

N° d'ordre : 51/2012 – M / EL

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Électronique et d'Informatique



MÉMOIRE

Présenté pour l'obtention du diplôme de MAGISTERE

Par : Lakhdari Abdelhalim

En : ÉLECTRONIQUE

Spécialité : Contrôle des processus et robotique

THEME

**Génération de mouvements optimaux pour un
robot mobile non holonome en utilisant les
techniques d'intelligence artificielle**

Soutenu publiquement le : 12 /07/ 2012 devant le jury composé de :

Mr. R.Toumi

Professeur à l'USTHB

Président

Mme.N.Achour

Professeur à l'USTHB

Directrice de mémoire

Mme. O.Azouaoui

Maître de Recherche /A CDTA

Examinatrice

Mr.R. Guernane

Maître de Conférences/B à l'UMBB

invité

Remerciements :

Je tiens d'abord à adresser mes plus vifs remerciements à Mme N.Achour , professeur à la Faculté d'électronique et d'informatique de L'USTHB, pour son aide et l'intérêt avec lequel elle a suivi mon travail.

Je tiens à remercier Mr R.Toumi , professeur à la Faculté d'électronique et d'informatique de L'USTHB, pour l'honneur qu'il me fait de présider ce jury.

Je tiens à remercier également Mme O.Azouaoui , Maître de recherche /A à CDTA pour l'honneur qu'elle me fait de participer à ce jury.

Je tiens à remercier aussi Mr R.Guernane , Maître de conférence/B à l'UMBB d'avoir accepter l'invitation de participer à l'évaluation de ce travail.

Enfin, j'exprime toute ma gratitude à ceux qui ont contribué à l'élaboration de ce mémoire chacun de sa manière.

Table des matières :

Introduction général	1
1- Contexte général et état de l'art de la planification de mouvement	4
Introduction	4
1.1. Notations et définitions	5
1.1.1 Espace de travail	5
1.1.2. Espace de configurations	5
1.1.3 Distances	6
1.1.4 Chemin et Trajectoire.....	7
1.1.5. Planification de trajectoire	8
1.2. Etat de l'art de la planification de mouvement.....	8
1.2.1. Méthodes de décomposition en cellules	8
1.2.1.1 Décomposition exacte	9
1.2.1.2. Décomposition approchée	10
1.2.2. Méthode des potentiels.....	11
1.2.3. La méthode du Roadmap (Graphe de visibilité).....	12
Conclusion	13
2 L'approche PRM : principe et variations	14
Introduction	14
2.1. Approche PRM de base.....	14
2.1.1 La phase d'apprentissage	15
2.1.1.1. L'étape de construction	15
2.1.1.2. L'expansion du graphe	17
2.1.2 La phase de recherche	20
2.1.3. Lissage de chemin	21
2.2. Variation autour de la méthode PRM de base	22
2.2.1 Les stratégies d'échantillonnage.....	22

2.2.1.1 Echantillonnage uniforme	23
2.2.1.2 Echantillonnage gaussien	23
2.2.1.3 Addition adaptative de nœuds.....	24
2.2.1.4 OBPRM: Obstacle based PRM	25
2.2.1.5 PRM basé sur la visibilité (visible-PRM)	25
Conclusion	26
3 La planification de mouvement et l'intelligence artificielle	27
Introduction	27
3.1 L'Algorithme meilleur d'abord (A^*).....	28
3.2 Les algorithmes génétiques	30
3.2.1 Codages et Initialisation	31
3. 2.2 L'évaluation des individus	32
3. 2.3 Sélection	32
3.2.4 Reproduction	34
3.2.4.1 Croisement	34
3.2.4.2 Mutation.....	35
3.2.5 L'évolution d'un algorithme génétique	37
3.3 Les algorithmes génétiques hiérarchiques.....	37
3.3.1 Forme du chromosome hiérarchique.....	38
3. 3.2 Opérateurs génétiques	39
Conclusion	39
4. Planification de mouvement non holonome	40
Introduction.....	40
4.1 Cinématique et notion de non-holonomie	41
4.1.1 Roulement sans glissement	41
4.1.2 Contraintes non holonomes	42
4.2 Modélisation du robot mobile de type voiture.....	45

4.3 Non holonomie du robot de type voiture	47
4.4 Planification de mouvement des robots non holonomes	48
4.4.1 Les chemins sans manœuvres	48
4.4.2 Les chemins avec manœuvres	49
Conclusion	50
5. Simulation et résultats obtenus	51
Introduction	51
5.1 Environnement	52
5.2 Le robot mobile.....	52
5.3 La construction du réseau PRM	53
5.3.1 Calcul de nombre des nœuds N_{max}	53
5.3.2 La phase d'apprentissage.....	54
5.3.3 Teste de collision	55
5.4 L'algorithme de recherche du plus court chemin.....	57
5.5 Suivi de chemin.....	59
5.6 Optimisation par les algorithmes génétiques.....	62
5.6.1 Codages et Initialisation	63
5.6.2 La fonction d'adaptation.....	64
5.6.3 Sélection	65
5.6.4 La reproduction des individus.....	65
5.6.4.1 Croisement	65
5.6.4.2 Mutation.....	66
5.7 L'algorithme général de planification de mouvement.....	66
5.8 Analyse de l'algorithme et résultats obtenus.....	68
Conclusion Générale	71
Références bibliographiques.....	72

Liste des figures :

Figure 1.1 : espace de configuration.....	5
Figure 1.2:construction d'un COB pour un robot triangulaire et un obstacle rectangulaire....	6
Figure 1.3 : Décomposition exacte	9
Figure 1.4 : Décomposition approchée	10
Figure 1.5 : Méthode de potentiel	11
Figure 1.6 : graphe de visibilité	12
Figure 2.1 : La construction du réseau PRM.....	16
Figure 2.2 : l'expansion du réseau	18
Figure 2.3 : La phase de recherche.....	21
Figure 2.4 : lissage de chemin	22
Figure. 2.5 : Les trois cas de figure pour l'implantation de l'échantillonnage.....	23
Figure 2.6: Configurations obtenues avec une distribution uniforme.....	24
Figure 2.7 : Roadmap obtenu par une distribution uniforme sur la surface des obstacles.....	25
Figure 2.8 :Domaine de visibilité d'une configuration q_i et le réseau construit à l'aide du Visible-PRM	26
Figure 3.1 : structure des solutions dans un algorithme génétique	31
Figure 3.2: exemple d'un chromosome dont des gènes codés en binaire	32
Figure 3.3 : sélection par la méthode de la loterie biaisée	33
Figure 3.4 : différents type de croisement	35
Figure 3.5: le processus de la mutation	36
Figure 3.6: la structure hiérarchique d'un chromosome.....	38
Figure 4.1 : description d'une roue	46
Figure 4.2: un robot de type voiture.....	42
Figure 4.3 : Exemple de plus court chemin de Dubins.....	49
Figure 4.4 plus court chemin de Dubins en présence d'obstacles.....	49
Figure 4.5 exemples de plus cours chemin de Reeds et Sheep.....	50
Figure 5.1:exemple d'un environnement à 4 obstacles.....	52

Figure 5.2: représentation du robot de type voiture	52
Figure 5.3: discrétisation de différents environnements avec différentes valeurs de d_{min}	55
Figure 5.4 : teste de collision	56
Figure 5.5 : La fonction isccw	56
Figure 5.6: chemins avant et après le raccourcissement	58
Figure 5.7 : chemin après discrétisation	59
Figure 5.8 : chemin après l'exécution successive des discrétisations et raccourcissements....	59
Figure 5.9 : Calcule des points de braquage.....	60
Figure 5.10 : suivie de chemin entre trois points	61
Figure 5.11 : trajectoire obtenue après l'exécution de la fonction : move.....	62
Figure 5.12: chemins obtenus par les PRM exécutés 10 fois.....	63
Figure 5.13 structure d'un individu	64
Figure 5.14 :le principe du croisement utilisé	65
Figure 5.15 : L'algorithme général de planification de mouvement	67
Figure 5.16 : chemins de la population initiale pour différents paramètres	68
Figure 5.17 Résultats pour l'environnement 1.....	69
Figure 5.18 Résultats pour l'environnement 2.....	70

Introduction générale

Le problème de la planification de mouvement était, pour longtemps, limité au domaine de la robotique de manipulation, mais le besoin de mettre les robots sur des tâches hors des sites industriels, fait apparaître les robots mobiles au cours des années 1960.

L'évolution importante des techniques d'intelligence artificielle permet à ces robots d'exécuter des tâches de plus en plus complexes, et parfois dangereuses telles que la manipulation sur des sites sensibles (nucléaires par exemple), les opérations d'intervention sur des sites accidentés, l'exploration des sites maritimes ou planétaires etc.

Depuis quelques années, un intérêt croissant est porté au sein de la communauté robotique à la planification et la génération de mouvements optimaux, un intérêt qui est né de la nécessité d'optimiser l'énergie consommée par les robots mobiles non holonomes de type voiture CLR (Car-Like-Robot) en développant des algorithmes dans ce sens, cela a une portée économique très importante.

Problématique :

Dans sa formulation de base, le problème de la planification de mouvement consiste à faire évoluer les systèmes robotiques de façon sûre dans des milieux réels en contrôlant les interactions entre le robot et son environnement.

Ces interactions peuvent prendre différents aspects :

- Actions de la part du système robotique (se positionner par rapport à un objet, manœuvrer pour se garer, etc.),
- Réactions vis-à-vis d'événements provenant du monde qui l'entoure (éviter des obstacles, poursuivre une cible mobile, ...etc.).

Pour un robot mobile non holonome évoluant dans un environnement donné, le problème général de la planification consiste à déterminer un mouvement lui permettant de se déplacer entre deux configurations données tout en respectant un certain nombre de contraintes et de critères, ceux-ci découlent de plusieurs facteurs de natures diverses et dépendent généralement des caractéristiques du robot, de l'environnement et du type de tâche à exécuter. Ainsi les contraintes relatives au robot concernent sa géométrie, sa cinématique et sa dynamique, et leur prise en compte peut être complexe selon l'architecture initiale considérée. Cette architecture peut correspondre à un mobile muni d'un système de manipulation tel qu'un manipulateur mobile, ou un véhicule simple ou tractant des remorques, ou encore à plusieurs systèmes de robots (système multi robots) tels qu'un groupe des robots mobiles de type voiture évoluant dans un réseau routier.

Les contraintes émanant de l'environnement concernent essentiellement la non-collision avec les obstacles fixes ou mobiles de l'environnement.

Le problème de la génération de mouvements optimaux pour un robot mobile non holonome peut se diviser en deux sous problèmes : le premier consiste à trouver des chemins optimaux, le deuxième consiste à adapter ces chemins pour être réalisables par le robot en prenant en compte les contraintes liées à ce dernier, ces chemins doivent donc satisfaire certaines conditions :

- ils doivent être sans collision et à longueur minimale.
- ils doivent être lisses et avec un minimum de virages.
- ils doivent être réalisables par le robot et en temps monotone.

Dans ce mémoire nous avons rapporté les travaux que nous avons élaborés, à savoir un algorithme de planification qui permet de calculer des mouvements optimaux en

utilisant des algorithmes probabilistes (PRM) associés à des techniques d'intelligence artificielle.

Organisation de mémoire :

Le premier chapitre donne une vue générale sur le problème de planification de mouvement, il présente des notions de base et les notations utilisées, ainsi qu'un état de l'art des grandes approches développées pour résoudre ce problème.

Le deuxième chapitre est consacré à la méthode des réseaux probabilistes (PRM), nous avons décrit d'une manière détaillée le principe général des PRM ainsi que les différentes variantes introduites autour de cette méthode.

Dans le troisième chapitre nous donnons la description de deux techniques d'intelligence artificielle : La première concerne les algorithmes génétiques qui sont un outil d'optimisation fortement appliqué dans le problème d'optimisation en robotique, la deuxième technique concerne l'algorithme bien connu de recherche du plus court chemin appelé *A étoile* (A^*).

Dans le quatrième chapitre, nous expliquons la notion de contraintes non holonomes, ensuite nous présentons une modélisation cinématique d'un robot mobile de type voiture, nous terminons ce chapitre en citant les différentes solutions de planification de mouvement des robots non holonomes.

Le cinquième chapitre est consacré aux différents résultats obtenus en utilisant l'algorithme de planification de mouvement élaboré, nous avons fait des tests sur plusieurs environnements et pour différentes configurations de départ et d'arrivée.

Nous terminons ce mémoire par une conclusion générale et les perspectives futures.

Chapitre I

Contexte général et état de l'art de la planification de mouvement

Introduction :

La planification de mouvement consiste à chercher une séquence de commandes qui permet au robot de se déplacer dans l'environnement qui l'entoure.

Dans ce chapitre nous donnons une vue générale sur le problème de la planification de mouvement ensuite nous présentons l'état de l'art des méthodes les plus courantes concernant la planification de mouvement en robotique.

1.1 Notations et définitions :

1.1.1 Espace de travail :

L'espace de travail noté $\mathcal{W}$ est l'espace dans lequel le robot évolue (dans notre cas c'est un espace euclidien à deux dimensions $\mathbf{R}^2$).

1.1.2 Espace de configurations :

Le concept de base de l'espace des configurations permet de reformuler le problème de planification de mouvement dans un nouvel espace qu'on note $\mathcal{CS}$, il est constitué de l'ensemble de toutes les configurations possibles du robot.

Une configuration q d'un robot $\mathcal{A}$ est une combinaison de n paramètres indépendants (où n est la dimension de $\mathcal{CS}$) spécifiant les différents paramètres (position, orientation, vitesse ...etc.) de chaque élément de $\mathcal{A}$ dans l'espace de travail $\mathcal{W}$, ainsi le problème se ramène facilement à la planification des mouvements d'un seul point dans ce nouvel espace.

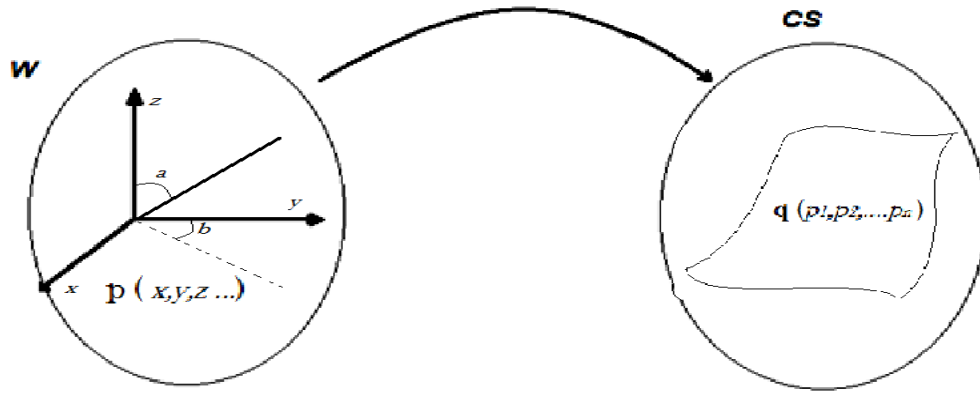


Figure 1.1 : *espace de configuration*

Un obstacle $\mathcal{OB}_i$ de $\mathcal{W}$ est représenté dans l'espace de configuration $\mathcal{CS}$ par un C-Obstacle $\mathcal{COB}$ défini par :

$$\mathcal{COB}_i = \{q \in \mathcal{CS} / \mathcal{A}(q) \cap \mathcal{OB}_i \neq \emptyset\} \quad (1.1)$$

Dans le cas d'un robot de taille et de forme quelconques, on peut se ramener au problème de planification d'un robot ponctuel en transformant l'environnement par un processus d'érosion de l'espace libre en créant des $\mathcal{COB}$, qui sont une représentation des obstacles réels dans l'espace des configurations. Les points de l'espace libre qui ne peuvent pas être atteints par une configuration sans collision sont éliminés, comme illustre la figure 1.2.

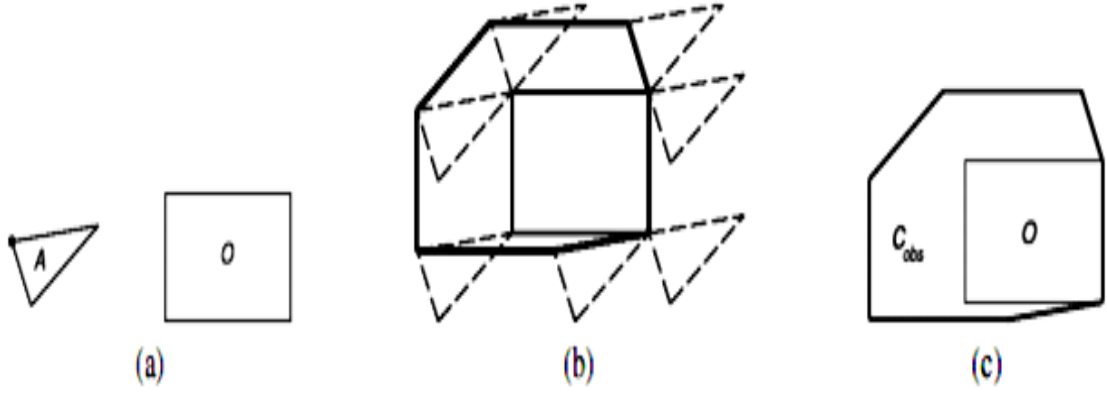


Figure 1.2 – construction d'un COB pour un robot triangulaire et un obstacle rectangulaire

Espace de configurations libres :

L'espace des configurations libres (sans collision) qu'on note $\mathcal{CS}_{libre}$, est l'espace dans lequel le robot se déplace, il est égal à l'espace total moins l'espace occupé par les obstacles, il peut être défini par :

$$\mathcal{CS}_{libre} = \mathcal{C} \setminus \bigcup_{i \in [1, m]} \mathcal{COB}_i \quad (1.2)$$

1.1.3 Distances :

La distance est une mesure positive qui peut être exprimé comme suit:

Distance entre deux configurations :

Soient q_1 et q_2 deux configurations de l'espace de configurations, p un point du robot et $q_i(p)$ la position de ce point dans l'espace de travail $\mathcal{W}$ lorsque le robot est dans la configuration q_i . La distance d_c entre q_1 et q_2 est définie comme la distance maximale entre deux points du robot dans les configurations q_1 et q_2 , elle est donnée par:

$$d_c(q_1, q_2) = \max_{p \in \mathcal{A}} \|q_1(p) - q_2(p)\| \quad (1.3)$$

Distance entre deux corps :

Soit $\mathcal{A}$ et $\mathcal{B}$ deux sous-espaces compacts de $\mathcal{W}$, représentant deux corps. On appelle distance entre ces deux corps la valeur :

$$d(\mathcal{A}, \mathcal{B}) = \min_{a \in \mathcal{A}, b \in \mathcal{B}} \|a - b\| \quad (1.4)$$

Distance entre un robot dans une configuration et un corps :

Si l'on note $\mathcal{O}$ un sous-espace de $\mathcal{W}$, et q une configuration du robot, on appelle distance entre le robot dans une configuration q et le sous-espace $\mathcal{O}$ la valeur :

$$d_{\mathcal{O}}(q, \mathcal{O}) = \min_{p \in \mathcal{R}, \sigma \in \mathcal{O}} \|q(p) - \sigma\| \quad (1.5)$$

1.1.4 Chemin et Trajectoire:**Trajectoire :**

Une trajectoire est une fonction continue de $[0, T] \subset \mathcal{R}$ dans $\mathcal{CS}$ qui associe à tout instant $t \in [0, T]$ une configuration $q(t)$ de $\mathcal{CS}$:

$$\begin{array}{ccc} q : & \longrightarrow & [0, T] \mathcal{CS} \\ t & \longrightarrow & q(t) \end{array} \quad (1.6)$$

Chemin :

Un chemin peut être défini comme étant l'ensemble des points par lesquels passe le robot au cours de son mouvement d'une configuration initiale à une configuration finale. Autrement dit le chemin est l'image d'une trajectoire dans $\mathcal{CS}$.

Trajectoire libre :

Une trajectoire est dite libre si toutes les configurations constituant cette trajectoire appartiennent à l'espace des configurations libre $\mathcal{CS}_{libre}$.

Trajectoire admissible :

Une trajectoire est dite admissible si elle est la solution du système d'équations différentielles correspondant au modèle cinématique du robot, incluant les contraintes sur les commandes pour des conditions initiales et finales données. En d'autres termes, cela signifie qu'il existe pour chacune de ces configurations une commande qui permet de ramener le robot de la configuration q_i à la configuration q_{i+1} .

De la même façon, un chemin admissible est l'image d'une trajectoire admissible.

1.1.5. Planification de trajectoire :

On appelle planification de trajectoire, le choix et la construction d'une trajectoire sans collision et admissible pour un robot entre une configuration de départ q_{in} , et une

configuration d'arrivée q_{fin} données. Pour deux configurations q_{in} et q_{fin} données, il peut exister plusieurs trajectoires entre q_{in} et q_{fin} .

1.2. Etat de l'art sur la planification de mouvement:

Durant ces vingt dernières années, le problème de la planification de mouvement des robots mobiles a intéressé les chercheurs de différents domaines, ce qui a permis l'émergence de plusieurs approches et implémentations.

L'introduction de la notion d'espace de configuration par Lozano Pérez [1] était le début de l'algorithmique de la planification de mouvement, il a réussi à transformer le problème de la recherche d'un chemin pour un système à n degrés de liberté dans l'espace euclidien en celui du mouvement d'un point évoluant dans l'espace de configuration de ce système.

Deux grandes approches de la planification de mouvement ont apparu, la première dite approche exacte et complète (déterministe) tels que :La décomposition cellulaire introduite par Schwartz [2], la méthode des champs de potentiels [3], le graphe de visibilité (roadmaps) [4]...etc. La seconde approche regroupe les méthodes probabilistes: les réseaux probabilistes introduits par Kavraki et Latombe sous le nom de PRM [5], et la méthode RRT (Rapidly-exploring RandomTree) décrite par Lavallée dans [6].

La complexité des premières méthodes augmente exponentiellement avec le nombre de degrés de liberté du robot alors que les secondes souffrent des inconvénients de la non complétude.

1.2.1. Méthodes de décomposition en cellules :

Les méthodes basées sur la décomposition en cellules utilisent une représentation discrète de l'environnement du robot, leur principe est de construire un recouvrement de CS_{libre} avec des régions disjointes nommées cellules, puis déterminer le graphe de connectivité entre ces cellules.

Le chemin reliant une configuration initiale q_{in} à une configuration finale q_{fin} est déduit de la séquence de cellules reliant celle contenant q_{in} à celle contenant q_{fin} , cette séquence étant obtenue à l'aide d'une recherche dans le graphe de connectivité.

Ces méthodes sont regroupées en deux catégories, décomposition exacte et décomposition approchée.

Décomposition exacte :

Cette première catégorie regroupe les méthodes de résolution dites exactes car la décomposition effectuée se base sur un recouvrement exact de l'espace libre du robot de ses contacts en cellules convexes. La complétude et la capacité de ce type de méthode à apporter une réponse quant à la résolution du problème général de planification ont été démontrées par Schwartz et Sharir [2].

Le problème de planification est formulé sous forme algébrique en considérant une décomposition de l'espace libre du robot sous forme de composantes cylindriques semi-algébriques (cellules de Collins)[7]. Par cette formulation, les auteurs aboutissent à l'un des résultats importants sur la complexité de la planification, à savoir qu'elle est polynomiale en la complexité de l'environnement et doublement exponentielle en la dimension de l'espace des configurations. La complexité de l'environnement est mesurée par le nombre et le degré des polynômes utilisés pour la description semi-algébrique de l'espace libre.

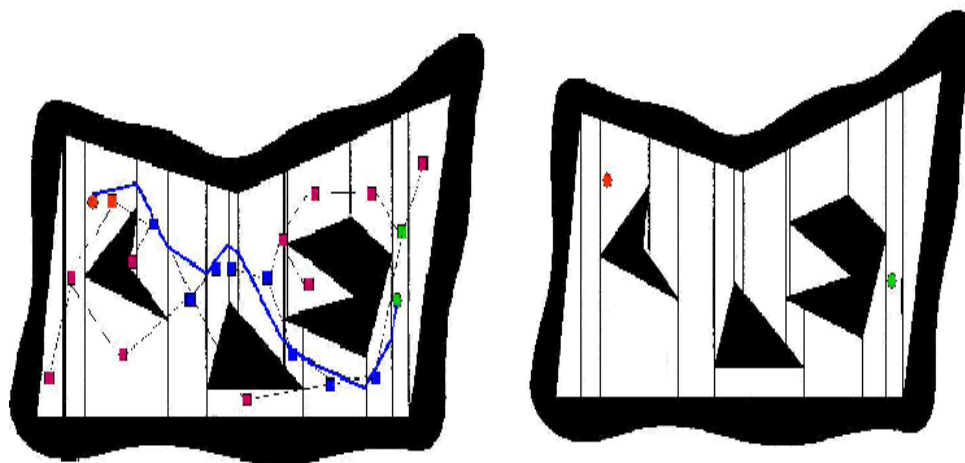


Figure1.3- Décomposition exacte

Il est important de noter que cette décomposition de l'espace libre est une méthode dite exacte, qui doit être distinguée des méthodes non-exactes.

Au premier rang de ces dernières, on peut citer la discrétisation de l'espace cartésien en cellules cubiques de tailles données, qui produit des solutions pour une résolution donnée.

Décomposition approchée :

Les méthodes dites approchées sont généralement appliquées afin de remédier à la complexité de la décomposition exacte. Elles se distinguent des précédentes par la simplicité de la structure des cellules utilisées pour la décomposition et l'approximation faite de l'espace libre du robot. La structure de celui-ci est généralement capturée dans une représentation hiérarchique en cellules élémentaires identiques et permettant l'adaptation de la taille de celles-ci à la géométrie des zones à recouvrir. Plusieurs types de décomposition de l'espace du robot sont généralement utilisés en pratique : représentation en *octrees*, découpage en tranches, ou encore en polyèdres [7].

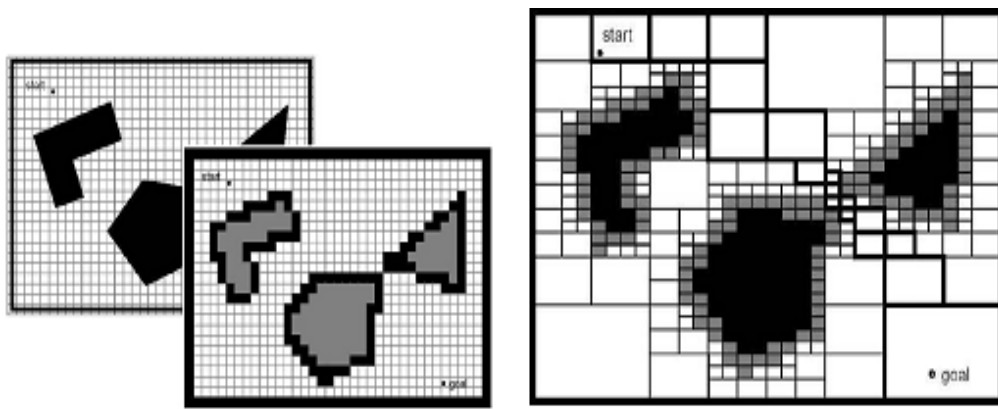


Figure 1.4 - Décomposition approchée

En comparant les deux approches de décomposition, il ressort que les méthodes exactes sont plus complètes en théorie que les méthodes approchées. Toutefois, la complexité de leur mise en œuvre, les rend moins appropriées en pratique que les techniques approchées même si celles-ci restent parfois limitées à des espaces de dimensions réduites.

1.2.2. Méthode des potentiels:

Cette méthode est largement utilisée dans la littérature de robotique, elle consiste à assimiler le robot à une particule contrainte à se déplacer dans un champ de potentiel fictif obtenu par la composition d'un premier champ attractif au but et d'un ensemble de champs répulsifs modélisant la présence des obstacles dans l'espace du robot.

Les déplacements de celui-ci sont alors calculés itérativement par un algorithme de descente du gradient du potentiel obtenu.

Cette méthode a été initialement introduite par Khatib [3], elle a été utilisée pour la commande et l'évitement d'obstacles pour un bras, puis étendue par la suite pour la planification de mouvement.

Si une méthode de type potentiel peut être facilement implantée et appliquée en temps réel pour des tâches de manipulation ou de navigation d'un robot mobile, elle reste sensible à l'occurrence de minimums locaux engendrant des configurations de blocage ou d'oscillations, ces minimums sont généralement liés à la géométrie et à la répartition des obstacles dans l'espace de travail du robot et surtout aux coefficients de pénalités qui leur sont associés pendant la construction du champ de potentiel.

Afin de remédier à ces problèmes, Barraquand et Latombe proposent dans [8] d'intégrer à la minimisation du potentiel appliqué au robot l'exploration d'une représentation hiérarchique en bitmap de l'espace des configurations.

L'évitement ou plutôt la sortie d'un minimum local est effectuée par l'application de mouvements aléatoires, dits browniens.

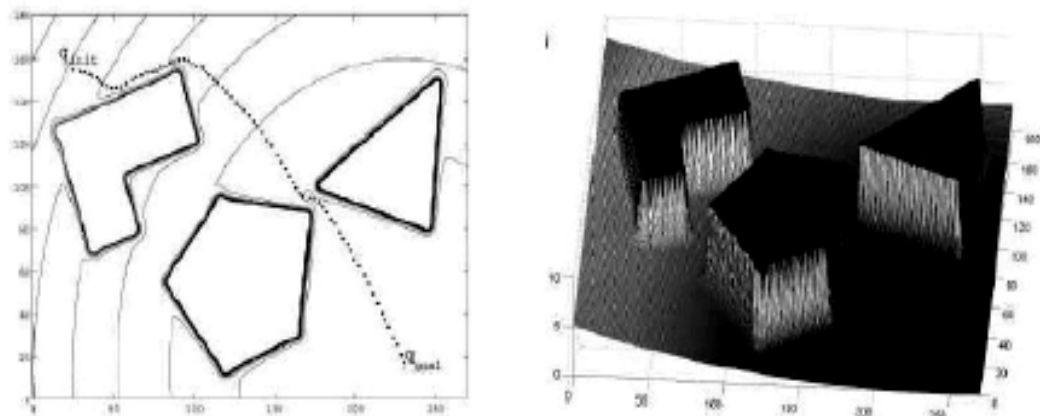


Figure 1.5 Méthode des potentiels

1.2.3. La méthode du Roadmap (Graphe de visibilité):

Cette approche représente la connexité de l'espace des configurations libres CS_{libre} par un graphe [4]. Le fonctionnement global de l'algorithme se déroule toujours sur deux étapes successives. Dans la première phase, la construction de la Roadmap est effectuée. Cette Roadmap se présente sous la forme d'un graphe non orienté $G = (N, V)$.

L'ensemble des nœuds N est un ensemble de configurations du robot dans l'espace libre CS_{libre} . V est l'ensemble des liens entre les nœuds du graphe deux à deux. Ces liens représentent des chemins sans collision simples dans CS_{libre} entre deux configurations du graphe.

Dans la deuxième phase, les configurations initiale et finale q_{in} et q_{fin} sont ajoutées au réseau avant d'essayer de trouver un chemin reliant les deux (voire figure 1.6).

On pourrait citer différents exemples comme l'approche par graphe de visibilité, l'approche par rétraction, la méthode Freeware, et la méthode Silhouette.

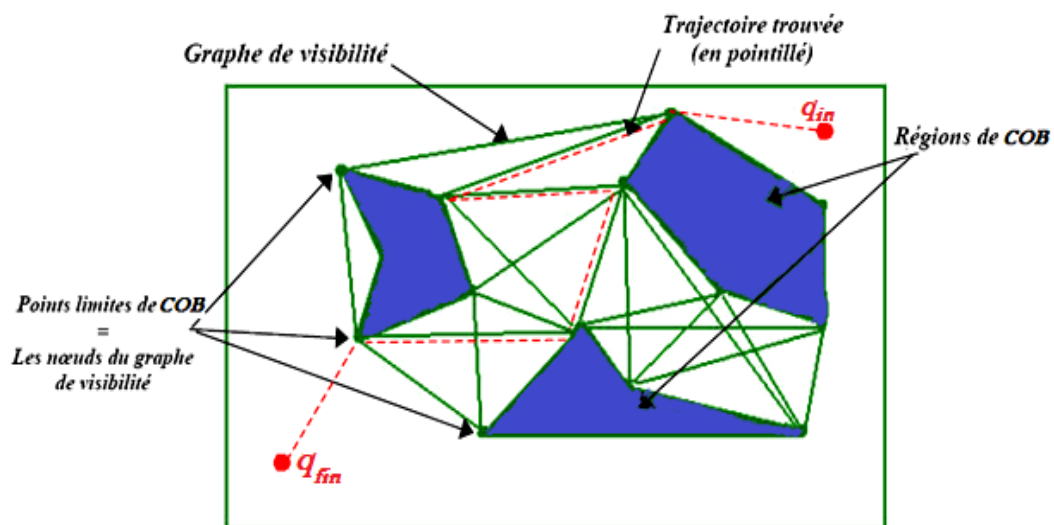


Figure 1.6- graphe de visibilité

On distingue cependant deux variantes différentes dans les méthodes Roadmap:

Les méthodes déterministes et les méthodes probabilistes. Ces deux modèles diffèrent dans l'implémentation de la première phase : la construction de la Roadmap.

Dans les approches déterministes, le graphe construit est toujours le même pour un même espace des configurations. Elles sont complètes mais compliquées et lentes. Elles ne sont pas applicables à tout type d'espace mais plutôt aux espaces de faibles dimensions (2 ou 3).

Dans la deuxième catégorie des méthodes Roadmap, on cite l'approches des réseaux probabilistes (ou PRM : Probabilistic RoadMap). Dans cette dernière, la construction du graphe utilise des stratégies probabilistes dans le parcours de l'espace CS_{libre} . Ces méthodes sont rapides par rapport aux méthodes exactes citées précédemment et permettent d'avoir de très bons résultats même dans des espaces complexes.

Conclusion :

Dans ce chapitre nous avons présenté les notions de base de planification de mouvement et nous avons cité les différentes approches existantes ; ces approches peuvent être des méthodes déterministes (exactes) qui présentent des outils efficaces quand il s'agit d'une planification dans des environnements simples ou de faibles dimensions.

Dans des environnements complexes, les méthodes déterministes sont très limitées à cause de leur complexité qui augmente avec la dimension et la complexité de l'environnement, cependant les méthodes probabilistes ont prouvé leur efficacité dans de tels environnements, c'est ce que nous verrons au chapitre deux.

Chapitre II

L'approche PRM : principes et variantes

Introduction

Au début des années 90 sont apparues les méthodes de planification aléatoire de mouvement. Ces méthodes, introduisent une idée permettant de fusionner la notion de complétude et celle de probabilité, autrement dit si une solution existe, la probabilité de réussite de l'algorithme tend vers un, lorsque le temps tend vers l'infini.

En première partie, nous donnons une description de l'approche PRM de base telle qu'elle a été introduite par Kavraki, Svestska, Latombe et Overmars en 1996 [5]. En deuxième partie, nous citons les différentes variantes de la méthode PRM classique en décrivant brièvement quelques-unes d'entre elles.

Nous finissons ce chapitre en citant l'application des techniques d'intelligence artificielle pour chercher le plus court chemin dans un graphe de PRM.

2.1. Approche PRM de base:

Les approches PRM s'inspirent des méthodes *Roadmap* classiques (graphe de visibilité) en capturant la connectivité de l'espace libre $\mathcal{CS}_{libre}$. Elles diffèrent cependant par la première phase de l'algorithme qui est la phase de construction du réseau Roadmap. En effet dans la méthode PRM, la construction utilise différentes techniques probabilistes permettant de réduire la taille du graphe pour ne représenter qu'un échantillon bien choisi de l'espace de recherche $\mathcal{CS}_{libre}$. Ceci réduit d'une façon importante le temps de calcul et l'espace mémoire consommé. Elles gardent la même structure générale de fonctionnement pour les autres étapes des méthodes Roadmap classiques.

Le principe des méthodes PRM est simple ; On cherche à construire un graphe $G = (N, V)$ capturant la connexité de $\mathcal{CS}_{libre}$. Les nœuds N sont des configurations libres générées aléatoirement selon une distribution uniforme. Les arcs de G , rassemblés dans l'ensemble V , sont des chemins libres (sans collision) reliant deux à deux des nœuds de N .

Des algorithmes de détection de collision [9] permettent de vérifier si les configurations considérées correspondent ou non à des configurations admissibles pour les déplacements du robot. La plupart des méthodes PRM ont la même structure générale. Elles sont habituellement divisées en deux phases : Une phase d'apprentissage et une phase de recherche.

2.1.1 La phase d'apprentissage :

L'objectif de la phase d'apprentissage est de construire un graphe avec assez de nœuds pour couvrir l'espace des configurations libres $\mathcal{CS}_{libre}$ d'une manière uniforme et ceci même dans les endroits les plus difficiles de cet espace. Nous allons détailler tout ceci par l'étude du fonctionnement interne de cette phase d'apprentissage.

La phase d'apprentissage est la phase dans laquelle des nœuds sont insérés et reliés au graphe. Elle est constituée habituellement de deux étapes : une étape de construction, et une autre d'expansion.

2.1.1.1. L'étape de construction :

Dans cette étape on utilise une stratégie d'échantillonnage pour générer aléatoirement des nouveaux nœuds dans l'espace des configurations libres et les insérer dans le graphe G .

Ensuite on utilise une stratégie de choix des voisins pour sélectionner un ensemble de nœuds voisins pour chaque nouveau nœud généré.

Un détecteur de collision permet de décider si un nœud appartient ou non à l'espace libre. Il est également utilisé pour tester si un chemin donné est inclus ou non dans l'espace libre.

Pour chaque nœud n de l'ensemble des voisins éventuels d'un nœud q , on fait appel à une méthode locale pour trouver un chemin admissible de q à n . Si la méthode locale réussit, elle ajoute un arc entre les nœuds correspondants dans le graphe, dans ce cas, le chemin est échantillonné en séquences de configurations qui sont itérativement testées.

Notons que pour les PRM classiques, la méthode locale est chargée de trouver un chemin sans collision entre deux configurations sans prendre en considération les différentes contraintes liées au robot.

Au début de la phase d'apprentissage, le graphe $G = (N, V)$ est initialement vide. On génère une configuration aléatoire dans $\mathcal{CS}_{libre}$. selon une certaine stratégie d'échantillonnage et d'une manière répétitive, ensuite on l'ajoute à l'ensemble des nœuds N . Nous donnerons des exemples de ces stratégies d'échantillonnages dans la deuxième partie de ce chapitre (variations autour de la méthode PRM de base).

Pour le nouveau nœud généré q , on sélectionne une portion N_q de N contenant les nœuds les plus proches de q . L'ensemble N_q inclut l'ensemble des nœuds qui gardent une distance avec q inférieure ou égale à une certaine distance seuil définie selon une certaine métrique instaurée sur $\mathcal{CS}_{libre}$. (nœuds voisins de q).

La figure 2.1 illustre l'étape de construction d'un réseau PRM :

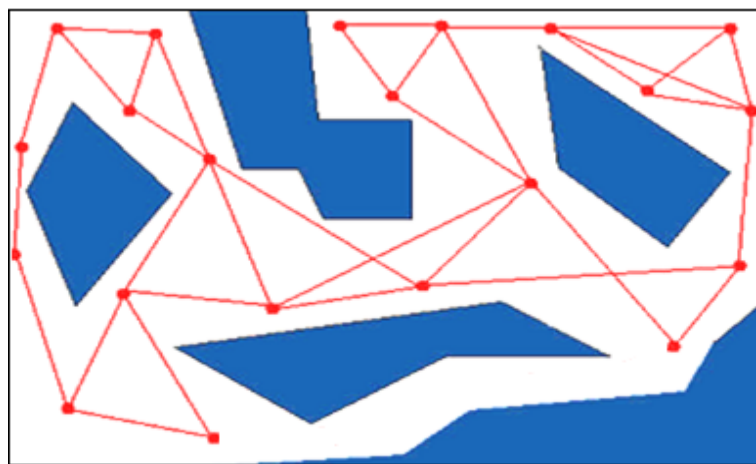


Figure 2.1- La construction du réseau PRM.

2.1.1.2. L'expansion du graphe :

Cette étape permet d'améliorer la connectivité du graphe généré dans l'étape de construction. Elle va intervenir surtout dans les endroits où G est déconnecté alors que $\mathcal{CS}_{libre}$ ne l'est pas, ce qui correspondrait dans la plupart des fois à des régions étroites, donc elle est très utile dans le problème de l'exploration des environnements contenant des passages étroits.

Après la phase d'apprentissage, le graphe G construit contient un ensemble N de n nœuds engendrés aléatoirement et reliés entre eux par des chemins sans collision. Si le nombre de nœuds engendrés est suffisant, l'ensemble N fournit une bonne couverture de l'espace libre. Le graphe G peut néanmoins ne pas refléter la connexité de l'espace libre si celui-ci contient des passages étroits.

Le principe d'expansion du graphe est simple, il s'agit de sélectionner les nœuds qui se trouvent dans les régions difficiles de l'espace libre et de faire une expansion à partir de ces derniers. Ce qui permettrait d'augmenter la densité des nœuds à l'intérieur de ces régions critiques et d'améliorer ainsi la connectivité de G sur tout l'espace.

Une expansion à partir d'un nœud q consiste à ajouter un ou des nœuds dans le voisinage de q (nœuds appartient à N_q). Pour déterminer les nœuds à partir desquels on doit exécuter l'expansion, on va associer un poids $w(q)$ qui est une mesure heuristique de la difficulté de la région dans laquelle se trouve q .

Ces poids sont calculés au début de l'étape d'expansion. Sachant que plus le poids d'un nœud n'est élevé, plus la chance que ce nœud soit choisi pour l'expansion diminue.

La mesure $w(q)$ peut être définie de différentes manières. Une des possibilités consisterait par exemple à compter le nombre de nœuds qui se trouve à une distance inférieure ou égale à un certain seuil. En effet si ce nombre est petit, c'est probablement que les obstacles occupent une grande partie du voisinage de q . On peut donc prendre $w(q)$ égal à l'inverse de ce nombre. On peut aussi considérer la distance de q à la composante connexe la plus proche qui ne le contient pas.

La mesure $w(q)$ doit avoir une grande valeur si q est situé dans une région difficile, de façon à lui donner une grande probabilité d'être choisi et d'être ainsi connecté mais elle doit également être peu coûteuse à calculer. Elle peut être proportionnelle au nombre de nœuds du graphe situés en dessous d'une certaine distance d de q .

En effet, si cette distance est petite, cela veut dire que q se trouve dans une région où deux composantes connexes n'ont pas réussi à se rejoindre bien qu'étant proches, et donc

probablement reliées par un passage étroit. Là aussi, on peut prendre $w(q)$ égale à l'inverse de cette distance. On peut également prendre en compte le comportement de la méthode locale lors de la phase d'apprentissage.

En effet, si lors de la construction du graphe la méthode locale a souvent été mise en échec en essayant de relier q au reste du graphe, alors c'est que q se situe probablement dans une de ces régions difficiles.

La figure 2.2 montre un exemple d'expansion d'un réseau PRM :

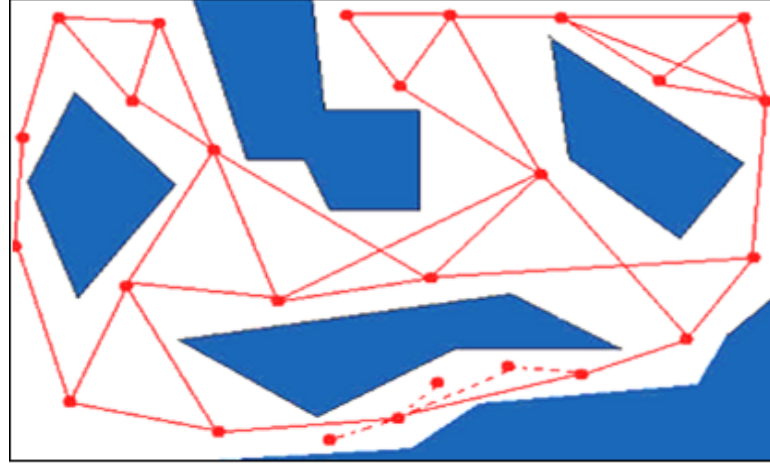


Figure 2.2- l'expansion du réseau.

Dans [10], L. Kavraki définit le degré d_q d'un nœud q comme le nombre de connections reliant q à d'autres nœuds de N à la fin de la phase d'apprentissage. Ce nombre peut être obtenu facilement. Il est raisonnable de penser que lorsque q se situera dans une région difficile de $\mathcal{CS}_{libre}$, d_q sera petit. On peut alors définir la fonction $w(q)$ de la manière suivante :

$$w(q) = \frac{1}{(d_q + 1)} \bigg/ \sum_{i=1}^N \frac{1}{d_i + 1}$$

Le choix de cette mesure dépend bien sûr du problème considéré, tout comme le choix du nombre des nœuds à ajouter.

La phase d'apprentissage peut être décrite dans l'algorithme suivant:

Algorithme de la phase d'apprentissage

// La construction du graphe $N \leftarrow \emptyset, V \leftarrow \emptyset, nb_{noeud} \leftarrow 0, G=(N, V)$ **Tant que** $nb_{noeud} < N_{max}$ **faire** *Tirer une configuration q en utilisant une stratégie d'échantillonnage* **Si** $q \in CS_{libre}$. **alors** $N \leftarrow N \cup \{q\}$ $nb_{noeud} \leftarrow nb_{noeud} + 1$ $N_q \leftarrow$ *un ensemble de voisins de q inclus dans N* **Pour** tout $n \in N_q$ **par ordre de distance croissante faire** **Si** n et q **ne sont pas connectés au G alors** **Si** *il existe un chemin sans collision entre n et q alors* $V \leftarrow V \cup \{(n, q)\}$ **Fin si** **Fin si** **Fin pour** **Fin si****Fin tant que****// L'expansion du graphe** **Pour** chaque nœud $q \in N$ **calculer $w(q)$** **Fin pour** **pour** $i = 1$ **jusqu'à** M *// M : nombre max de nœuds à ajouter.* *Tirer aléatoire un nœud q de N tel que $P(q \text{ soit choisi}) = w(q)$* *Tirer au hasard une configuration $v \in CS_{libre}$ telle que $v \in Vois(q)$* $N \leftarrow N \cup \{v\}$ *Sélectionner dans N les voisins N_v de v* **Pour** tout $n \in N_v$, **par ordre de distance $d(n, v)$ croissante faire** **Si** *il existe un chemin local sans collision entre v et n* $V \leftarrow V \cup \{(v, n)\}$ **Fin si** **Fin pour** **Fin pour**

Le choix des paramètres de cet algorithme (fonction et seuil de distance, nombre de nœuds ou durée d'apprentissage maximum) dépend évidemment du type du problème traité et ils ne peuvent donc pas être fixés de manière générique.

Le choix des voisins d'une nouvelle configuration engendrée q est un paramètre important. En effet, la manière dont on choisira quels nœuds du graphe seront voisins de q influencera sur la connectivité du graphe.

La méthode classique de sélectionner des voisins de q est de considérer tout nœud n du graphe situé à une distance $d(q, n)$ inférieure à un seuil d_{max} comme voisin de q . Le choix de la distance d_{max} devient donc un paramètre critique de l'algorithme puisque si elle est très petite, le graphe risque d'être discontinu, par contre si elle est grande le graphe sera complexe et coûteux en terme de temps de calcul.

D'autre part, il est clair que du fait des nombreux tests de collisions nécessaires pour juger si un chemin est faisable ou non, ainsi que les appels à la méthode locale plusieurs fois lors de la phase de construction du graphe sont les opérations les plus coûteuses en termes de temps de calcul. Il est donc évident que plus la construction des chemins locaux et leur validation seront simples et rapides, plus l'algorithme s'exécutera rapidement.

On peut éviter les appels à la méthode locale qui n'apporteront pas de nouvelle information sur l'espace des configurations en ne considérant comme voisins d'une configuration q que les nœuds qui ont une probabilité élevée d'être reliés à q par la méthode locale.

2.1.2 La phase de recherche :

Dans la phase de recherche, nous allons utiliser le graphe $G = (N, V)$ construit dans la phase précédente pour trouver des chemins reliant une configuration de départ à une configuration d'arrivée.

Différents algorithmes de recherche de chemins sont utilisés dans cette phase, leur principe général peut être décrit de la manière suivante :

Tout d'abord, une configuration de départ et un but q_i et q_f sont données, l'algorithme tente de les relier à deux nœuds n_i et n_f du graphe en utilisant un planificateur local. En cas de succès, il reste à effectuer une recherche dans le graphe pour trouver un chemin reliant n_i et n_f . Ce chemin, quand il existe, est une séquence de chemins locaux, il correspond à une trajectoire admissible.

Au cas où l'algorithme échoue, la phase de recherche est abandonnée, avec un échec pour la planification.

La figure 2.3 illustre un chemin trouvé après la phase de recherche :

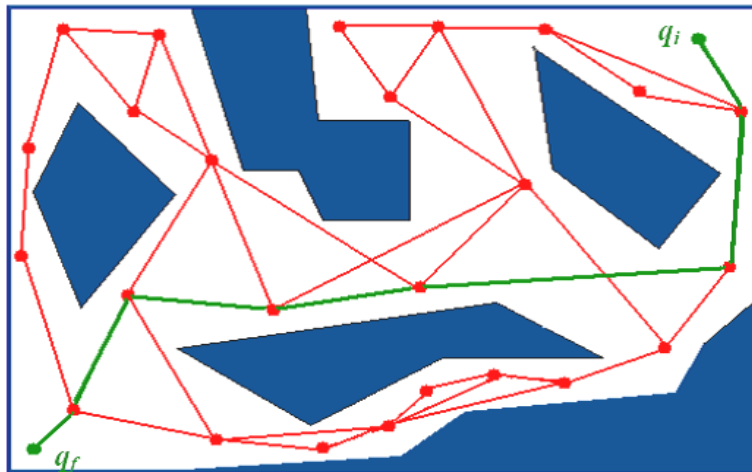


Figure 2.3- La phase de recherche.

Lorsque la phase d'apprentissage d'un environnement est terminée, la phase de recherche peut être répétée autant de fois qu'on le désire, avec des configurations initiales et finales différentes.

Une troisième phase peut être présentée dans les méthodes PRM. Elle consiste à faire un lissage aux chemins trouvés dans la phase de recherche.

2.1.3. Lissage de chemin :

Les chemins construits dans la phase de recherche peuvent être longs et irréguliers à cause du caractère probabiliste des PRM, pour être exécutés par le robot les chemins doivent être lisses et optimisés.

Un processus de raccourcissement (Shortcutting en anglais) est utilisé pour l'optimisation du chemin en éliminant les parties irrégulières.

La méthode classique consiste à remplacer des "morceaux" du chemin en faisant appel à une méthode locale qui cherche un chemin admissible entre les nœuds de la trajectoire (deux à deux) commençant par la configuration initiale. Si la méthode locale trouve un chemin admissible entre deux nœuds et si la longueur de ce dernier est inférieure à celui calculé précédemment entre les mêmes nœuds, alors on le remplace par le nouveau chemin calculé.

Le lissage de chemin a pour but d'adapter ce dernier pour être exécutable par le robot, ceci est fait par l'élimination des pics inclus dans le chemin, selon les différents critères associés au robot.

En répétant plusieurs fois cette opération, pendant une durée donnée ou jusqu'à ce que le gain du lissage soit négligeable, on peut obtenir des chemins lisses et beaucoup plus courts que les chemins construits initialement par les PRM.

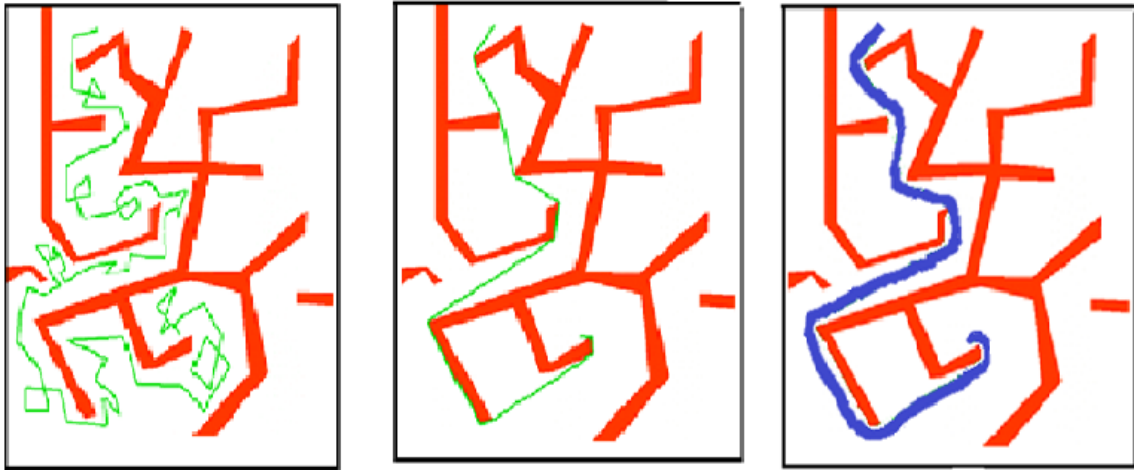


Figure 2.4- lissage de chemin

2.2. Variation autour de la méthode PRM de base :

Depuis l'apparition des méthodes PRM, un certain nombre d'algorithmes de planification de chemins fortement inspirés de l'algorithme PRM ont été développés.

La plupart de ces algorithmes ont été élaborés en remplaçant des parties de l'algorithme original par d'autres stratégies destinées à des problèmes spécifiques de planification de mouvement, particulièrement pour les problèmes dans lesquels le robot doit passer par des passages étroits.

Dans la section suivante, des variations autour la méthode PRM sont décrites.

2.2.1 Les stratégies d'échantillonnage

Les stratégies d'échantillonnage sont utilisées pour générer de nouvelles configurations pendant la phase d'apprentissage. L'approche la plus utilisée est l'échantillonnage aléatoire qui consiste à échantillonner l'espace d'une manière uniforme. Elle fonctionne très bien lorsque la densité d'obstacles est plutôt uniforme. Dans certaines régions, plus particulièrement dans les passages étroits, il est difficile d'obtenir une quantité suffisante d'échantillons aléatoires car la densité de configurations est très faible et le nombre total de configurations pour satisfaire tous l'espace de configuration devient énorme.

2.2.1.1 Echantillonnage uniforme

Dans la stratégie d'échantillonnage uniforme, des configurations sont produites aléatoirement, le processus d'échantillonnage est répété jusqu'à ce que soit trouvée une configuration appartenant à $\mathcal{CS}_{libre}$. En effet, dans ce cas l'échantillonnage uniforme de l'espace des configurations produit une forte densité de nœuds dans chaque composante connexe de l'espace libre avant de réussir à les connecter, par contre il y a une faible probabilité de tirer des configurations à l'intérieur des passages étroits.

2.2.1.2 Echantillonnage gaussien

L'idée de l'échantillonnage gaussien est de faire distribuer les configurations de telle sorte que leur concentration soit élevée près des obstacles selon une distribution gaussienne qui diminue avec la distance aux obstacles [11], cette distribution des nœuds est réalisée avec l'algorithme suivant :

Au début nous choisissons une configuration aléatoire q_1 et une distance aléatoire d en respectant une distribution gaussienne, puis nous choisissons une autre configuration aléatoire q_2 à une distance d de q_1 . On teste ces deux configurations si elles sont en collision, dans ce cas elles seront rejetées toutes les deux, par contre si une des configurations et une seule est libre elle sera ajoutée au graphe (figure 2.5).

La concentration des nœuds près des limites des obstacles est contrôlée par l'écart-type de la distribution gaussienne qui est utilisée pour choisir la distance d .

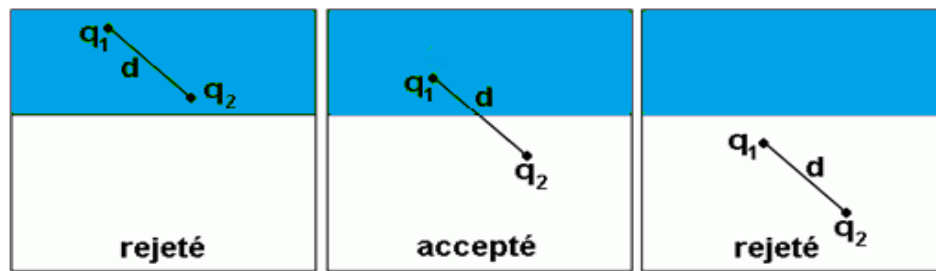


Figure. 2.5– Les trois cas de figure pour l'implantation de l'échantillonnage.

Il est clair que cette méthode d'échantillonnage est plus coûteuse, en termes de temps de calcul, que l'échantillonnage uniforme, néanmoins la qualité de l'échantillonnage obtenu fait que cette méthode est plus efficace que la méthode classique, car elle minimise le nombre de nœuds nécessaires pour capturer la connexité du graphe (on échantillonne seulement aux endroits près des obstacles), par conséquent l'appel à la méthode locale sera minimisé.

Lorsque les obstacles sont convexes, les régions situées près d'au moins deux obstacles (ce qui forme des passages étroits) sont difficiles à capturer, mais l'échantillonnage

Gaussien génère des configurations le long de tous les obstacles, et pas seulement dans les passages étroits. On pourrait éviter cela simplement en tirant non plus deux mais trois configurations distantes de d et en n'en gardant une que si celle-ci est dans l'espace libre, et les deux autres en collision avec des obstacles différents seront rejetés (figure 2.6).

Le nombre de nœuds nécessaires pour capturer la connexité diminue sensiblement, mais il semblerait que cela entraîne une augmentation du temps de calcul pour trouver des configurations utiles, sans que cela soit compensé par la réduction du temps de calcul dû au plus faible nombre de nœuds nécessaires. Cela entraîne aussi une dépendance de l'échantillonnage par rapport à la structure de l'environnement, ce qui peut être gênant (lorsque l'environnement est constitué d'une multitude de petits obstacles par exemple, ou lorsqu'il n'y a qu'un seul obstacle, auquel cas on ne peut pas appliquer cet échantillonnage).

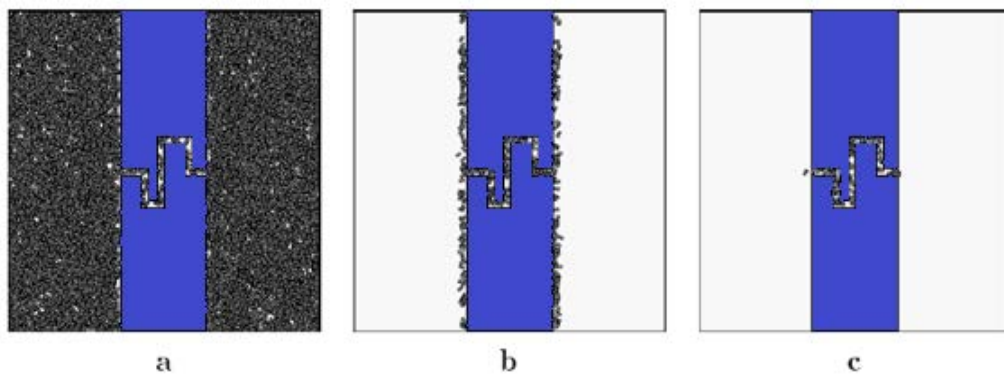


Figure 2.6 **a)** Configurations obtenues avec une distribution uniforme. **b)** Configurations obtenues avec l'échantillonnage Gaussien, en tirant les configurations deux par deux. **c)** Configurations obtenues avec l'échantillonnage Gaussien, en tirant les configurations trois par trois.

2.2.1.3 Addition adaptative de nœuds

Cette méthode est semblable à la technique d'échantillonnages uniformes [12 et 13], mais un seul nouveau nœud est ajouté au graphe, à condition qu'il se trouve dans une région faiblement couverte de l'espace des configurations, ou alors s'il est plus proche de deux composantes non connexes différentes du graphe.

De cette façon nous ajoutons seulement les nœuds qui ont une grande chance d'améliorer la couverture du graphe. Si un nœud donné est situé dans une région faiblement couverte de CS_{libre} , il faut alors déterminer le nombre de ses voisins; un faible nombre de voisins signifie que le nœud doit appartenir à une région faiblement couverte. Si un nœud a deux voisins qui ne sont pas reliés au graphe, il est alors situé près de deux composantes différentes du graphe.

2.2.1.4 OBPRM: Obstacle based PRM

Cette variante de la méthode PRM est décrite dans [14]. Elle propose une méthode de type PRM, pour des robots évoluant dans des environnements encombrés. Les nœuds sont tirés sur ou près des surfaces des obstacles dans l'espace des configurations (figure 2.7). Une stratégie de génération d'OBPRM fournit les informations qui rendent possible les connexions de ces nœuds.

Diverses stratégies d'échantillonnage et de connexion ont été évaluées dans le contexte d'OBPRM dans [15]. Cette méthode est un ensemble d'heuristiques permettant d'améliorer la capture de passages étroits.

La méthode OBPRM propose trois stratégies pour construire une distribution de nœuds dans toutes les régions de $\mathcal{CS}_{libre}$. Ces stratégies utilisent l'information fournie par l'environnement pour guider la génération des nœuds. On obtient trois types de configurations différentes (configurations au contact, configurations libres (proche des surfaces) et enveloppes).

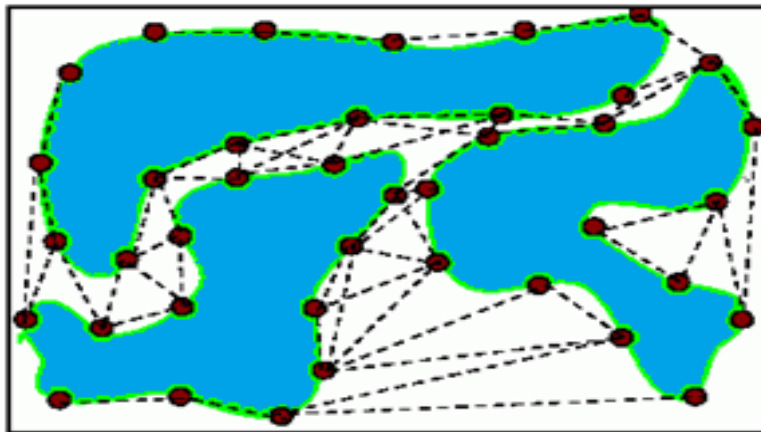


Figure 2.7–Roadmap obtenu par une distribution uniforme sur la surface des obstacles.

2.2.1.5 PRM basé sur la visibilité (visible-PRM) :

L'idée de cette variation est de forcer l'algorithme à explorer en priorité les régions de l'espace libre qu'il n'a pas encore couvertes, c'est à dire les domaines de visibilité des nœuds déjà existants, pour les connecter entre eux. Cette méthode introduite dans [16 et 17] a pour but de construire un réseau adapté d'une façon incrémentale, couvrant tout l'espace libre tout en capturant sa connexité.

De cette manière, tout l'espace finira par être exploré, et toutes les zones pouvant être reliées le seront. Un exemple simple de réseau adapté basé sur la visibilité est donné dans la figure (2.8).

En pratique, on peut constater que cette approche permet de couvrir l'espace libre avec un graphe beaucoup plus petit que le PRM classique, en outre le gain de temps est minimisé ainsi que l'espace mémoire.

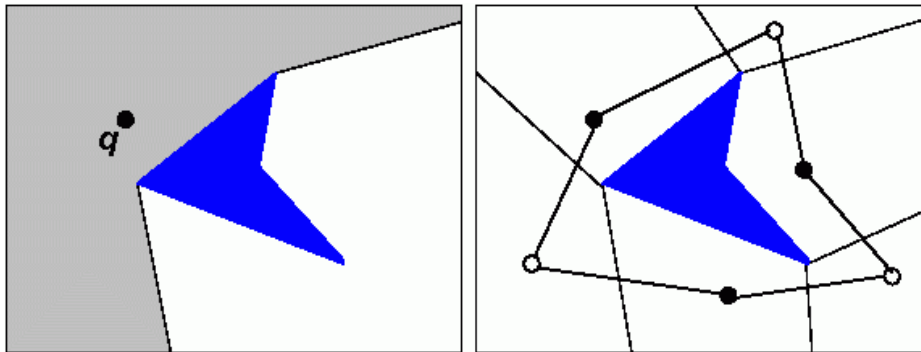


Figure 2.8 *Domaine de visibilité d'une configuration q , et le réseau construit à l'aide du Visible-PRM.*

Puisque des nœuds sont produits aléatoirement de la même façon que le PRM classique, cette méthode n'est pas meilleure que le PRM pour traiter les problèmes de passages étroits. Pour couvrir l'espace libre d'un environnement, cette méthode engendre un grand nombre de configurations libres mais les rejette lorsqu'elles ne sont visibles que d'une composante connexe du réseau et d'une seule. Ceci constitue une énorme perte d'informations.

Conclusion :

A cause de leur simplicité de mise on œuvre, les PRM ont permis de résoudre une large classe de problèmes complexes de planification de mouvement. Elles ont beaucoup évolué et ont été améliorées depuis leur première version de base pour constituer aujourd'hui des approches très efficaces pour la résolution du problème de la planification de trajectoire.

Chapitre III

La planification de mouvement et l'intelligence artificielle.

Introduction

Le calcul de trajectoires optimisées pour les robots mobiles constitue un problème majeur pour les roboticiens. Les techniques d'intelligence artificielle sont fortement implémentées pour résoudre ce type de problème.

Dans son concept général, le problème d'optimisation de trajectoire se divise naturellement en deux phases : recherche des chemins admissibles puis recherche de celui à coût optimal parmi ces derniers.

Plusieurs approches sont introduites dans ce domaine, pour notre part ceux qui nous intéressent sont l'algorithme de parcours de graphe A étoile (A^*) pour rechercher des chemins courts (faisables et non faisables) dans un réseau PRM et les algorithmes génétiques pour l'optimisation de ces chemins afin d'avoir le plus court chemin faisable.

Dans ce chapitre nous allons décrire ces deux techniques.

3.1 L'Algorithme meilleur d'abord ou A^* :

L'algorithme de recherche A^* a été proposé pour la première fois en 1968 par Peter E.Hart, Nils Nilsson, et Bertram Raphael. De par sa simplicité il est souvent exhibé comme un exemple typique des algorithmes de l'intelligence artificielle.

L'algorithme A^* est une extension de l'algorithme de *Dijkstra*, qui sert à trouver le plus court chemin dans un graphe, entre un nœud initial et un nœud final donnés, en utilisant une fonction d'évaluation heuristique sur chaque nœud pour estimer le meilleur chemin y passant, et visite ensuite les nœuds par ordre de cette évaluation heuristique.

Le A^* est un algorithme simple, il ne nécessite pas un prétraitement et il ne consomme que peu de mémoire.

L'algorithme A^* se trouve aussi sous le nom d'algorithme de recherche *meilleur d'abord* (BFS : Best First Search) car il a été créé pour que la première solution trouvée soit l'une des meilleures.

Principe

Dans l'algorithme A^* on commence d'un nœud de départ choisi. On associe à ce nœud un coût (généralement zéro pour le nœud initial). Ensuite on estime la distance qui sépare ce nœud du but à atteindre. La somme du coût et de l'évaluation représente le coût heuristique assigné au chemin menant à ce nœud. Le nœud est alors ajouté à une file d'attente prioritaire, couramment appelée : ***open***. Si elle est vide, il n'y a aucun chemin du nœud initial au nœud d'arrivée, ce qui interrompt l'algorithme. Si le nœud retenu est le nœud d'arrivée, on reconstruit le chemin complet et on arrête.

Si le nœud n'est pas le nœud d'arrivée, de nouveaux nœuds sont créés pour tous les nœuds voisins admissibles (la sélection des voisins dépend du problème à traiter). On calcule le coût de chaque nœud voisin à partir de la somme du coût de son ancêtre et du coût estimé pour atteindre ce nouveau nœud.

L'algorithme maintient également la liste des nœuds qui ont été visités dans une liste appelé habituellement ***close***. Si un nœud nouvellement produit est déjà dans cette liste avec un coût égal ou inférieur, aucune opération n'est faite sur ce nœud ni sur son homologue se trouvant dans la liste.

Ensuite, l'évaluation de la distance du nouveau nœud au nœud d'arrivée est ajoutée au coût pour former l'heuristique du nœud. Ce nœud est alors ajouté à la liste ***open***.

Une fois les trois étapes ci-dessus réalisées pour chaque nouveau nœud voisin, le nœud original pris de la liste ***open*** est ajouté à la liste des nœuds vérifiés. Le

prochain nœud est alors retiré de la liste **open** et le processus recommence. Si on note :

- s_{deb} : le nœud initial.
- s_{fin} : le nœud de but à atteindre.
- $h(s)$: la fonction heuristique estimant le coût de s à s_{fin} .
- $k(s, s')$: le coût du passage de s à s' (voisin de s).
- **Closed** : l'ensemble des nœuds visités.
- **Open** : l'ensemble des nœuds générés mais non encore développés.
- $g(s)$: le coût du trajet de s_{deb} à s .
- $f(s')$: le coût total estimé du trajet de s_{deb} à s' en passant par s .
- $père(s)$: le nœud ayant généré s .

L'algorithme A^* peut réduit dans le pseudo code ci-dessous :

L'Algorithme A^* :

Entrées: $G = (S, A)$, s_{deb} , s_{fin}

Sortie : plus court chemin entre s_{deb} et s_{fin} .

Open = { s_{deb} } ; **close** = $\emptyset$; $g(s_{deb}) \leftarrow 0$; $f(s_{deb}) \leftarrow h(s_{deb})$;

Pour toutes $s \in S$, $g(s) = \infty$

Tant que : **Open** $\neq \emptyset$ faire

Extraire de **Open** l'élément s tel que $f(s)$ est minimale;

Insérer s dans **close** ;

Si $s = s_{fin}$, alors **trv**=1 ; //Solution trouvée

Sinon **trv**=0 // Développement de s

Pour tout s' successeur de s faire

Si s' n'appartient pas à (**close** $\cup$ **Open**) ou $g(s') > g(s) + k(s, s')$ alors

$g(s') = g(s) + k(s, s')$;

$f(s') = g(s') + h(s')$;

$Père(s') = s$;

Insérer s' dans **Open** ;

Fin Si

Fin Pour

Fin Si

Si **trv**=0

retourner échoue // Aucune Solution a été trouvée

Fin Si

Fin Tant Que.

3.2 Les algorithmes génétiques :

Les algorithmes génétiques introduits par John Holland en 1962 [18], font partie de la famille des algorithmes heuristiques, ils tentent de simuler le processus d'évolution naturelle suivant le modèle darwinien dans un environnement donné. Ils utilisent un vocabulaire similaire à celui de la génétique naturelle, cependant, les processus naturels auxquels ils font référence sont beaucoup plus complexes.

Les algorithmes génétiques sont utilisés pour obtenir une solution approchée d'un problème d'optimisation lorsqu'il n'existe pas d'algorithme donnant une solution exacte.

Dans les algorithmes génétiques, les individus sont représentés par des chromosomes constitués de gènes qui contiennent les caractères héréditaires de chaque individu, les individus sont soumis à des processus de *sélection*, *croisement*, et de *mutation* qui s'inspirent des processus naturels de même nom.

Pour un problème d'optimisation donné, un individu représente un point de l'espace d'états (une solution), on lui associe la valeur du critère à optimiser (*évaluation*). On génère ensuite de façon itérative des populations d'individus sur lesquelles on applique des processus de *sélection*, de *croisement* et de *mutation*.

La sélection a pour but de choisir les éléments de la population courante qui vont participer à la construction de la nouvelle population selon le critère considéré.

Le croisement et la mutation assurent l'exploration de l'espace d'états. On commence par la génération d'une population aléatoire d'individus. Pour passer d'une génération i à la génération $i + 1$, les opérations suivantes sont effectuées :

Dans un premier temps, la population est reproduite par le processus de **sélection** où les bons individus se reproduisent mieux que les mauvais. Ensuite, on applique un processus de **croisement** aux paires d'individus (les parents) d'une certaine proportion de la population pour produire des nouveaux (les enfants). Un opérateur de **mutation** est également appliqué à une certaine proportion de la population.

Enfin, les nouveaux individus sont évalués et intégrés à la population de la génération suivante. Un mécanisme de génération de la population initiale et d'opérateurs permet de diversifier la population au cours des générations et d'explorer l'espace de recherche.

Nous allons décrire les différentes étapes des algorithmes génétiques dans la section suivante :

3.2.1 Codages et Initialisation :

Comme première étape, un ensemble de solutions aléatoires est créé, cette étape d'initialisation est très importante pour assurer la convergence des solutions.

Le codage :

Pour utiliser un algorithme génétique sur un problème d'optimisation nous devons disposer d'un principe de codage des individus. Une population est un ensemble d'individus dont chacun représente une solution probable au problème posé, chaque individu est représenté par un ensemble de chromosomes, un chromosome est une suite de gènes, chaque gène représente l'un des paramètres caractéristiques de la solution (*figure 3.1*), on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome et chaque gène sera repérable par sa position.

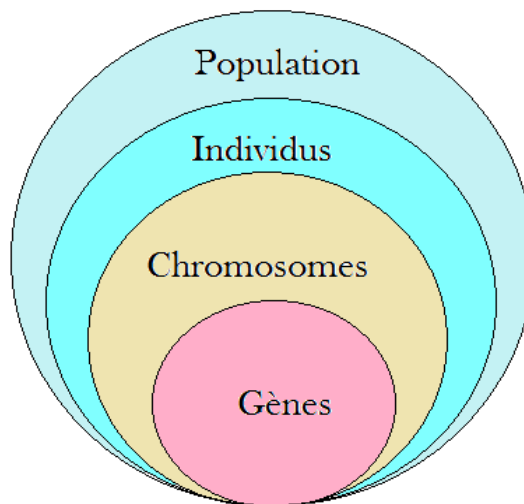


Figure 3.1: structure des solutions dans un algorithme génétique

Chaque gène peut être codé de différentes manières. Il existe principalement trois types de codage : le codage binaire, le codage réel et le codage en base n .

Codage binaire

Ce codage a été le premier à être utilisé dans le domaine des AG. Il consiste à représenter les gènes par une séquence de bits 0 ou 1.

Un exemple de chromosome dont des gènes sont codés en binaire peut être décrit par la figure 3.2:

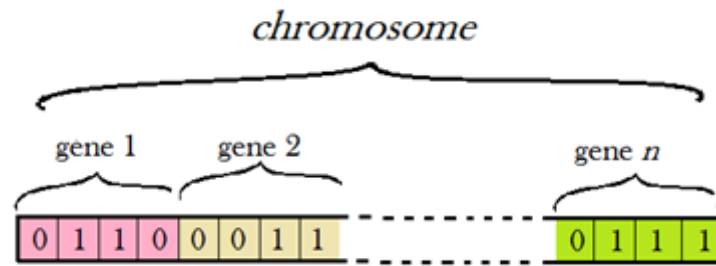


Figure 3 .2: exemple de chromosome dont des gènes codés en binaire.

Codage réel

Ce type de codage représente chaque chromosome par un vecteur dont les composantes sont des valeurs réelles des paramètres du processus d'optimisation.

Avec ce type de codage, la procédure d'évaluation des chromosomes est plus rapide vu l'absence de l'étape de transcodage (du binaire vers le réel).

Codage en base n

Dans ce type de codage, les gènes constituant un chromosome sont des chiffres exprimés dans une base de numération n , ce qui permet de représenter n valeurs discrètes.

3.2.2 L'évaluation des individus :

Dans les algorithmes génétiques on définit une fonction f qui va associer à chaque individu une valeur qui indique son évaluation selon le critère à optimiser.

Cette fonction est appelée fonction d'adaptation (fitness), elle est souvent exprimée par la composition de deux fonctions g et h :

$$f = g \circ h$$

Avec:

$\circ$: loi de composition de fonctions

g : fonction de transformation, pouvant être linéaire, exponentielle, logarithmique, etc.

h : fonction objectif ou de coût, elle dépend de l'objectif recherché.

3.2.3 Sélection :

Dans cette étape on définit quels seront les individus de la population P qui vont être dupliqués dans la nouvelle population P' , la fonction d'adaptation (ou fitness) permet

de déterminer quelles sont les meilleures solutions. Ces solutions participeront à l'étape de reproduction alors que les mauvaises solutions sont supprimées.

On distingue trois types essentiels de méthodes de sélection différentes :

- La méthode de la "loterie biaisée" (*Roulette de Wheel*) de Goldberg.
- La méthode "élitiste".
- La sélection par tournois.

a) La méthode de la loterie biaisée ou roulette de Wheel :

Dans cette méthode, chaque individu a une chance d'être sélectionné proportionnelle à sa performance, donc plus les individus sont adaptés au problème, plus ils ont des chances d'être sélectionnés, chaque individu se voit attribué un secteur d'une roue dont l'angle est proportionnel à son adaptation, (fitness).

On fait tourner la roue et quand elle cesse de tourner on sélectionne l'individu correspondant au secteur désigné par un curseur qui pointe sur un secteur particulier de celle-ci après qu'elle se soit arrêté de tourner (voire figure 3.3).

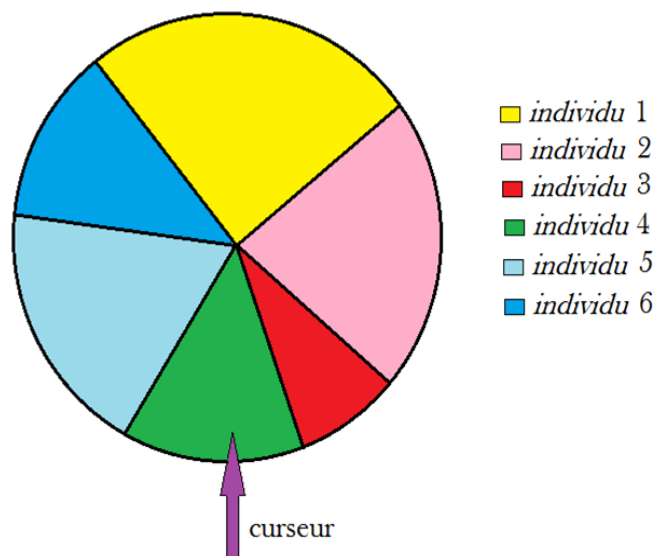


Figure 3.3 sélection par la méthode de la loterie biaisée.

b) La méthode élitisme :

Cette méthode consiste à conserver les meilleurs individus de chaque génération, elle empêche les individus les plus performants et leurs bonnes combinaisons de disparaître au cours de la sélection.

Dans cette méthode, on prend les $\frac{n}{2}$ meilleurs individus d'une population P après l'avoir triée de manière décroissante selon la fonction fitness de ses individus. Ces

individus vont subir les opérateurs de croisement et de mutation pour construire la nouvelle génération P' .

c) La sélection par tournois :

Le principe de cette méthode est d'effectuer un tirage aléatoire avec remise de deux individus de la population P , ensuite on compare leurs fonctions fitness, celui dont la fitness est la plus élevée sera choisi à participer dans la génération suivante (P') avec une probabilité comprise entre 0.5 et 1.

On répète la procédure jusqu'à ce que la nouvelle population soit complète.

3.2.4 Reproduction :

Les meilleures solutions sont combinées afin de donner naissance à de nouvelles solutions qui sont ajoutées à l'ensemble des solutions précédentes, cette étape peut se réaliser par deux processus, le croisement et la mutation :

3.2.4.1 Croisement :

Après avoir appliqué le processus de sélection sur la population P on se retrouve donc avec une population P' de $\frac{n}{2}$ individus et on doit doubler ce nombre pour que notre nouvelle génération soit complète.

Le processus de croisement utilisé par les algorithmes génétiques est une imitation du processus de croisement naturel, il permet de produire des individus qui héritent partiellement des caractéristiques des parents en faisant recombinaison des informations présentes dans le patrimoine génétique des deux parents.

Le processus de croisement peut être fait en un point ou plusieurs points comme l'illustre la figure 3.4 :

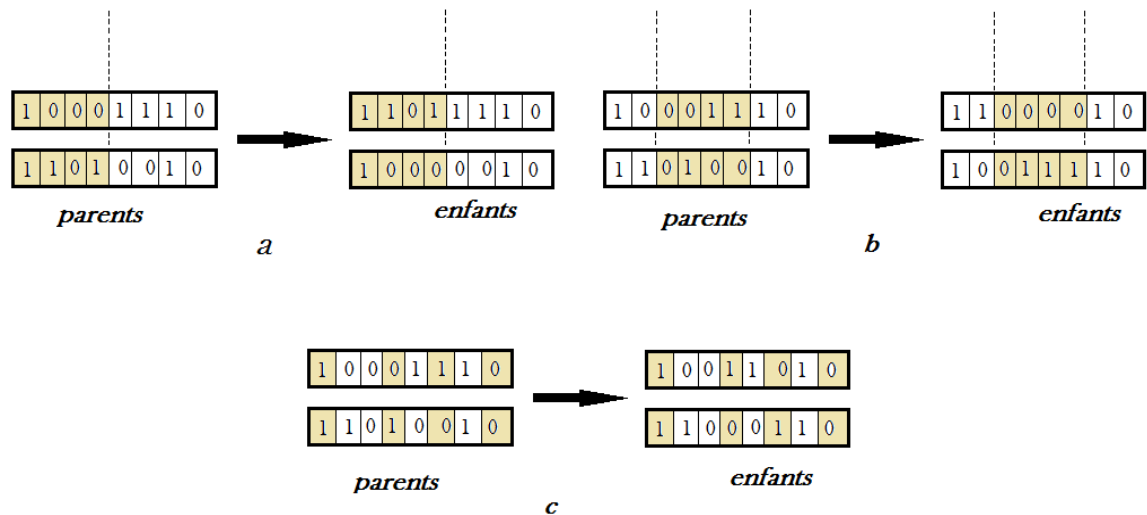


Figure 3.4 : différents type de croisement :

a : croisement en un point ; **b :** en deux points ; **c :** en plusieurs points.

Il existe aussi une autre méthode de croisement utilisée dans le cas des problèmes modélisés par un codage réel, il s'agit du croisement barycentrique [28] où les gènes des descendants sont des combinaisons linéaires des gènes des parents (avec un coefficient de pondération qui définit le taux de participation de chacun des parents aux descendants).

Le croisement de deux gènes g_1 et g_2 donne deux gènes $\hat{g}_1$ et $\hat{g}_2$ telle que :

$$\begin{cases} \hat{g}_1 = \alpha g_1 + (1 - \alpha)g_2 \\ \hat{g}_2 = \alpha g_2 + (1 - \alpha)g_1 \end{cases}$$

Où $\alpha \in [0, 1]$ est un coefficient de pondération.

3.2.4.2 Mutation :

Afin d'éviter des extremums locaux, les solutions sont parfois mutées afin de renouveler l'ensemble des solutions.

Le processus de mutation consiste simplement à introduire un changement à un gène avec une probabilité très faible, généralement comprise entre 0.01 et 0.001.

Pour un gène codé en binaire il suffit d'inverser un de ses bit qui peut être choisi aléatoirement ou en une position bien particulière.

On peut donc illustrer le processus de mutation dans la figure suivante :

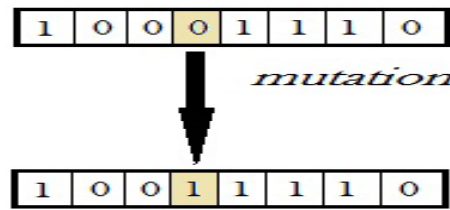


Figure 3.5: le processus de la mutation

Dans le cas d'un codage réel, il existe principalement deux types de mutation: mutation uniforme et mutation non uniforme [20]. Supposons que la probabilité de mutation est fixée, une sélection aléatoire d'un gène g_k permet de décider s'il va être modifié ou non, nous supposons aussi que le gène prend ses valeurs dans un intervalle $[g_{min}, g_{max}]$.

Pour la mutation uniforme, qui est une simple extension de la mutation binaire, on remplace le gène g sélectionné par une valeur quelconque tirée aléatoirement dans l'intervalle $[g_{min}, g_{max}]$.

Pour la mutation non uniforme, le calcul de la nouvelle valeur d'un gène est un peu plus complexe. Le gène g subit des modifications importantes durant les premières générations, puis graduellement décroissantes au fur et à mesure que l'on progresse dans le processus d'optimisation.

Pour une génération n , une valeur binaire tirée aléatoirement va décider du signe du changement (positif ou négatif).

La nouvelle valeur g' du gène g est donnée par :

$$g' = \begin{cases} g + \Delta(n, g_{max} - g) & \text{si } rand = 0 \\ g - \Delta(n, g - g_{min}) & \text{si } rand = 1 \end{cases}$$

Où $\Delta(n, g)$ est une fonction qui définit l'écart entre la nouvelle valeur et la valeur initiale à la génération n et $rand$ est nombre aléatoire qui prend les valeurs 0 ou 1.

Si l'intervalle de variations du gène g n'est pas connu, une mutation gaussienne est souvent utilisée. Le principe de base de ce type de mutation est d'ajouter un bruit gaussien centré $N(0, \delta)$ au gène que l'on désire faire muter :

$$g' = g + N(0, \delta).$$

Où δ représente la variance.

L'opérateur de mutation offre aux algorithmes génétiques une propriété très importante : l'ergodicité (tous les points de l'espace de recherche peuvent être atteints), il est généralement effectué avec une probabilité très faible (de l'ordre de

1%), d'ailleurs un algorithme génétique peut converger sans l'opérateur de croisement et certaines applications fonctionnent de cette manière [21].

3.2.5 L'évolution d'un algorithme génétique :

L'étape de reproduction est répétée jusqu'à faire émerger des bonnes solutions. Plusieurs critères d'arrêt de l'algorithme sont possibles : le nombre de générations peut être fixé à priori (temps constant) ou l'algorithme peut être arrêté lorsque la population n'évolue plus suffisamment rapidement.

Le principe de fonctionnement des algorithmes génétiques peut se résumer dans l'algorithme suivant :

Principe d'un algorithme génétique

Soit P une population initiale.

Répéter

- Faire évaluer les individus de la population P selon le critère à optimiser.
- Sélectionner dans P un ensemble des individus à l'aide d'une méthode de sélection.
- croiser les individus sélectionnés deux à deux afin d'obtenir une nouvelle population P' .
- Appliquer un opérateur de mutation aux individus de la population P' .
- Remplacer P par la nouvelle population P' .

Jusqu'à ce que la règle d'arrêt soit satisfaite.

3.3 Algorithmes Génétiques Hiérarchiques (AGH)

Malgré que les algorithmes génétiques standards (AGS) fournissent de très bons résultats pour la résolution des problèmes d'optimisation difficiles où d'autres techniques ont échoué, ils présentent quand même des limites dans les problèmes qui s'intéressent non seulement à l'optimisation des paramètres du système, mais aussi à l'optimisation de sa structure (topologie).

Dans l'AGS la structure des chromosomes doit être fixé ou prédéfini, mais dans certains problèmes d'optimisation il est difficile de le faire, ce qui conduit à l'utilisation des AGH qui surmontent ces contraintes [22].

Les AGH diffèrent des AGS dans la structure des chromosomes. Ils présentent une structure hiérarchique qui se compose de deux types de gènes: les gènes de paramètres et les gènes de contrôle. Les gènes de paramètres portent l'information génétique du chromosome (ensemble des valeurs qui définissent les différentes caractéristiques de l'individu), tandis que les gènes de contrôle ont un rôle d'activation ou de désactivation des gènes de paramètres.

3.3.1 Forme du chromosome hiérarchique :

La structure générale d'un chromosome hiérarchique se compose d'un seul niveau de gènes paramétriques et un ou plusieurs niveaux de gènes de contrôle comme l'illustre la figure 3.6.

Cette structure est basée sur le principe naturel de l'ADN où les gènes paramétriques (analogie aux gènes structuraux en ADN) et les gènes de contrôle sont sous forme de bits.

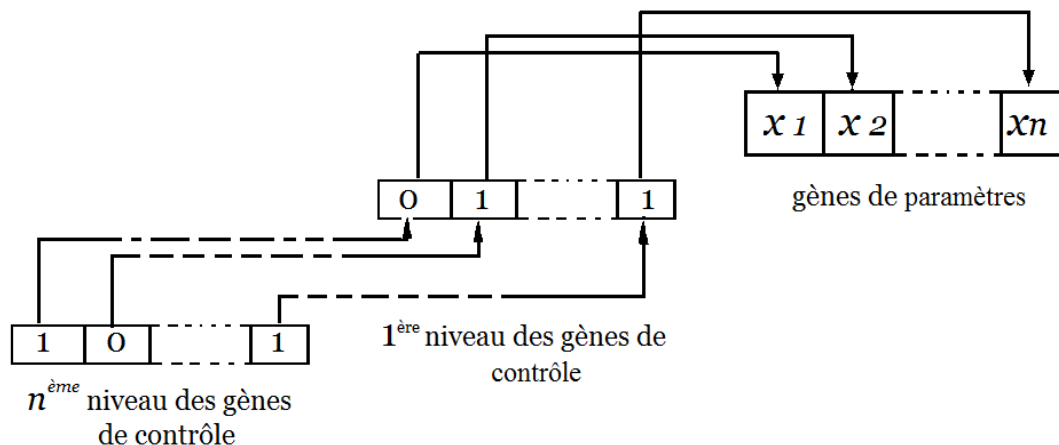


Figure 3.6 la structure hiérarchique d'un chromosome

L'activation des gènes paramétriques est régie par la valeur du premier niveau du gène de contrôle, qui est régi à son tour par la valeur du deuxième niveau du gène de contrôle, et ainsi de suite.

Un gène de contrôle g_i^n (le $i^{\text{ème}}$ gène du $n^{\text{ème}}$ niveau) prend une valeur binaire, s'il est égal à 1 ($g_i^n = 1$), le gène de paramètre (ou de contrôle) associé à ce gène sera activé, et désactivé si $g_i^n = 0$.

Un gène de contrôle peut être associé à un seul gène ou bien à tout un bloc de gènes. La structure d'un chromosome contient à la fois des gènes actifs et inactifs. Il faut noter que les gènes inactifs existent toujours dans le chromosome et peuvent être transmis ensuite aux nouvelles générations. Cette architecture hiérarchique implique que le chromosome contient plus d'informations que celle de la structure conventionnelle des AGS. Par conséquent, la puissance artificielle est largement améliorée.

3.3.2 Opérateurs génétiques :

Les opérateurs génétiques de base peuvent être utilisés sans aucune contrainte. La mutation et le croisement peuvent être appliqués indépendamment à chaque niveau de gènes de contrôle. Cependant, les opérateurs génétiques qui affectent les niveaux hauts des gènes de contrôle et produisent des changements des gènes actifs mènent à des changements multiples dans les niveaux inférieurs. C'est la raison pour laquelle, un AGH est capable d'obtenir le meilleur ensemble de paramètres d'un système avec une topologie ou structure minimisée.

Conclusion :

Dans ce chapitre nous avons décrit deux techniques d'intelligence artificielle, la première est l'algorithme de recherche du plus court chemin **A étoile**, cet algorithme est utilisé dans notre cas pour trouver le plus court chemin qui relie deux configurations dans un réseau PRM. La deuxième technique concerne les Algorithmes génétiques qui présentent un outil d'optimisation très utilisé dans les problèmes de planification de mouvement.

Chapitre IV

Planification de mouvement pour un robot mobile non holonome

Introduction :

Dans les approches décrites dans les chapitres précédents, la planification de mouvement est formulée dans un espace défini sur un ensemble de paramètres décrivant le mouvement du robot et qui sont indépendants les uns des autres.

Ces approches fournissent des résultats probants pour les problèmes de navigation d'un robot assimilé à un point évoluant dans son espace de configuration, elles effectuent la planification en ne prenant en compte que les contraintes géométriques liées à l'environnement (les obstacles). Ceci conduit à penser que tout mouvement planifié par l'une de ces méthodes est exécutable par le robot. Ainsi, la faisabilité d'une trajectoire devient une notion équivalente à l'existence du chemin.

Cependant pour certains systèmes tels qu'un robot mobile de type voiture (CLR), le problème de planification de mouvement faisable ne peut être abordé par les aspects considérés jusqu'ici. Ceci est dû à l'existence d'une liaison particulière entre certains

des paramètres de configuration et de vitesse du robot rendant la génération d'un mouvement faisable encore plus difficile.

Les robots mobiles de type voiture soumis à des contraintes cinématiques liées au roulement sans glissement (contraintes dites non holonomes) font que le robot suit une certaine trajectoire imposée par sa structure mécanique, par conséquent tout chemin libre ne correspond pas nécessairement à une trajectoire faisable. Ce type de robots mobiles sont dits non holonomes.

Il existe un autre type de robots mobiles qui peuvent effectuer des mouvements dans toutes les directions, ces robots sont dits holonomes.

Dans ce chapitre nous essayons d'expliquer brièvement la notion de contrainte non holonome, ensuite une modélisation cinématique d'un robot mobile de type voiture sera présentée, et nous terminons ce chapitre en citant les différentes approches de planification de mouvement non holonome existantes.

4.1 Cinématique d'un robot mobile et notion de non-holonomie :

4.1.1 Roulement sans glissement :

Le déplacement des véhicules à roues est dû aux frottements aux points de contact entre les roues et le sol. Dans des conditions parfaites, le roulement de la roue sur le sol est supposé sans glissement (r.s.g.), c'est-à-dire que la vitesse relative de la roue par rapport au sol au point de contact est nulle.

Théoriquement, pour satisfaire cette condition, il faut vérifier les hypothèses suivantes:

- Le contact entre la roue et le sol est ponctuel.
- Les roues sont indéformables, de rayon r .

La roue est donc assimilée à un disque rigide d'épaisseur supposée nulle, elle appartient à un plan vertical à un sol supposé plan, ce qui assure un contact ponctuel entre la roue et le sol, c'est ce qu'illustre la figure 4.

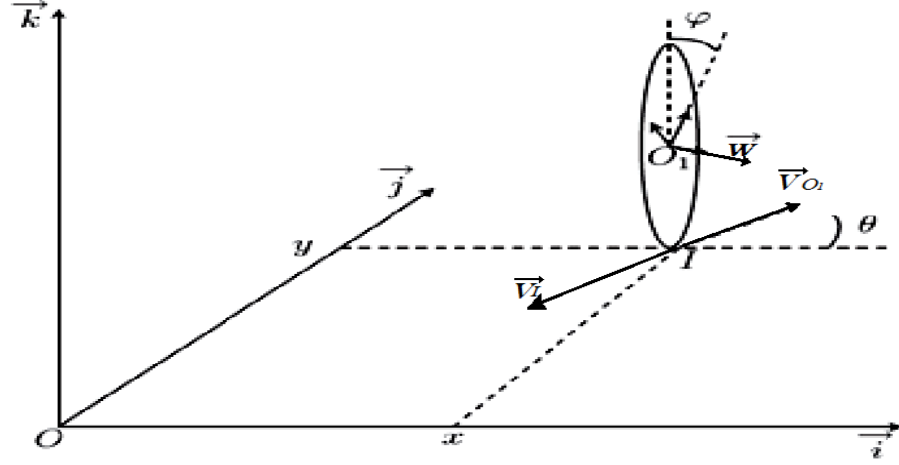


Figure 4.1 : description d'une roue

Mathématiquement, Le roulement sans glissement peut s'exprimer par :

$$\vec{V}_I = \vec{V}_{O_1} + \vec{W} \wedge \overrightarrow{O_1 I} = \vec{0} \quad (4.1)$$

Avec $\vec{W} = \dot{\varphi}$ est la vitesse de rotation de la roue.

Si on pose (x, y) les coordonnées du point O_1 dans le repère fixe $\mathcal{R}_{(O, \vec{i}, \vec{j}, \vec{k})}$, et r le rayon de la roue, θ l'angle entre le plan de la roue et le plan $(O, \vec{i}, \vec{k})$, la relation (4.1) sera :

$$\begin{aligned} \vec{V} \left(I / \mathcal{R}_{(O, \vec{i}, \vec{j}, \vec{k})} \right) &= \dot{x} \vec{i} + \dot{y} \vec{j} + (\dot{\theta} \vec{k} + \dot{\varphi} (-\sin \theta \vec{i} + \cos \theta \vec{j})) \wedge (-r \vec{k}) \\ &= (\dot{x} - r \dot{\varphi} \cos \theta) \vec{i} + (\dot{y} - r \dot{\varphi} \sin \theta) \vec{j} \\ &= \vec{0} \end{aligned} \quad (4.2)$$

D'après l'équation (4.2) Les contraintes dont les roues sont soumises peuvent s'exprimer :

$$\begin{cases} \dot{x} - r \dot{\varphi} \cos \theta = 0 \\ \dot{y} - r \dot{\varphi} \sin \theta = 0 \end{cases} \quad (4.3)$$

4.1.2 Contraintes non holonomes :

En général, on dit qu'un mobile est soumis à une contrainte cinématique si son mouvement est contraint par une liaison G qui est une équation du premier ordre en fonction du temps t , des paramètres de sa configuration q et de leurs dérivées $\dot{q}$ par rapport à t , cette liaison réduit l'ensemble des mouvements possibles du robot, autrement dit ses degrés de liberté.

La liaison $\mathbf{G}$ s'exprime comme une contrainte linéaire sur les vitesses, elle peut se mettre sous la forme :

$$\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{0} \quad (4.4)$$

Dans le cas où $\mathbf{G}$ est une fonction intégrable, c'est à dire qu'il existe une fonction $\mathbf{F}$ telle que $\frac{\partial \mathbf{F}}{\partial \mathbf{q}} = \mathbf{G}$, on dit que $\mathbf{G}$ est une contrainte holonome et le robot soumis à ce type de contraintes est un robot holonome.

Dans le cas où $\mathbf{G}$ est une fonction non-intégrable, on dit que $\mathbf{G}$ est une contrainte non holonome et le robot soumis à cette contrainte est un robot non holonome.

Ce type de contrainte empêche le système d'effectuer certains mouvements instantanément, il réduit l'espace des vitesses accessibles par le robot sans réduire son espace accessible.

En général, il n'est pas évident de juger si une contrainte est intégrable ou non, l'application du théorème de Frobenius [26] nous permet de savoir si un système est holonome ou non. Ce théorème utilise la notion de ce qu'on appelle : crochet de Lie et Algèbre de Lie.

Crochet de Lie et Algèbre de Lie :

Soit $\mathcal{F}$ une application de classe $\mathcal{C}^\infty$ de $\mathcal{R}^n$ dans $\mathcal{R}^n$. On peut associer à cette application une équation différentielle tel que :

$$\begin{array}{ccc} \mathcal{F} : \mathcal{R}^n & \longrightarrow & \mathcal{R}^n \\ x & \longrightarrow & \mathcal{F}(x) = \frac{\delta x}{\delta t} . \end{array}$$

$\mathcal{F}(x)$ peut être considérée comme une équation différentielle qu'on appellera « champ de vecteur » (si δt est le temps, $\mathcal{F}$ représente donc le vecteur vitesse).

Le crochet de Lie est un opérateur qui associe à deux champ de vecteurs $\mathcal{F}$ et G , un nouveau champ de vecteurs noté $[\mathcal{F}, G]$ tel que :

$$[\mathcal{F}, G] = \frac{\delta G(x)}{\delta x} \mathcal{F}(x) - \frac{\delta \mathcal{F}(x)}{\delta x} G(x)$$

Le crochet de lie de deux vecteurs définit la variation d'un champ de vecteur par rapport à l'autre.

L'Algèbre de Lie est un espace vectoriel muni du crochet de Lie, engendré par des champs de vecteur et leurs crochets de Lie.

Théorème de Frobenius

Soit un système de configuration q , de dimension n , soumis à un ensemble de contraintes indépendantes s'écrivant sous la forme $A^T(q)\dot{q} = 0$.

Soit $B(q) = (b_1(q) \ b_2(q) \dots b_m(q))$ une matrice de rang plein m , orthogonale à $A(q)$ sur tout l'espace des configurations. Soit enfin l'algèbre de Lie $\mathcal{L}ie \{ b_1(q) \ b_2(q) \dots b_m(q) \dots b_p(q) \}$ de dimension p , avec $m \leq p \leq n$, alors, le système possède $n - p$ contraintes intégrables.

Prenant l'exemple de la roue, on peut réécrire le système (4.3) de la manière suivante :

$$\begin{cases} -\dot{x} \sin \theta + \dot{y} \cos \theta = 0 \\ \dot{x} \cos \theta + \dot{y} \sin \theta = -r \dot{\phi} \end{cases} \quad (4.5)$$

Si on pose $q = [x \ y \ \theta \ \phi]^T$ la configuration de la roue et $\dot{q} = [\dot{x} \ \dot{y} \ \dot{\theta} \ \dot{\phi}]^T$ sa dérivée, d'après les équations du système (4.5) on déduit donc :

$$A(q) = \begin{pmatrix} -\sin \theta & \cos \theta \\ \cos \theta & \sin \theta \\ 0 & 0 \\ 0 & r \end{pmatrix}$$

$B(q)$ est une matrice orthogonale à $A(q)$ (c à d : $A^T(q) B(q) = 0$) donc :

$$B(q) = \begin{pmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \\ -\frac{1}{r} & 0 \end{pmatrix}$$

Si on pose : $b_1(q) = [\cos \theta \ \sin \theta \ 0 \ -1/r]^T$ et $b_2(q) = [0 \ 0 \ 1 \ 0]^T$, On calcule alors les crochets de Lie successifs à partir de $b_1(q)$ et $b_2(q)$:

$$b_3(q) = [b_1(q), b_2(q)] = (\sin \theta \quad -\cos \theta \quad 0 \quad 0)^T$$

$$b_4(q) = [b_2(q), b_3(q)] = (\cos \theta \quad \sin \theta \quad 0 \quad 0)^T$$

On constate que $\{ b_1(q) \ b_2(q) \ b_3(q) \ b_4(q) \}$ engendrent une algèbre de Lie de dimension 4, égale à celle de q (c à d : $p = n = 4$). $n - p = 0$, donc on peut conclure qu'il n'existe pas de contrainte intégrable.

Les équations du système (4.5) caractérisent le r.s.g. d'une roue sur le sol, ce sont donc des contraintes non holonomes.

4.2 Modélisation du robot mobile de type voiture :

Considérons un robot de type voiture $\mathcal{A}$ dont la structure mécanique est munie de deux roues motrices à orientation fixe en arrière et de deux roues directionnelles en avant, le robot se déplace sur une surface considérée plane.

Le modèle le plus simple d'un robot mobile de type voiture est un rectangle avec un point de référence $\mathbf{M}$ qui est le point médian de l'axe des roues arrière et une orientation donnée par l'angle θ qui correspond à la direction de l'axe principal du robot par rapport au référentiel $\mathcal{F}(\mathbf{O}, \mathbf{i}, \mathbf{j})$ de l'environnement $\mathcal{W}$.

La configuration $\mathbf{q}$ du robot peut être définie par le quadruplet $\mathbf{q} = [x, y, \theta, \phi]$ de $\mathcal{R}^2 \times [0, 2\pi[\times [-\phi_{max}, \phi_{max}]$,

Où x et y définissent la position du point référence $\mathbf{M}$ dans $\mathcal{F}(\mathbf{O}, \mathbf{i}, \mathbf{j})$, et ϕ représente l'angle de braquage du robot, l'espace des configurations $\mathcal{CS}$ du robot est donc de dimension quatre.

Pour une orientation des roues donnée le mouvement du robot est soit une translation pure, soit une rotation pure. Dans ce dernier cas, les axes des quatre roues se croisent en un même point $\mathbf{C}$ qui est le centre instantané de rotation du robot (CIR).

On définit alors l'angle $\phi \leq \phi_{max}$, l'angle de braquage de $\mathcal{A}$, qui est égal à l'angle entre l'axe principal de $\mathcal{A}$ et la perpendiculaire de la droite passant par $\mathbf{C}$ et le point de référence $\mathbf{M}$ (droite tangente en $\mathbf{M}$ à l'arc parcouru par la voiture).

ϕ_{max} est l'angle de braquage maximum (généralement compris entre 18° et 45°). L'angle de braquage des roues directionnelles impose sur le robot à suivre un mouvement tangent à sa direction en un point donné comme l'illustre la figure 4.2 :

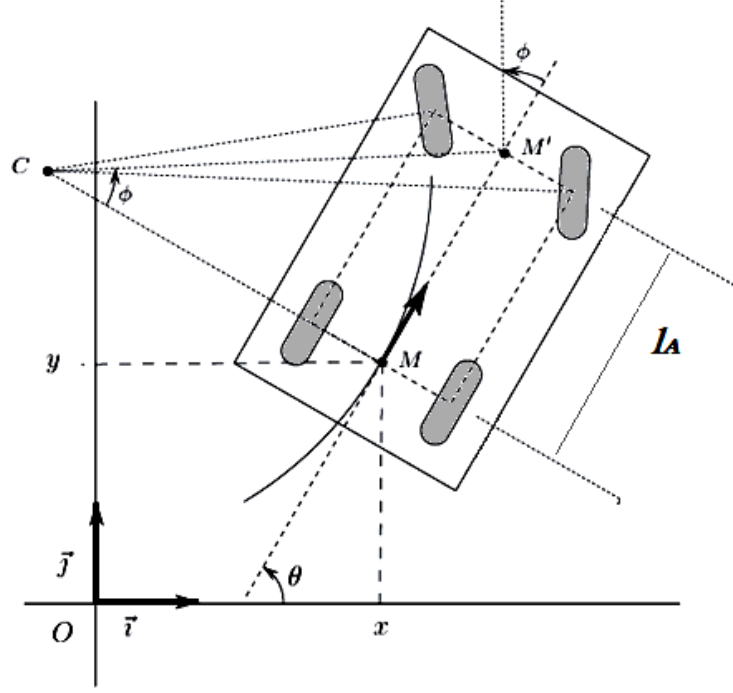


Figure 4.2: un robot de type voiture.

On définit aussi le rayon de braquage ρ (ou rayon de courbure) comme étant la distance entre C et M .

$$\rho = \frac{l_A}{\tan \phi} \quad (4.6)$$

Où l_A représente la distance entre les deux axes des roues avant et arrière

Une autre contrainte est due au fait que ϕ est majoré par ϕ_{max} pour des raisons mécaniques ($\phi < \phi_{max}$).

Par conséquent, le rayon de braquage est compris entre le rayon de giration minimum $\rho = \rho_{min}$ correspondant à $\phi = \phi_{max}$ et l'infini correspondant à $\phi = 0$.

Si on pose : $q = [x \ y \ \theta \ \phi]^T$, le modèle cinématique correspondant au robot mobile de type voiture peut s'exprimer comme suit :

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = \frac{v}{\rho} \tan \phi \\ \dot{\phi} = \omega \end{cases} \quad (4.7)$$

Avec $v = \rho \dot{\phi}$ la vitesse linéaire, $\dot{\theta}$ la vitesse angulaire du point de référence M du robot, et ω la vitesse de braquage des roues directrices.

Du point de vue du conducteur, une voiture possède deux commandes : l'accélération et la direction, et puisque nous ne intéressons qu'à la cinématique du robot, le système (4.7) peut être donc écrit sous la forme :

$$\dot{q} = A(q) \dot{q} + B(q) U \quad (4.8)$$

$$\text{Avec } A(q) = 0, \quad B = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ \frac{1}{l} \tan \phi & 0 \\ 0 & 1 \end{bmatrix} \quad \text{et} \quad U = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

4.3 Non holonomie du robot de type voiture :

Dans le problème classique de la planification de mouvement, un robot peut se déplacer dans n'importe quelle direction, c'est à dire que le robot possède autant de degrés de liberté que de paramètres de configuration. Or en réalité le robot de type voiture est soumis lors de ses déplacements à des contraintes non holonome qui réduisent l'ensemble des mouvements qu'il peut réaliser.

La non holonomie du robot de type voiture vient de celle de la roue qui ne peut pas faire une translation instantanée parallèlement à son axe, un tel déplacement nécessite une ou plusieurs manœuvres, et la liaison G peut être écrite ainsi :

$$G = -\dot{x}(t) \sin \theta(t) + \dot{y}(t) \cos \theta(t) = 0 \quad (4.9)$$

Si on pose $b_1(q) = [0 \quad 0 \quad 0 \quad 1]^T$ et $b_2(q) = [\cos \theta \quad \sin \theta \quad \frac{1}{l} \tan \phi \quad 0]^T$, le système (4.8) peut donc s'écrire :

$$\dot{q} = b_1(q) \omega + b_2(q) v \quad (4.10)$$

Un calcul rapide des crochets de lie donne :

$$b_3(q) = [b_1(q), b_2(q)] = (0 \quad 0 \quad \frac{1}{l \cos^2 \phi} \quad 0)^T$$

$$b_4(q) = [b_2(q), b_3(q)] = (\frac{\sin \theta}{l \cos^2 \theta} \quad \frac{-1}{l \cos \theta} \quad 0 \quad 0)^T$$

$\{b_1(q) \quad b_2(q) \quad b_3(q) \quad b_4(q)\}$ engendrent une algèbre de Lie de dimension 4, égale à celle de q , on peut conclure qu'il n'existe pas de contrainte intégrable.

4.4 Planification de mouvement des robots non holonomes :

Les algorithmes utilisant des méthodes classiques (citées précédemment) étant difficiles à implémenter pour la planification de mouvement des robots mobiles non holonomes, il faudra donc trouver d'autres méthodes capables d'adapter ces derniers à ce type des problèmes.

Bien que de nombreuses solutions aient été proposées, il a toujours fallu prendre en compte de nouveaux facteurs, ainsi on a voulu étendre ce problème à des robots mobiles de type voiture, ce qui a conduit à prendre en compte les contraintes cinématiques (non holonomie).

Énormément de travaux ont été publiés autour de ce sujet, Lamiroux et Laumond dans [23] décrivent des adaptations des méthodes existantes pour résoudre ce type des problèmes.

Les méthodes locales les plus connues sont les courbes de Reeds and Shepp [24], et les courbes de Dubins [25] pour les modèles cinématiques de véhicule, ces courbes sont très spécifiques à un modèle donné et ne peuvent être utilisées dans le cas où le modèle du véhicule est quelconque.

Nous présentons dans cette section les approches principales de planification de mouvement non holonome.

4.4.1 Les chemins sans manœuvres :

Le problème de planification d'un chemin sans manœuvres a été abordé pour la première fois par Dubins [25]. Dans un environnement qui ne contient aucun obstacle, Dubins a montré qu'une solution de longueur minimale peut être composée de la concaténation d'au plus trois portions élémentaires de segments de droites et d'arcs de cercle dont le rayon est minoré (figure 4.3).

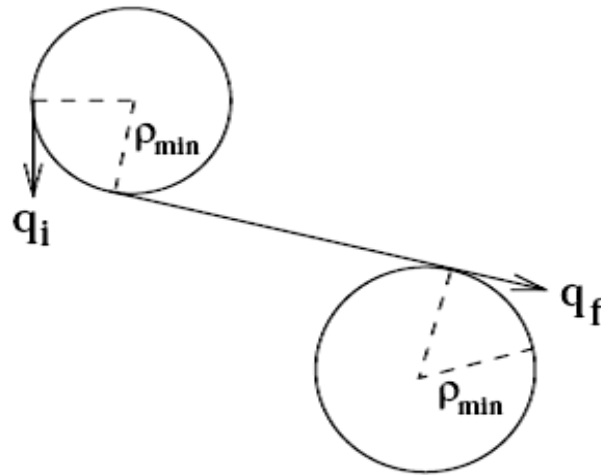


Figure 4.3 : Exemple de plus court chemin de Dubins.

Les courbes de Dubins ne peuvent être utilisées comme telles quand l'environnement est encombré d'obstacles, ce problème a été abordé par Laumond dans [28], il a considéré un robot circulaire non holonome se déplaçant dans un environnement contenant un nombre fini d'obstacles, les chemins engendrés sont des chemins sans manœuvres composés, comme dans le cas de Dubins, de segments de droites et d'arcs de cercle de rayon minoré (figure 4.4).

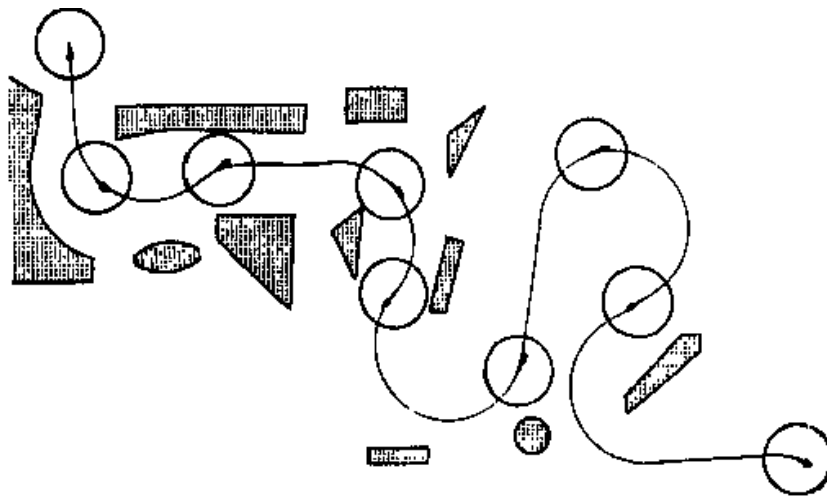


Figure 4.4 plus court chemin de Dubins en présence d'obstacles.

4.4.2 Les chemins avec manœuvres :

Dans le cas où les manœuvres sont permises, Reeds et Shepp dans [25] ont pu montrer un résultat similaire à celui de Dubins. Pour le robot de Reeds et Shepp, les trajectoires optimales présentent des points de rebroussement où le robot passe de la marche avant à la marche arrière. En l'absence d'obstacle, il existe un plus court chemin joignant deux configurations, ce chemin est constitué d'au plus cinq arcs, qui sont des arcs de

cercle de rayon minoré ou des segments de droites, et contient au plus deux points de rebroussement.

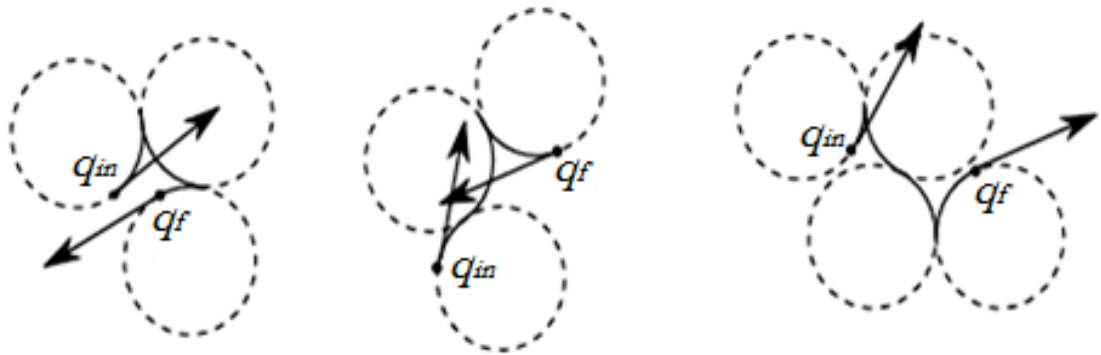


Figure 4.5 exemples de plus courts chemins de Reeds et Sheep.

Dans un environnement contenant des obstacles, les chemins de Reeds et Sheep ne sont pas utiles, Laumond dans [28] a développé un algorithme afin de résoudre ce problème. Cet algorithme est constitué de trois phases. La première consiste à générer un chemin holonome sans collision aux obstacles. Une fois déterminé, ce chemin est transformé en un chemin faisable par un processus itératif consistant à subdiviser la solution initiale en chemins de longueur optimale sans collision qui vérifient les contraintes de la non-holonomie.

Cette seconde phase ne garantit pas l'optimalité en nombre de manœuvres, une troisième phase est appliquée afin d'éliminer d'une manière aléatoire certaines manœuvres "inutiles".

Conclusion :

Dans ce chapitre nous avons vu que la planification de mouvement pour un robot mobile non holonome (CLR) est difficile même en l'absence d'obstacles, cette difficulté vient des contraintes de non holonomie qui exige de ce type de robot de suivre certaines trajectoires imposées par l'angle de braquage des roues directionnelles. Par conséquent, tout chemin libre ne représente pas nécessairement un chemin faisable pour les robots mobiles non holonomes.

Chapitre V

Simulation et résultats obtenus

Introduction :

Ce chapitre décrit l'essentiel de notre travail effectué dans le cadre de la résolution du problème de génération des mouvements optimaux pour un robot mobile non holonome de type voiture. Il décrit les différentes étapes suivies pour résoudre ce problème ainsi que les difficultés qu'on a rencontrées allant de la représentation de l'environnement où se déplace le robot jusqu'à l'exécution des mouvements calculés. Dans ce cadre, l'algorithme général élaboré est basé sur la méthode des réseaux probabilistes PRM associée à un algorithme de recherche du plus court chemin entre une configuration de départ et une configuration d'arrivée. Ce chemin n'est pas nécessairement faisable à cause des contraintes de non holonomie, il a fallu donc l'adapter pour qu'il soit faisable par le robot, pour ce faire nous avons utilisé un algorithme de suivi de chemin pour extraire un ensemble de commandes menant le robot d'une configuration initiale à une configuration finale, ensuite nous avons fait appel aux algorithmes génétiques dans le but d'optimiser ces commandes.

5.1. Environnement :

L'environnement dans lequel le robot se déplace est supposé à deux dimensions est parfaitement connu, il peut contenir un nombre indéterminé d'obstacles supposés polygonaux et fixes, les configurations de départ et d'arrivée sont prises aléatoirement (Figure 5.1).

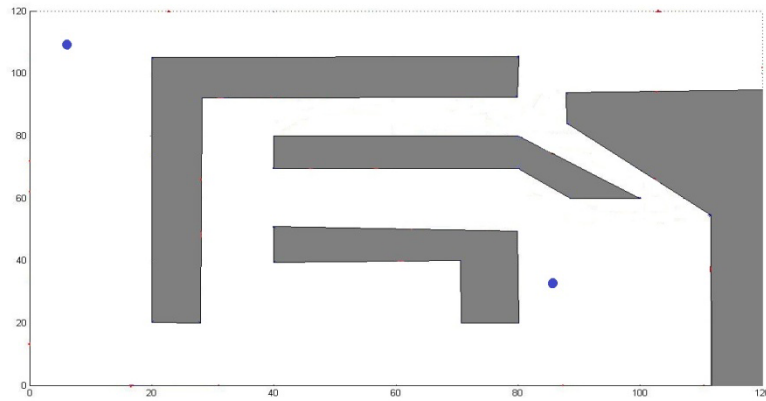


Figure 5.1: exemple d'un environnement a 4 obstacles.

5.2. Le robot mobile:

Le robot considéré dans notre cas est un robot mobile de type voiture (CLR), c'est un système mécanique constitué d'une plate-forme rectangulaire (châssis) qui correspond au corps principal du robot, et de quatre roues (représentées par des petits rectangles noirs), les deux roues avant sont des roues directionnelles ayant un angle de braquage supposé inférieur ou égal à 40° et les deux roues arrières sont des roues motrices fixées sur un axe dont le centre est pris comme le point référence du robot.

La structure générale du robot est représentée dans la figure 5.2

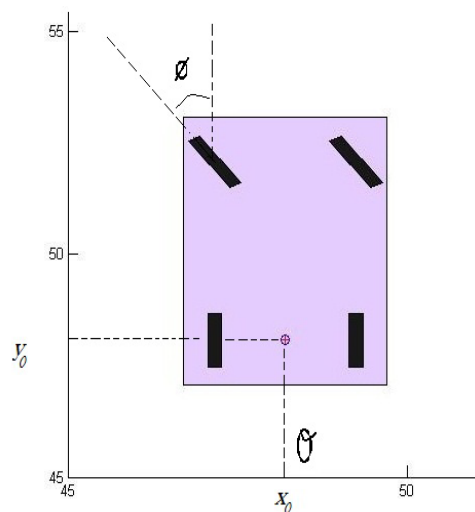


Figure 5.2 : représentation du robot de type voiture.

Le robot est supposé être une voiture de Dubins dont la vitesse est toujours positive (il ne peut pas reculer) et est égale à une valeur maximale V_{max} .

5.3. La construction du réseau PRM :

Le premier problème que nous avons rencontré est la construction du PRM ainsi que le choix des paramètres de discrétisation de l'environnement, en particuliers le nombre de nœuds N_{max} et la distance d'échantillonnage d_{min} qui est la distance minimale entre deux nœuds du réseau.

5.3.1. Calcul du nombre de nœuds N_{max} :

Le choix de la taille du réseau (nombre de nœuds N_{max}) est très important pour satisfaire le maximum de l'espace libre.

Pour calculer le nombre N_{max} , nous avons utilisé une méthode basée sur le calcul du rapport de la surface libre sur la surface totale de l'environnement pour une distance d'échantillonnage d_{min} donnée. Le principe est de générer M configurations aléatoires, le rapport surface libre sur surface totale sera donc égal au rapport du nombre de configurations libres sur le nombre total des configurations.

On répète ce processus K fois et on calcule la valeur moyenne des rapports obtenus pour avoir une meilleure approximation, le nombre N_{max} sera donc la valeur entière du rapport de la surface libre sur la surface du disque qui englobe un carré de côté d_{min} .

Le pseudo code suivant décrit le principe de calcul de N_{max} .

Procédure de calcul de N_{max} :

Entrées: d_{min} , M , K , $surf$. // $surf$: surface totale de l'environnement.

Sortie: N_{max} ,

$k=1$.

Répéter K fois :

$m=1$, $libre=0$.

Tant que : $m \leq M$ faire

Générer une configuration aléatoire q .

Si q est une configuration libre **alors**

$libre = libre + 1$.

Fin si.

$m = m + 1$.

Fin tant que.

$$r(k) = \frac{libre}{M}$$

$k = k + 1$.

Fin répéter

$$Q = \frac{1}{K} \sum_{k=1}^K r(k).$$

$$N_{max} = \text{entier} \left(Q \times \frac{surf}{d_{min}^2 \times \pi / 2} \right).$$

5.3.2 Phase d'apprentissage :

Pour construire le réseau du PRM, nous avons utilisé la méthode d'échantillonnage uniforme, où les nœuds sont générés aléatoirement à condition que la distance entre chaque deux nœuds soit supérieure ou égale à une distance minimale d_{min} .

Pour un nœud q_i donné, on considère comme voisin de q_i tout nœud situé à une distance inférieure ou égale à d_{max} qui représente la distance maximale entre deux nœuds voisins.

La distance d_{max} est choisie proportionnellement à d_{min} ($d_{max} = 2 d_{min}$) pour assurer une bonne connexion au réseau ;

Le réseau construit contient donc un nombre N_{max} de nœuds dont chacun est relié à ses voisins par des segments.

Les figures (5.3 : a, b, c et d) illustrent les différents réseaux obtenus dans différents environnements et avec différentes valeurs de d_{min} :

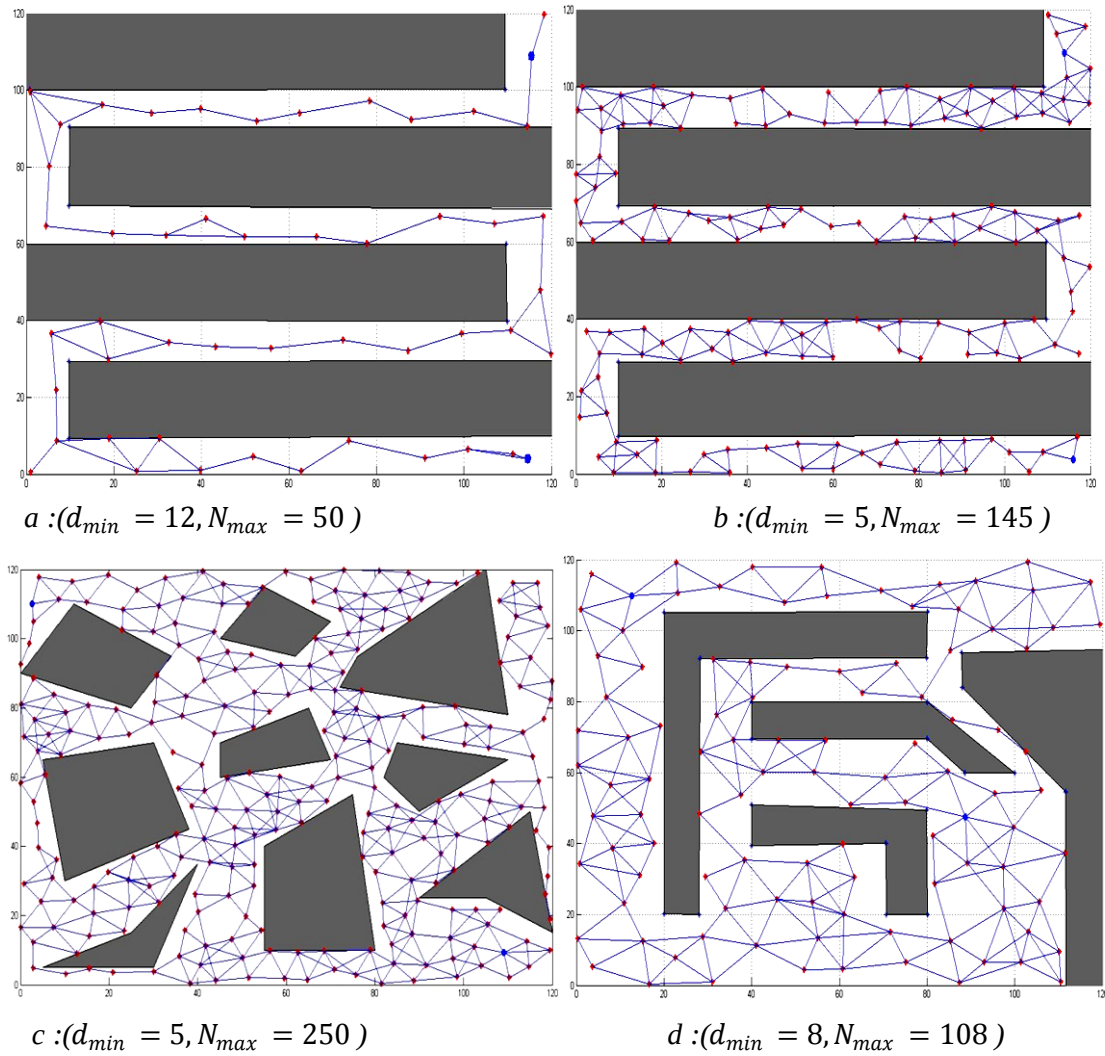


Figure 5.3: discrétisation de différents environnements avec différentes valeurs de d_{min}

5.3.3 Teste de collision :

Pour savoir si un chemin entre deux nœuds correspond à un chemin libre ou non, nous avons utilisé un processus de teste de collision, ce processus retourne 1 si le segment qui relie ces deux nœuds est en croisement avec l'un des cotés des obstacles, et retourne 0 si ce n'est pas le cas.

On dit que deux segments sont en collision si les bornes de l'un des segments sont dans des côtés différents du deuxième segment (figure 5.4), Ceci est vérifié en utilisant une fonction appelée *iscw* (anti sens horaire) qui utilise le produit vectoriel pour juger si le parcours entre trois points est dans le sens horaire ou non (figure 5.5)

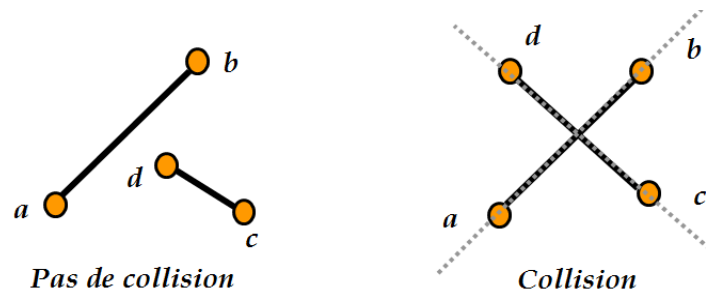
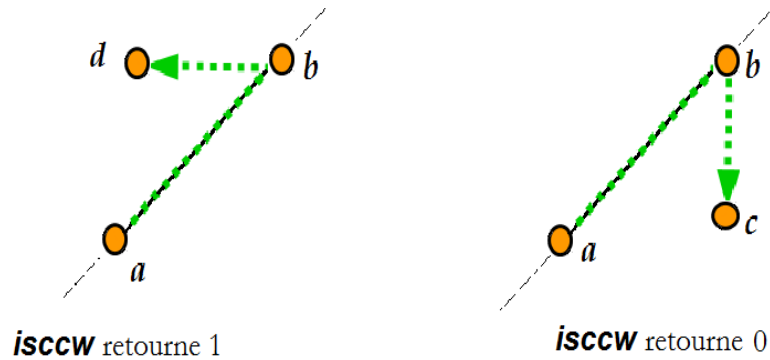


Figure 5.4 : teste de collision

Figure 5.5 : La fonction **isccw**

Le pseudo-code suivant décrit le principe de détection de collision entre deux segments :

Procédure de Teste de collision

Entrées: segment **ab**, segment **cd**.

Fonction : $s = \text{isccw}(a, b, c)$ // is counter clockwise : anti sens horaire

$p = \overrightarrow{ab} \wedge \overrightarrow{bc}$ // produit vectoriel

Si $p \geq 0$

$s = 1$;

Si non

$s = 0$;

Fin si

Fin

Fonction : $c = \text{collision}(\text{segment } ab, \text{segment } cd)$

$s = \text{xor}(\text{isccw}(a, b, c), \text{isccw}(a, b, d)) \& \text{xor}(\text{isccw}(c, d, a), \text{isccw}(c, d, b));$

// xor : ou exclusive

Fin

5.4 L'algorithme de recherche du plus court chemin :

Pour trouver le plus court chemin reliant la configuration de départ et celle d'arrivée on a utilisé un algorithme constitué de trois étapes :

Dans la première étape, il s'agit d'appliquer l'algorithme A* pour chercher le plus court chemin contenu dans le réseau construit précédemment (ce chemin est une concaténation des arcs inclus dans le réseau PRM),

À cause du caractère probabiliste des PRM, le chemin calculé peut contenir des parties irrégulières. On effectue donc dans une deuxième étape un processus de raccourcissement pour éliminer les parties irrégulières du chemin calculé.

Le pseudo code suivant décrit la procédure de raccourcissement utilisée :

Procédure de raccourcissement

Entrées: $\text{chemin}(p_1, p_2, p_3, \dots, p_n)$.

$dc = 1$; // initialisation

Tant que : $dc > 0$ **faire :**

$c_1 = \text{cout du chemin avant le raccourcissement}$ // longueur du chemin

$i = 1$;

$l = \text{taille du chemin}$

Tant que : $i < l - 1$ **faire :**

Si il existe un chemin libre entre p_i et p_{i+2} Alors :
 $\text{supprimer } p_{i+1} \text{ du chemin.}$

$l = l - 1$.

Fin Si

$i = i + 1$;

Fin Tant que

$c_2 = \text{cout du chemin après le raccourcissement}$

$dc = c_1 - c_2$

Fin Tant que

La figure 5.6 illustre les chemins obtenus avant et après l'application du processus de raccourcissement :

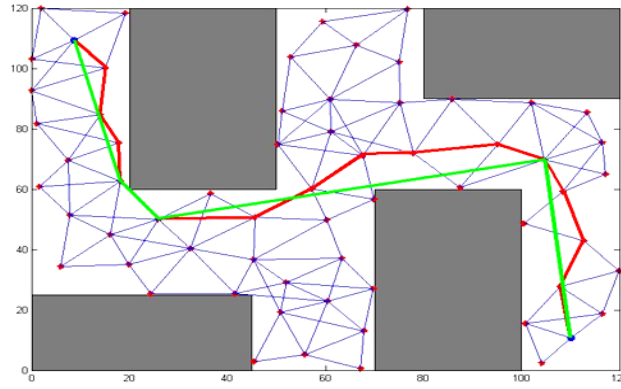


Figure 5.6: chemins avant et après le raccourcissement :

Le chemin calculé après l'exécution du raccourcissement n'est pas nécessairement le plus court, afin d'améliorer ce dernier on fait appel en troisième étape à une procédure de discrétisation qui va diviser ce chemin en plusieurs parties dont chacune est de longueur supérieure ou égale à un pas de discrétisation prédéfini.

Le principe de discrétisation d'un chemin est décrit dans le pseudo code suivant :

Procédure de discrétisation

Entrées: $path(p_1(x_1, y_1), p_2(x_2, y_2) \dots p_n(x_n, y_n))$ // chemin à discrétiser,
 l // pas de discrétisation.

Sortie : $pathd$ // chemin discret.

$pathd = \emptyset$; // Initialisation du chemin discret.

Pour $i = 1$ jusqu'à $n - 1$ // n : le nombre des nœuds construisent le chemin $path$.

$d = \text{distance entre } p_i \text{ et } p_{i+1}$;

$nb = \text{entier} \left(\frac{d}{l} \right) + 1$; // le nombre des nœuds qui peuvent être insérés entre
 // p_i et p_{i+1} .

$dt = \frac{1}{nb}$;

$t = 0$;

$j = 1$;

Tant que : $t \leq 1$ **faire :**

$x_d = t \times x_{i+1} + (1 - t)x_i$;

$y_d = t \times y_{i+1} + (1 - t)y_i$;

$p_d = \text{noeud}(x_d, y_d)$;

Ajouter p_d au $pathd$.

$t = t + dt$;

Fin Tant que

$c_2 = \text{coût du chemin après le raccourcissement}$

$dc = c_1 - c_2$

Fin Pour

La figure (5.7) illustre un chemin soumis à une procédure de discrétisation :

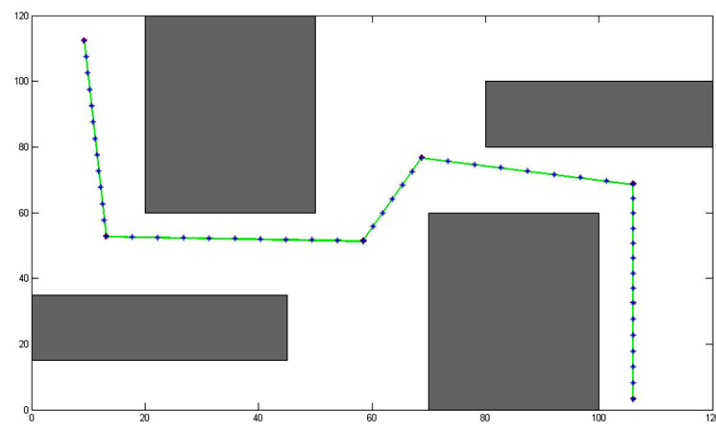


Figure 5.7: chemin après discrétisation

La deuxième et la troisième étape sont répétées successivement N fois ou jusqu'à ce que le gain du raccourcissement soit négligeable.

La figure 5.8 illustre un exemple d'un chemin obtenu après l'exécution des procédures de discrétisation et de raccourcissement :

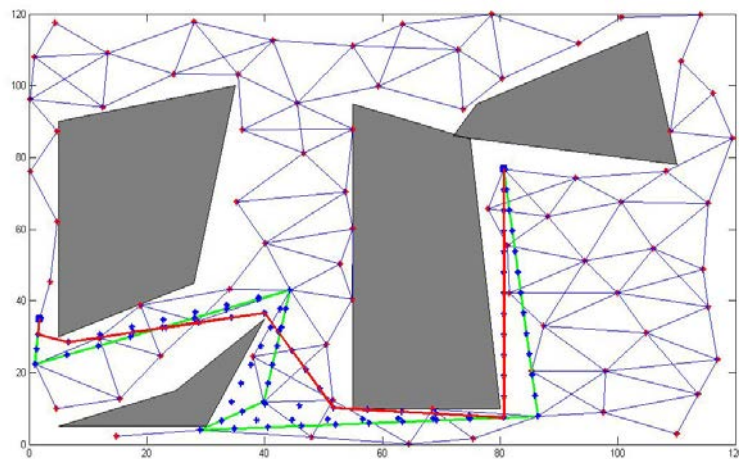


Figure 5.8 : chemin après l'exécution successive de la discrétisation et du raccourcissement.

5.5. Suivi de chemin

Jusqu'à cette étape le chemin calculé restait infaisable à cause des contraintes de non holonomie du robot mobile, il a fallu donc calculer les commandes qui vont faire que le robot suive ce chemin mais à condition qu'on prenne en compte les contraintes du robot, pour cela nous avons eu besoin d'une discrétisation du modèle du robot décrit au système (4.7).

Pour un angle de braquage fixe, la discrétisation du modèle du robot peut s'écrire de la manière suivante :

$$\begin{cases} x_{i+1} = x_i + \Delta t \frac{v}{2l} \cos(\theta_i + \tan\phi \Delta t) \\ y_{i+1} = y_i + \Delta t \frac{v}{2l} \sin(\theta_i + \tan\phi \Delta t) \\ \theta_{i+1} = \theta_i + \Delta t \frac{v}{\ell} \tan\phi. \end{cases} \quad (5.1)$$

Pour le suivi de chemin, on a supposé que la vitesse du robot est fixe et à valeur maximale, et l'angle de braquage ne peut prendre que trois valeurs $\{-\phi_{max}, 0, \phi_{max}\}$. Une fonction de suivi de chemin permet d'extraire les différents angles de braquage le long de ce dernier.

Comme première étape, cette fonction va calculer deux points de braquage (début et fin de braquage) pour chaque trois points successif du chemin, ensuite elle va diviser le chemin qui relie ces trois points en deux sous chemins, le premier est un segment de droite allant du premier point au point de début de braquage, et le deuxième est un arc de rayon minimal (r_{min}) qui relie les deux points de braquage, le deuxième point sera remplacé par le point de fin de braquage et le processus recommencera pour trois nouveaux points.

Les deux points de braquage sont calculés géométriquement comme indique la figure 5.9 :

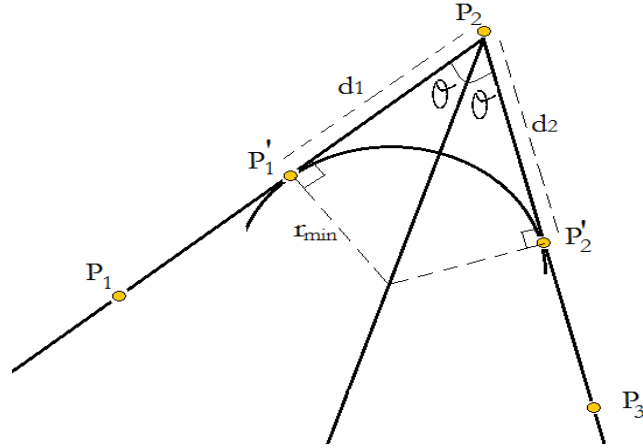


Figure 5.9 : Calcul des points de braquage.

Tel que :

$$d_1 = d_2 = \frac{r_{min}}{\tan \theta} \quad (5.2)$$

Avec :

r_{min} : le rayon minimale correspond à $\pm\phi_{max}$.

θ : le moitié de l'angle entre les deux segments.

La figure 5.10 explique le fonctionnement du suivi de chemin :

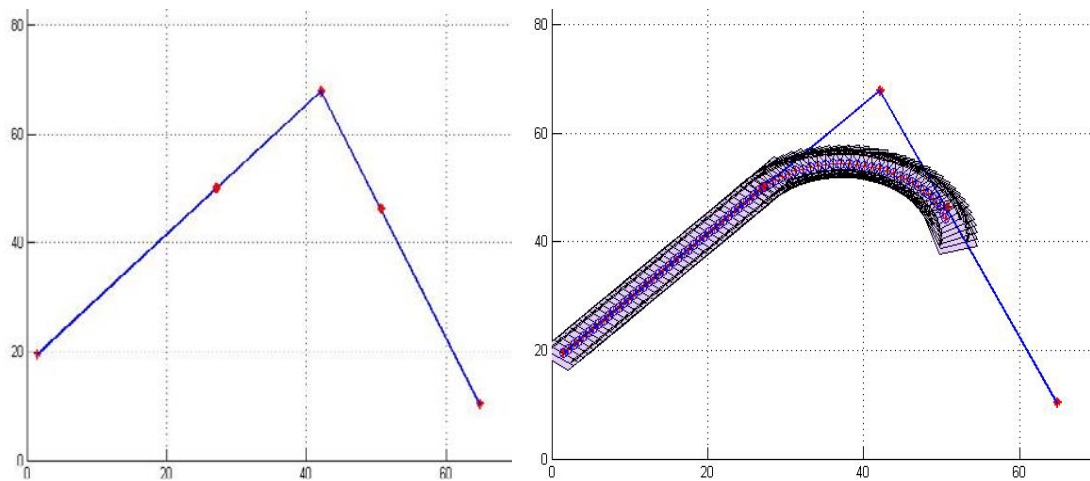


Figure 5.10 : suivi de chemin entre trois points.

Pendant le suivi de chemin, on va extraire les différents angles de braquage dans chaque intervalle Δt de temps.

Le pseudo code suivant va expliquer le principe du suivi de chemin :

Algorithme de suivi de chemin

Entrées: $\text{chemin}(p_1(x_1, y_1), p_2(x_2, y_2) \dots p_n(x_n, y_n)), V_{\max}, \phi_{\max}, \Delta t$

Sortie: ϕ . //une suite d'angles de braquage.

Pour tout triplet p_i, p_{i+1} et p_{i+2}

Calculer p'_1 et p'_2 les deux points de braquage.

$d_1 = \text{distance entre } p_i \text{ et } p'_1$;

$nb = \text{entier}(\frac{d_1}{\Delta t V_{\max}})$ //le nombre des pas entre p_i et p'_1 .

Pour $j = 1$ jusqu'à nb

$\phi(j) = 0$.

Fin Pour

$d_2 = \text{longueur de l'arc } p'_1 \text{ et } p'_2$;

$nb_1 = nb + \text{entier}(\frac{d_2}{\Delta t V_{\max}})$ // le nombre des pas entre p'_1 et p'_2 .

Pour $j = nb + 1$ jusqu'à nb_1

$\phi(j) = s \times \phi_{\max}$. // s le sens de braquage.

Fin Pour

$p_{i+1} == p'_2$.

Fin Pour

Jusqu'à cette étape nous ne possédons qu'une série d'angles de braquage, l'exécution de la trajectoire est effectuée à l'aide d'une fonction qui va calculer les paramètres

(x_i, y_i, θ_i) de la configuration q_i à chaque instant t_i en utilisant le modèle discret du robot et la série des angles de braquage calculée précédemment.

La fonction de l'exécution de la trajectoire est décrite dans le pseudo code suivant :

Fonction d'exécution de la trajectoire

Fonction $(x, y, \theta) = \text{move}(q_{\text{start}}(x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}}), \phi, V_{\text{max}})$

$x(1) = x_{\text{start}}, y(1) = y_{\text{start}}, \theta(1) = \theta_{\text{start}}.$

$n = \text{longueur de } \phi ;$

Pour $i = 1$ jusqu'à n

$$x(i+1) = x(i) + \Delta t \frac{v}{2l} \cos(\theta(i) + \tan\phi(i) \Delta t)$$

$$y(i+1) = y(i) + \Delta t \frac{v}{2l} \sin(\theta(i) + \tan\phi(i) \Delta t)$$

$$\theta(i+1) = \theta(i) + \Delta t \frac{v}{\ell} \tan\phi(i).$$

Fin Pour

La figure 5.11 illustre une trajectoire obtenue après l'exécution de la fonction *move* :

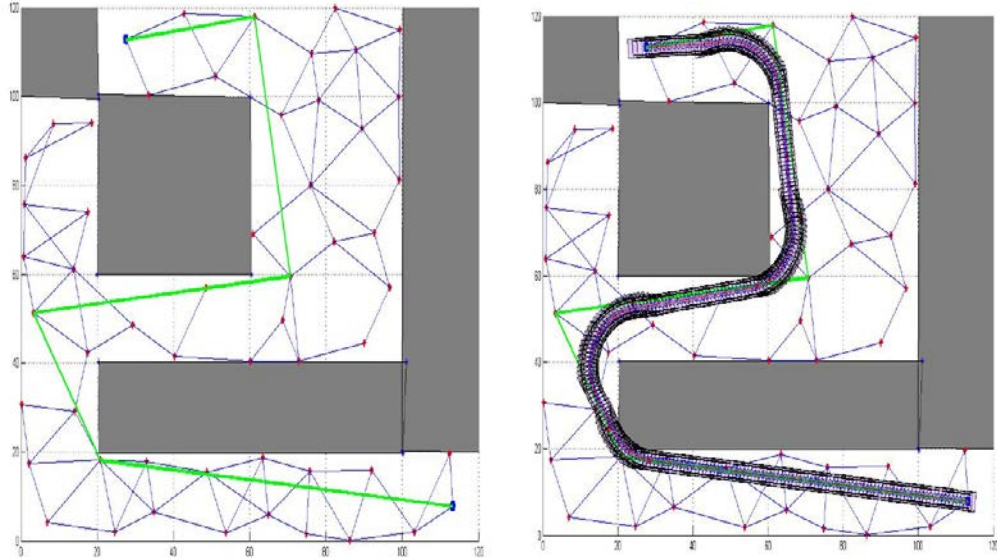


figure5.11 trajectoire obtenue après l'exécution de la fonction : *move*

5.6. Optimisation par les Algorithmes Génétiques :

Lors de l'exécution de la trajectoire, le robot risque d'être en collision avec les obstacles (figure 5.11), même si n'est pas le cas, le chemin calculé n'est pas nécessairement le meilleur, on doit donc chercher une séquence d'angles de braquage

correspondant au plus court chemin sans collision. Pour cela, nous allons utiliser les Algorithmes Génétiques Hiérarchiques AGH qui vont jouer sur les deux critères : collision et longueur de chemin pour calculer le chemin optimal sans collision.

5.6.1 Codages et Initialisation :

Pour générer la population initiale, nous devons répéter toutes les étapes précédentes allant de la construction du PRM jusqu'à la fonction de suivi du chemin et cela dans le but d'avoir plusieurs chemins reliant les deux positions de départ et d'arrivée, ces chemins vont se distinguer selon le réseau PRM construit à cause du critère probabiliste des PRM. On obtient donc pour chaque chemin une séquence des angles de braquage extraites au cours de suivi du chaque chemin ; l'ensemble de ces séquences vont construire les individus de la population initiale des Algorithmes génétiques.

La figure 5.12 montre 10 chemins obtenus par la méthode des PRM exécutée 10 fois :

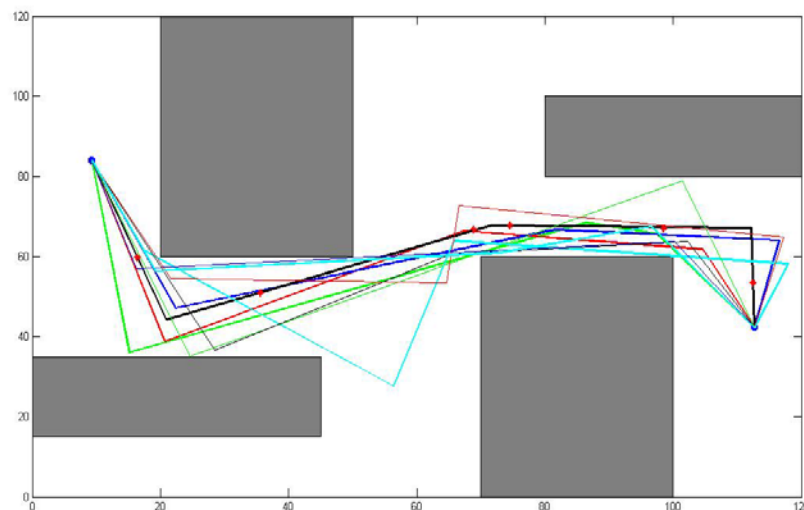


Figure 5.12 chemins obtenus par les PRM exécutée 10 fois.

Chaque individu est constitué de deux chromosomes, le premier contient des gènes portant les valeurs des angles ϕ_i extraites au cours du suivi de chaque chemin ; le deuxième contient l'angle θ_{start} de la configuration de démarrage.

La taille du premier chromosome diffère d'un individu à un autre selon le chemin dont il extrait ses gènes (angles de braquage).

Pour surmonter ce problème, on a ajouté au premier chromosome de chaque individu un nombre de gènes (initialisés à zéro) jusqu'à ce que sa taille soit égale à la taille du plus grand chromosome, ensuite on a associé à chaque gène de paramètres un gène g_i

de contrôle qui va désactiver le gène de paramètres s'il est un gène ajouté ($g_i = 0$), ou bien il va l'activer, sinon ($g_i = 1$).

La structure d'un individu peut donc être représentée par la figure 5.13 :

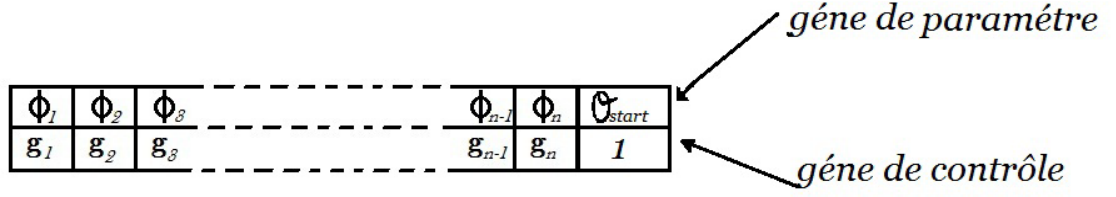


figure 5.13 : structure d'un individu

5.6.2 Fonction d'adaptation (fitness function):

L'évaluation des individus est effectuée comme suit :

Nous calculons un chemin de référence en appliquant sur l'un des chemins calculés précédemment la procédure de raccourcissement et de discrétisation (avec un pas de discrétisation très petit) suffisamment de fois jusqu'à avoir le plus court chemin qui relie les configurations de départ et d'arrivée.

Ensuite nous définissons le potentiel v_i d'une configuration q_i qui est proportionnel à la distance aux obstacles et à la trajectoire de référence du point de référence du robot lorsqu'il est à la configuration q_i , ainsi le chemin de référence va exercer un champ attractif sur la solution par contre les obstacles vont exercer un champ répulsif et l'évaluation de chaque gène sera proportionnelle à la résultante de ces deux champs et au nombre de gènes de paramètre actifs ($g_i = 1$) qui est inversement proportionnel à la longueur du chemin associé.

Le potentiel v_i d'une configuration q_i peut s'exprimer par :

$$v_i = \begin{cases} \frac{1}{n} \left(\frac{K_a}{(d_i+1)^2} - \frac{K_r}{(dr_i)^2} \right) & \text{si } dr_i \leq d_0 \\ \frac{1}{n} \frac{K_a}{(d_i+1)^2} & \text{si } dr_i > d_0 \end{cases} \quad (5.3)$$

Avec :

K_a et K_r sont des constantes positives.

d_i : est la distance entre la configuration q_i et la trajectoire de référence.

dr_i : est la distance entre la configuration q_i et le plus proche obstacle.

d_0 : est la distance d'influence des obstacles.

n : est le nombre de gènes de paramètre actifs.

L'évaluation d'un individu est égal donc à la somme des potentiels v_i de ses gènes de paramètre actifs.

La fonction d'adaptation peut donc s'exprimer par :

$$f = \sum_{i=1}^n g_i \times v_i \quad (5.4)$$

5.6.3 La sélection:

La sélection des individus est effectuée en utilisant la méthode élitiste où les individus sont triés de manière décroissante selon l'évaluation de chaque individu.

Les individus les plus performants (qui ont une évaluation grande) sont sélectionnés pour participer à la prochaine génération.

5.6.4 La reproduction des individus:

Les meilleures solutions sont combinées afin de donner naissance à de nouveaux individus qui sont ajoutés à l'ensemble des individus précédents, cette étape peut se réaliser par deux processus : croisement et mutation.

5.6.4.1 Le croisement:

On fait un tirage au sort avec remise de deux individus sélectionnés qui vont subir deux types de croisement :

Le premier est un croisement barycentrique effectué sur les gènes des paramètres avec un coefficient de pondération pris aléatoirement dans l'intervalle [01].

Le deuxième est un croisement classique (échange des gènes) effectué sur les gènes de contrôle, la figure 5.14 décrit le principe du croisement utilisé :

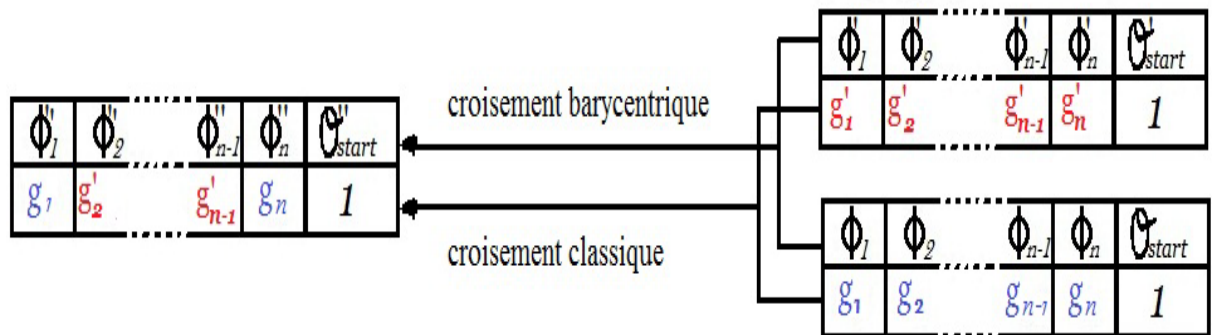


figure 5.14 : le principe du croisement utilisé

Où :

$$\begin{cases} \theta'' = \alpha \theta' + (1 - \alpha) \theta \\ \theta''_{start} = \alpha \theta'_{start} + (1 - \alpha) \theta_{start} \end{cases} \quad (5.5)$$

On répète le croisement jusqu'à obtenir un nombre suffisant d'individus qui vont créer la nouvelle population.

5.6.4.2 La mutation:

Pour le processus de mutation, nous avons utilisé une mutation uniforme en ajoutant un bruit gaussien d'une variance très petite (de l'ordre de $\frac{\phi_{max}}{25}$) aux gènes du premier chromosome avec une probabilité de l'ordre de 20%.

Pour un angle de braquage ϕ , le processus de mutation peut s'exprimer comme suit :

$$\phi' = \phi + N\left(0, \frac{\phi_{max}}{25}\right).$$

L'angle ϕ' est obtenu après que la mutation soit acceptée si $\phi' \in [-\phi_{max} \phi_{max}]$.

Nous répétons les opérateurs de sélection, croisement et mutation jusqu'à obtenir le plus court chemin faisable.

5.7 L'algorithme général de planification de mouvement :

L'algorithme général de planification de mouvement est résumé dans l'organigramme de la figure 5.15 :

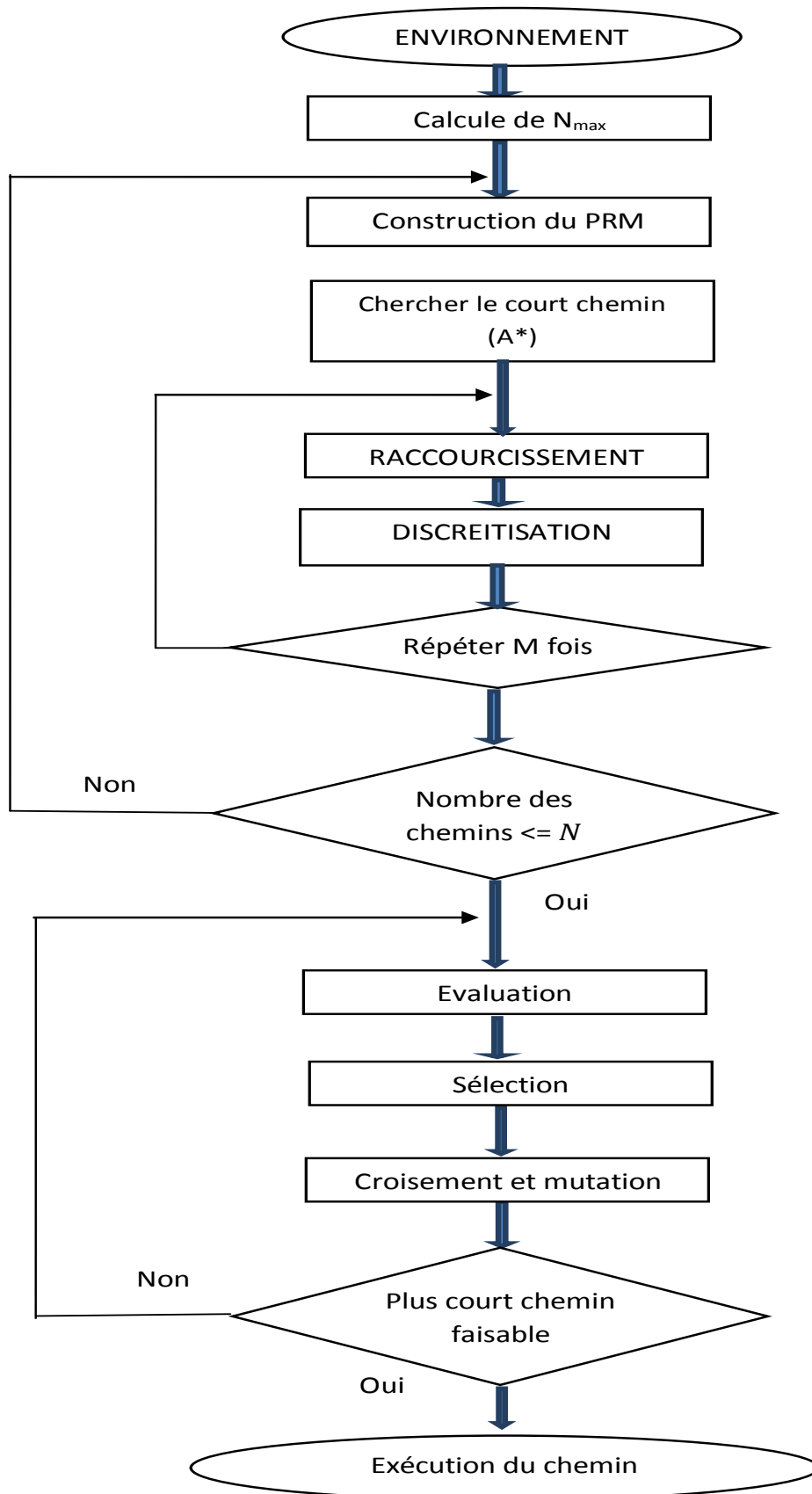


figure 5.15 : L'algorithme général de planification de mouvement

5.8. Analyse de l'algorithme et des résultats obtenus:

Dans cette section nous allons commenter les différents résultats obtenus par l'application de l'algorithme de planification élaboré et exécuté sous l'environnement Matlab.

Comme tous les algorithmes heuristiques, notre algorithme peut prendre un temps très grand pour converger vers la solution optimale, comme il peut converger rapidement, ceci dépend de plusieurs paramètres, le premier concerne la population initiale qui dépend de son côté des réseaux PRM construits ($N_{max}, d_{min} \dots$), on a également le pas de discrétisation l et le nombre M de fois qu'on applique la procédure de raccourcissement et de discrétisation (figure 5.16) :

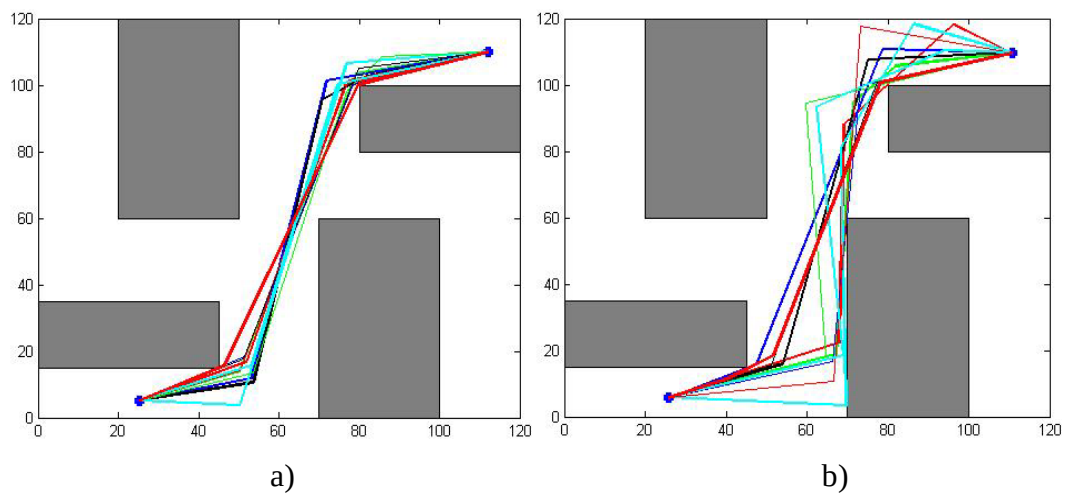


Figure 5.16 chemins de la population initiale pour différents paramètres

(a: $l = 2, d_{min} = 8, M = 20$ et b: $l = 10, d_{min} = 15, M = 5$).

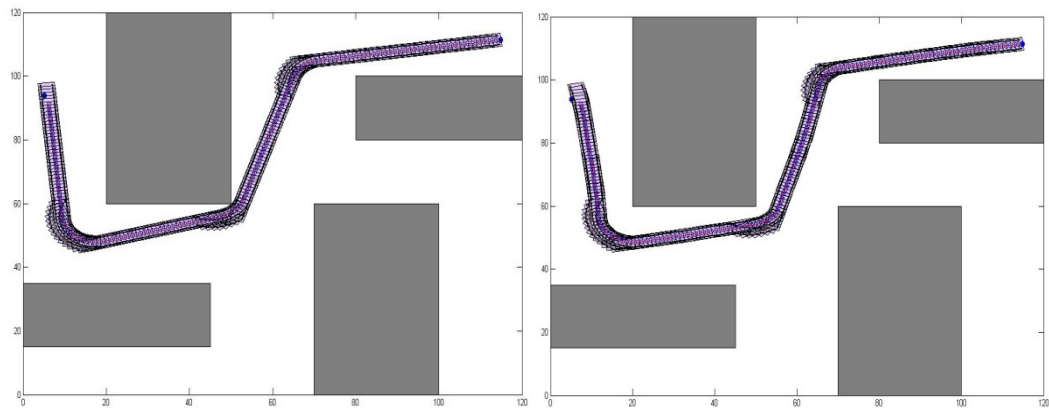
Dans le cas où les individus de la population initiale sont semblables, l'espace de recherche de la solution sera très petit (il sera difficile d'améliorer les individus au cours de l'opération de croisement), l'opérateur de mutation joue un rôle très important pour élargir cet espace de recherche et donner naissance à des individus qui peuvent être plus performants.

La population initiale peut aussi contenir un chemin très proche de l'optimal, par conséquent la solution va converger rapidement.

Le choix des paramètres de la fonction d'évaluation (K_a, K_r, d_0) joue un rôle très important dans la convergence de la solution, en effet plus K_a, K_r, d_0 augmentent plus la solution se rapproche du chemin de référence et s'éloigne des obstacles.

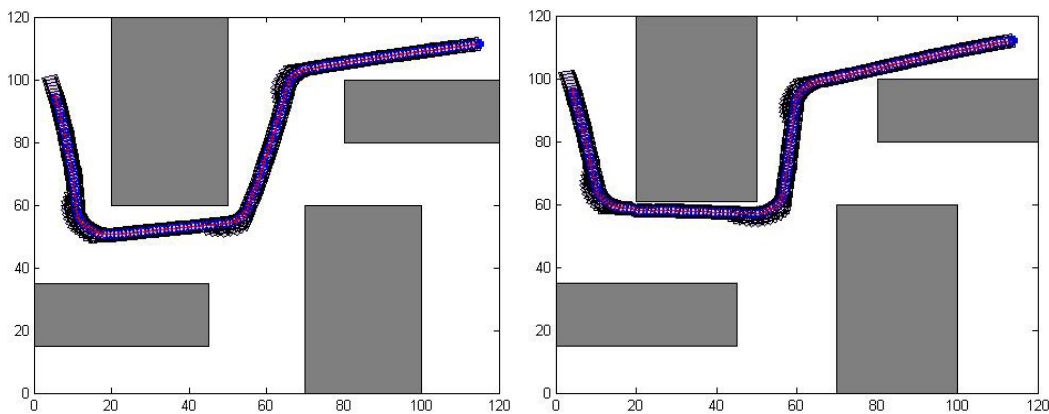
Prenons deux environnements différents, et deux différentes configurations de départ et d'arrivée, les résultats obtenus sont illustrés par la figure 5.17 et 5.18:

Environnement 1 :



a1) trajectoire après 10 générations

a2) trajectoire après 50 générations

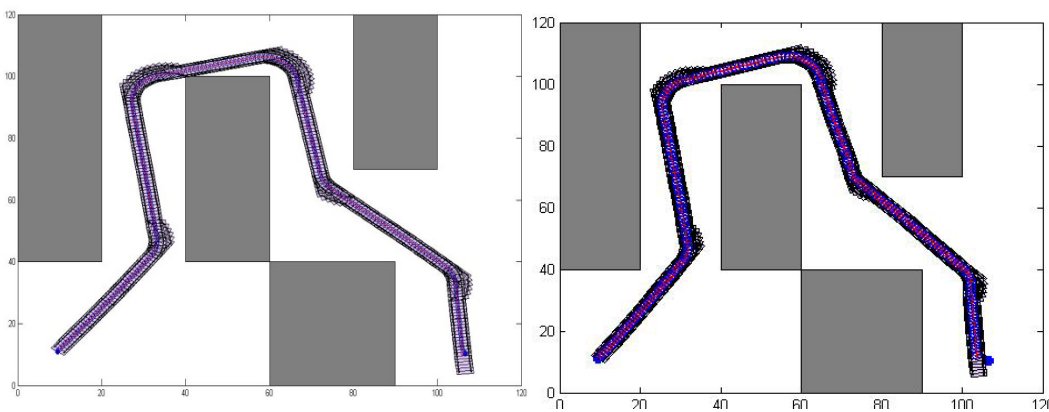


a3) trajectoire après 100 générations

a4) trajectoire après 200 générations

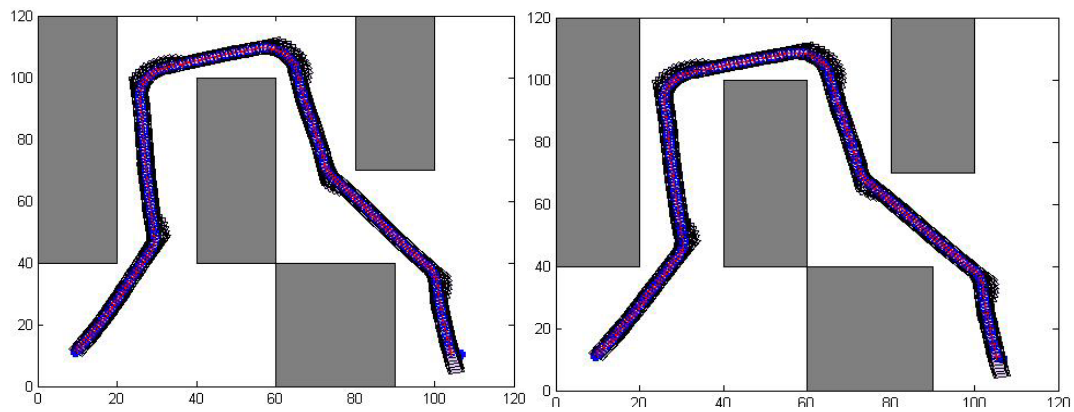
Figures 5.17 Résultats pour l'environnement 1

Environnement 2 :



b1) trajectoire après 10 générations

b2) trajectoire après 50 générations



b3) trajectoire après 100 générations b4) trajectoire après 300 générations

Figures 5.18 Résultats pour l'environnement 2

L'évolution des trajectoires obtenues pour les deux environnements est donnée par les tableaux 5.1 et 5.2.

Génération	10	20	50	100	200
Evaluation	-180,42	-192,41	-220,71	-250,07	-330,35
Longueur de la trajectoire	170	170	170	169	163
Collisions	OUI	NON	NON	NON	NON

Tableau 5.1 L'évolution de la trajectoire dans l'environnement 1

Génération	10	20	50	100	300
Evaluation	-531,24	-664,82	-776,07	-963,33	-997,46
Longueur de la trajectoire	242	240	240	240	240
Collisions	OUI	NON	NON	NON	NON

Tableau 5.2 L'évolution de la trajectoire dans l'environnement 2

Pour le premier environnement, nous remarquons que l'algorithme a convergé vers la solution optimale après 200 générations, et l'évolution de la solution est sensible d'une génération à l'autre, par contre l'évolution de la solution dans le deuxième environnement est très lente de la 50^{ème} génération jusqu'à la génération 300 ; ceci est dû généralement à la population initiale ou encore à la distribution des obstacles qui peut contenir des passages étroits. Donc on peut constater qu'il est difficile de définir une condition d'arrêt de cet algorithme parce que la solution peut stagner jusqu'à ce que l'opérateur de mutation améliore la solution et ceci peut être après un nombre indéterminé de générations ; donc nous devons répéter les opérateurs génétiques suffisamment de fois jusqu'à avoir la meilleure solution.

Conclusion générale

Dans ce mémoire, nous avons rapporté nos travaux qui entrent dans le cadre de la planification de mouvement pour un robot mobile non holonome. Nous nous sommes particulièrement intéressés au problème de la génération de mouvements optimaux pour un robot mobile non holonome (robot type voiture) qui se déplace dans un environnement supposé plan et dont des obstacles sont supposés fixes et polygonaux.

Pour ce type de robot, un chemin libre ne correspond pas nécessairement à un chemin faisable à cause des contraintes de non holonomie qui imposent au robot de suivre le chemin avec une vitesse toujours tangente à ce dernier.

Nous avons montré dans ce mémoire que le problème de la génération de mouvements optimaux peut être résolu en trois étapes :

La première étape concerne le calcul de chemins libres sans prise en compte des contraintes de non holonomie du robot, ceci est effectué en utilisant la méthode des réseaux probabiliste PRM associée à un algorithme de recherche du plus court chemin (A star), ces chemins sont par la suite améliorés en utilisant des algorithmes de raccourcissement et de discrétisation.

En deuxième étape nous utilisons un algorithme de suivi de chemin basé sur le modèle d'un robot mobile non holonome pour extraire les différentes commandes nécessaires pour mener le robot d'une position initiale à une position finale en suivant les chemins calculés dans la première étape.

Enfin à la troisième étape nous faisons appel aux algorithmes génétiques pour calculer les commandes qui permettent de générer un mouvement optimal du robot depuis le point de départ jusqu'au point d'arrivée.

Nous avons montré que l'algorithme élaboré donne des résultats satisfaisants dans des environnements à structure différente et pour de différentes configurations de départ et d'arrivée. Puisqu'il est basé sur des méthodes heuristiques, cet algorithme est coûteux en termes de temps de calcul et d'espace mémoire, cela ne constitue pas un inconvénient car c'est un algorithme de planification hors-ligne car les environnements sont supposés statiques. Nous envisageons justement d'adapter ces méthodes au cas d'environnements dynamiques.

Références bibliographiques:

- [1]: Lozano-Perez, Spatial planning: A configuration space approach” .IEEE Transactions on Robotics and Automation, 32 :108–120.1983.
- [2]: J. T. Schwartz and M. Sharir. On the piano mover’s problem. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics, Academic Press 4*, pages 298–351, 1983.
- [3]: O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5(1):90–98, 1986.
- [4]:N.J.Nilsson. A mobile Automation :An Application of Artificial Intelligence Techniques, the first International Joint Conference on Artificial Intelligence, 1969
- [5]: L. Kavraki, P. Svestka, J. C. Latombe, et M. Overmars. "Probabilistic roadmaps for fast path planning in high dimensional configuration spaces" . IEEE Transactions on Robotics and Automation,vol.12(4),pages 566-580, 1996.
- [6]: S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept., Iowa State University,1998.
- [7] : P. Tournassoud. Géométrie et intelligence artificielle pour les robots. Traité des nouvelles technologies, série robotique. Hermès, Paris (F), 1988.
- [8]: J.C. Latombe .Robot motion planning, The Kluwer international series in engineering and computer science_ Kluwer Academic Publishers 1991.
- [9]: M. Lin, D. Manocha, J. Cohen, and S. Gottschalk. Collision detection: algorithms and applications. Workshop on Algorithmic Foundations of Robotics, 1996.
- [10]: L. Kavraki. *Random networks in configuration space for fast path planning*. PhD thesis, Stanford University, 1995.
- [11]: V, and F. Van der Steppen. The gaussian sampling strategy for probabilistic roadmap planners. Int. Conf. on Robotics and Automation, pages 1018–1023, 1999.
- [12]: P. Svestka and M. H. Overmars. Motion planning for car-like robots using a probabilistic learning approach. Int. Journal of Robotics Research, 16(2):119–143, 1997.
- [13]: P. Svestka. *Robot motion planning using probabilistic roadmaps*. PhD thesis, Utrecht University, 1997.
- [14]: N. M. and Y. Wu. a randomized roadmap method for path and manipulation planning. Int. Conf. on Robotics and Automation, pages 113–120, 1996.

- [15]: N. M. Amato, B. O. Burchan, L. K. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3d workspaces. Workshop on Algorithmic Foundations of Robotics, pages 155–168, 1998.
- [16] : C. Nissoux. Visibilité et méthodes probabilistes pour la planification de mouvement en robotique. PhD thesis, Université Paul Sabatier, 1999.
- [17]: J. P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Journal of Advanced Robotics*, 14(6):477– 494, 2000.
- [18] John Holland. Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machinery*, 3, 1962.
- [19] : Wu, C.J. et G.Y. Lin. Design of fuzzy logic controllers using genetic algorithms. In: *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 6. SAAA. pp. 104–109. Tokyo 1999
- [20] : Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag. 1992
- [21]: Hwang, H.S. Automatic design of fuzzy rule base for modeling and control using evolutionary programming. *IEE Proceedings on Control Theory and Applications*, 1999.
- [22]: Tang, K.S., C.Y. Chan, K. F. Man et S. Kwong Genetic structure for non topology and weights optimization. In: *Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. IEE. pp. 250–255. 1995.
- [23]: J. P. Laumond, S. Sekhavat, and F. Lamiraux, Robot Motion Planning, ch. Guidelines in Nonholonomic Motion Planning for Mobile Robots. Springer, 1998.
- [24] Reeds et Shepp, 90] : J.A. Reeds and L.A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 1990.
- [25]: L.E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 1957.
- [26]: F. W. Warner. Foundations of differentiable manifolds and lie groups. Graduate Texts in Mathematics. Springer-Verlag, 1983.
- [27] J.P. Laumond . Finding collision-free smooth trajectories for a non holonomic mobile robot. The International Joint Conference on Artificial Intelligence , 1 120- 1 123 , Milan , Italy, 1987
- [28] J-P.Laumond, P.E. Jacobs, M. Taix, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. Robotics and Automation*, Oct 1994.

Résumé

Le problème de la génération de mouvements optimaux constitue depuis longtemps un des principaux axes de recherche en robotique mobile, Bien que de nombreuses solutions aient été proposées il a toujours fallu prendre en compte de nouveaux facteurs et contraintes qui peuvent être liées soit à l'environnement (obstacles), soit au robot lui-même. Pour un robot mobile de type voiture, une contrainte cinématique dite non holonome oblige le robot à suivre une trajectoire imposée par l'angle de braquage de ses roues directionnelles.

Dans ce mémoire, nous sommes intéressés à développer un algorithme de planification de mouvement dans le but de générer de mouvements optimaux pour un robot mobile non holonome en utilisant la méthode des réseaux probabiliste (PRM) associée à deux techniques d'intelligence artificielle : l'algorithme A* pour la recherche du plus court chemin et les algorithmes génétique pour l'optimisation.

Abstract

The problem of optimal motion planning has long been a major focus of research in robotics. Although many solutions have been proposed, we must always take in consideration new factors and constraints which can be related either to environment (obstacles) or to the robot itself.

For a car like robot, a kinematic constraint called non holonomic forces the robot to follow a trajectory imposed by the steering angle of the steerable wheels.

In this thesis we are interested in developing an algorithm of motion planning to generate optimal motion for a non holonomic robot using the probabilistic roadmap approach (PRM) with two artificial intelligent techniques: the A* for finding the shortest path, and the genetic algorithms for optimization.