement vérifier que :

$$\Delta d_j = x_c + y_c \cot\tfrac{\alpha}{2} \quad \text{si } \alpha_j > 0$$
$$\Delta d_j = 0 \quad \text{si } \alpha_j < 0$$

car dans le cas de la boucle extérieure, l'origine de la courbe est confondue avec le nœud.
l'instant absolu t_j de passage du mobile par le point c relatif au nœud n_j peut s'exprimer de la manière suivante

$$t_j = t_{j-1} + tc_{j-1} + tc_j + (d(n_{j-1}, n_j) - \Delta d_{j-1} - \Delta d_j)/v_0 \quad \text{(IV.53)}$$

Nous avons introduit cette expression de t_j dans la fonction d'évaluation afin d'affecter une composante temporelle à chaque nœud n_j. Il est possible à présent d'évaluer la rapidité des chemins, ce qui introduit un aspect de plus à minimiser dans le critère d'évaluation :

$$time = C_t t_{j=nu} \quad \text{(IV.54)}$$

nu étant le nombre de nœuds utiles qui sont pris en considération lors de l'évaluation (ceux qui sont à l'intérieurs du cercle de vision, voir navigation on-line), et C_t est une constante.

Nous connaissons aussi la commande appliquée au robot :

1. Sur la portion de la courbe de lissage, la commande est du type bang-bang : à chaque roue est appliquée une accélération maximale opposée à celle appliqué à l'autre roue, ceci lors du mouvement sur la première clothoïde (première moitié de la courbe), puis les deux accélérations (des deux roues) s'inversent et restent constantes (de valeur maximale, toujours) durant le mouvement sur la seconde clothoïde.

2. Le mouvement le long des segments de droites, quant à lui, étant uniforme, est à accélération donc à commande nulle.

On peut donc introduire aussi l'aspect dynamique dans l'évaluation des chemins, en vue de minimiser la consommation de l'énergie mécanique. De cette façon, notre fonction d'évaluation est complète et considère l'optimisation du chemin sous tous ses aspects.

L'énergie consommée est proportionnelle à la longueur de l'arc formé par les deux clothoïdes (puisque, comme il vient d'être dit, l'accélération y est constante, et nulle ailleurs). L'aspect quantifiant l'énergie consommée s'écrit donc :

$$energy = C_e \sum_{j}^{nu} 2tc_j \qquad \text{(IV.55)}$$

Il est à noter que lors de l'introduction des facteurs temps et énergie, dans le critère pondéré, il faut uniformiser les unités , les rendant compatibles avec les unités des autres aspects de la fonction d'évaluation, qui sont des distances ou quelque chose qui s'en rapproche. On multiplie donc le temps par la vitesse v_0 (qui n'influence d'ailleurs en rien l'évaluation puisqu'elle est constante). Quant à l'énergie, elle est mesurée en terme de longueur de l'arc.

La fonction d'évaluation devient :

$$eval_{feas} = v_d dist + v_c clear + v_s smooth + v_t time + v_e energy \qquad \text{(IV.56)}$$

IV.3.5. Evitement des obstacles mobiles :

Considérons un environnement parfaitement inconnu à priori, et qui comporte éventuellement des objets en mouvement (également inconnus), sur lesquels on émet les hypothèses suivantes :

- comme les obstacles fixes les obstacles mobiles sont polygonaux.
- leur mouvement est parfaitement prédictible, c'est à dire qu'on peut déterminer leurs positions respectives à n'importe quel instant tout au moins dans le futur proche.

Dans l'algorithme de navigation et planification on-line que nous avons développé jusqu'à présent, l'environnement avoisinant est analysé à chaque déplacement du robot et le résultat de cette analyse est utilisé pour la planification de déplacement futur. De même, dans le nouvel

algorithme adapté aux obstacles mobiles, l'environnement est analysé à chaque pas.

Seulement cette fois, la vitesse et l'accélération des obstacles présents à l'intérieur du cercle de vision sont mesurées. Si elles sont nulles, alors, la recherche de chemin se fait par la procédure on-line normale, si non, la procédure d'évitement d'obstacle devient une action de prédiction –vérification, i.e., la carte de l'environnement dans laquelle se fera la recherche de chemin (par l'algorithme évolutif toujours) n'est plus la carte remise à jour à base d'informations fournies par les capteurs, mais une carte prédite à un instant $t + \Delta t$ au futur, t étant l'instant présent.

La difficulté réside à choisir l'instant de prédiction. Si les segments du chemin générés par l'algorithme évolutif étaient de longueur prédéterminée, nous aurions su à quel instant approximatif le mobile atteindra le nœud prochain. Nous aurions alors, chargé la carte du robot par la configuration des obstacles comme elle serait à ce moment-là, et Δt aurait été, dans ce cas-là, le temps de déplacement jusqu'à ce nœud, à peu de chose prés. Seulement nous n'avons, à priori, aucune idée de l'éloignement du prochain nœud du chemin qui sera généré par le processus évolutif sur la base de l'environnement prédit. Nous avons donc décidé de prédire à un instant assez proche, puis lorsque le chemin dans l'environnement prédit est généré, sa validité est remise en question. Voici comment procède l'algorithme :

- les coordonnées des sommets des obstacles ou des parties d'obstacles connus par le robot (qui sont ou ont été, à un moment donné, à l'intérieur de son cercle de vision) sont calculés pour l'instant $t + \Delta t$, à l'aide des vitesses et accélérations de ces obstacles que le robot aurait évalué préalablement (la détermination des équations de mouvement des objets de l'environnement est un problème à part que nous ne traiterons pas ici).
- la carte du robot est chargée par la configuration calculée.
- Le processus d'évolution est lancé à la recherche du meilleur chemin, sur la base de la fonction d'évaluation décrite plus haut qui affecte à chaque nœud n_j le temps t_j de passage du robot par ce nœud.
- La réalisabilité du premier segment $\overline{n_1 n_2}$ du meilleur chemin est testée pour une configuration d'obstacles à l'instant t_2 du passage du robot au nœud n_2. Pour que le segment soit réalisable il ne suffit pas d'avoir une configuration sûre (exempte de collision) au moment de départ du premier nœud et au moment d'arrivée au nœud d'après, il faut aussi s'assurer que les trajectoires des obstacles mobiles ne croisent pas le segment dans l'intervalle de temps entre les deux

moments considérés. C'est pour cette raison que nous avons choisi un temps de prédiction assez faible.

- Si le premier segment est réalisable, on peut passer au test du segment d'après, (s'il ne sort pas du cercle de rayon $k_{\max}$ (pas maximal que peut effectuer le robot, à ne pas confondre avec son champ de vision).
- Dans le cas contraire, le processus d'évolution est relancé pour trouver un chemin meilleur après avoir diminué le temps de prédiction.

IV.4. Simulations et résultats :

Nous avons exécuté les algorithmes pour différentes configurations d'obstacles.

La complexité de l'algorithme fait que beaucoup de paramètres sont à ajustaer, notamment :

- Taille de la population (variant entre 20 et 50 chromosomes).
- Nombre de générations (variant entre moins de 100 et 500 générations).
- Intervalle d'adaptation des probabilités de choix des opérateurs et la fréquence de cette adaptation (faisant l'adaptation toutes les 20 à 100 générations, de 3 à 10 fois durant le processus d'évolution global).
- Nombre maximal de nœuds dans la population initiale, qui dépendant de la complexité de l'environnement (de 10 à 20 nœuds intermédiaires).
- Coefficients des fonctions d'évaluation. Paramètres de pénalisation et autres grandeurs comme la distance de sécurité, l'angle de braquage minimal...
- Seuil d'activation de certains opérateurs.

La gestion de tous ces paramètres influe considérablement sur l'efficacité de l'algorithme.

Parmi les différents jeux de ces paramètres, nous avons retenu ceux qui donnent des résultats que nous avons jugé acceptables.

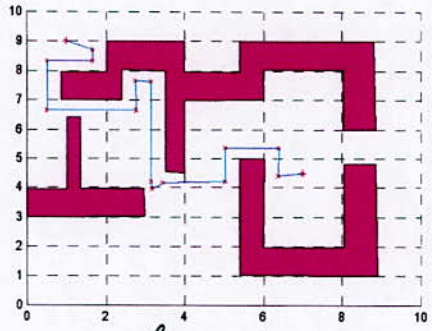


fig. IV.4.1

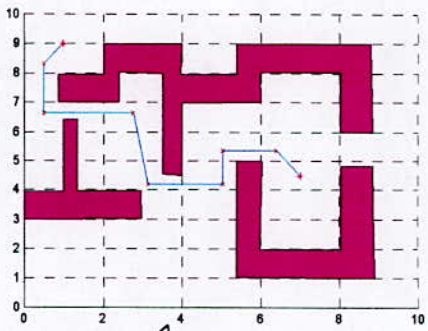


fig. IV.4.2

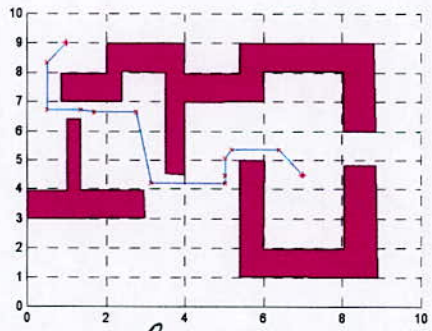


fig. IV.4.3

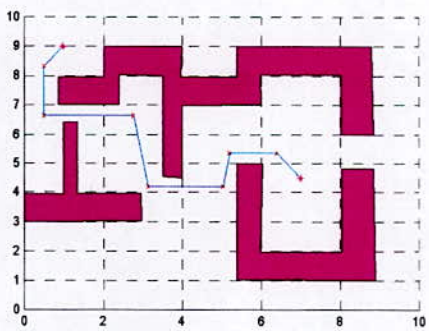


fig. IV.4.4

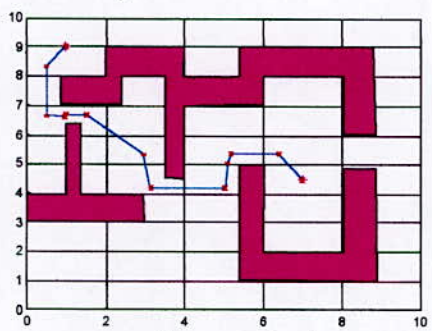


fig. IV.4.5

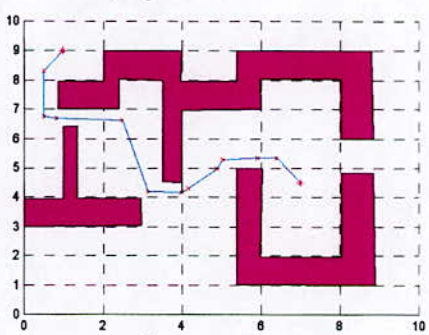


fig. IV.4.6

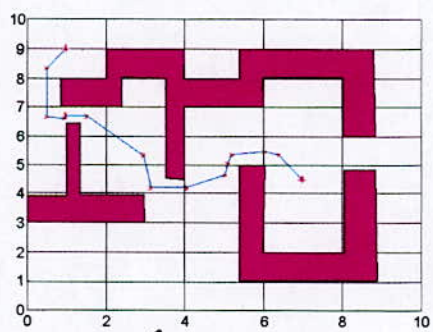


fig. IV.4.7

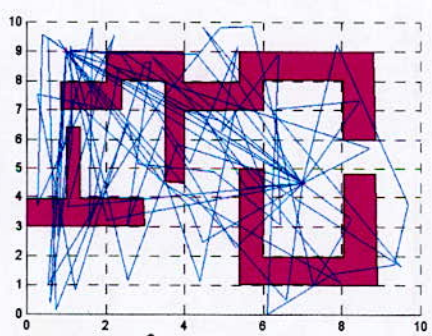


fig. IV.4.8

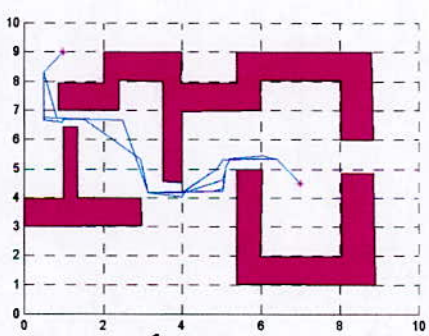


fig. IV.4.9

figure IV.4

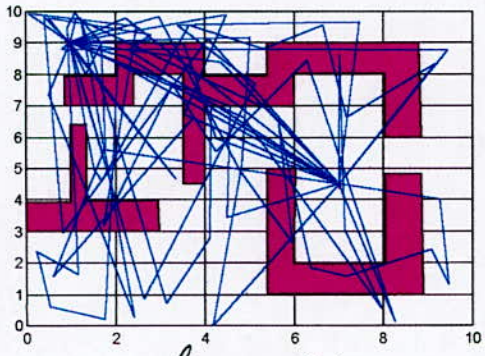


fig. IV. 5. 1

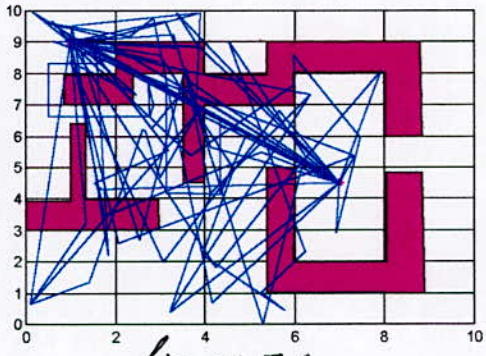


fig. IV. 5. 2

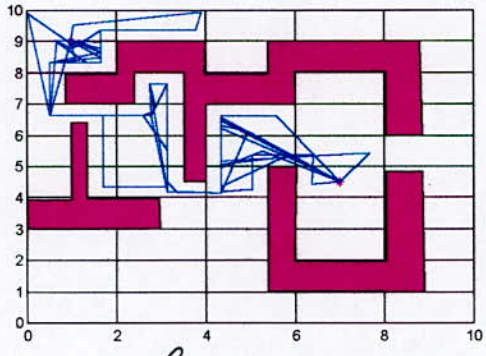


fig. IV. 5. 3

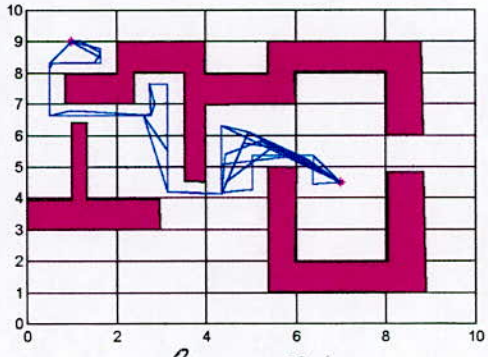


fig. IV. 5. 4

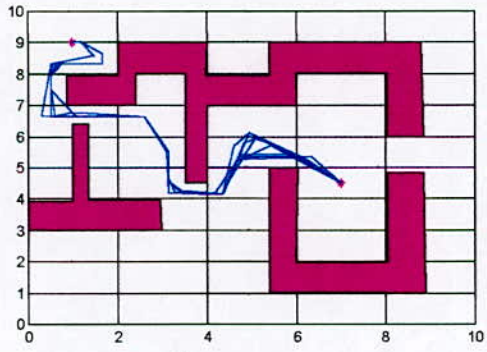


fig. IV. 5. 5

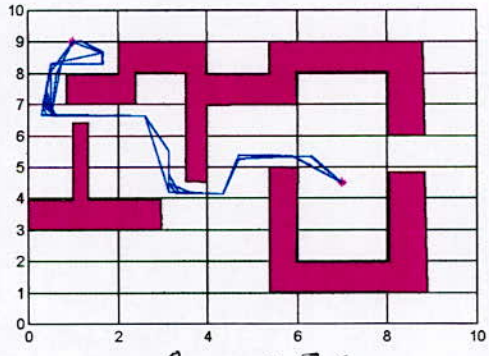


fig. IV. 5. 6

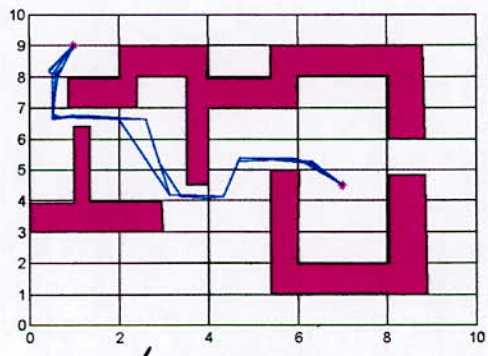


fig. IV. 5. 7

Figure IV. 5

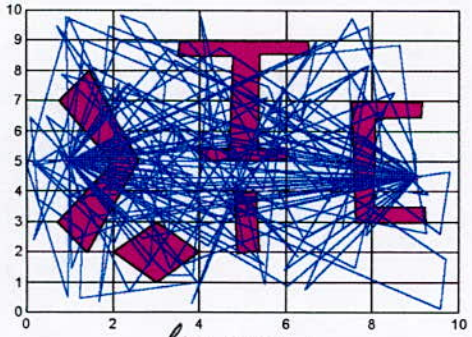


fig. IV.6.1

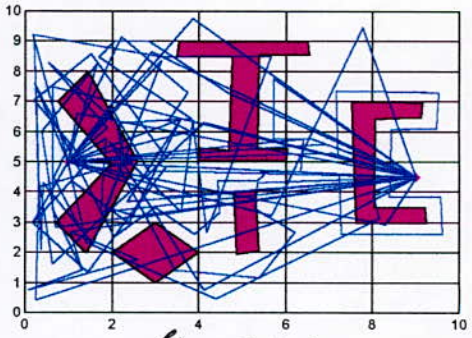


fig. IV.6.2

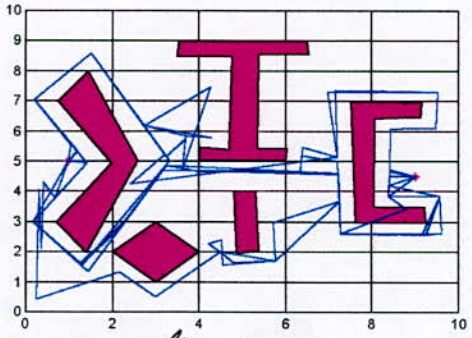


fig. IV.6.3

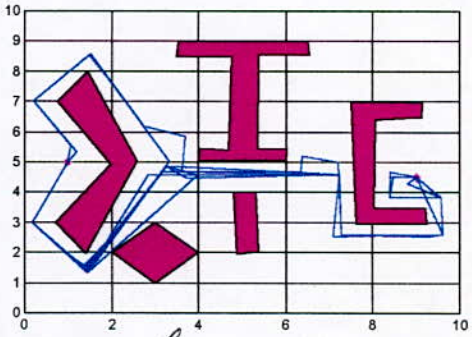


fig. IV.6.4

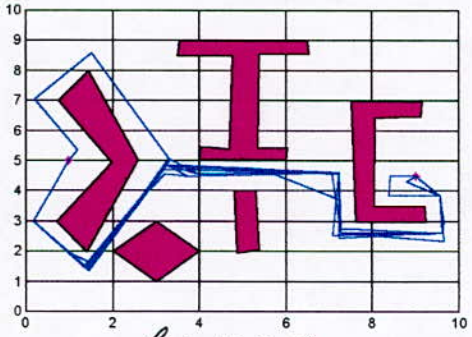


fig. IV.6.5

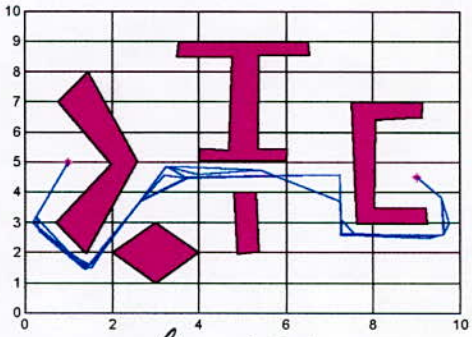


fig. IV.6.6

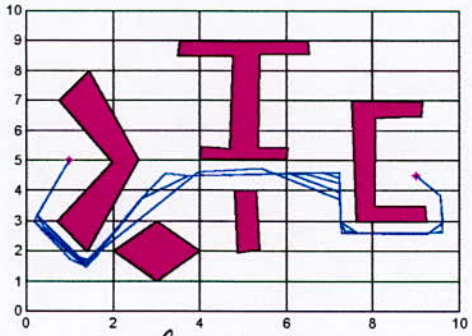


fig. IV.6.7

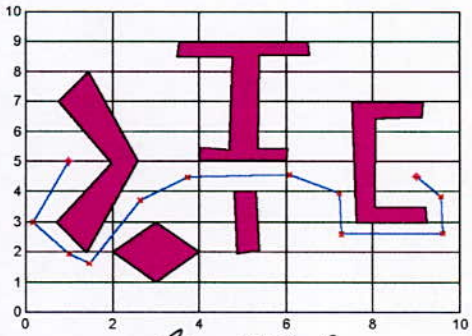


fig. IV.6.8

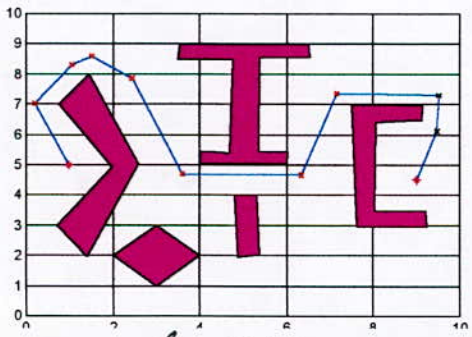


fig. IV.6.9

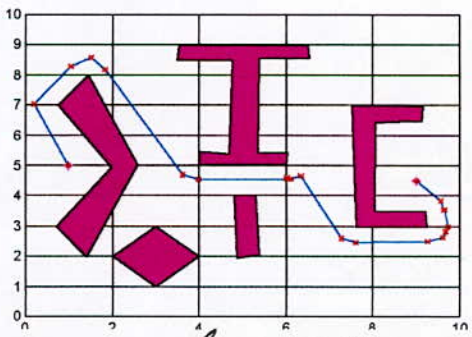


fig. IV.6.10

figure IV.6

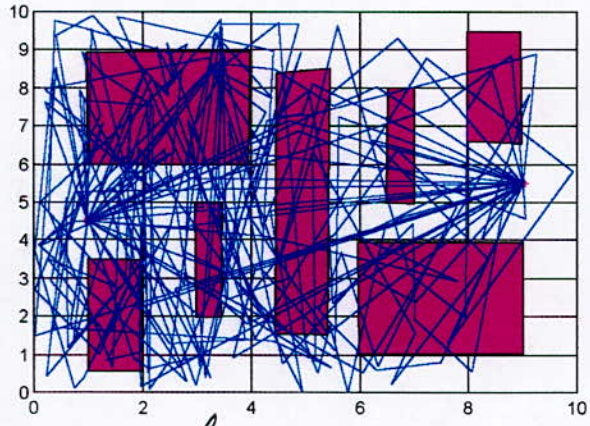


fig. IV. 7. 1

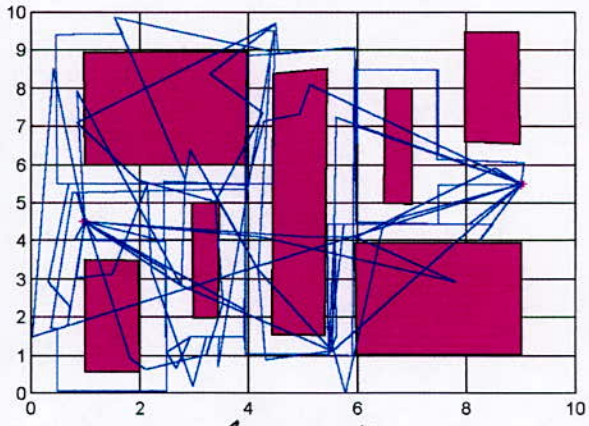


fig. IV. 7. 2

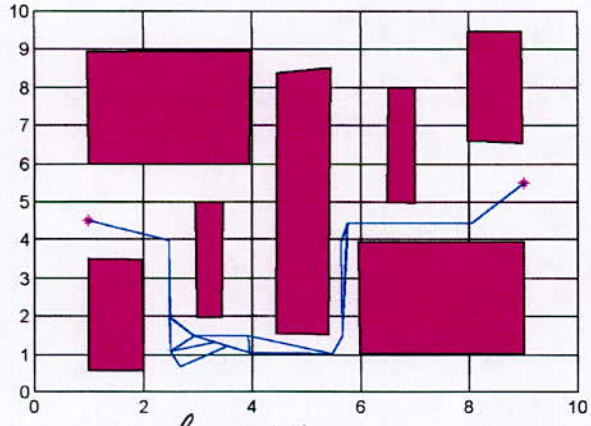


fig. IV. 7. 3

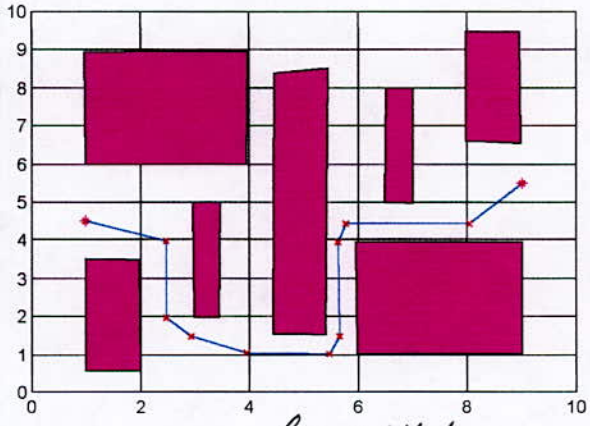


fig. IV. 7. 4

figure IV. 7

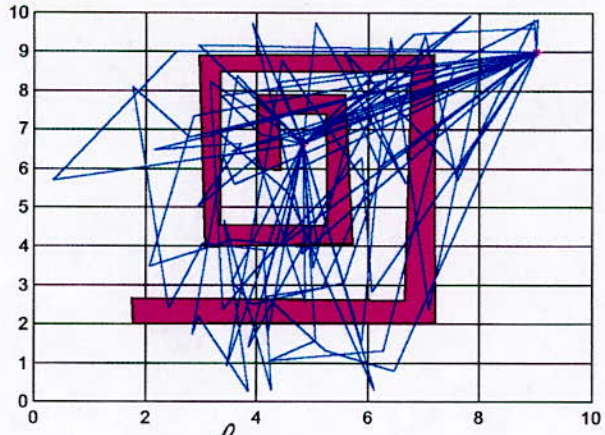


fig. IV.8.1

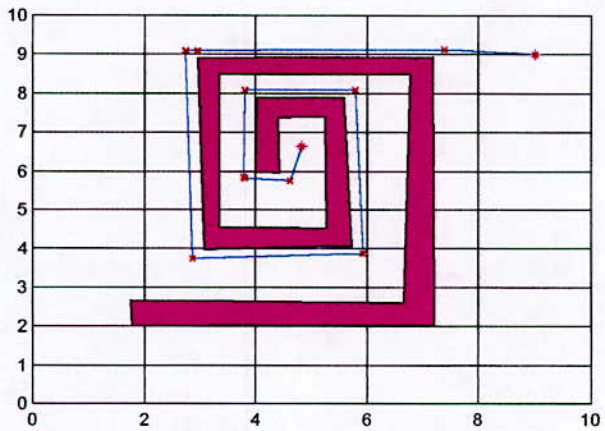


fig. IV.8.2

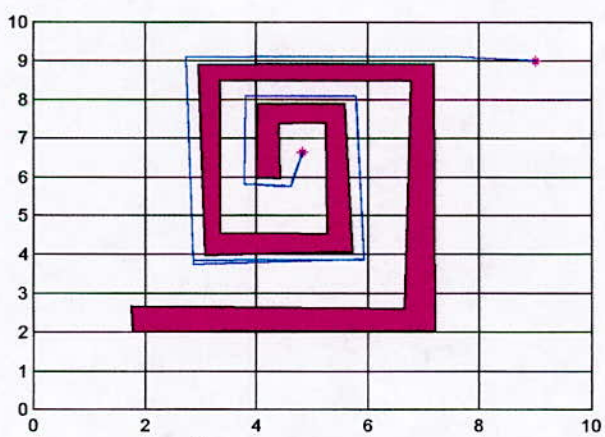


fig. IV.8.3

Figure IV.8

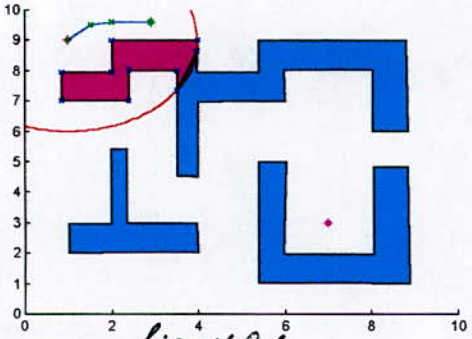


fig. IV.9.1

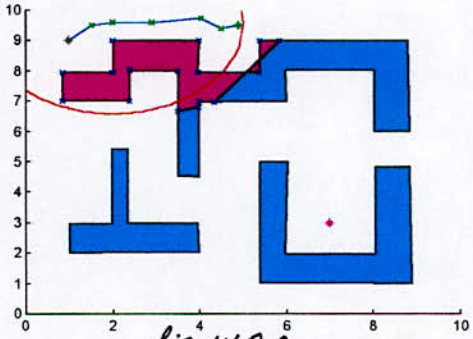


fig. IV.9.2

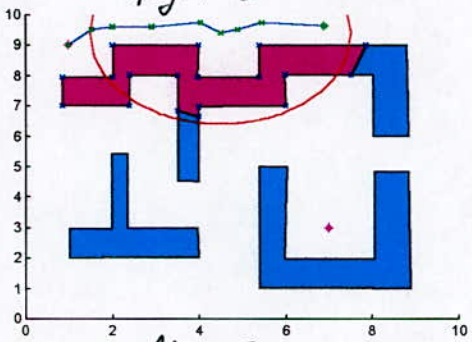


fig. IV.9.3

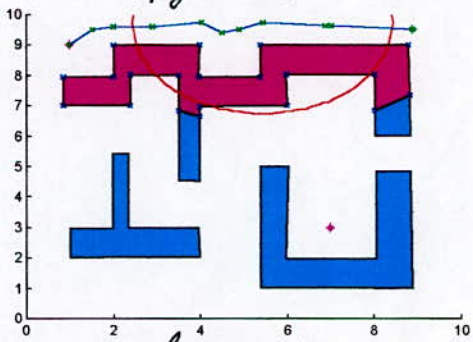


fig. IV.9.4

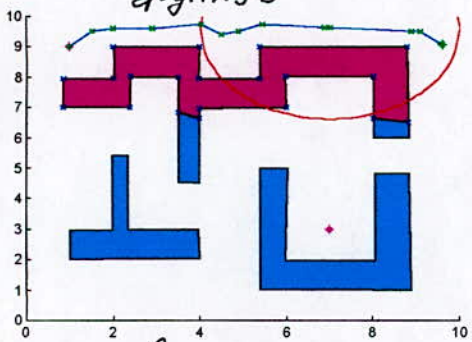


fig. IV.9.5

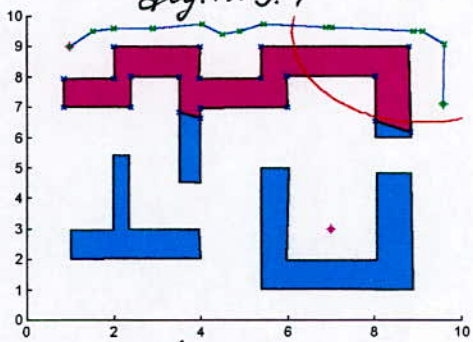


fig. IV.9.6

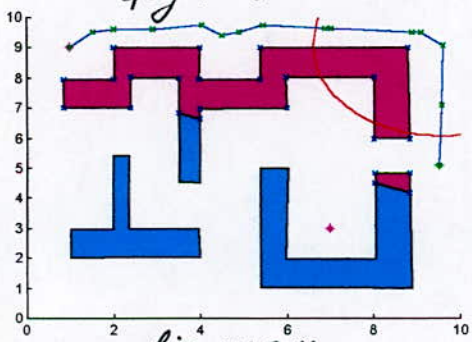


fig. IV.9.7

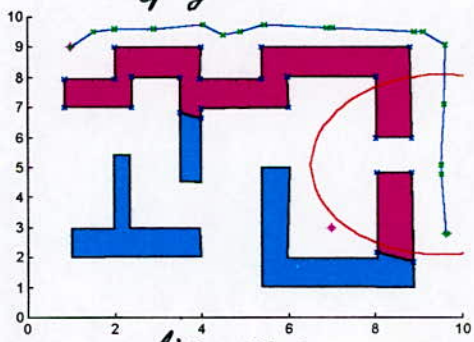


fig. IV.9.8

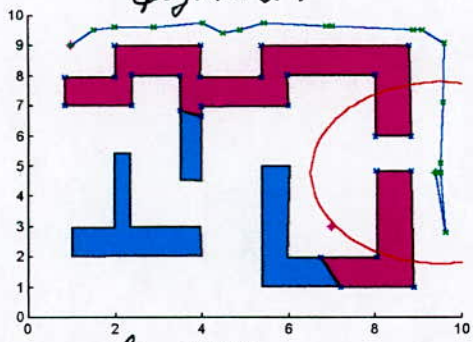


fig. IV.9.9

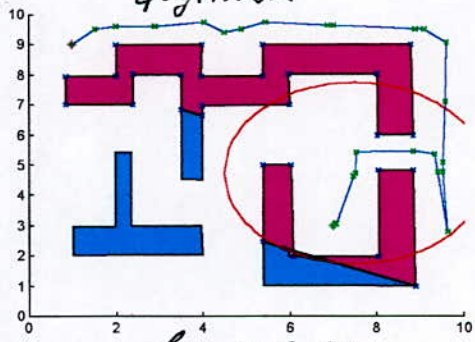


fig. IV.9.10

figure IV.9.

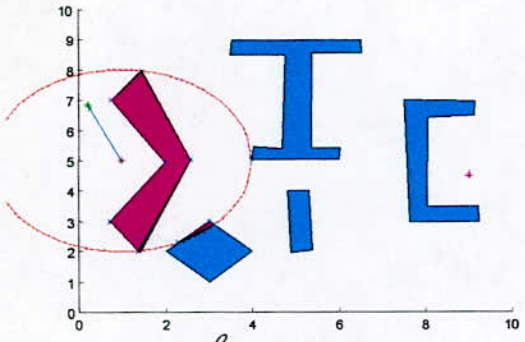


fig. IV. 10. 1

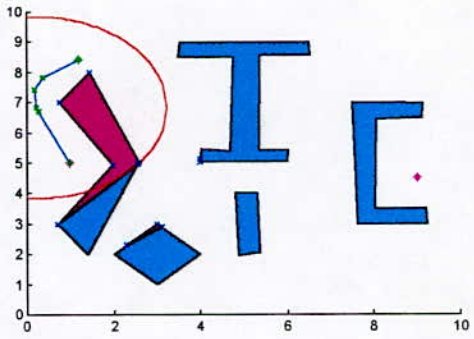


fig. IV. 10. 2

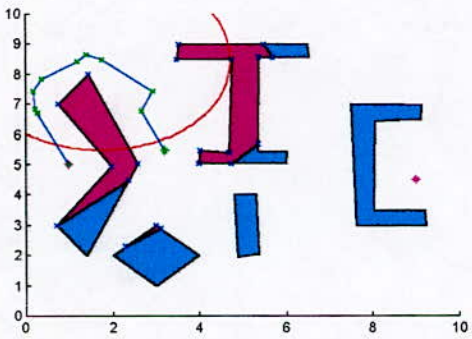


fig. IV. 10. 3

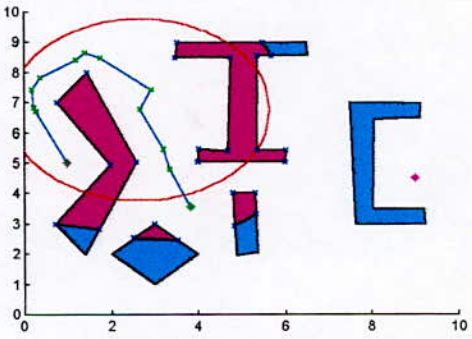


fig. IV. 10. 4

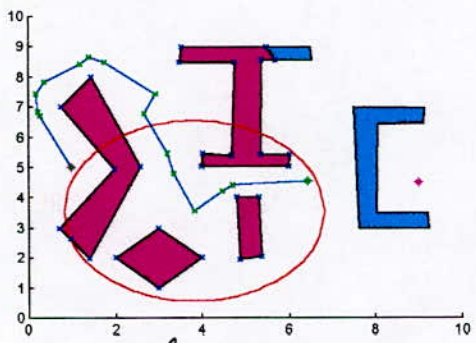


fig. IV. 10. 5

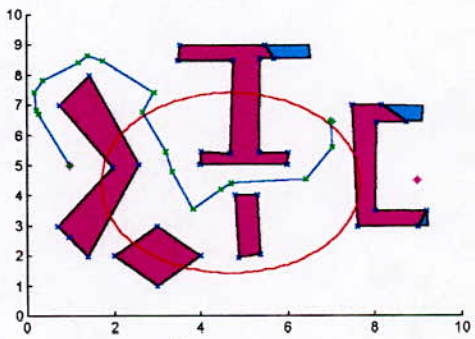


fig. IV. 10. 6

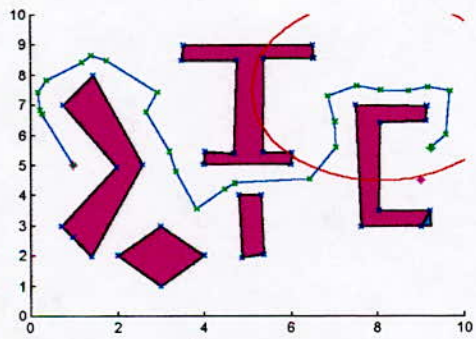


fig. IV. 10. 7

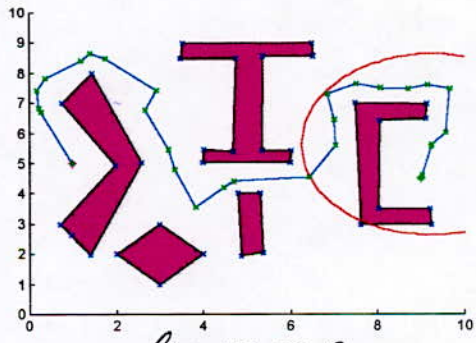


fig. IV. 10. 8

figure IV. 10.

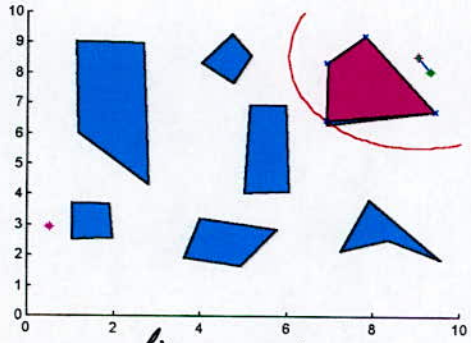


fig. IV.11.1

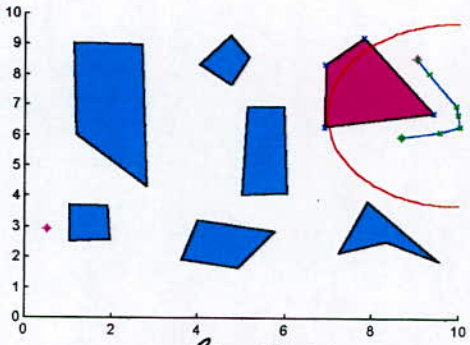


fig. IV.11.2

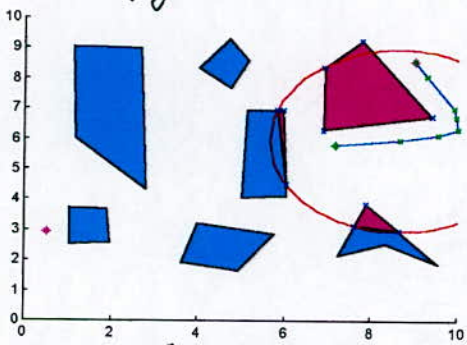


fig. IV.11.3

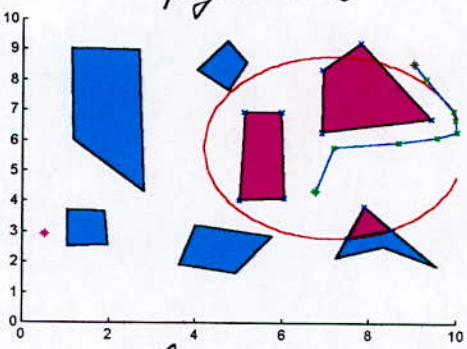


fig. IV.11.4

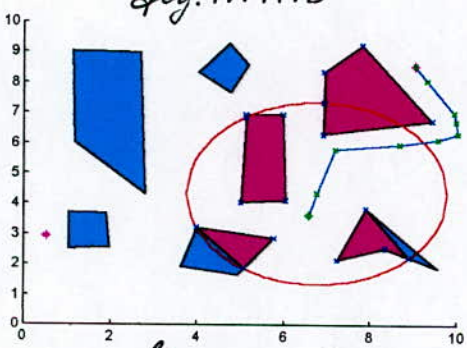


fig. IV.11.5

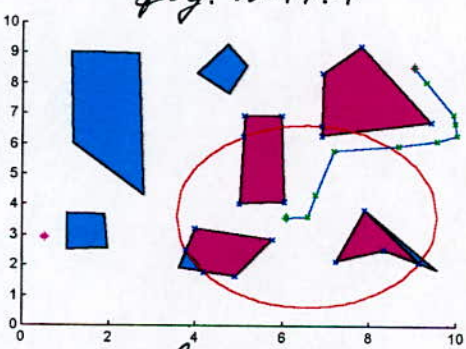


fig. IV.11.6

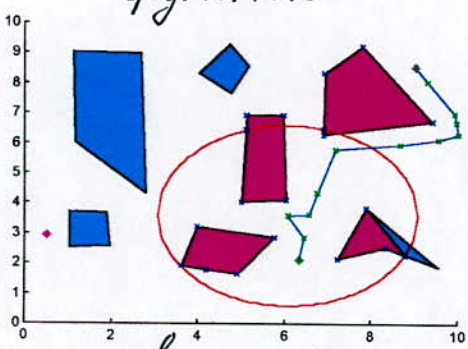


fig. IV.11.7

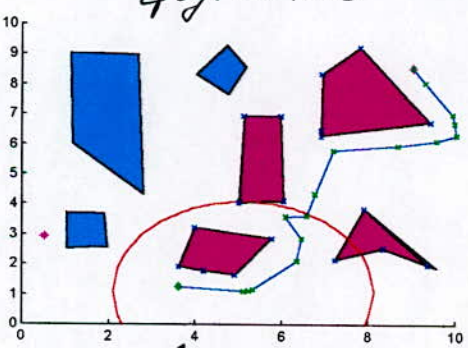


fig. IV.11.8

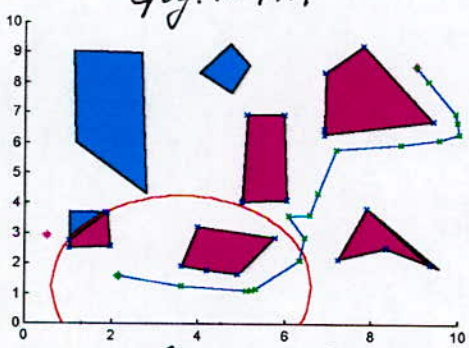


fig. IV.11.9

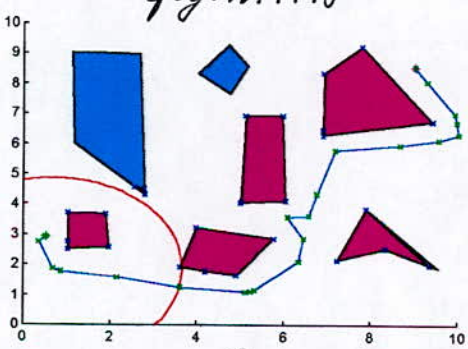


fig. IV.11.10

figure IV.11.

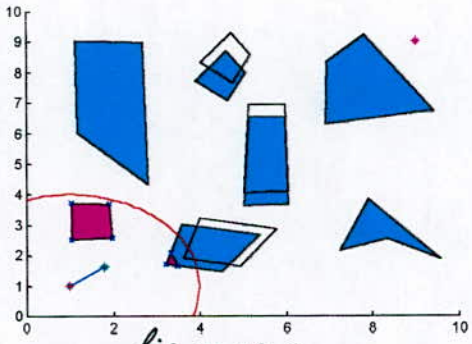


fig. IV. 12. 1

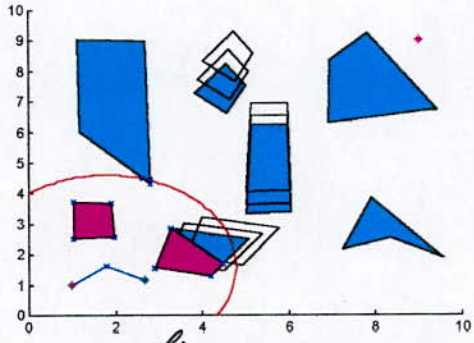


fig. IV. 12. 2

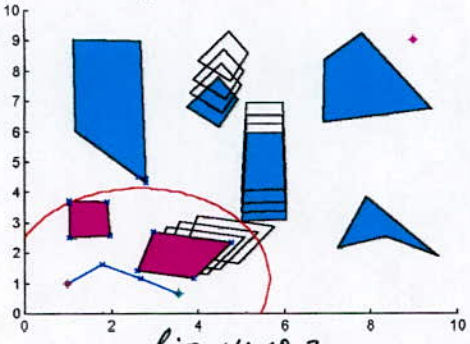


fig. IV. 12. 3

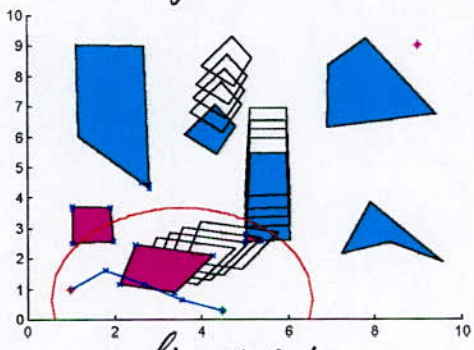


fig. IV. 12. 4

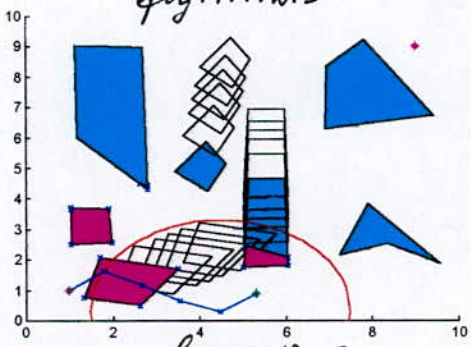


fig. IV. 12. 5

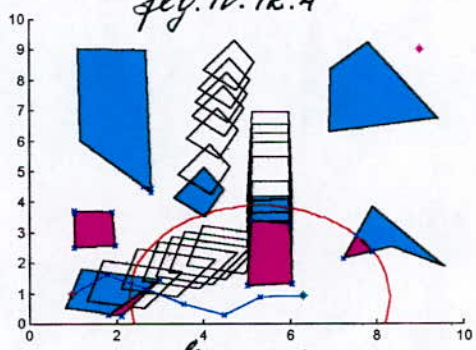


fig. IV. 12. 6

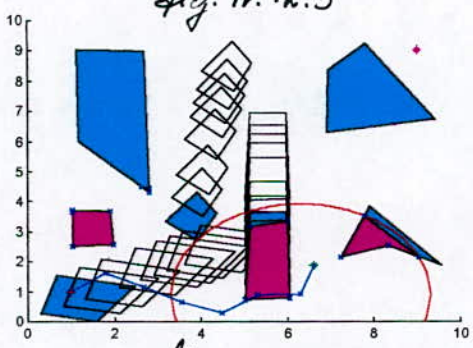


fig. IV. 12. 7

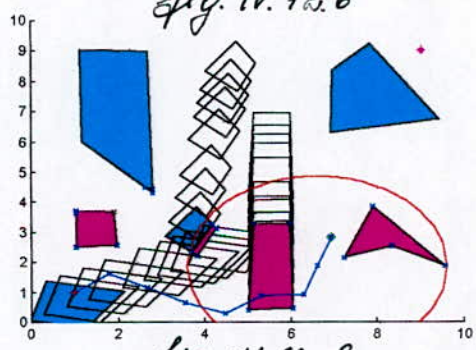


fig. IV. 12. 8

fig. IV. 12. 9

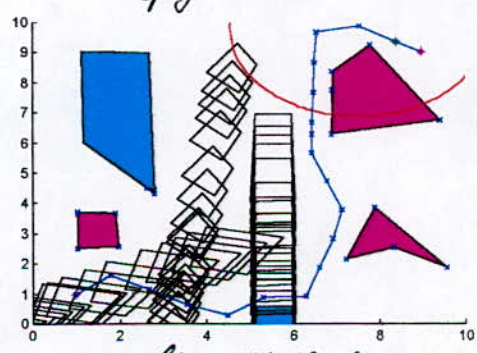


fig. IV. 12. 10

Figure IV. 12.

IV.4.1. Planification *off-line* :

1ère configuration d'obstacles :

Les figures (IV.4.1 à IV.4.7) montrent le meilleur chemin de toute la population de 20 chromosomes, toutes les 50 générations qui constituent la période d'adaptation.

On peut remarquer la contribution de certains opérateurs durant certaines périodes de l'évolution (qui correspondent aux intervalles d'adaptation des probabilités de ces opérateurs), le rôle de chaque opérateur est quantifié par la valeur relativement faible ou élevée de la probabilité qui lui est attribuée, et qui est illustrée dans le tableau ci-dessous :

Nbre de Générations	repair	mutate 1	mutate 2	delete	insert-delete	swap	smooth	crossover
50	0.0356	0.0677	0.0520	0.0008	0.2684	0.0473	0.4473	0.0810
100	0.0433	0.2933	0.0547	0.0010	0.1386	0.0548	0.3159	0.0985
150	0.1605	0.1439	0.0856	0.0581	0.1461	0.0917	0.2634	0.0507
200	0.0652	0.2128	0.0760	0.0335	0.2492	0.0723	0.1845	0.1065
250	0.0426	0.1780	0.1949	0.0069	0.2552	0.0454	0.1880	0.0891
300	0.0390	0.0644	0.3383	0.0009	0.0938	0.0365	0.3383	0.0888
350	0.0523	0.0134	0.4399	0.0012	0.1255	0.0690	0.1797	0.1189

A partir des valeurs données par ce tableau, on peut déterminer les opérateurs qui ont été les plus performants durant une période donnée (c'est ceux qui présentent la probabilité la plus élevée, bien entendu).

On ne peut reconnaître les opérateurs les plus agissants en ne suivant que l'évolution du meilleur chemin, il faut observer les changement accourus à tous les chemins pour constater que tel ou tel opérateur a produit de l'effet bénéfique.

Les figures (IV.4.8. et IV.4.9.) montrent la population initiale (aléatoirement générée) et la population finale, produit des 350 générations d'évolution. Ces deux figures soulignent l'effet global du processus d'évolution et son habilité dans la recherche de la solution.

Pour voir de plus près l'évolution de la population entière, passons à la seconde série de figures (IV.5.1 à IV.5.7), qui à été simulée à partir d'une population de 30 individus, et pour une période d'adaptation des probabilités égale à 30 générations. On y voit bien que l'ensemble des chemins de la population convergent à fur et à mesure que l'évolution avance, vers le meilleur chemin.

2ème configuration d'obstacles :

Dans cette configuration d'obstacles, l'algorithme est d'abord appliqué pour une population de 50 chromosomes et pour une période d'adaptation de 50 générations, l'évolution de l'ensemble de la population est exposée dans

les figures (IV.6.1 à IV.6.7), puis le meilleur chemin issu du processus global est illustré dans la figure (IV.6.8) .

Nous voudrions montrer aussi que le processus évolutif ne fournit pas toujours le chemin optimal : en lançant l'algorithme plusieurs fois, avec les mêmes conditions et les mêmes paramètres, on peut obtenir à chaque fois des résultats tout à fait différents, comme il est montré dans les figures (IV.6.7. à IV.6.9), qui ont été simulées dans les mêmes conditions que la série précédente. Il arrive en fait que la recherche soit bloquée sur un chemin qui ne soit pas optimal : c'est le fameux problème du minimum local duquel même les algorithmes évolutifs ne sont pas exempts (bien qu'ils sont supposés l'être selon ce qui est énoncé dans la littérature).

Autres configurations d'obstacles :

Une recherche de chemin est encore effectuée dans deux autres configurations d'obstacles, histoire d'insister sur la diversité des modèles de l'environnement sur lesquels les algorithmes évolutifs sont applicables. Il sont en fait supposés être applicables à toutes sortes d'environnements à obstacles polygonaux (qu'ils soient convexes ou non). La figure (IV.7.1) décrit la population initiale composée de 30 chemins aléatoires. La figure (IV.7.2) montre l'état de la population après une période d'adaptation des probabilités des opérateurs qui est de 50 générations, la figure (IV.7.3) montre la population après 350 générations d 'évolution, et la figure (IV.7.4) désigne le meilleur chemin obtenu par une telle évolution. On peut facilement constater que ce chemin est presque optimal du point de vue longueur, dégagement par rapport aux obstacles, de plus, il est assez lisse.

La dernière configuration (voir série de figures (IV.8.1 à IV.8.3) est assez particulière : le meilleur chemin est trouvé dès 50 premières générations de l'évolution, et avant 200 générations, l'ensemble de la population composée de 30 chemins deviennent tous identiques au meilleur chemin. On doit cette rapide et globale performance à l'opérateur *repair* qui a pratiquement fait tout le travail, cet opérateur est très important dans la transformation de chemins non faisables en chemins faisables, et tant qu'il existe des chemins non faisables dans la population, cet opérateur est toujours performant, i.e., lorsqu'il est appliqué, sa *performance* (voir fonction d'évaluation) n'est jamais nulle, des huit opérateurs c'est le seul qui garantie l'amélioration du chemin sur lequel il est appliqué, et de ce fait peut être considéré comme le plus efficace, mais il n'est pas dit qu'il est meilleur que tel ou tel autre opérateur, en fait, aucun des opérateurs ne peut remplacer l'autre, il sont tous complémentaires et nécessaires, sauf peut être *swap* qui durant toutes nos simulations n'a manifesté que très peu de performances.

IV.4.2. Navigation *on line* :

Les figures (IV.9), (IV.10) et (IV.11) montrent l'évolution du robot dans un milieu tout à fait inconnu à priori.

Les parties d'obstacles en rouge, représentent la carte de l'environnement dont dispose le robot au moment où il planifie d'exécuter le dernier pas (avant d'arriver à ce pas).

Prenons la série de figures (IV.9.1 à IV.9.10), la première figure montre que le robot a dès le départ, entrepris de suivre le chemin qui n'est pas le plus court (en comparant au chemin qui à été suivi lors de la planification off-line) , c'est tout à fait justifié car il à mesuré son chemin en fonction du seul obstacle dont il disposait sur sa carte (voir la partie en rouge sur la figure (IV.90.1)). Sur la figure (IV.9.7) on peut voir que le robot à dépassé l'ouverture qu'il aurait dû prendre pour rejoindre son objectif. On peut expliquer cela aussi par le fait qu'on n'apercevant qu'un tout petit bout de l'obstacle contournant le point but il a préféré le contourné de l'autre côté (du mauvais côté, en réalité), et ce n'est qu'après quelques pas (voir fig.IV.9.9), lorsqu'il a mesuré l'étendu de cet obstacle, qu'il a décidé de revenir sur ses pas au lieu de contourner l'obstacle et qui a visiblement été la meilleure chose à faire.

Les figures deux autres (IV.9) et (IV.10) montrent la navigation en temps réel ans deux autres configurations d'obstacles.

IV.4.3. Navigation en temps réel dans un environnement dynamique :

La figure (IV.10) montre l'évolution du robot dans un environnement parsemés de quatre obstacles fixes et trois obstacles mobiles, en translations de vitesses du même ordre de la vitesse du robot.

Les sept premières figures montrent l'habilité du robot traversant une zone assez dangereuse où évoluait les trois obstacles en se dirigeant vers lui. On distingue sur la série de figures deux moments critiques :

Le premier, lorsqu'il a évité un obstacle qui se dirigeait droit sur lui à une vitesse égale à la moitié de celle du robot, le robot l'a évité de justesse.

Le second, lorsqu'il a évité l'obstacle rectangulaire qui est venu aussi s'approcher d'assez près.

Le reste du chemin, une fois la zone dangereuse passée, n'était que navigation *on-line*, au milieu d'obstacles statiques.

N.B : la trajectoire du robot intersecte parfois quelques obstacles, il ne faut pas se méprendre sur ce point, il ne s'agit que de la trace de cette trajectoire, car le robot étant déjà ailleurs à ce moment-là.

IV.5. Conclusion :

Dans ce chapitre, l'algorithme de navigation évolutif de base a été étudié, plusieurs défauts sont identifiés et des solutions pour y remédier ont été proposées. Ces dernières ont permis d'avoir des résultats acceptables.

Fort de cette première expérience, nous avons proposé en première étape un algorithme amélioré permettant la navigation *on-line* et prenant en compte le caractère dynamique du mobile et éventuellement celui des obstacles, ainsi que le type de mouvement exécuté par le mobile.

Pour cela, nous avons reformulé la fonction d'évaluation ainsi que certains critères de choix d'opérateurs. Pour ce faire, nous avons développé une série d'heuristiques par opérateur, un nouvel index de performance et une nouvelle fonction d'évaluation pour la planification *off-line*, avec une version modifiée pour la navigation *on-line* et dont nous avons proposé une extension pour la navigation *on-line* en présence d'obstacles mobiles, dans le but d'incorporer des notions d'optimisation de temps et d'énergie. Dans ce dernier cas, nous avons effectué le lissage de trajectoire, qui a rendu cette dernière approche possible, et a permis d'obtenir des résultats intéressant dans le cas des obstacles mobiles non traité par l'algorithme de base.

Néanmoins, cette dernière technique reste encore à améliorer et à affiner, car ce n'était là qu'une approche extrêmement primitive.

CHAPITRE V

CONCLUSION GÉNÉRALE

Ce travail présente une extension (et une variante) à une approche de planification et navigation basée sur l'algorithme évolutif, présentée dans [29-32].

Cette technique présente un algorithme unique de recherche de chemin sous-optimal aussi bien dans un environnement parfaitement connu statique et structuré que dans un environnement non totalement connu à priori (toujours statique et structuré).

Nous avons étudié cette approche, puis s'imprégnant des grandes lignes de la procédure, nous avons développé un algorithme visant à réaliser les mêmes effets, sans pour autant avoir à suivre le même sentier. Quoi qu'il en soit, nous avons dû improviser pour combler le manque d'informations sur la méthode d'origine. Nous avons aussi détecté certaines failles dans les quelques peu d'éléments qui ont été présentés avec précision :

- l'expression de l'index de performances qui régit les seuils d'activation des opérateurs a dû être sérieusement remise en question, car son application lors de l'implémentation, telle qu'elle a été présentée, inhibait parfois certains opérateurs bloquant ainsi le processus d'évolution.

- le critère d'évaluation des chemins a été aussi passé en revue, car constituant la base de jugement de la qualité des solutions disponibles, il permet de diriger la recherche dans le sens désiré en avantageant les chemins présentant de "bonnes" caractéristiques, et en pénalisant ceux qui en présentent des "mauvaises". Nous avons quelques peu reconstruit ce critère en redéfinissant les "bonnes" ainsi que les "mauvaises" caractéristiques (ou gènes dans un langage plus approprié).

Des heuristiques ont été développées pour améliorer l'effet des opérateurs les rendant les plus efficaces possibles.

L'algorithme de navigation en temps réel a été révisé et conditionné de façon à donner un chemin plus facilement réalisable en pratique. La fonction d'évaluation a été modifiée pour ce cas de navigation, ne prenant en compte que les portions utiles des chemins lorsque la recherche s'effectue dans un environnement totalement inconnu. Ceci a permis de réduire la taille de la mémoire nécessaire et d'accélérer considérablement le processus d'évaluation qui représente l'opération qui consomme le plus de temps dans l'ensemble de l'algorithme.

L'implémentation de l'algorithme on-line, dans des environnements sans aucune connaissance à priori, et à de différentes configurations d'obstacles, a présenté, comme le montrent les simulations, des résultats satisfaisants. En se basant sur ce fait il nous est possible d'affirmer que l'algorithme fait non seulement preuve d'une habilité à éviter les obstacle (recherche locale), mais

aussi d'une bonne recherche globale de solution (bonne dans les limites du raisonnable, car lorsqu'il n'y a aucune connaissance sur l'environnement, il est parfois très difficile voire impossible de déterminer le meilleur chemin, pour cela il n'y a qu'à voir la différence entre un chemin trouvé par la technique off-line et celui trouvé en on-line dans le même environnement et avec les mêmes point de départ et arrivé).

L'aspect cinématique d'un problème strictement géométrique jusqu'à présent, a été étudié. Cet aspect, nous l'avons introduit en incorporant une technique de lissage de trajectoire par clothoïdes, étape qui relève de la génération (exécution) de trajectoire, sortant du domaine de planification mais qui est nécessaire à ce stade pour rendre cette planification possible dans un environnement dynamique.

En effet, la description cinématique de la trajectoire a permis de réaliser un évitement efficace d'obstacles mobiles par une technique de prédiction et vérification.

D'autre part, une sommaire description de la commande a permis d'ajouter deux aspects d'optimisation supplémentaires au problème : le chemin obtenu par le processus d'évolution n'est plus le meilleur de la population en terme de longueur, de sûreté et de "lisseur" (*smoothness*) seulement, mais aussi en terme de rapidité et consommation d'énergie.

Il reste cependant à affiner et améliorer la technique de prédiction/ vérification citée plus haut, en essayant d'incorporer un processus d'adaptation du pas de prédiction et du pas maximal de déplacement du mobile, afin d'augmenter l'efficacité de l'algorithme et l'optimalité du chemin obtenu. Ceci faute de temps n'a pu être inclus et est certainement une voie à développer dans le futur.

Références :

[1] A.Pruski, « Robotique mobile, planification de trajectoire ».
[2] R. Chavez, A. Meystel, « structure of intelligence for an automous vehicle », Fort Belvoir Research Center, University of Florida, 1984 IEEE
[3] E.W. Dijkstra, « A Note with Two Problems in Connexion with Graphs. », Numerishe Mathematik 1, 269-271 (1959).
[4] B. Glavina, « Solving Findpath by Combination of Goal-directed and Randomized search », IEEE, conf. on Robotics and Automation (1990).
[5] J. Barraquand, J.C. Latombe, « A Monte-carlo Algorithm for Path Planning With Many Degrees of Freedom », IEEE on Robotics and Automation, (1990).
[6] S.R. Maddila, C.Y. Yap, « Mooving a Polygon Rround a Corner in a Coridor », ACM, (1986).
[7] C.Y. Yap, « How to Move a Chair through a Door », IEEE Journal of Robotics and Automation, vol RA , No 3, (1987).
[8] Brooks and Lozano-Pérez, « a subdivision algorithme in configuration space for findpath with rotation », MASSACHUSETTS INSTITUTE OF TECHNOLOGY, ARTIFICIAL INTELLIGENCE LABOROTORY, A.I.MEMO NO.684, DECEMBER, 1982
[9] R.A. Brooks, « Solving The Find-Path Problem by Good Representation of Free Space », IEEE Transactions on Systems, Man, and Cybernetics, may, 1983.
[10] J. D. Boissonnat & al.ed. «Technique de le robotique, T.II : Perception et planification »
[11] R.L. Drysdale, « Generalized Voronoi diagrams and geomeric searching », Department of Computer sciences, Stanford University, (1979).
[12] Lozano-Pérez, T. « spatial planning : configuration space approche », trans. Computers (February, 1983)
[13] F. Avenaim, J.D.Boissonat, B. Faverjon, «a practical exact motion planning algorithme for polygonal objects avmist polygonal obstacles», in IEEE int. conf. on robotics and automation, Philadelphia, april (1988)
[14] J. Borensrein, Y. Koren, «Real-Time Obstacle Avoidance for Fast Mobile Robot ». Transactions on Systems, Man, and Cybernetics, vol. 19,No5, sebtember/october (1989)
[15] J.L. Crowley, « Navigation for Intelligent Robot », IEEE Journal of Robotics and Automation, (1985)
[16] Brady, Hollenbach, Johnson, Lozano-Pérez, and Masson, editors, Robot Motion, Task Planning.
[17] Susan BONNER and Robert B. KELLEY, « A novel représentation for planning 3-D collision-free paths », IEEE Transactions On Systems, Man, and Cybernetics, vol.20, No. 6 NOVEMBER/DECEMBER 1990
[18] J.G. Lamadrid, « Avoidance system for moving obstacles », Computer Science Department, 136 Lind Hall, University of Minnesota, SPIE vol. 727 Mobil Robots(1986)
[19] O. Khatib, « Real Time Obstacle Avoidance for Manipulators and Mobile Robots », Artificial Intelligence Laborotary, Stanford University, CA 94305 (1985).
[20] Van-Duc Nguyen, « The Find-Path Problem in the Plane », Massachusetts Institute of Technology, Artificial Intelligence Labororatory, A0I. Memo No. 760, february ,1984.

[21] T. S. , K. Qiu and J.J. Nitao, « An Obstacle Avoidance Algorithme for an Autonomous Land Vehicle », SPIE vol. 727 Mobile Robots(1986)
[22] Brady, Hollenbach, Johnson, Lozano-Pérez, and Masson, editors, Robot Motion, Task Planning.
[23] K.S. Fu, R.C. Gonzalez, C.S.G. Lee, ROBOTICS : Control, Sensing, Vision, and intelligence.
[24] Mujtaba, M.S. « Discussion of trajectory calculation methods ,» in Exploratory study of computor integrated assembly systems, Binford, T. O. et. Al., Stanford University, Artificial Intelligence Laboratory,AIM 285.4, 1977.
[25] S. Fleury, P. Souères, J.P. Laumond, and R. Chatila, « Primitives for Smoothing mobile Robot Trajectories », IEEE Trnsactions on Robotics and Autumation, vol. 11, No. 3, (june 1995).
[26] M. Bucaille, « l'Homme, d'où vient-il ? ».
[27] J.M. Renders, « Algorithmes génétiques et réseaux de Neurones ».
[28] T. Bäck, U. Hammel, and H.P. Schwefel, « Evolutionary Computation : Comments on the History and Current State », IEEE Trans on Evolutionary Computation, vol1, No1, April 1997.
[29] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, « Adaptive Planner/Navigator for mobile Robot », IEEE Trans on Evolutionary Computation, vol1, No1, April 1997.
[30] J.Xiao, « Evolutionary planner/Navigator in a mobile robot envirenment », in Handbook of Evolutionary Computation, T. Bäck, D. Fogel, and Z. Michaelewicz, Eds. New york : Oxford Univ., Press and Institute of Physics, 1997
[31] H. Lin, J. Xiao, and Z. Michalewicz, « Evolutionary Algorithme for Path Planning in Mobile Robor Environment »,1994 IEEE.
[32] H. Lin, J. Xiao, and Z. Michalewicz, « Evolutionary Navigator for Mobile Robot », Computer Science Department, University of North California – Charlotte.
[33] M. Zhao, N. Ansari and E.S.H. Hou, « Mobile Manipulator Path Planning by a Genetic Algorithm », Proceedings in the1992 IEEE/RSJ International Conference on Intelligent Robots and Systems.
[34] R . Chatila, « Mobile Robot Navigation, Space Modeling and decisional process », in Robotic Research, 1985
[35] L. Pontryagine, V. Boltyansky, R. Gamkrelidze, and E. Mischenko, « the mathematical theory of optimal process », New York, Wily,1964.