REPUPLIQUE ALGERIENNE DEMOCRATIQUE ET POLPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Automatique

Projet de fin d'études

Pour l'obtention du
Diplôme D'ingénieur d'état et du diplôme de Master en Automatique

THEME

Planification de la trajectoire d'un robot mobile autonome dans un environnement statique et/ou dynamique

Etudié par :

HAZERCHI Mohamed
TAZIR Mohamed Lamine

Proposé et dirigé par :

MC A. O. Azouaoui (CDTA)
Pr. H. Chekireb (ENP)

Juin 2012

Ecole Nationale Polytechnique 10, avenue Hassen Badi BP182 El-Harrach
16200 Alger (Algérie)

Remerciements

Louange à Allah le tout puissant qui nous a accordé le savoir, le droit chemin, l'opportunité de poursuivre nous études et la force pour réaliser ce modeste travail.

Au terme de notre cycle d'études, il nous paraît opportun de nous acquitter d'un devoir noble, celui de remercier tous ceux qui ont contribué par leur assistance tant morale que physique à notre cursus universitaire et à la réalisation de ce mémoire.

*Nos remerciements s'adressent particulièrement à nos encadreurs : **Mme O. Azzouaoui** de centre de recherche des technologies avancées (**CDTA**) et **Mr H. Chekireb** de l'**ENP**, qui ont bien voulu diriger ce sujet. Pour leurs disponibilités, aides et bonnes humeurs durant toutes les étapes de ce projet. Leurs dévouements, conseils scientifiques et suivis, nous ont permis de mener notre travail à terme.*

Nous remercions les membres du jury d'avoir bien voulu étudier ce travail et participer à la commission d'examen afin de le juger. Qu'ils en soient louablement gratifiés.

*Notre reconnaissance va tout naturellement à Mr **Djekoune Oualid**, Mr **Brahimi Mohamed**, Mr **Kraim Khaireddine** et Mr **Soulaimene Houcine**. Leurs conseils et soutiens nous ont été d'un apport précieux.*

Nous remercions également tous ceux qui de près ou de loin nous ont aidés ou qui on contribue à l'élaboration du présent mémoire.

Merci infiniment à tous.

Dédicaces

Je dédie ce modeste travail :
À Celle à qui mon cœur depuis ma naissance n'a pu qu'éprouver qu'amour et reconnaissance
, à celle qui a donné un sens à mon existence
À ma chère Mère.
À la mémoire de mon père, qui a été un idéal père
Merci Père.
À mes frères et mes sœurs pour leurs soutiens tout au long de mes études
À toutes ma famille.
À mes meilleurs amis Taher et Amine
À mon très sérieux binôme Lotfi
À tous mes amis et à tous ceux qui m'ont aidé de près ou de loin dans la réalisation de ce travail.

À tous ceux que j'aime et qui m'aiment.

À celle que j'aime et qui m'aime

Mohamed

DEDICACES

En signe d'amour, de gratitude et de respect, je dédie ce modeste travail :

Aux êtres les plus chers au monde qui n'ont pas cessé de me fournir leurs soutiens et leurs encouragements : Mon père qui m'a éclairé mon chemin et qui m'a encouragé et soutenue dans les moments difficiles de ma carrière ; Ma chère mère, mon petit ange, la lumière de mes yeux, l'être que j'aime et que je chéris plus que tout au monde, celle qui a fait de moi ce que je suis.

J'espère qu'ils sont fiers de leur fils,
ils sont mes plus chers êtres, que
Dieu les protège.

Je le dédie aussi, à :

mes frères,
mes oncles, mes tantes
mes cousines et mes cousins et
toute ma famille, de loin ou de proche.

Mon très cher et sérieux binôme Mohamed, À tous mes amis de L'ENP et tous mes amis de la cité universitaire « Bouraoui » avec qui j'ai appris comment s'accrocher à ses rêves malgré toutes les souffrances. À tous mes amis que j'ai connus pendant toute ma vie. À tous ceux que j'aime et qui m'aiment. Enfin, last but not least, je remercie vivement mes chers amis ***Dr Kraim Khairedine*** *et* ***Brahimi Mohamed*** *qui m'ont donné l'enthousiasme dans la recherche et qui n'ont jamais cessé de m'encourager.*

Thank you for all.

Mohamed Lamine

Alger, le 15 Juin 2012.

ملخص:
تستخدم الروبوتات المتنقلة على نطاق واسع في العديد من المجالات الصناعية. و يعد البحث في تخطيط مسار الروبوتات المتنقلة هو واحد من أهم المواضيع في مجال البحوث المتعلقة بالروبوت المتحرك. تخطيط مسار الروبوت المتحرك هو العثور على طريق حر من دون الاصطدام مع العقبات الموجودة في البيئة، لتحريك الروبوت من نقطة البداية إلى وجهة معينة،مع استيفاء بعض معايير التحسين. معظم الطرق الموجودة و المستعملة لهذا الغرض، مثل الرسم البياني الرؤية، وتحلل الخلية، وحقل محتمل , تركزفي غالبها على البيئات الثابتة، التي توجد فيها عقبات ثابتة فقط. رغم انه في الأنظمة التطبيقية مثل العلوم البحرية، والصناعة النووية والتعدين، الروبوتات تواجه عادة بيئات ديناميكية ، والتي تحتوي على عقبات متحركة وعقبات ثابتة. ونظرا لتعقيد البيئات الديناميكية فان الابحاث في هذا المجال محدودة. وقد تم نشر عدد قليل من المقالات و الأبحاث في هذا المجال مقارنة مع مئات المقالات في ما يخص البيئات ثابتة.
هذا البحث يطورطريقة لتخطيط مسار لروبوت متنقل في البيئات الديناميكية باستعمال الخوارزميات الجينية مع خوارزمية مفهوم وقت الانتظار(CTA) العقبات الموجودة في البيئة ذات ابعاد واشكال واماكن مختلفة.
تنقسم طريقة التخطيط هذه إلى مرحلتين:
خارج الخط Offline ترتكز على قمم العقبات الثابتة المضخمة مما يقلص فضاء البحث.
في هذه المرحلة نستعمل الخوارزميات الجينية التي تستعمل في احد اطوارها خوارزمية Dijkstra هذه المرحلة تعمل على ايجاد مسار امثل للروبوت من بين مجموعة من المسارات الممكنة و الغير ممكنة
في المرحلة الثانية (على الخط) يبدأ الروبوت بالتنقل متبعا المسارالامثل بمساعدة لواقطه , و في حالة اكتشافه للعقبات الدينميكية يبدا بحساب امكانية التصادم معه ,و اذا كان هناك تصادم سيقع تتم مناداة خوارزمية 'مفهوم وقت الانتظار' (CTA) التي تقوم بحساب مكان و مدة الانتظار من اجل تجنب التصادم.
في حالة عدم وجود امكانية التصادم يكمل الروبوت اتباع مساره الامثل الى هدفه.
كلمات مفتاحية: الروبوت المتنقل ,تخطيط مسار ، البيئة الديناميكية، الخوارزمية الجينية، خوارزمية مفهوم وقت الانتظار.

Résumé

Les robots mobiles sont largement utilisés dans beaucoup des domaines industriels. La recherche sur la planification de la trajectoire pour les robots mobiles est un des sujets les plus importants dans le domaine de la robotique mobile. Il consiste à trouver un chemin libre sans collision avec les obstacles de l'environnement, pour faire déplacer le robot du point de départ au point de destination, tout en satisfaisant certains critères d'optimisation. La plupart des méthodes existantes de la planification, telles que le graphe de visibilité, la méthode de la décomposition en cellules, et la méthode du champ de potentiel sont conçues à des environnements statiques, dans lesquels où il y a seulement des obstacles stationnaires. Cependant, dans les cas réels tels que la recherche maritime, l'industrie minière, le secteur médical et l'industrie nucléaire, les robots font face habituellement à des environnements dynamiques, dans lesquels les obstacles statiques et dynamiques existent. En raison de la complexité des environnements dynamiques, la recherche dans ce domaine est limitée. Nombre limité de papiers ont été publié dans ce secteur en comparaison avec des centaines de rapports qui ont été publié dans la planification de chemin dans les environnements statiques dans la littérature ouverte.

Dans ce travail, on développe une méthode de planification de la trajectoire pour un robot mobile, basée sur les algorithmes génétiques et le concept de temps d'attente (CTA) dans un environnement encombré d'obstacles qui ont des formes, des tailles et des endroits arbitraires. L'algorithme proposé est applicable à des environnements statiques, partiellement dynamiques aussi bien que des environnements dynamiques.

L'approche est divisée en deux phases, statique et dynamique. Dans la phase statique, le Calcul est basé sur les sommets des obstacles stationnaires élargis, réduisant de ce fait l'espace de recherche et le temps de calcul d'une manière significative. L'approche utilise les algorithmes génétiques combinés avec l'algorithme de Dijkstra pour trouver le chemin optimal en créant une population initiale des chemins faisables et infaisables. Puis, elle trouve le chemin optimal pour le robot. Une fois que le chemin est prêt, le robot commence à se déplacer entre les obstacles stationnaires avec des capteurs qui peuvent détecter le moindre changement au voisinage du robot. La phase dynamique est appliquée une fois que les obstacles mobiles sont détectés par ces capteurs. Quand l'obstacle mobile entre dans l'intervalle de détection du robot, les capteurs le détecteront et le robot peut acquérir les informations de cet obstacle mobile tel que sa vitesse et sa direction. Puis, le robot utilise ces informations pour calculer la possibilité de collision avec cet obstacle mobile. S'il y en a, le robot fait appel à l'algorithme du CTA pour éviter la collision avec cet obstacle. L'algorithme calcule le temps d'attente nécessaire au passage de l'obstacle et la position d'attente où le robot peut attendre en toute sécurité. Dans le cas où il n'y a pas de collision entre le robot et l'obstacle, le robot utilise le chemin obtenu par les algorithmes génétiques pour atteindre son but.
Mots clés : robot mobile, planification de la trajectoire, environnement dynamique, algorithmes génétiques, algorithme du concept de temps d'attente

Abstract

Mobile robots are widely used in many industrial fields. Research on path planning for mobile robots is one of the most important aspects in mobile robots research. Path planning for a mobile robot is to find a collision-free route, through the robot's environment with obstacles, from a specified start location to a desired goal destination while satisfying certain optimization criteria. Most of the existing path planning methods, such as the visibility graph, the cell decomposition, and the potential field are designed with the focus on static environments, in which there are only stationary obstacles. However, in practical systems such as Marine Science Research, the Nuclear and Mining Industry, robots usually face dynamic environments, in which both moving and stationary obstacles exist. Because of the complexity of the dynamic environments, research on path planning in the environments with dynamic obstacles is limited. Limited numbers of papers have been published in this area in comparison with hundreds of reports on path planning in stationary environments in the open literature.

This research develops a genetic algorithm based approach with algorithm of waiting time concept for path planning of a mobile robot in dynamic environments.

The designed structure of our approach is divided into two stages : off-line calculation for stationary obstacles and online calculation when moving obstacles are detected. Offline calculation for the path on stationary obstacle based on the vertices of the static obstacles. Using the genetic algorithms combined with the algorithm of Dijkstra to find the optimal path by creating an initial population of feasible and unfeasible paths. Then, it finds the optimal path for the robot. Once the path is ready, the robot starts to travel through the stationary obstacles with the sensor that could detect the 360 degree direction of the robot.

Online path calculation once the moving obstacles are detected by the sensor. As moving obstacle enters the detection range of the robot, the sensor will detect the obstacle and the robot acquire the moving information such as speed and moving direction of the obstacle from the sensor of the robot. Then the robot uses the motion information to calculate the possibility of the moving obstacle clashing with the robot. If the moving obstacle will clash with the robot, the robot use the algorithm of waiting time concept to avoid the collision with this obstacle. The algorithm calculates the necessary time to the passage of the obstacle and the position of waiting where the robot can wait in full safety. If it is calculated that the moving obstacle will not collide with the robot, the robot will use the current path plan to travel through the map.

Keywords : mobile robot, path planning, dynamic environment, genetic algorithm, algorithm of waiting time concept

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La robotique est la discipline s'intéressant au fonctionnement et à l'utilisation des robots pour effectuer des tâches de manière automatique. À ses débuts, les connaissances et les moyens techniques ne permettaient d'envisager que des tâches extrêmement simples, dans des environnements maitrisés et connus auxquels le robot est dédié [27]. Cependant de nos jours, l'objectif de la robotique est beaucoup plus ambitieux. II consiste à construire des robots capables d'évoluer dans des environnements variés, connus ou non. Les robots doivent également réaliser des tâches de plus en plus complexes, de la manière la plus autonome possible. Les applications vont de l'exploration spatiale à la chirurgie, en passant par l'aide à la personne. De même, les robots mobiles sont largement utilisés dans beaucoup de domaines industriels dangereux où il peut y avoir des dangers pour l'homme, tels que l'industrie nucléaire et l'industrie minière [51].

Pour réaliser les tâches dans un monde physique, un système robotique agit sur son environnement par le mouvement. Sa capacité à planifier ses mouvements est donc une composante essentielle de son autonomie. Actuellement, la recherche sur divers algorithmes pour la planification de trajectoire d'un robot mobile reste toujours un défi ouvert. Trouver une trajectoire sûre et sécurisée dans un environnement dangereux pour le robot mobile est une exigence essentielle pour le succès de tous les systèmes robotiques mobiles. Par conséquent, la recherche sur des algorithmes de planification de trajectoire pour déplacer le robot du point de départ au point d'arrêt sans collision avec les obstacles de l'environnement est une condition fondamentale pour la sûreté du robot dans un tel environnement. De plus, pour réduire le temps de calcul, le retard de la communication et la consommation d'énergie, le chemin prévu devrait être optimal en termes de longueur [27].

Les algorithmes génétiques sont parmi les méta-heuristiques qui ont été utilisées dans le domaine de la planification de trajectoire ; ce sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle [23]. Il faut dire qu'ils fournissent d'excellentes performances à de faibles coûts.
Dans ce mémoire, le choix des algorithmes génétiques, pour planifier le chemin le plus court pour le robot, a été guidé par les raisons énumérées ci-dessous :

– Les algorithmes génétiques ont démontré que ce sont des procédures efficaces pour la résolution des problèmes d'optimisation multicritères.
– ils sont un outil puissant pour rechercher dans les espaces de dimension élevés [50].
– C'est l'une des premières méthodes de l'intelligence artificielle à être utilisée dans le domaine de la robotique et qui a donné de bons résultats particulièrement dans la planification de trajectoire et la navigation (exemple, le robot Aïbo de la société SONY [25]).

Récemment, des approches basées sur les algorithmes génétiques et le recuit simulé ont été développées pour la planification de trajectoire du robot dans un environnement dynamique [47] et [51]. Cependant, ces méthodes ont toujours des inconvénients qui se résument ainsi :

– Le temps de calcul pour éviter les obstacles mobiles est très grand pour les deux méthodes.
– Ces deux approches ont besoin de re-planifier le chemin du robot entre le point où il est et le point de destination à chaque fois qu'il rencontre un obstacle dynamique.
– Les deux approches considèrent l'environnement dynamique comme un environnement statique pour chaque instant "t".
– L'efficacité de la première approche n'est pas suffisante pour les problèmes à grande échelle [13].
– Le recuit simulé utilise une solution unique plutôt qu'une population qui contient une collection de solutions, ce qui implique qu'il prend beaucoup de temps pour trouver la solution optimale que les algorithmes génétiques.

Par conséquent, une méthode améliorée pour la planification de trajectoire dans un environnement dynamique s'avère nécessaire.

Dans ce mémoire, on se propose de développer une approche basée sur les algorithmes génétiques combinés avec l'algorithme du 'Concept du Temps d'Attente (CTA)' pour la planification de trajectoire d'un robot mobile dans un environnement dynamique. L'approche utilise les sommets des obstacles statiques comme un espace de recherche. En utilisant les algorithmes génétiques, elle recherche un chemin optimal pour le robot. Quand un obstacle mobile est détecté avec un risque de collision avec le robot, la deuxième étape qui est la planification dynamique est activée. L'approche résoud quelques inconvénients produits dans certaines méthodes précédentes comme :

1. L'approche prend en considération la forme de chaque obstacle dans le calcul, ce qui peut fournir des résultats plus précis que d'autres méthodes comme [49],[52] et [53].
2. L'approche vise à surmonter les principaux inconvénients des travaux existants dé-

crits dans le chapitre I, et utilise une simple représentation de l'environnement. Cette représentation exploite la méthode du graphe de visibilité généré initialement à partir des sommets des obstacles statiques, limitant de ce fait l'espace de recherche, ce qui nous amène à diminuer d'une manière significative le temps de calcul.

3. Elle utilise une représentation variable de la longueur du chromosome, où chaque gène représente un simple sommet d'obstacle afin de réduire au maximum l'utilisation de l'espace mémoire du calculateur en comparaison avec [3].
4. L'approche n'a pas besoin de re-planifier le chemin du robot entre le point où il est et le point de destination à chaque fois qu'il rencontre un obstacle dynamique. Elle fait appel directement à l'algorithme du CTA pour éviter cet obstacle.
5. L'approche ne considère pas l'environnement dynamique comme un environnement statique pour chaque instant "t", ce qui réduit d'une manière considérable le temps de calcul et l'espace mémoire du calculateur.

Notre mémoire s'articule autour de quatre chapitres :

- Le chapitre I résume toutes les notions de base nécessaires à la compréhension du domaine de la robotique mobile, ainsi qu'un panorama de techniques classiques et récentes appliquées dans la planification de la trajectoire d'un robot mobile autonome.
- Le chapitre II introduit, tout d'abord, les notions et les concepts sur les algorithmes génétiques et détaille, par la suite, les différentes étapes et les opérateurs utilisés par les AGs. Leurs principaux avantages et limitations sont abordés ainsi que quelques travaux qui utilisent les algorithmes génétiques pour la planification.
- Le chapitre III explique en détail notre approche de planification, y compris l'application des algorithmes génétiques pour trouver le chemin optimal, puis l'application de l'algorithme du CTA pour éviter les obstacles dynamiques.
- Le chapitre IV comporte les simulations pour tester et valider l'approche mise en œuvre, et l'interprétation des résultats pour différents environnements, et discute également l'influence des différents paramètres sur les résultats obtenus.
- Enfin, nous concluons le manuscrit par une synthèse des résultats obtenus ainsi que par la proposition de quelques perspectives.

Chapitre 1

généralité sur la robotique mobile

1.1 robot mobile

La robotique est une filière pluridisciplinaire qui implique de nombreuses thématiques telles que la mécanique, la mécatronique, l'automatique électronique, l'automatique, l'informatique ou l'intelligence artificielle[8]. En fonction du domaine d'origine des auteurs, il existe donc diverses définitions du terme robot, mais elles tournent en général autour de celle-ci :

Un robot *est une machine équipée de capacités de perception, de décision et d'action (figure 1.1)qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a*[8].

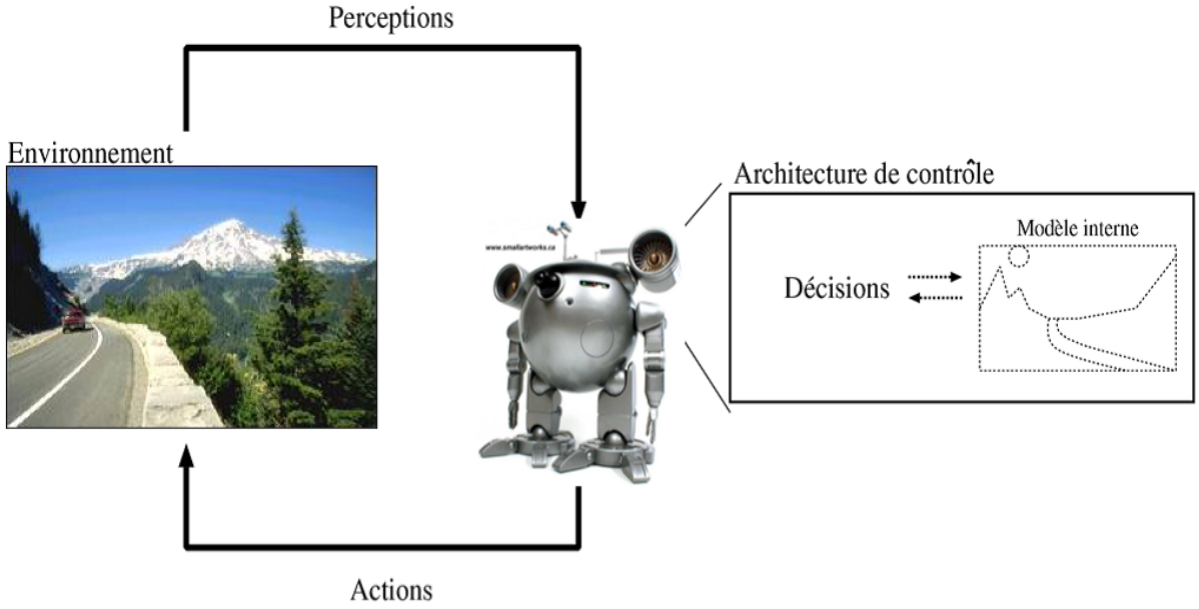


FIGURE 1.1 – Schéma des interactions d'un robot avec son environnement [8]

1.2 Les grandes classes de robots mobiles à roues

De manière générale, on regroupe sous l'appellation robots mobiles l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs. L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues. Les autres robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens.

1.2.1 Robots mobiles de type unicycle

Description

On désigne par unicycle un robot actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité. Le schéma des

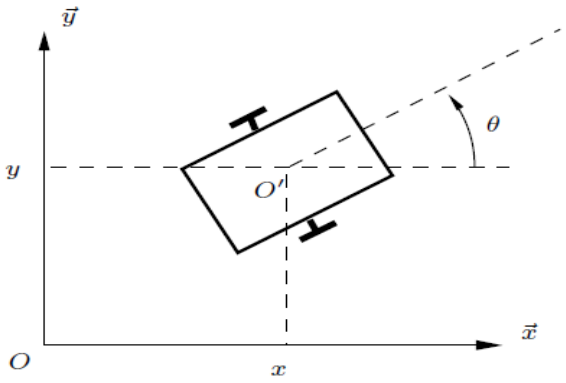


FIGURE 1.2 – Robot mobile de type unicycle [4]

robots de type unicycle est donné à la figure 1.2. On y a omis les roues folles, qui n'interviennent pas dans la cinématique, dans la mesure où elles ont été judicieusement placées(voir figure1.3)[4].

FIGURE 1.3 – Pioneer P3-DX, ActiveMedia Robotics, 2004 [2]

modèle cinématique :

$$\begin{cases} \dot{x} &= v cos(\theta) \\ \dot{y} &= v sin(\theta) \\ \dot{\theta} &= \omega \end{cases} \quad (1.1)$$

où θ : l'angle d'orientation du robot.
ω : vitesse de rotation du robot autour de son centre instantané de rotation.
v : vitesse linéaire du robot et $\dot{x}, \dot{y}$:sont les vitesses linéaires suivant les axes x et y.

1.2.2 Robots mobiles de type tricycle et de type voiture

Ces robots partagent des propriétés cinématiques proches, raison pour laquelle on les a regroupés dans ce paragraphe .

Description

Considérons tout d'abord le cas du tricycle, représenté à la figure 1.4. Ce robot est constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe

longitudinal du robot. Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et l'orientation de la roue orientable. De ce point de vue, il est donc très proche d'une voiture. C'est d'ailleurs pour cela que l'on étudie le tricycle, l'intérêt pratique de ce type de robot restant limité [4]

modèle cinématique :

$$\begin{cases} \dot{x} &= vcos(\theta) \\ \dot{y} &= vsin(\theta) \\ \dot{\theta} &= (v/D)tan(\psi) \\ \dot{\psi} &= \mu \end{cases} \quad (1.2)$$

où θ : l'angle d'orientation du robot.

$\dot{\psi}$: vitesse d'orientation imposée à la roue orientable.

v : vitesse linéaire du robot et $\dot{x}, \dot{y}$:sont les vitesses linéaires suivant les axes x et y.

D : la distance orthogonale entre le centre de la roue orientable et l'axe des roues fixe Le

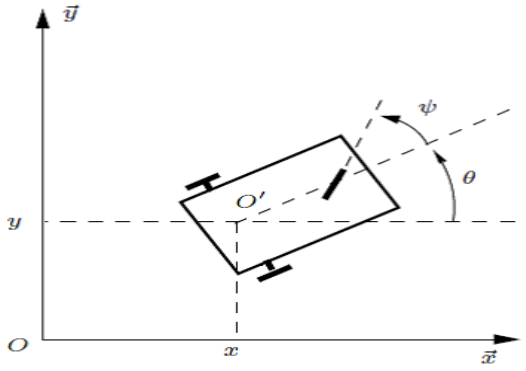


FIGURE 1.4 – Robots mobiles de type tricycle [4]

cas du robot de type voiture est très similaire à celui du tricycle. La différence se situe au niveau du train avant, qui comporte deux roues au lieu d'une (figure 1.5). Cela va de soit, on rencontre beaucoup plus souvent ce type de systèmes. On parle de robot dès lors que la voiture considérée est autonome [36][7], donc sans chauffeur ni télépilotage. Il s'agit là d'un des grands défis issus de la robotique mobile.

1.2.3 Robots mobiles omnidirectionnels

Un robot mobile est dit omnidirectionnel(figure 1.6) si l'on peut agir indépendamment sur les vitesses : vitesse de translation selon les axes x et y et vitesse de rotation autour de z. D'un point de vue cinématique on montre que cela n'est pas possible avec des roues fixes ou des roues centrées orientables [12]. On peut en revanche réaliser un robot omnidirectionnel en ayant recours à un ensemble de trois roues décentrées orientables

FIGURE 1.5 – Robots mobiles de type voiture [4]

FIGURE 1.6 – Représentation d'un robot mobile omnidirectionnel [4]

ou de trois roues suédoises disposées aux sommets d'un triangle équilatéral (voir figure 1.7). Du point de vue de la transmission du mouvement, ceci ne va pas sans poser de problème[4].

1.3 notion de base

L'aspect matériel -de base- d'un robot mobile autonome comporte trois types d'organes [11] :

— les organes de perception

— les unités de traitement

— les organes de locomotion

D'autres organes tels que des bras manipulateurs peuvent lui être ajoutés pour une application particulière.

1.3.1 perception

La notion de perception en robotique mobile est relative à la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir dans le monde qui l'entoure. Alors que pour des taches de manipulation on peut considérer

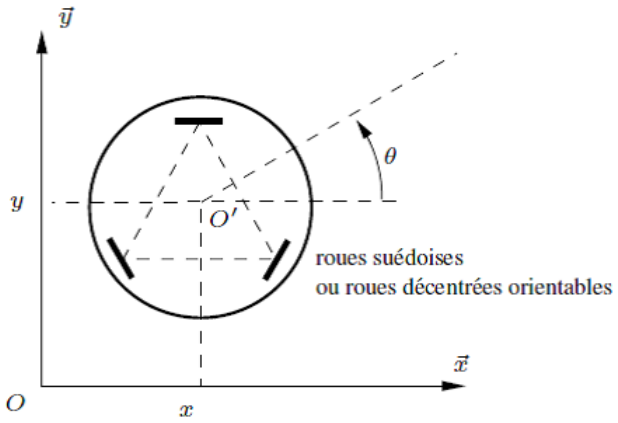


FIGURE 1.7 – Robot mobile omnidirectionnel Nomadic XR4000 [4]

que l'environnement du robot est relativement connu, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée[4].

1.3.2 décision

Elle constitue l'intelligence du robot. A l'instar de l'Homme, le raisonnement du robot lui permet de décider d'une action appropriée à une situation donnée, compte-tenu d'une mission à réaliser. Plusieurs tâches élémentaires s'exécutent en parallèle pour synthétiser un comportement. La façon dont elles interagissent est définie par l'architecture décisionnelle du robot. Celle-ci comprend notamment un contrôleur d'exécution, dont le but est de lancer l'exécution d'un groupe de tâches donné à intervalles de temps réguliers. Chaque cycle (ou itération) du contrôleur correspond à une décision du robot, et génère un ordre (appelé consigne) envoyé à la partie action. On parle de traitement itératif[11] .

1.3.3 action

Cette fonction est également appelée « contrôle » du robot. L'action essentielle du robot mobile est de se déplacer. Par conséquent, la consigne est un déplacement désiré et le contrôle regroupe l'ensemble des opérations permettant au robot d'effectuer ce déplacement. Il se fait en deux étapes :
1. La génération de la commande (ensemble de paramètres de contrôle des effecteurs) supposée impliquer le déplacement escompté.
2. L'application de cette commande, c'est-à-dire sa conversion en signaux directement utilisables par les effecteurs

1.3.4 Navigation

La navigation d'un robot mobile consiste à trouver un mouvement qui amène le robot d'une configuration initiale à une configuration finale[4]. Généralement, ceci exige deux facultés : la planification de trajectoires et la réaction pour éviter des obstacles. De ce fait, pour différencier les techniques de navigation, on peut distinguer en principe trois approches

– Approches délibératives (**globales**) :
ces approches travaillent sur une représentation complète des C-obstacles et de l'espace libre. Le temps de calcul de l'espace des configurations croit exponentiellement avec le nombre de degrés de liberté. Ainsi les méthodes globales ne sont applicables qu'à des robots ayant un faible nombre de degrés de liberté. Le fonctionnement est basé sur un cycle rigide de modélisation de l'environnement, planification des actions au sein de cette représentation tout en évitant les obstacles, puis exécution du plan.
– Approche réactive (**locales**) :
Elle ne suppose aucun modèle a priori de l'environnement. Cette architecture s'appuie sur le couplage étroit entre les capteurs et les actionneurs, pour générer en continu les commandes. Elle consiste à offrir un ensemble de primitives plus réactives. Elles correspondent à des sous-tâches (convergence vers une cible, suivre un mur, éviter un obstacle) dont l'enchaînement est du ressort d'un planificateur de tâches ayant décomposé la tâche globale.
– Approche hybride : utilise une combinaison des deux méthodes précédentes.

1.3.5 localisation,cartographie et planification

Le processus complet qui permet à un robot de mémoriser son environnement, puis de s'y déplacer pour rejoindre un but, peut être découpé en trois phases : la cartographie, la localisation et la planification. Ces trois phases permettent d'informer le robot sur sa position actuelle et sur l'environnement qui l'entoure et de définir la stratégie pour atteindre le but[8].

- La cartographie est la phase qui permet la construction d'une carte reflétant la structure spatiale de l'environnement à partir des différentes informations recueillies par le robot.
- Une telle carte étant disponible, la localisation permet alors de déterminer la position du robot dans la carte qui correspond à sa position dans son environnement réel.
- Enfin, la planification est la phase qui permet, connaissant la carte de l'environnement et la position actuelle du robot, de prévoir les mouvements à effectuer afin de rejoindre un but fixé.

1.3.6 L'architecture de contrôle

La manière dont un robot gère le cycle Perception/Décision/Action est définie par son architecture de contrôle, qui la plupart du temps va faire appel à un modèle interne de l'environnement qui lui permettra de planifier ses actions à long terme[4].

définition

Une architecture de contrôle logicielle est un ensemble de composants logiciels mis on œuvre selon une politique, permettant leur organisation, communication et interaction[8]. Ces composants collaborent entre eux pour réaliser le contrôle du robot, qui sera d'ordonner les actions à entreprendre par les composants physiques afin d'aboutir à un but précis

1.3.7 L'espace des configurations

Une configuration (au sens géométrique) est un ensemble de paramètres qui fixent de façon unique la position d'un robot dans l'espace. L'espace des configurations est défini comme le produit cartésien des espaces correspondant à chacun de ces paramètres. Par définition, l'état du robot est représenté dans cet espace par un point. En général, les régions occupées par des obstacles on les nomme les C-obstacles. La planification d'une tâche consiste à trouver une trajectoire pour le point représentant le robot, de la configuration initiale à la configuration finale en évitant les C-obstacles[35].

1.4 Classifications des méthodes de planification de la trajectoire du robot mobile

La planification de trajectoire pour les robots mobiles est un des aspects les plus importants dans la recherche de la navigation de robot. Les méthodes de planification de trajectoire de robot peuvent être classifiées dans différentes sortes basées sur différentes situations. Selon l'environnement où le robot est localisé, les méthodes de planification de trajectoire peuvent être classifiées selon les deux types suivants (Figure 1.8) [13] :

1. Planification de la trajectoire du robot dans un environnement statique qui contient seulement les obstacles statiques.
2. planification de la trajectoire du robot dans un environnement dynamique qui contient des obstacles statiques et dynamiques.

Chacun des deux types pourrait être encore divisé en deux sous-groupes selon la connaissance entière de l'environnement :

1. planification de la trajectoire du robot dans un environnement complètement connu dans lequel le robot connaît déjà l'endroit des obstacles avant qu'il commence à se

déplacer. La trajectoire pour le robot pourrait être le résultat optimisé global parce que l'environnement entier est connu.

2. planification de la trajectoire du robot dans un environnement partiellement connu ou incertain, dans lequel le robot perçoit l'environnement à l'aide de ses capteurs pour acquérir les informations locales de l'endroit, de la forme et de la taille des obstacles, et utilise alors ces informations pour procéder à une planification locale de la trajectoire [13].

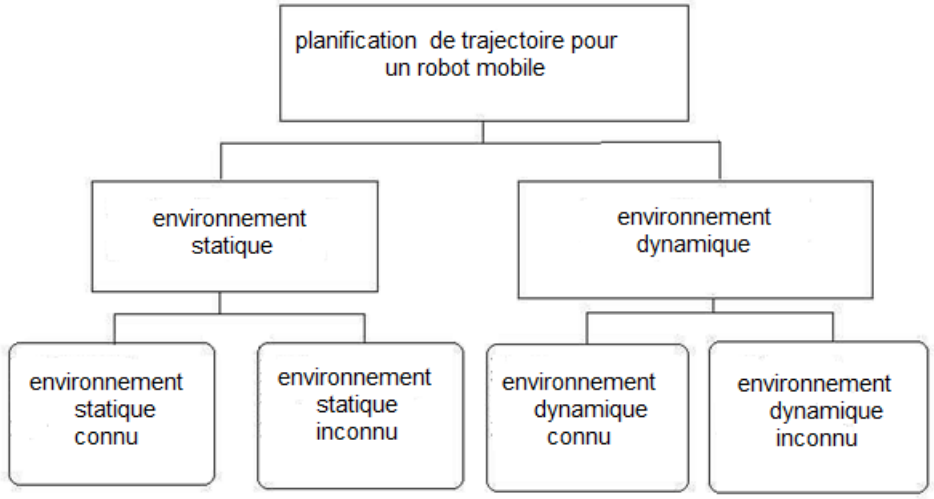


FIGURE 1.8 – classification des méthodes de planification [8]

1.4.1 planification dans un environnement statique connu

Dans un environnement statique et connu, le robot sait l'information entière de l'environnement avant qu'il commence à se déplacer. Par conséquent le chemin optimal peut être calculé avant le démarrage du robot.
Les techniques de planification du chamin pour un environnement statique et connu sont très traitées. Les méthodes représentatives pour la planification du chemin pour l'environnement statique et connu incluent :
-Méthode de Graphe de Visibilité
- Diagramme de Voronoi
-The méthode de décomposition en cellules
-Méthode de champs de potentiel
Les quatre premières méthodes sont plutôt des modèles de l'environnement qui peuvent être utilisés par des techniques de planification comme A*, dijkstra...
Certaines techniques utilisent l'intelligence artificielle qui consiste à apprendre des relations inspirées de la nature, ces approches comportent les algorithmes génétiques, l'algorithme de recuit simulé, et autre méthodes d'optimisation. Ces algorithmes ont été

également utilisés pour obtenir le chemin optimale pour le robot.
Davidor [34] a développé un algorithme génétique avec un opérateur modifié de croisement pour optimiser le chemin de robot.
Nearchou [3] a utilisé le nombre de sommets produits dans le graphe de visibilité pour construire des chromosomes de longueur fixe dans lesquels la présence d'un sommet dans le chemin est indiqué par un arrangement de bits au lieu approprié. Son algorithme est capable de déterminer une solution proche-optimale.
Cai et Peng [14] ont développé un mécanisme de codage décimal de longueur constante pour remplacer le mécanisme de codage de longueur variable et d'autres mécanismes de codage binaires de longueur constante utilisés dans l'approche génétique pour la planification de trajectoire de robot.

graphe de visibilité[16] [26] [35]

L'idée principale de cette méthode est de créer un graphe G avec les sommets des obstacles polygonaux, la configuration initiale q_i,et la configuration finale q_f .
Deux nœuds du graphe sont connectés entre eux s'il existe dans l'espace physique une trajectoire en ligne droite sans collision permettant d'aller de l'un à l'autre. Ces lignes décrivent l'espace libre. Des translations sont exécutées le long des chemins libres et les rotations sont réalisées aux intersections, c'est-à-dire dans les nœuds. La planification devient alors un problème de théorie des graphes. Il faut trouver le chemin (roadmap) entre les nœuds du graphe qui connecte le nœud initial au nœud final.

étapes

le déroulement de cette méthode se fait en deux étapes Constituants son algorithme général :
-construire le graphe G à partir du chemin et des nœuds q_i et q_f
-chercher un chemin de qi vers qf dans le graphe G
Un exemple de la construction du graphe G à partir des obstacles et des configurations q_i et q_f est montré dans la figure 1.9. La résolution du graphe est dépendante du nombre de nœuds et de la fonction heuristique de recherche.

Inconvénients Cette méthode permet de trouver le chemin le plus court dans le graphe G de la configurationq_i à la configuration q_f. Il est possible de trouver une solution avec cette méthode si elle existe, cependant la méthode présente les inconvénients suivants :

- chaque fois que la configuration initiale et la configuration finale sont modifiées, le graphe G doit être en partie recalculé. (les arcs pour les nœuds initial et final)

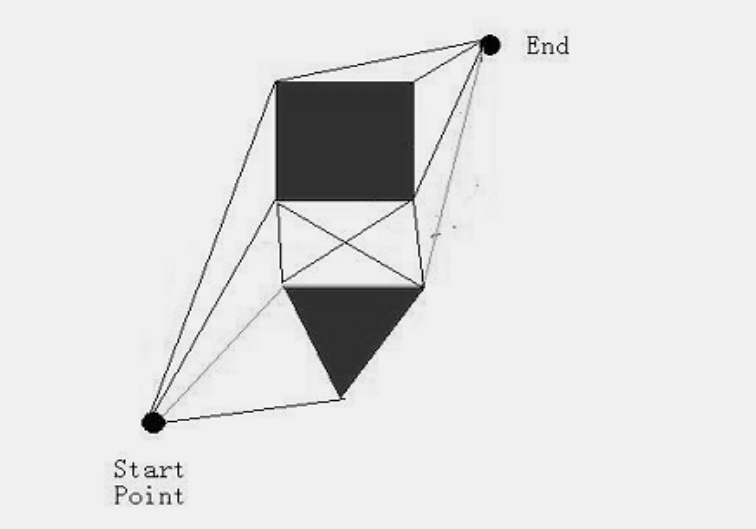


FIGURE 1.9 – graphe de visibilité [13]

– le cout de construction de G est très élevé

– les chemins trouvés sont des chemins qui amènent le robot près des obstacles

– la méthode peut générer la solution la plus courte dans le graphe, mais elle n'est pas nécessairement la plus courte dans l'espace.

L'algorithme de construction du graphe G :

L'algorithme de la construction du graphe de visibilité est décrit par Le pseudo-code ci-dessous [13] :

1 : pour chaque paire des nœuds u, v
2 : si segment (u, v) est un côté d'obstacle
3 : faire insérer le segment (u, v) dans G ;
4 : sinon
5 : pour chaque côté d'obstacle e
6 : si le segment (u, v) coupe e
7 : aller à (1)
8 : insérer le segment (u, v) dans G

Techniques utilisées pour la recherche du chemin

Des techniques comme l'algorithme A* et tous les algorithmes d'optimisation tels que l'algorithme génétique (comme le cas de notre technique) et l'algorithme de recuit simulé peuvent être utilisés pour calculer le chemin optimal pour le robot. De la même manière des techniques pour la réduction du nombre de nœuds de G peuvent être utilisées [35].

Algorithme A* [44]

L'algorithme A* est un algorithme de recherche dans un graphe : cette recherche est guidée par une heuristique, qui donne pour chaque nœud du graphe un minorant de la distance restant à parcourir avant d'atteindre le nœud final. A chaque étape de la recherche, l'algorithme propage l'ensemble des nœuds que l'on sait atteindre depuis le nœud de départ, en favorisant les nœuds les plus Intéressants (ceux dont la somme entre la distance déjà parcourue depuis le nœud de départ et l'estimation de la distance restant à parcourir est minimale).
L'inconvénient de cet algorithme est son coût, et ce pour deux raisons. D'une part, lorsque la construction de l'arbre de recherche atteint un espace occupé par un obstacle, elle se propage très lentement le long de cet obstacle. De plus, l'algorithme n'échoue qu'après avoir étendu autant que possible l'arbre de recherche, avec une précision , ce qui demande de nombreuses itérations. Cet inconvénient est compensé par le fait que son coût peut être diminué en augmentant , c'est-à-dire en diminuant la précision de la recherche.

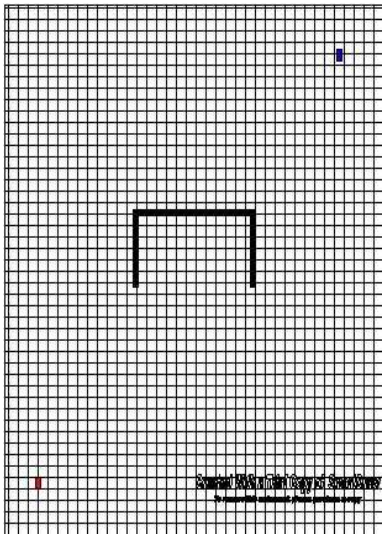


FIGURE 1.10 – Représentation de l'environnement pour l'algorithme A* [8]

Algorithme de Dijkstra[18]
L'algorithme proposé par Dijkstra, permet de résoudre le problème de recherche du chemin optimal, de longueur minimale, dans un environnement schématisé par un graphe. Il assure la recherche de tous les chemins possibles du nœud source vers le nœud destination (ou but). Un seul chemin dont le coût est minimum est retenu. Le principe de cet algorithme repose sur le fait que si un nœud P quelconque du graphe est situé sur le chemin minimal du nœud source S au nœud destination D, alors le chemin minimal de S à P fait partie du chemin minimal de S à D. Dijkstra a eu l'idée de marquer chaque nœud visité et empêché ainsi son algorithme de revenir en arrière.
L'algorithme de Dijkstra possède une complexité en temps, malgré cela, il est intéressant puisqu'il ne nécessite pas le développement de l'ensemble des chemins possibles. Par contre l'ensemble des nœuds doit être atteint avant d'affirmer que la solution optimale a

été trouvée.

Algorithme de recuit simulé

Le recuit simulé est une méta-heuristique inspirée d'un processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lent et de réchauffage (recuit) qui tendent à minimiser l'énergie du matériau. Elle est aujourd'hui utilisée en optimisation pour trouver les extrema d'une fonction.

Déroulement du processus

Le recuit simulé s'appuie sur l'algorithme de Metro-polis-Hastings, qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système. Partant d'une solution donnée, en la modifiant, on en obtient une seconde. Soit celle-ci améliore le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système, soit celle-ci le dégrade. Si on accepte une solution améliorant le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une « mauvaise » solution permet alors d'explorer une plus grande partie de l'espace de solution et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local.

État initial de l'algorithme

La solution initiale peut être prise au hasard dans l'espace des solutions possibles. À cette solution correspond une énergie initiale E = E0. Cette énergie est calculée en fonction du critère que l'on cherche à optimiser. Une température initiale T = T0 élevée est également choisie. Ce choix est alors totalement arbitraire et va dépendre de la loi de décroissance utilisée.

Itérations de l'algorithme

À chaque itération de l'algorithme, une modification élémentaire de la solution est effectuée. Cette modification entraîne une variation E de l'énergie du système (toujours calculée à partir du critère que l'on cherche à optimiser). Si cette variation est négative (c'est-à-dire qu'elle fait baisser l'énergie du système), elle est appliquée à la solution courante. Sinon, elle est acceptée avec une probabilité $e^{-\Delta E/T}$. Ce choix de l'exponentielle pour la probabilité s'appelle règle de Metro-polis. On itère ensuite selon ce procédé en gardant la température constante.

Programme de recuit

Deux approches sont possibles quant à la variation de la température : Pour la première, on itère en gardant la température constante. Lorsque le système a atteint un équilibre thermodynamique (au bout d'un certain nombre de changements), on diminue la température du système. On parle alors de paliers de température. La seconde approche fait baisser la température de façon continue. On peut alors imaginer toute sorte de loi de décroissance. La plus courante étant $T_i + 1 = XT_i$ avec $X < 1$ (assez couramment $X = 0.99$). Dans les

2 cas, si la température a atteint un seuil assez bas fixé au préalable ou que le système devient figé, l'algorithme s'arrête. La température joue un rôle important. À haute température, le système est libre de se déplacer dans l'espace des solutions ($e^{-\Delta E/T}$ proche de 1) en choisissant des solutions ne minimisant pas forcément l'énergie du système. À basse température, les modifications baissant l'énergie du système sont choisies, mais d'autres peuvent être acceptées, empêchant ainsi l'algorithme de tomber dans un minimum local.

Avantages

L'avantage principal de cet algorithme, c'est sa capacité d'échapper des minimums locaux.

Inconvénients

Les principaux inconvénients du recuit simulé résident dans le choix des nombreux paramètres, tels que la température initiale, la loi de décroissance de la température, les critères d'arrêt ou la longueur des paliers de température. Ces paramètres sont souvent choisis de manière empirique.

la methode de diagramme de voronoi[35]

Il s'agit de l'utilisation des espaces libres obtenus par les diagrammes de Voronoi. Dans un espace de travail polygonal muni d'obstacles polygonaux, le diagramme de Voronoi est un réseau de segments linéaires et paraboliques. Une fois le diagramme de Voronoi obtenu, le chemin le plus court peut être trouvé, grâce aux techniques de la théorie des graphes. Trois types de nœuds sont identifiés dans le graphe, les nœuds de raccordement, les nœuds terminaux et les pseudo-nœuds.
Sur la figure 1.11 :

-les nœuds 1, 2, 11 et 12 sont des nœuds terminaux et correspondent aux nœuds morts.
-Les nœuds 3 à 10 sont des nœuds de raccordement car ils comportent trois arcs ou plus.
- Finalement les pseudo-nœuds sont les nœuds artificiels c'est-à-dire les nœuds correspondant aux configurations initiale et finale.
Les nœuds de raccordement et les nœuds terminaux constituent le chemin [16] [26]. Comme dans le graphe de visibilité, si les configurations initiale ou finale sont modifiées alors le graphe doit être recalculé.

La technique du graphe de Voronoi est similaire à l'utilisation de l'espace des configurations. Elle peut être appliquée à des problèmes compliqués de planification de trajectoires, ou l'espace libre est réduit [26]. Ces problèmes, ne peuvent pas parfois être résolus par les méthodes des espaces libres.

Avantages

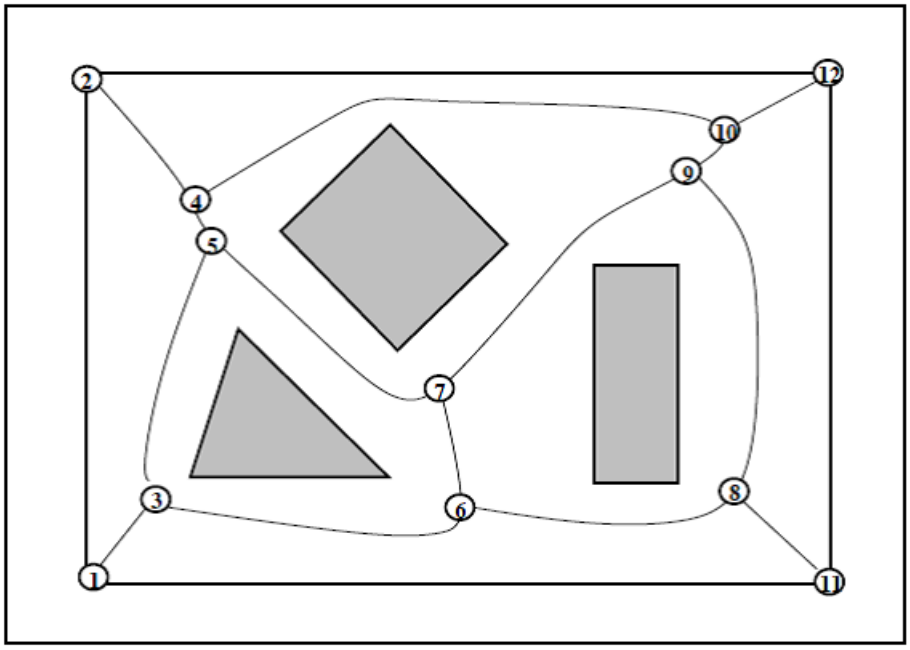


FIGURE 1.11 – graphe de Voronoi [35]

-Cette méthode permet de trouver les trajectoires qui amènent le robot par des chemins éloignés des obstacles.
-La technique semble avoir des avantages sur les autres méthodes qui utilisent les espaces libres. En particulier, il est possible d'obtenir des solutions plus rapides puisque le cout de calcul du graphe est moins cher.
-Dans la plupart des techniques qui utilisent les espaces libres, uniquement des rotations sont faites dans les endroits où il y a une intersection des chemins libres L'utilisation des diagrammes de Voronoi permet de faire des rotations et des translations en même temps.

La méthode avec décomposition en grilles [35]

La décomposition de grilles est la division de l'espace de navigation en sous-espaces appelés cellules. La planification d'une trajectoire est alors la génération d'un chemin entre deux sous-espaces. Chaque cellule libre d'obstacles représente un nœud. Deux nœuds sont connectés entre eux si les deux cellules qui représentent les nœuds sont adjacentes. Les méthodes de décomposition de grilles peuvent être divisées en exactes et approximatives. Pour les méthodes exactes il s'agit d'avoir des cellules complètement libres, c'est-à-dire l'espace libre exact est représenté à par les cellules, voir figure 1.12. Les méthodes approximatives font une représentation partielle de l'espace libre, voir figure 1.13. Il y a plusieurs techniques pour faire la division de l'espace de navigation. Par exemple dans la figure 1.12, l'espace est décomposé en cellules polygonales. Les Cellules sont construites en dessinant des lignes verticales à partir des sommets des obstacles.
Une autre technique est la division récursive de l'espace comme cela est montré dans la figure 1.13. La taille du graphe d'états obtenu peut cependant beaucoup varier en fonc-

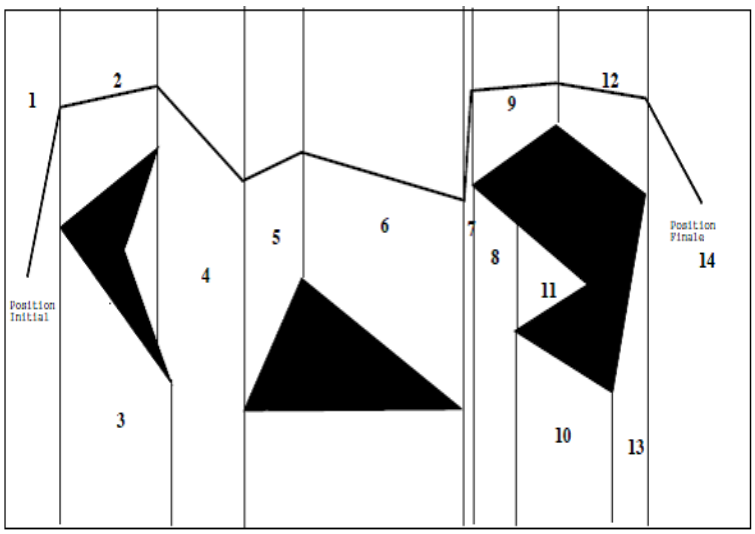


FIGURE 1.12 – décomposition exacte [35]

tion du procédé de découpage utilisé. Le problème de cette méthode est de trouver la mesure des cellules, car des cellules petites font un graphe plus complexe. Par contre, des cellules plus grandes peuvent amener l'algorithme à ne pas trouver la solution quand elle existe[35].

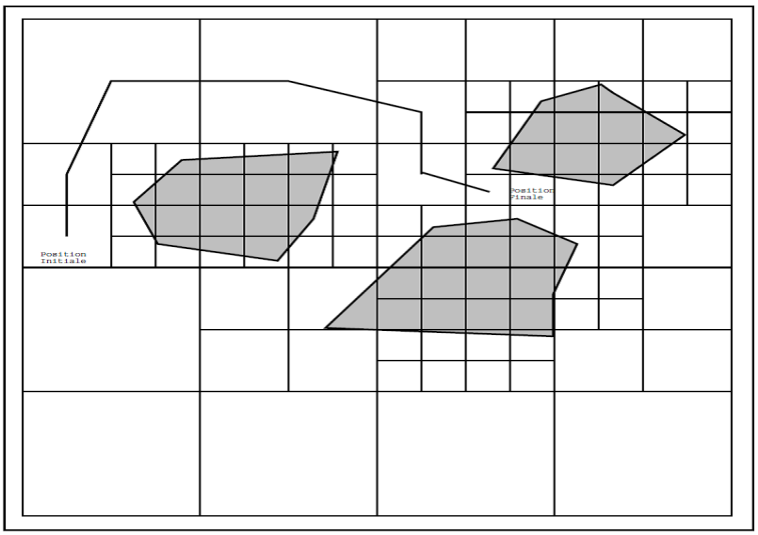


FIGURE 1.13 – décomposition approximative [35]

1.4.2 Planification dans un environnement statique inconnu

Dans un environnement statique et inconnu, la navigation sera plus difficile que dans le cas statique où l'environnement est connu .Cette difficulté est due à l'incertitude d'information pour l'environnement, tel qu'il est difficile d'obtenir des résultats optimaux et le robot doit utiliser des informations locales pour calculer son chemin. Quelques méthodes de planification du chemin dans l'environnement statique et inconnu sont présentées ci-dessous :

La méthode de champs de potentiel

Cette méthode a été proposée par KHATIB en 1986 pour un robot manipulateur [16] [11]. L'idée principale de cette méthode est de considérer le robot comme un point matériel sous l'influence de deux forces, l'une répulsive due aux obstacles et l'autre attrayante due au point de destination. L'ensemble de forces peut être représenté par un champ de courbes équipotentielles [16] comme montré dans la figure 1.14 : Le robot est muni alors

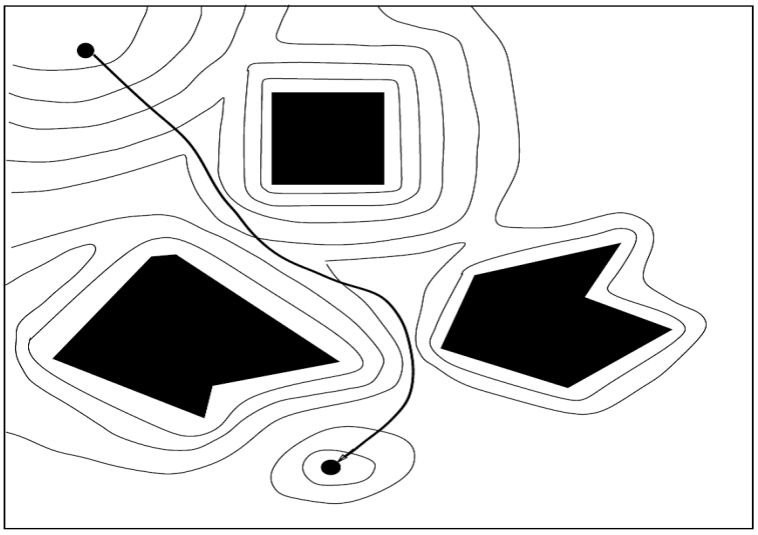


FIGURE 1.14 – méthode de champs de potentiel [35]

d'une énergie potentielle en se trouvant dans un point plus haut que l'objectif. L'énergie potentielle obligera la particule à descendre par un chemin qui ne touche pas d'obstacle car la force répulsive est infinie en chaque point qui forme un obstacle (voir figure 1.14). La fonction potentielle prend un minimum local correspondant à la configuration finale. De cette manière la trajectoire atteindra la configuration finale. En comparaison avec les autres méthodes, cette technique est très efficace, par contre il est possible de tomber dans des minimums locaux qui ne sont pas la configuration finale, si la fonction n'est pas bien définie. La méthode peut être aidée par des heuristiques pour éviter ce problème Pour remédier à ce problème, Park and Lee [21] ont proposé une combinaison de la méthode de champs de potentiel avec l'algorithme du recuit simulé

La méthode stochastique

Il s'agit d'une technique qui combine les méthodes potentielles avec une recherche aléatoire [16]. Pour un espace des configurations E_c de dimension n l'axe de coordonnées est dénoté par (θ_1,θ_2 ,...,θ_n). Cet espace est divisé en cellules où une augmentation dans la direction de l'axe θ_i est représentée par $\Delta\theta_i$. Soit U la fonction potentielle, q un point (θ_1,θ_2 ,...,θ_n) dans l'espace et q_f la configuration finale alors q $\in$ espace des configurations, $U(q) >= 0$. Si U(q)=0 alors q est la configuration finale cherchée. Le planificateur construit et cherche un graphe G tel que les nœuds du graphe soient des minimums locaux

de la fonction potentielle U. Deux minimums locaux sont connectés si le planificateur a construit un chemin entre eux. L'algorithme commence la recherche avec la configuration initiale q_i . A partir de cette configuration il exécute un mouvement "meilleur voisin" c'est-à-dire l'algorithme passe de la configuration q_i à q_1 tel que q_1 est un voisin de q_i. La configuration q_1 est égale à q_i $+\Delta$ θ_i. et U (q_i) $<$ U(q_1). Le planificateur fait de même pour q_2,..., q_{loc}. où q_{loc} est un minimum local. Si U(q_{loc})=0 alors l'algorithme a trouvé une solution. Sinon si U(q_{loc})>0 le planificateur essaie d'échapper au minimum local par l'exécution d'un ensemble de mouvements aléatoires, à partir de q_{loc}. Chaque mouvement aléatoire est suivi d'une recherche "meilleur voisin" qui atteint un minimum local. Si ce minimum est différent de q_{loc}, il est remplacé comme un successeur de qloc dans le graphe G. A ce moment, les deux minimums adjacents sont connectés par le chemin qui a été trouvé. par la recherche d'un mouvement aléatoire et d'une recherche "meilleur voisin". Le graphe G est augmenté jusqu'à trouver la configuration finale. Si la recherche est terminée avec succès, le chemin construit est transformé en un chemin régulier. Cette transformation réalisée une optimisation de la trajectoire engendrée. Les expériences de cette méthode montrent qu'elle est très efficace. L'algorithme engendre des chemins différents s'il est lancé plusieurs fois sur un même problème. Cependant il n'est pas possible d'identifier quand il n'y a pas de solution au problème car un temps limité d'exécution doit être pris en compte dans l'algorithme

Autres méthodes

Ying and Eicher [24], ont proposé une méthode de planification en temps réel basée sur les capteurs pour un robot sous-marin dans un environnement statique et inconnu. L'environnement est constitué par des obstacles statiques de diverses formes. Un algorithme tridimensionnel de planification du chemin est développé pour le robot dans un tel environnement. Ils ont utilisé le sonar pour détecter les obstacles statiques dans l'environnement.

Park et Lee [21] ont proposé un concept virtuel modifié de colline basé sur la méthode des champs de potentiel pour échapper à des minimums locaux dans l'environnement local. Une colline virtuelle avec le potentiel supplémentaire est ajoutée à l'obstacle pour piéger le robot quand il est emprisonné en un minimum local. Le potentiel supplémentaire est ajouté au potentiel global, qui peut repousser le robot du minimum local. Cependant, à cause de l'insuffisance de l'information des obstacles dans l'environnement, le résultat global d'optimisation ne peut pas être acquis et seulement le résultat optimal local est atteint.

Récemment, Chang et Yamamoto [42] combinent l'approche de diagramme de Voronoi avec la méthode de champs de potentiel pour éviter le problème d'emprisonnement du

robot (minimum local) dans la dernière méthode. Le problème est résolu en définissant des buts secondaires entre la configuration initiale et finale (cible) sur la carte construite.

Les problèmes de planification dans un environnement statique inconnu ont été largement abordés au cours de ces dernières années. Dans la littérature, il y a beaucoup de travaux qui ont été développés dans ce sens, certains sont des combinaisons de méthodes connues et d'autres de nouvelles techniques.
Actuellement, le problème de planification du chemin est toujours une matière préférée pour la recherche en robotique et en intelligence artificielle. La planification dans l'environnement statique inconnu sera toujours un objet de recherche à l'avenir [13].

1.4.3 Planification de la trajectoire dans un environnement dynamique connu

Des centaines d'articles traitants le cas de l'environnement statique ont été publiés dans différentes revues scientifiques [13], par contre on trouve très peu de documentation qui traite le cas dynamique. En pratique l'environnement où le robot navigue n'est pas tout le temps statique, il y'a toujours des obstacles qui bougent d'une manière déterministe ou aléatoire.
Ce dernier cas est plus compliqué que le premier, cette complexité augmente avec l'augmentation du nombre d'obstacles dynamiques. Par conséquent, les algorithmes traditionnels de planification de trajectoire qui utilisent les modèles de l'environnement, tels que le graphe de visibilité, le diagramme de voronoi et la méthode de grilles, ne sont pas performants dans les environnements dynamiques. Dans ce qui suit on va citer quelques approches trouvées dans la littérature :

En 2005, Chestnutt, Lau et Cheung [15] ont utilisé un algorithme A* modifié pour calculer la trajectoire pour un robot de humanoïde de Honda ASIMO. La méthode de planification de chemin (A* modifié) est appliquée à de vrais robots plutôt que simulée. Une grille des cellules est utilisée pour représenter l'environnement.
Une méthode de navigation hybride [13] a été proposée en 2003. Cette méthode comporte deux modules : global et local, qui peut combiner deux méthodes de navigation pour exploiter les informations de la carte globale et les informations locales qui viennent des capteurs.
(1) Le module global indique les positions globales de la trajectoire. Il utilise les informations préalables sur l'environnement de navigation et choisit les points critiques pour le passage avant de commencer la navigation réelle. Ce module utilise l'algorithme A* pour déterminer la trajectoire optimal vers le but.
(2) Le module local effectue la navigation elle-même, en se basant sur des données cou-

rantes des capteurs, ce qui facilite l'évitement des obstacles statiques ou dynamiques. Ce module utilise une représentation neuro-floue de la méthode de champs de potentiels. Premièrement, le module global trouve la trajectoire optimal et propose des positions intermédiaires. Ces points intermédiaires seront transférer vers le module local point par point. Séquentiellement, le robot va chercher le point indiqué et évite par réaction Les obstacles présentés dans l'environnement, en utilisant une fonction potentielle Précédemment fourni au module local.

1.4.4 Planification de la trajectoire dans un environnement dynamique inconnu

La navigation dans un environnement dynamique et inconnu, c'est le cas le plus difficile pour le robot mobile. Ce cas est fréquemment rencontré dans la pratique,c'est le cas pour un robot mobile de type voiture autonome, ou un robot sous-marin qui est chargé de compléter sa tâche tout en évitant les obstacles statiques (les roches ...) et aussi des obstacles mobiles (la vie sous-marine).A cause de cette complexité, il est très difficile (ou parfois impossible) de planifier la trajectoire globale. Le robot doit exploiter les informations perçues localement par les capteurs et planifier sa trajectoire en temps réel. Quelques travaux récents sont présentés ci-dessous :

Algorithme D*[13]

Le D * (version dynamique de A*) c'est la méthode typique utilisée pour la planification de chemin dans les environnements inconnus et dynamiques. Le nom D* a été choisi à cause de sa ressemblance avec l'algorithme A* à l'exception qu'il est dynamique dans le sens où les paramètres de coût entre les cellules ou carrés peut changer durant le processus de résolution du problème de planification de la trajectoire. L'algorithme génère une trajectoire globale en utilisant un modèle déjà établi de l'environnement. En cours de suivi de cet trajectoire, le système mobile peut rencontrer un obstacle qui n'était pas défini au préalable, dans ce cas, le système peut re-planifier son chemin pour atteindre son but sans heurter l'obstacle. D * n'exige pas la re-planification complète de trajectoire lorsque des nouvelles informations entrent. Il met à jour le coût de l'arc (les cases voisines) de trajectoire localement quand l'environnement change. Ce qui permet de réduire considérablement le temps de calcul. Cependant, D* essaye toujours d'obtenir la trajectoire globalement optimale si possible.

Appartenant à la famille des algorithmes de re-planification rapide, l'algorithme D* a comme avantage, par rapport à l'algorithme A*, qu'il peut être utilisé en ligne . Son inconvénient réside dans le fait que la cinématique des systèmes mobiles est très peu prise

en compte.

Autres travaux

En 2006, N. Lave et Feng [13] ont présenté une combinaison de la méthode des champs de potentiel avec l'algorithme de colonie de fourmi pour trouver une trajectoire dans un environnement dynamique. Le mouvement des obstacles est pris en compte en changeant les valeurs du champ de potentiel local. La méthode utilise premièrement un champ de potentiel dynamique évalué pour modéliser l'environnement dynamique, puis elle utilise une méthode d'optimisation de colonie de fourmi pour rechercher la trajectoire par une planification locale.

En 2007, Wang, Sillitoe et Mulvaney [46] ont présenté un planificateur à base des algorithmes génétiques pour déterminer les solutions optimales ou proche-optimales pour un robot mobile dans des environnements dynamiques. Les obstacles sont décrits en tant que polygones élargis et les dimensions du robot sont négligées (considérée comme un point). La méthode utilise premièrement la planification offline pour planifier un chemin, après le robot commence à voyager à l'aide de ses capteurs .La méthode online de planification de chemin est activée pour calculer un chemin alternatif si des obstacles sont détectés. A cause de la complexité et de l'incertitude de l'environnement, les travaux pour la planification de la trajectoire complète sont très limités.

Zheng et Zhao [39] ont proposé une approche basée sur les champs de potentiel artificiel en 2006. La méthode planifie des chemins pour cinq robots mobiles dans un environnement inconnu dynamique. Les robots passeront le secteur d'intersection dans l'ordre selon leurs priorités, ce qui est prédéfini avant le mouvement. La méthode suppose que les obstacles dans la carte ont des tailles identiques. Elle coordonne avec succès les mouvements des robots multiples, et fait passer les robots par les obstacles statiques et les autres robots sans collision. La nouveauté de la recherche est l'application de la méthode de champ de potentiel à la planification des trajectoires pour plusieurs robots dans le même environnement dynamique.

Tarokh [22] développe une approche intelligente hybride à la planification de la trajectoire pour un robot de haute mobilité naviguant dans un terrain accidenté. La planification de la trajectoire se compose de la caractérisation de l'environnement en utilisant un cadre de logique floue, et un planificateur d'algorithme génétique à deux étages. Un planificateur global détermine la trajectoire qui optimise une combinaison de la rugosité de terrain et de la courbure de trajectoire. Un planificateur local utilise l'information des capteurs, et

en cas de détection des obstacles inattendus effectue une re-planification en ligne pour l'éviter.

Hui Miao [13] a proposé un algorithme de planification avec le recuit simulé valable pour un environnement dynamique inconnu .cet approche utilise les sommets des obstacles statiques comme un espace de recherche. En utilisant des méthodes de recherche heuristique, elle cherche une trajectoire initiale faisable. Puis, l'approche utilise l'algorithme de recuit simulé pour trouver la trajectoire optimale du robot. Lorsqu'un obstacle est détecté avec une possibilité de collision avec le robot, ces deux étapes sont activées de nouveau pour une planification dynamique.

Jusqu'à maintenant, la plupart des activités de recherches sur la planification de trajectoire de robot dans les environnements dynamiques ont été en général théoriques. Certaines méthodes de ce type ont été proposées pour traiter la question de la planification de chemin de robot dans l'environnement incertain. Cependant, à cause de la complexité de l'environnement dynamique, ces méthodes ont toujours des limitations concernant leurs hypothèses, leurs efficacités algorithmiques et d'autres inconvénients

1.5 conclusion

Dans ce chapitre, on a présenté quelques notions de base fréquemment utilisé en robotique mobile, avant d'aborder la classification des méthodes de planification de trajectoire en se basant sur la nature de l'environnement. Cette classification contient aussi les grandes lignes concernant quelques travaux et techniques récents dans l'optimisation et la planification de trajectoire pour un robot mobile. Dans le chapitre suivant, on présente les principales notions des algorithmes génétiques et leurs applications dans la planification de trajectoire.

Chapitre 2

Les algorithmes génétiques et ses applications dans la planification de la trajectoire

2.1 Introduction

Dans le but de résoudre des problèmes complexes, des idées glanées à partir de mécanismes naturels ont été exploités pour développer des heuristiques inspirées de la nature. L'optimisation naturelle a été largement investiguée durant la dernière décennie ouvrant la voie à de nouveaux paradigmes de calcul comme les réseaux de neurones, les systèmes immunitaires artificiels ou encore les systèmes évolutionnaires. L'objectif ultime est de développer des systèmes qui ont la capacité d'apprendre de façon incrémentale tout en s'adaptant à leur environnement. La génétique artificielle est un paradigme récent qui tente de capturer des caractéristiques intéressantes des systèmes évolutionnaires naturels, comme le croisement, la mutation, et la sélection. Un algorithme génétique est défini comme un système adaptatif inspiré de la génétique et de l'évolution naturelle pour la résolution de problèmes [30]. Les caractéristiques intéressantes des systèmes évolutionnaires ont encouragé leur adaptation au domaine informatique pour la résolution de problèmes du monde réel allant de la sécurité des réseaux jusqu'à la détection de changements en passant par le web mining [23][30]. En particulier, les études sur les capacités d'adaptation des algorithmes génétiques ont conduit à la création de modèles informatiques pour la navigation autonome des robots mobiles.

Dans ce chapitre nous présentons les algorithmes génétiques en indiquant leurs caractéristiques, puis on introduit ces algorithmes en expliquant leur passage vers l'artificiel. On présente par la suite les grandes lignes, en montrant les différentes composantes de ces algorithmes et les différents mécanismes utilisés par ces systèmes. Enfin, on aborde les applications des algorithmes génétiques dans différents domaines et en particulier dans le domaine de la robotique.

2.2 Origines

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle tels que croisements, mutations, sélection. [23]. Ces algorithmes ont déjà une histoire relativement ancienne puisque les premiers travaux de John Holland [17] sur les systèmes adaptatifs remontent à 1962. Holland a étudié les systèmes évolutifs et, en 1975, il a introduit le premier modèle formel des algorithmes génétiques (the canonical genetic algorithm AGC) dans son livre *Adaptation in Natural and Artificial Systems* [34]. Il expliqua comment ajouter de l'intelligence dans un programme informatique avec les croisements (échangeant le matériel génétique) et la mutation (source de la diversité génétique). Ce modèle servira de base aux recherches ultérieures et sera plus particulièrement repris par Goldberg [9] qui publiera en 1989, un ouvrage de vulgarisation des algorithmes génétiques, et a ajouté à la théorie des algorithmes génétiques les idées suivantes :

- un individu est lié à un environnement par son code d'ADN.
- Une solution est liée à un problème par son indice de qualité.

2.3 Analogie avec la biologie

Puisque les algorithmes génétiques ont une inspiration biologique[6], il convient de rappeler au préalable quelques termes de génétique. Les organismes vivants sont constitués de cellules, dont les noyaux comportent des chromosomes qui sont des chaines d'ADN comme montré à la figure 2.1. L'élément de base de ces chromosomes (le caractère de la chaine d'ADN) est un gène. Sur chacun de ces chromosomes, une suite de gènes constitue une chaine qui code les fonctionnalités de l'organisme (la couleur des yeux, ...). La position d'un gène sur le chromosome est son locus comme il est représenté dans la figure 2.2. L'ensemble des gènes d'un individu est son génotype et l'ensemble du patrimoine génétique d'une espèce est le génome. Les différentes versions d'un même gène sont appelées allèles [1][6][29][34].

On trouve aussi, dans les algorithmes génétiques, une analogie avec la théorie de l'évolu-

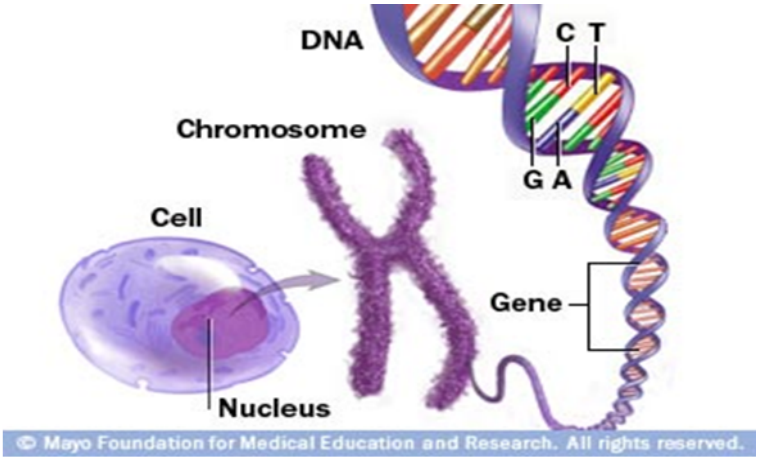


FIGURE 2.1 – Analogie avec la biologie

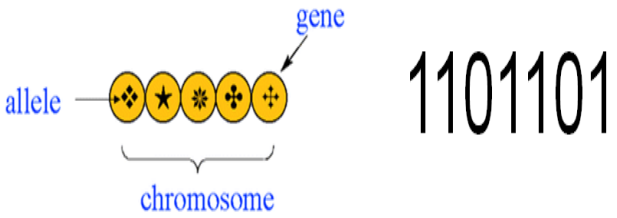


FIGURE 2.2 – représentation d'un chromosome

tion qui propose qu'au fil du temps, les gènes conservés au sein d'une population donnée soient ceux qui sont le plus adaptés aux besoins de l'espèce vis-à-vis de son environnement [6][25][29][30][34].

2.4 Codage des chromosomes

Pour les algorithmes génétiques, un des facteurs importants est la façon dont les solutions sont codées, c'est-à-dire les structures de données qui représentent les gènes [25]. Il y a trois principaux types de codage utilisables, et on peut passer de l'un à l'autre relativement facilement :

2.4.1 Codage binaire

C'est le plus utilisé. Chaque gène dispose du même alphabet binaire 0, 1, les raisons pour lesquelles ce type de codage est le plus utilisé sont tout d'abord historiques. En effet, lors des premiers travaux de Holland, les théories ont été élaborées en se basant sur ce type de codage [25], de plus ce type de codage a pour intérêt de permettre de créer des opérateurs de croisement et de mutation simples [29].

2.4.2 Codage réel

Ce codage peut-être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle (figure 2.3).

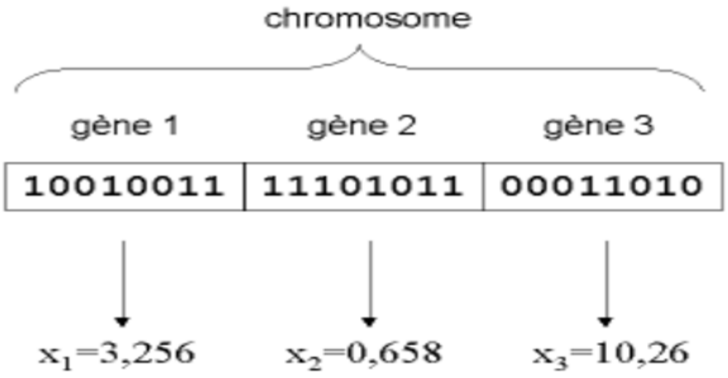


FIGURE 2.3 – codage réel

2.4.3 Codage de Gray

Dans le cas d'un codage binaire, on utilise souvent la "distance de Hamming" comme mesure de la dissimilarité entre deux éléments de population [29], cette mesure compte les différences de bits de même rang de ces deux séquences. Et c'est là que le codage binaire commence à montrer ses limites [30]. En effet, deux éléments voisins en termes de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un "codage de Gray". Le codage de Gray est un codage, qui a comme propriété qu'entre un élément n et un élément $n+1$, donc voisin dans l'espace de recherche, un seul bit diffère [30].

2.5 Principe

Les algorithmes génétiques fournissent des solutions aux problèmes n'ayant pas de solutions calculables en temps raisonnable de façon analytique ou algorithmique [34]. Selon cette méthode, des milliers de solutions (génotypes) plus ou moins bonnes sont créés au hasard [30], puis sont soumises à un procédé d'évaluation de la pertinence de la solution mimant l'évolution des espèces [30]. Les plus "adaptés", c'est-à-dire, les solutions au problème qui sont les plus optimales survivent davantage que celles qui le sont moins, et la population évolue par générations successives en croisant les meilleures solutions entre elles et en les faisant muter, puis en relançant ce procédé, un certain nombre de fois, afin d'essayer de tendre vers la solution optimale.
Le déroulement d'un algorithme génétique peut être découpé en quatre parties [29] :

1. La création de la population initiale,
2. L'évaluation des individus,
3. La création de nouveaux individus,
4. Réitération du processus.

On peut représenter ces étapes dans l'organigramme de la figure 2.4 :

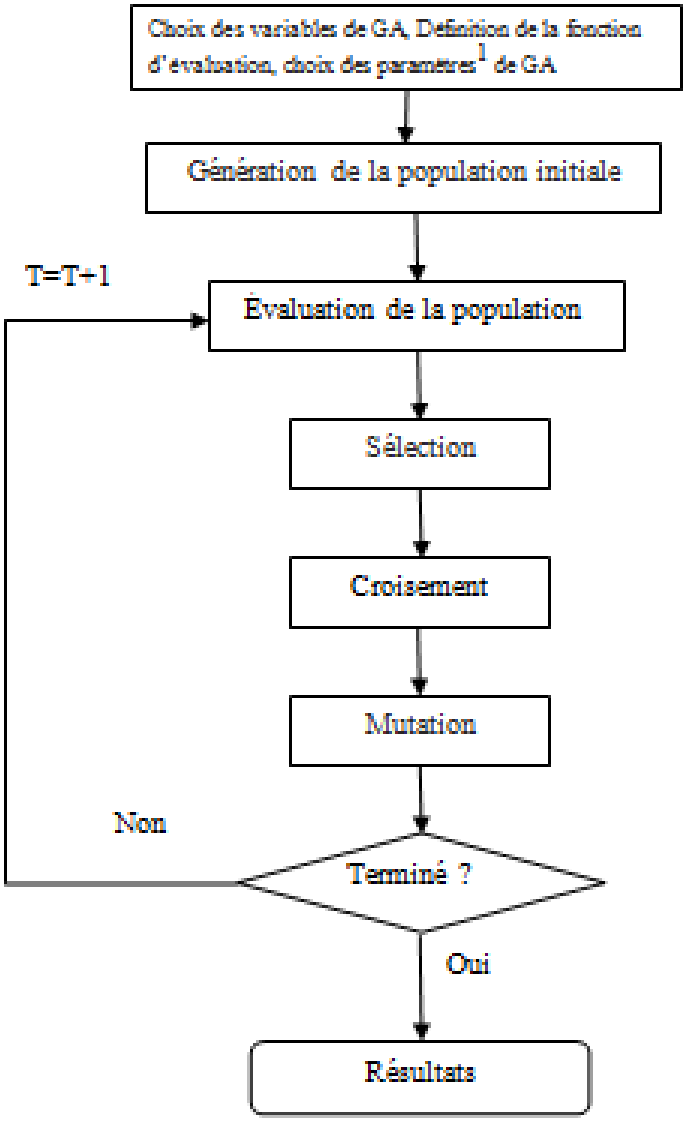


FIGURE 2.4 – organigramme de AGs

2.5.1 La création de la population initiale

Pour démarrer un algorithme génétique, il faut lui fournir une population à faire évoluer [29][30]. La manière dont le programmeur va créer chacun des individus de cette population est entièrement libre [23]. Il suffit que tous les individus créés soient ***de la forme d'une solution potentielle*** [34], et il n'est nullement besoin de songer à créer de bons individus. Ils doivent juste fournir une réponse, même mauvaise, au problème posé.

Et le hasard dans tout cela ?

Il est tout à fait possible de créer les individus de manière aléatoire. Cette méthode amène un concept très utile dans les algorithmes génétiques : la diversité [23][34]. Plus les individus de la population de départ seront différents les uns des autres, plus nous aurons de chance d'y trouver, non pas la solution parfaite, mais de quoi fabriquer les meilleures solutions possibles.

2.5.2 L'évaluation des individus

Une fois que la population initiale a été créée, nous allons en sortir les individus les plus prometteurs, ceux qui vont participer à l'amélioration de notre population [23]. Nous allons donc attribuer une 'note' ou un indice de qualité à chacun de nos individus appelés fitness. La méthode d'évaluation des individus est laissée au programmeur en fonction du problème qu'il a à optimiser ou à résoudre.

Cette étape intermédiaire d'évaluation peut même devenir une étape importante du processus d'amélioration de notre population. En effet, les différents individus ne sont pas toujours comparables, il n'est pas toujours possible de dire qu'un individu est meilleur ou moins bon qu'un autre. C'est le cas des problèmes multicritères, lorsqu'une solution dépend de plusieurs paramètres. On peut toujours dire qu'un nombre est supérieur ou non à un autre, mais on ne peut pas dire si un vecteur est supérieur ou non à un autre. La notion de supériorité pour les vecteurs n'existe pas. On peut comparer leurs normes, mais pas les vecteurs eux-mêmes [1].

2.5.3 La création de nouveaux individus

La sélection

Cet opérateur est très important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir.

On trouve essentiellement trois types de méthodes de sélection différentes :

- La méthode de la roulette
- La sélection par tournois

• La méthode d'élitisme

La roulette « Roulette wheel selection »[23] La sélection des individus par le système de la roulette s'inspire des roues de loterie (figure 2.5), ce principe consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa tness [29][34]. On reproduit ici le principe de tirage aléatoire utilisé dans les roulettes de casinos avec une structure linéaire. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1. On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on "regarde" quel est le segment sélectionné. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent adressés que les petits [29].

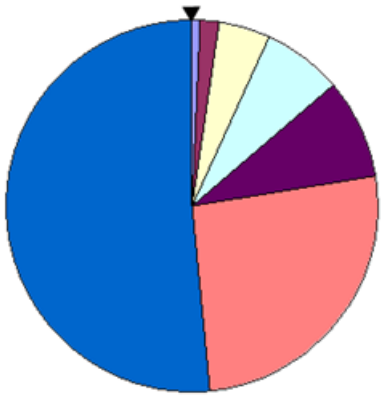


FIGURE 2.5 – schéma de roulette

La sélection par tournoi [30][29][34] Le principe de la sélection par tournoi augmente les chances pour les individus de piètre qualité de participer à l'amélioration de la population. Le principe est très rapide à implémenter. Un tournoi consiste en une rencontre entre plusieurs individus pris au hasard dans la population. Le vainqueur du tournoi est l'individu de meilleure qualité . Comme on peut choisir aussi de conserver les 2 meilleurs individus ou les 3 meilleurs, selon la stratégie de tournois, on peut créer beaucoup de tournois, créer des tournois avec beaucoup de participants ou bien mettre en avant ceux qui gagnent les tournois haut la main. On peut faire participer un même individu à plusieurs tournois. Une fois de plus, on est totalement libre quant à la manière d'implémenter cette technique de sélection.

L'élitisme [6] Cette méthode de sélection permet de mettre en avant les meilleurs individus de la population. Ce sont donc les individus les plus prometteurs qui vont participer à l'amélioration de notre population . Cette méthode a l'avantage de permettre une convergence (plus) rapide des solutions, mais au détriment de la diversité des individus.

On prend en effet le risque d'écarter des individus de piètre qualité, mais qui auraient pu apporter de quoi créer de très bonnes solutions dans les générations suivantes. Cette méthode est extrêmement sensible à la présence des optimums locaux, c'est-à-dire à la présence de solutions paraissant optimales tant que l'on ne s'en éloigne pas trop - le véritable optimum pouvant alors être situé dans une tout autre partie du domaine de recherche.
Une autre possibilité relevant aussi du domaine de l'élitisme, pour s'assurer que les meilleurs individus feront effectivement partie de la prochaine génération, est tout simplement de les sauvegarder pour pouvoir les rajouter à coup sûr dans la population suivante.
Le nombre d'individus à sélectionner en vue d'un croisement ou d'une mutation est encore une fois laissé à l'appréciation de l'utilisateur. Cependant, il n'est pas nécessaire (et pas recommandé non plus) de modifier tous les individus d'une population.

Le croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants [23]. Pour effectuer l'opérateur de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. On échange ensuite les deux sous-chaines terminales de chacun des deux chromosomes, ce qui produit deux enfants C1 et C2 [1][30][29][34](voir figure 2.6).
On peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaines, mais en 3, 4 , etc... [1][10] (Voir figure 2.7).

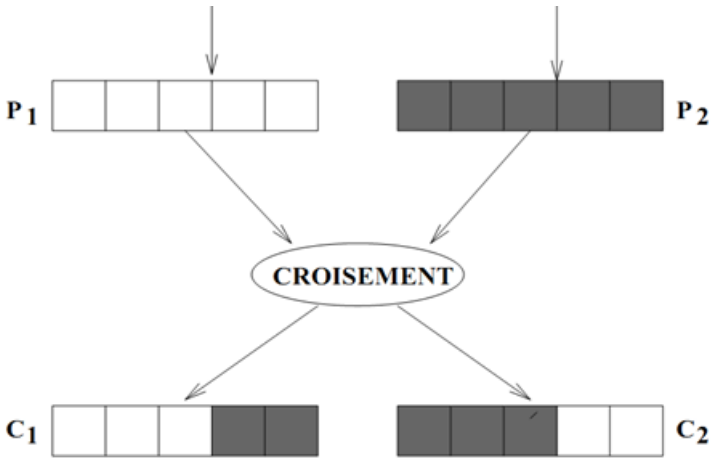


FIGURE 2.6 – croisement simple

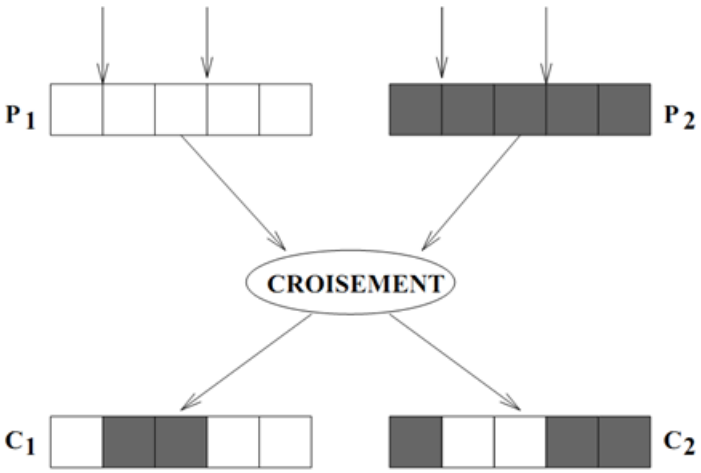


FIGURE 2.7 – croisement multi-points

La mutation

Une autre solution que le croisement pour créer de nouveaux individus est de modifier ceux déjà existants. Une fois de plus, le hasard va être d'une grande utilité. Il peut s'avérer efficace de modifier aléatoirement quelques individus de la population en modifiant un gène ou un autre [29] comme il est montré dans la figure 8. Rien ne dit que l'individu muté sera meilleur ou moins bon, mais il apportera des possibilités supplémentaires qui pourraient bien être utiles pour la création de bonnes solutions [23]. De même que pour les croisements, il n'est pas recommandé de faire muter tous les individus. Il est possible de faire muter un individu de plusieurs manières. Une seule contrainte : l'individu muté doit être de la forme d'une solution potentielle.

Cet opérateur dispose de 4 grands avantages :

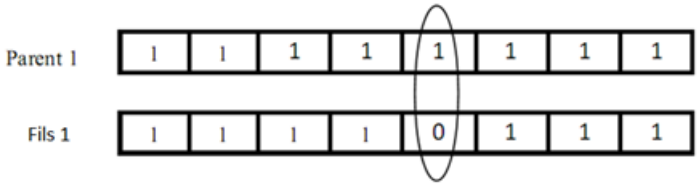


FIGURE 2.8 – mutation

1- Il garantit la diversité de la population, ce qui est primordial pour les algorithmes génétiques [6][30].

2- Il permet d'éviter un phénomène connu sous le nom de dérive génétique. On parle de dérive génétique quand certains gènes favorisés par le hasard se répandent au détriment des autres et sont ainsi présents au même endroit sur tous les chromosomes. Le fait que l'opérateur de mutation puisse entrainer de manière aléatoire des changements au niveau de n'importe quel locus permet d'éviter l'installation de cette situation défavorable [29].

3- Il permet de limiter les risques d'une convergence prématurée [23][34] causée par exemple par une méthode de sélection élitiste imposant à la population une pression sélec-

tive trop forte. En effet, dans le cas d'une convergence prématurée on se retrouve avec une population dont tous les individus sont identiques, mais ne sont que des optimums locaux. Tous les individus étant identiques, le croisement ne changera rien à la situation. En effet, l'échange d'informations par le croisement entre des individus strictement identiques est bien sûr totalement sans conséquence ; on aura beau choisir la méthode de croisement qu'on veut, on se retrouvera toujours à échanger des portions de chromosomes identiques et la population n'évoluera pas. L'évolution se retrouvant bloquée, on n'atteindra jamais l'optimum global.
La mutation entrainant des inversions de bits de manière aléatoire permet de réintroduire des différences entre les individus [6][34].
4- L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace [34]. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implantations fonctionnent de cette manière. Grâce à cette propriété, on est donc sûr de pouvoir atteindre l'optimum global [5][34].

Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur sur le plan théorique.

2.5.4 Réitération du processus

Dans cette étape, si un critère d'arrêt est atteint, l'algorithme s'arrête et là, les meilleurs individus sont sélectionnés. Dans le cas contraire, on recommence le processus depuis l'étape de l'évaluation[20]. Plusieurs procédures peuvent être employées pour terminer l'exécution d'un AG et sont étudiées dans [29].

2.6 Applications

Il y a plusieurs applications des algorithmes génétiques où un grand nombre de problèmes du monde réel ont été résolus par ces algorithmes. En général, les algorithmes génétiques peuvent être appliqués pratiquement à n'importe quel problème qui a un grand espace de recherche, comme le puzzle technique. On peut les utiliser aussi comme une base pour l'étude de la théorie d'apprentissage '*machine Learning*' [34] ou en tant que modèle informatique d'innovation et de créativité. Parmi les domaines d'applications des algorithmes génétiques, on trouve :
- ***Les systèmes automatisés*** [6] : évasion de missile, poursuite [34].

- ***La Médecine*** : (par exemple dans la détection du cancer du sein).
- ***L'optimisation combinatoire*** : Problème de voyageur de commerce [34].
- ***Les domaines de l'ingénierie*** : (comme l'électronique dans la conception des circuits [6], mécanique, civil, production, aéronautique et robotique).
- ***Les topologies de distribution de réseau informatique*** [34].
- ***La recherche Internet.***
- ***La robotique*** : planification de trajectoire de robot et navigation. Par exemple, la société SONY a utilisé les Algorithmes Génétiques dans son robot Aïbo. En effet, ce robot a « appris » à marcher via un dispositif expérimental où son système de commande a été soumis à une évolution artificielle. Différents modes de commandes ont été testés, les plus performants ont été croisés et le résultat a été très positif. De génération en génération, le robot s'est redressé, puis a commencé à marcher en chutant souvent pour finir par marcher d'un pas assuré [25].

2.7 Avantage des AGs

1. les algorithmes génétiques sont un outil très puissant pour l'optimisation [6]. En plus du fait de s'inspirer de l'évolution des espèces et d'avoir une « légitimité» biologique, ils disposent d'une base théorique solide qui permet leur mise en application dans divers domaines [34].
2. Ils ont l'énorme avantage de pouvoir être appliqués dans un grand nombre de domaines de recherche de solution, lorsqu'il n'est pas nécessaire d'avoir la solution optimale, qui prendrait par exemple trop de temps et de ressources pour être calculée ou tout simplement s'il n' y a pas une procédure capable de la trouver de manière théorique [5][34].
3. Il existe de très nombreuses variantes d'algorithmes génétiques, et cela à cause de la disponibilité de plusieurs options à chaque étape (plusieurs modes de codage, plusieurs possibilités de mutation, plusieurs méthodes de croisement, etc.) [29].
4. Ils sont aussi particulièrement adaptés à l'évaluation de problèmes multicritères et à la réalisation de recherches de valeurs optimales selon plusieurs objectifs simultanés [34].
5. Il y a toujours une réponse et la réponse devient meilleure avec le temps.
6. Il sont facilement modifiés pour différents problèmes [34].
7. L'optimum global est facile à découvrir [34].
8. Ils manipulent les grands problèmes de recherche et les mal-compris facilement [34].

2.8 Limites

Comme toute méta-heuristique, les algorithmes génétiques ont quelques limites telles que :

1. Le temps de calcul : les algorithmes génétiques sont plus lents que d'autres méta-heuristiques ,[6], ils nécessitent de nombreux calculs, en particulier au niveau de la fonction d'évaluation [25].
2. Ils sont le plus souvent difficiles à mettre en œuvre : des paramètres comme la taille de la population ou le taux de mutation sont parfois difficiles à déterminer. Or le succès de l'évolution en dépend et plusieurs essais sont donc nécessaires, ce qui limite encore l'efficacité de l'algorithme. En outre, choisir une bonne fonction d'évaluation est aussi critique. Celle-ci doit prendre en compte les bons paramètres du problème. Elle doit donc être choisie avec soin [25].
3. On ne peut pas assurer que la solution trouvée soit la meilleure, même après un nombre important de générations. Mais on peut juste être sûr que l'on s'est approché de la solution optimale (pour les paramètres et la fonction d'évaluation choisie), sans la certitude de l'avoir atteinte [25][34].
4. Un autre problème important est celui des optimums locaux. En effet, lorsqu'une population évolue, il se peut que certains individus qui à un instant occupent une place importante au sein de cette population deviennent majoritaires. À ce moment, il se peut que la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants, mais trop éloignés de l'individu vers lequel on converge. Pour vaincre ce problème, il existe différentes méthodes comme l'ajout de quelques individus générés aléatoirement à chaque génération, des méthodes de sélection différentes de la méthode classique [25].

2.9 Application des algorithmes génétiques dans la planification de la trajectoire des robots mobiles

Les robots imitent le mouvement biologique, et les algorithmes génétiques imitent l'évolution biologique. Les deux thèmes semblent être parfaitement liés, et beaucoup de chercheurs ont établi cette liaison. Plusieurs approches ont étudié l'utilisation des algorithmes génétiques pour la planification de trajectoire pour un robot mobile.

Davidor [43] a développé en (1991) un algorithme génétique spécifique pour l'optimisation de la trajectoire du robot. Les caractéristiques principales de son algorithme est

l'utilisation des structures dynamiques pour les chromosomes et l'utilisation d'un opérateur de croisement modifié appelé "analogous crossover". Chaque chromosome dans la population représente une trajectoire différente pour le robot. Le but de l'algorithme proposé est de réduire au minimum le décalage entre le chemin réel et désiré.

Shibata et Fukuda (1993) ont proposé une combinaison des algorithmes génétiques avec la logique foule pour déterminer un chemin optimal pour un robot mobile qui se déplace et prend des charges sur son chemin [38].

Hoi-Shan Lin et ses collaborateurs [19] en (1994) ont présenté un algorithme évolutionnaire pour le problème de planification de chemin des robots mobiles, dans un environnement qui peut contenir des obstacles qui ont des formes et des endroits inconnus. Leur algorithme détermine d'abord un chemin global, à partir du point de départ au point de destination, puis il exécute ensuite un processus de réparation de ce chemin tout en évitant des collisions possibles avec des obstacles connus ou inconnus.

Cai et Peng [48] ont développé en (1997) un mécanisme de codage décimal pour des chromosomes de longueur constante. Les sommets des obstacles sont aléatoirement numérotés, et la longueur de chaque chromosome est fixée pour être la somme de tous les sommets de l'environnement. Quand la valeur du gène est différente de zéro ceci correspond au nombre réel du sommet choisi, tandis que lorsqu'elle est égale à zéro ce sommet n'est pas choisi. L'ordre trouvé de ces gènes ayant des valeurs différentes de zéro est alors simplement celui des nœuds intermédiaires du chemin produit. Nearchou en (1998) [3] a utilisé le nombre de sommets produit par le graphe de visibilité pour construire des chromosomes de longueur fixe. Si un sommet du graphe de visibilité est choisi, on note dans un bit correspondant à ce sommet le nombre "1", et dans le cas contraire "0". Un nouvel opérateur a été appliqué pour augmenter la performance de l'approche. L'algorithme était capable de déterminer une solution proche-optimale.

Le Planificateur/Navigateur évolutionnaire (EP/n) [19, 37, 40, 41] [46] incorpore les connaissances heuristiques de l'environnement dans les opérateurs de l'AG développé par TAILOR appelé "tailored GA" pour produire des solutions faisables de haute qualité à partir d'une grande population de chemins dans un environnement connu.

En 2007, Wang et Sillitoe [47] ont proposé un planificateur avec les algorithmes géné-

tiques. Le planificateur est capable de déterminer rapidement la solution optimale ou la proche-optimale dans un environnement dynamique. La méthode utilise les sommets des obstacles comme un espace de recherche pour créer une trajectoire à travers les obstacles mobiles.

Premièrement, elle intègre la vitesse du robot dans les gènes du chromosome, qui optimise le temps du déplacement et la longueur de la trajectoire. Avant que le robot commence le mouvement, toutes les informations concernant les obstacles mobiles sont disponibles pour lui. Le robot utilise les algorithmes génétiques pour calculer la solution optimale (en termes de temps ou de distance) puis commence à se déplacer.

Les hypothèses de cette méthode sont :

• les obstacles sont entourés par des polygones avec des sommets.

• la vitesse des obstacles est fixe.

• les dimensions physiques du robot sont négligés (le robot est représenté par un point matériel)

La solution est représentée par un chromosome (figure 2.9), qui est constitué par une série

Start gene	gene1		gene L-2	goal gene
x_0	x_1	....	x_{l-2}	x_g
y_0	y_1		y_{l-2}	y_g
s_0	s_1		s_{l-2}	s_g
0/1	0/1		0/1	0/1

FIGURE 2.9 – Représentation de d'un chromosome [13]

des gènes. Le nombre minimum de gènes est deux (point de départ et point d'arrivée), le nombre maximum des gènes est N+2 (nombre de sommets plus point de départ et point d'arrivée) Chaque gène contient quatre cases comme il est représenté dans la figure 9, les deux premières étant les coordonnées x et y du sommet, la troisième case contenant la vitesse du robot dans le segment de chaque gène. La vitesse du robot est choisie parmi un ensemble de vitesses discrètes prédéfinies. La dernière case indique la faisabilité du segment ; si le segment qui connecte deux sommets consécutifs coupe un obstacle, la case de faisabilité prend la valeur 1 pour indiquer l'infaisabilité du segment, et elle prend 0 dans le cas contraire.

La fonction d'évaluation contient deux valeurs : la longueur de la trajectoire et le temps de voyage.

Quatre opérateurs ont été utilisés dans cette méthode : ce sont l'opérateur du croisement, l'operateur de mutation, l'operateur de réparation et la mutation de vitesse du robot, la figure 10 montre les trois premiers opérateurs.

La procédure commence par la création d'une population initiale puis leur évaluation. L'opération de génération commence par la sélection aléatoire d'un opérateur génétique. La sélection des parents (pour l'operateur de croisement) sera faite en utilisant la roulette (roulette Wheel sélection). Pour former une nouvelle génération, les nouveaux fils

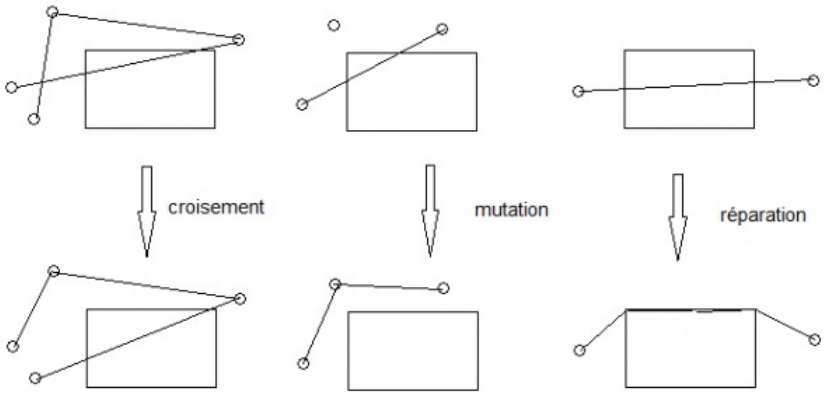


FIGURE 2.10 – Les opérateurs des algorithmes génétiques [28]

remplacent les individus qui ont des mauvaises valeurs dans la fonction d'évaluation. Ce processus continue jusqu'à la satisfaction d'une condition d'arrêt, qui peut être définie par le nombre de générations ou une valeur prédéfinie d'un critère. Lorsque l'évolution est terminée, le meilleur chromosome représente la trajectoire à suivre (optimale ou proche-optimale). Cette méthode est très efficace surtout pour les environnements statiques.
Pour les environnements dynamiques, dans la littérature, on trouve peu de contributions qui traitent la navigation du robot en utilisant les algorithmes génétiques ou qui incorporent la vitesse du robot en tant qu'élément du problème de planification. Une version modifiée d'EP/n, nommée EP/n++, a été proposé dans [32][21][33] comme un système interactif d'aide à la décision pour un bateau qui voyage sans collision sur la base de l'information obtenue à partir d'un radar. Les variables principales d'EP/n++ sont les paramètres du bateau, sa vitesse variable, et la contrainte du temps des autres bateaux mobiles. Le système d'EP/n++ maintient la même structure d'EP/n et a le même ensemble d'opérateurs génétiques, excepté un nouvel opérateur additionnel développé pour agir sur la vitesse du bateau. Les solutions sont obtenues à partir des individus de longueur variable, qui représentent des chemins possibles pour le bateau. La planification en ligne peut être activée en réponse aux changements du mouvement d'autres bateaux.

2.10 Conclusion

Introduits en 1975 par John Holland, les algorithmes génétiques sont relativement récents et de ce fait, les fondements théoriques sont encore en cours de développements [25]. Cependant, les algorithmes génétiques présentent un fort potentiel d'applications pratiques. D'ailleurs, ils sont de plus en plus utilisés, et ce, dans de multiples domaines. Il faut dire qu'ils fournissent d'excellentes performances à de faibles coûts. En effet, les algorithmes génétiques permettent d'explorer des domaines possédant de très nombreuses solutions. Pour une population d'origine de N individus, un algorithme génétique per-

met d'explorer N^3 solutions possibles (*soit un milliard de possibilités pour une population d'origine d'un millier d'individus*). Malheureusement, quelques inconvénients viennent ternir ce tableau. En effet, l'utilisation de ces algorithmes est souvent coûteuse en temps de calcul (les algorithmes sont lents).

Mais malgré tout ça, les algorithmes génétiques restent parmi les techniques intéressantes à utiliser lorsque toutes autres méthodes échouent ou quand nous ne connaissons absolument rien de l'espace de recherche. Néanmoins, même dans le cas où des techniques spécialisées existent, il est souvent intéressant de les hybrider avec les algorithmes génétiques afin de gagner probablement quelques améliorations. Il est important toujours de garder un point de vue objectif et ne pas considérer que les algorithmes génétiques soient des remèdes pour résoudre tous les problèmes d'optimisation. Les algorithmes génétiques fonctionnent et donnent d'excellents résultats s'ils sont appliqués correctement sur des problèmes appropriés.

le but de ce chapitre est de fournir une vue abstraite sur les propriétés fondamentales et les principaux applications des algorithmes génétiques.

Dans le chapitre suivant, on va présenter notre approche de planification de trajectoire pour un robot mobile dans un environnement dynamique, qui est basé essentiellement sur les algorithmes génétiques.

Chapitre 3

Planification de trajectoire à base des algorithmes génétiques et du concept de temps d'attente

3.1 introduction

Dans le monde réel, un robot mobile peut rencontrer des obstacles de n'importe quelle nature tels que des obstacles statiques, mobiles (dynamiques) ou des environnements partiellement dynamiques (un environnement comportant des obstacles statiques et dynamiques) [27]. Il y a beaucoup d'algorithmes pour résoudre les problèmes de planification de chemin tout en évitant les obstacles, néanmoins la plupart des algorithmes sont applicables à des formes spécifiques d'obstacles ou appropriés aux environnements statiques seulement.

Ce chapitre présente une méthode de planification de trajectoire pour un robot mobile, basée sur les algorithmes génétiques et le concept de temps d'attente dans un environnement encombré d'obstacles qui ont des formes, des tailles et des endroits arbitraires. L'algorithme proposé est applicable à des environnements statiques, partiellement dynamiques aussi bien que les environnements dynamiques.

L'approche est divisée en deux parties, statique et dynamique. Dans la partie statique, elle utilise les sommets des obstacles stationnaires comme un espace de recherche. En utilisant les algorithmes génétiques, elle trouve le chemin optimal pour le robot. Quand un obstacle mobile est détecté avec la possibilité de collision avec le robot, la partie dynamique est activée, dans cette partie, l'approche applique le concept de temps d'attente (CTA) pour éviter cet obstacle.
Dans ce chapitre, on explique les deux parties statique et dynamique de l'approche utilisée avec tous les détails, y compris la modélisation de l'environnement, la structure de l'approche/algorithme, la génération de la population initiale, la recherche du chemin optimal et le calcul de la position et du temps d'attente pour les obstacles dynamiques.

3.2 hypothèses de la méthode

La complexité du problème de planification de mouvement est grande [27], ainsi une variété de simplifications ont été suggérées pour réduire cette complexité [54]. Le but recherché ici est de maitriser cette complexité et de permettre la modélisation d'un environnement dynamique dans un espace de dimension réduite où s'applique les outils classiques de la géométrie algorithmiques [11].
Dans notre approche, les hypothèses suivantes sont faites :

1. Le robot se déplace sur un terrain plat et l'environnement est global (la position et la vitesse des obstacles sont connues).
2. Il existe des obstacles stationnaires et mobiles dans l'environnement dynamique.

3. Chaque obstacle est enfermé par un rectangle ou un carré puis amplifié par une valeur déterminée, à partir de la distance minimum avec laquelle le robot peut approcher les obstacles sans collision, tenant compte de ses dimensions physiques [27]. Comme les obstacles sont enfermés dans des frontières rectangulaires ou carrées, les obstacles de n'importe quelle forme comme les obstacles de corps convexe, concave et courbé peuvent être traités par l'algorithme proposé.
4. l'espace de recherche est les sommets des obstacles statiques.
5. la trajectoire de mouvement des obstacles dynamiques est constituée par une série de segments des lignes droites.
6. Les paramètres de mouvement des obstacles dynamiques, tels que la vitesse et la direction, peuvent être détectés par le robot si l'obstacle est dans l'intervalle de perception du robot.
7. le robot peut changer sa vitesse et sa direction à tout moment.

3.3 architecture globale

Notre algorithme procède en deux phases : phase statique en se basant sur les obstacles statiques, et phase dynamique quand des obstacles mobiles sont détectés (figure 3.1).

Étape 1 : dans la phase statique, le Calcul est basé sur les sommets des obstacles stationnaires. L'approche utilise les algorithmes génétiques pour trouver le chemin optimal en créant une population initiale des chemins faisables et infaisables. Puis, elle trouve le chemin optimal pour le robot. Une fois que le chemin est prêt, le robot commence à se déplacer entre les obstacles stationnaires avec des capteurs qui peuvent détecter le moindre changement au voisinage du robot.

Étape 2 : la phase dynamique est appliquée une fois que les obstacles mobiles sont détectés par les capteurs.
Quand l'obstacle mobile entre dans l'intervalle de détection du robot, les capteurs le détecteront et le robot peut acquérir les informations de cet obstacle mobile tel que la vitesse et la direction. Puis, le robot utilise ces informations pour calculer la possibilité de collision avec cet obstacle mobile. S'il y a collision comme montré à la figure 3.2, le robot fait appel à l'algorithme du CTA pour éviter la collision avec cet obstacle. L'algorithme calcule une position d'attente et le temps d'attente nécessaire où le robot peut attendre en toute sécurité, comme montré à la figure 3.3.
Dans le cas où il n'y a pas de collision entre le robot et l'obstacle, le robot utilise le chemin obtenu par les algorithmes génétiques pour atteindre son but. Comme la figure 3.4 montre, le robot calcule à partir des informations perçues par les capteurs que l'obstacle en « T »

ne le heurtera pas. Par conséquent, le robot ne fait pas appel à l'algorithme CTA.

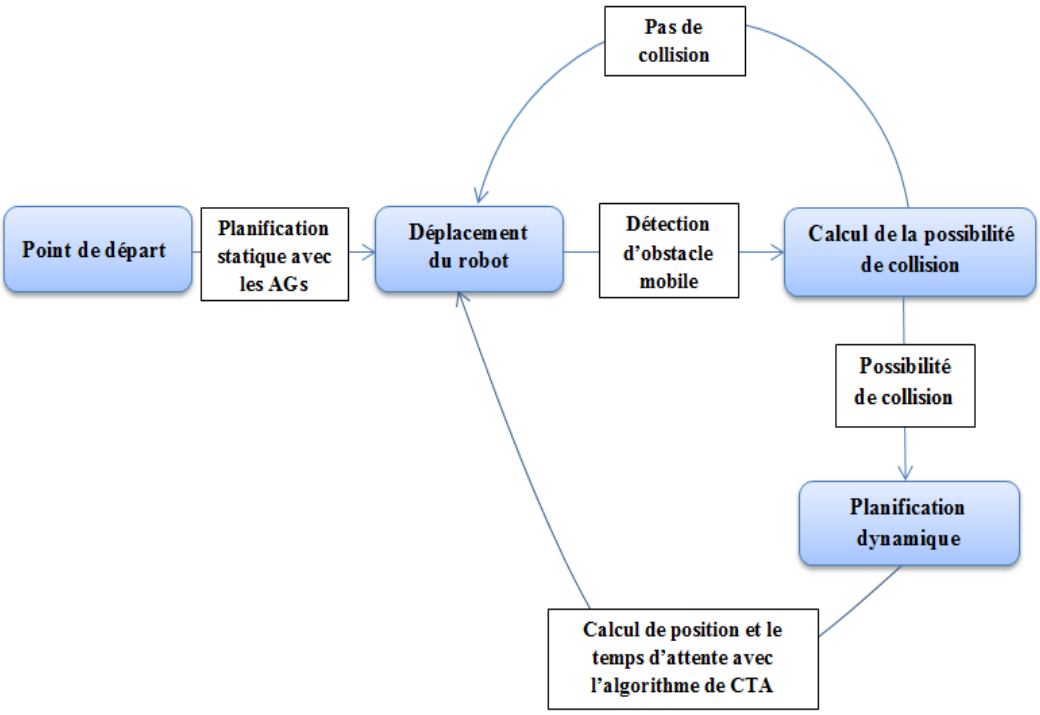


FIGURE 3.1 – Diagramme de transformation d'état de notre approche [13]

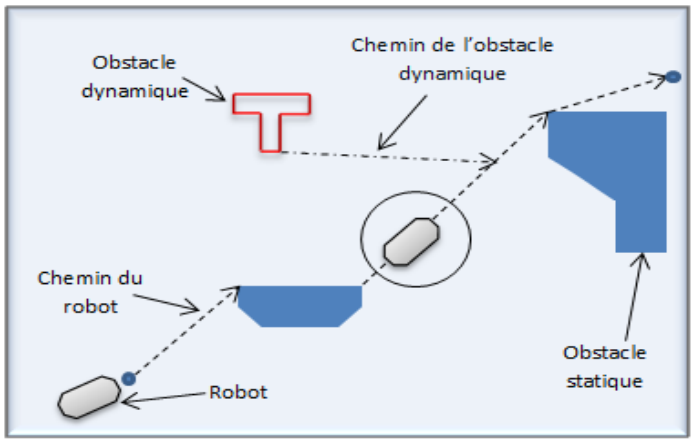


FIGURE 3.2 – scénario 1 de l'obstacle mobile

3.4 modélisation mathématique

3.4.1 Modélisation de l'environnement

L'environnement est représenté par des polygones avec des sommets et des segments. Le modèle mathématique de la localisation des segments et des sommets des obstacles dans l'environnement n'est pas compliqué, il utilise le modèle mentionné au chapitre 1 section

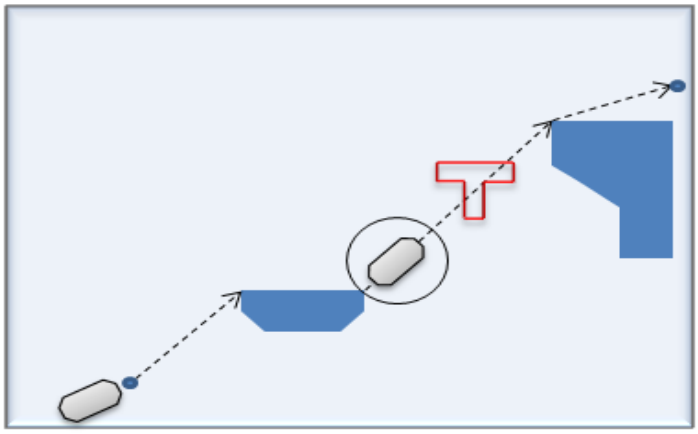


FIGURE 3.3 – la planification dynamique

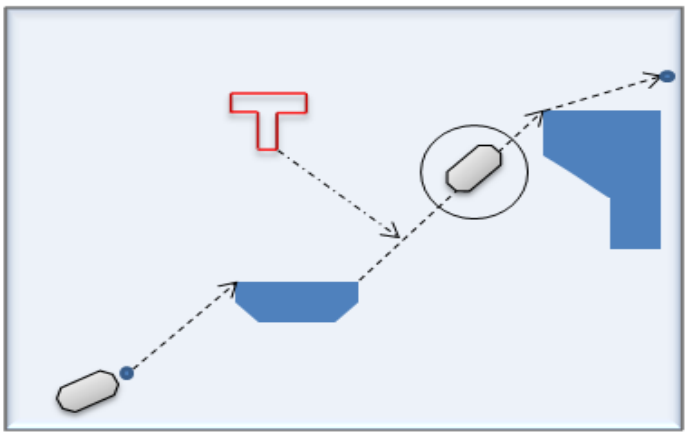


FIGURE 3.4 – scénario 2 de l'obstacle mobile

(4.1.1). Des modélisations des environnements réels sont données dans [47][27]. Tous les obstacles dans la carte sont enfermés par des rectangles ou des carrés puis amplifiés par une valeur déterminée, et le robot peut approcher les obstacles sans collision, par conséquent la dimension du robot est négligée, et il est considéré comme un seul point matériel [27] [13][45][46]. Dans la figure 3.5, les polygones noirs représentent les obstacles statiques et les triangles rouges sont les obstacles mobiles. Les sommets des polygones statiques agrandis forment l'espace de recherche du robot.

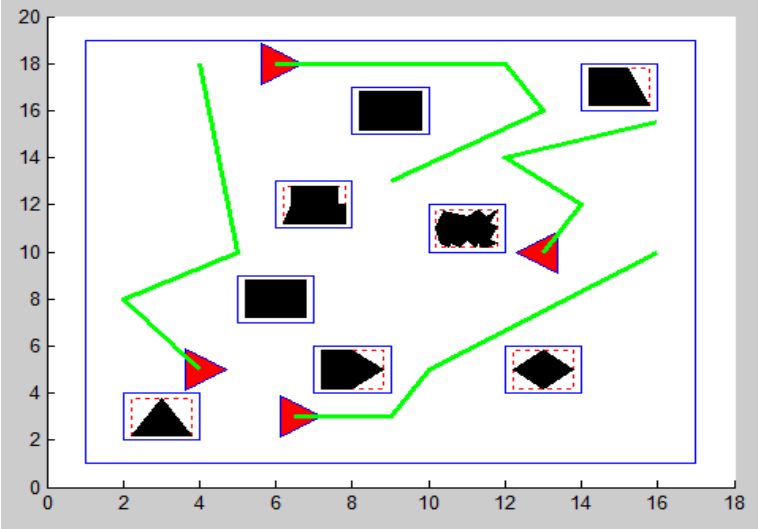


FIGURE 3.5 – Modélisation de l'environnement

Création des deux matrices d'adjacence 'A' et 'B'

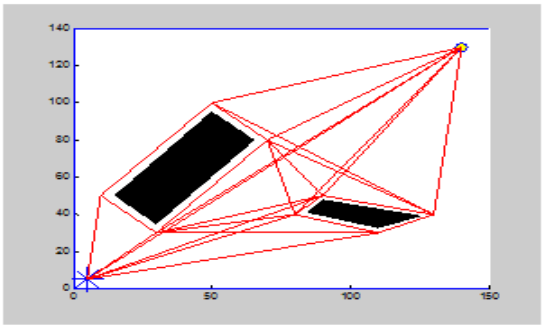
matrice A

C'est une matrice carrée, d'ordre N où N est le nombre total de sommets des obstacles de l'environnement. Les lignes et les colonnes de cette matrice sont les sommets des obstacles. Chaque élément (i,j) de cette matrice définit la longueur d'un segment direct du sommet « i » au sommet « j » (s'il n'y a pas une intersection entre le segment [i j] et un obstacle). Dans le cas contraire, l'élément (i, j) doit contenir '-1'.

matrice B

Cette matrice a les mêmes caractéristiques que la matrice A, sauf que dans cette matrice on met le nombre d'obstacles intersectés si le segment [i, j] est infaisable, sinon on met '0'.

Dans la figure 3.6, la partie (a) c'est la représentation d'un environnement qui contient deux obstacles et les deux figures (b) et (c) représentent les deux matrices d'adjacence B et A respectivement. On peut remarquer que cet environnement contient huit sommets avec les deux points de départ et d'arrivée, donc le nombre de lignes de ces deux matrices sera égal à 10.



(a) Représentation de tous les chemins possibles dans l'environnement

B =

0	0	1	0	1	1	2	1	0	1
0	0	0	1	1	1	0	2	1	0
1	0	0	0	0	0	0	1	1	0
0	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0
2	0	0	1	1	0	0	0	1	0
1	2	1	0	0	1	0	0	0	1
0	1	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0

(b) Matrice B

A =

-1.0000	64.0312	-1.0000	28.2843	-1.0000	-1.0000	-1.0000	-1.0000	45.2769	-1.0000
64.0312	-1.0000	28.2843	-1.0000	-1.0000	-1.0000	100.0000	-1.0000	-1.0000	94.8683
-1.0000	28.2843	-1.0000	64.0312	41.2311	36.0555	72.1110	-1.0000	-1.0000	86.0233
28.2843	-1.0000	64.0312	-1.0000	50.9902	63.2456	-1.0000	80.0000	35.3553	148.6607
-1.0000	-1.0000	41.2311	50.9902	-1.0000	14.1421	-1.0000	31.6228	82.7647	108.1665
-1.0000	-1.0000	36.0555	63.2456	14.1421	-1.0000	41.2311	-1.0000	96.1769	94.3398
-1.0000	100.0000	72.1110	-1.0000	-1.0000	41.2311	-1.0000	22.3607	-1.0000	90.5539
-1.0000	-1.0000	-1.0000	80.0000	31.6228	-1.0000	22.3607	-1.0000	107.9352	-1.0000
45.2769	-1.0000	-1.0000	35.3553	82.7647	96.1769	-1.0000	107.9352	-1.0000	183.9837
-1.0000	94.8683	86.0233	148.6607	108.1665	94.3398	90.5539	-1.0000	183.9837	-1.0000

(C) Matrice A

FIGURE 3.6 – Modélisation de l'environnement

3.4.2 le modèle mathématique de l'algorithme du CTA

Le modèle mathématique de l'algorithme du concept de temps d'attente est décrit comme suit [27] :

1. le premier point de croisement entre le chemin du robot proposé par les algorithmes génétiques et la trajectoire d'un obstacle mobile est calculé avant d'examiner la possibilité de collision.

2. en se basant sur le temps "t" exigé au robot pour parcourir la distance de sa position actuelle au premier point d'intersection, l'endroit instantané de l'obstacle mobile peut être calculé, et par conséquent, un intervalle de sécurité pour cet obstacle peut être obtenu.
   en se basant sur le temps « t » exigé au robot pour parcourir la distance de sa position actuelle au premier point d'intersection, ce temps « t » peut être calculé par l'équation suivante :

$$t = d/v \tag{3.1}$$

   Où « v » est la vitesse du robot et « d » représente la distance entre le point d'intersection et sa position actuelle, qui peut être calculée en connaissant les coordonnées de ces deux points par cette équation :

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.2}$$

Avec (x_1, y_1) représente les coordonnées de la position actuelle du robot et (x_2, y_2) représente les coordonnées du point d'intersection. À cet instant « t », l'endroit instantané de l'obstacle mobile peut être calculé, et par conséquent, un intervalle de sécurité pour cet obstacle peut-être obtenu, cet intervalle est calculé en se basant sur les dimensions de cet obstacle et sa vitesse.

3. si le temps "t" appartient à cet intervalle, alors une collision se produira entre le robot et l'obstacle mobile, dans le cas contraire aucune collision ne se produira entre les deux.

3.5 Planification statique

3.5.1 La représentation génétique

Chaque chemin est représenté par un chromosome avec un certain nombre de gènes intermédiaires (sommets) égale à « l », où « l » a une valeur minimale de deux (correspondant à un chemin contenant seulement les points de départ et de fin) et une valeur maximale de N+2, où N représente le nombre de sommets de tous les obstacles statiques dans l'environnement. La figure 3.7 présente la structure d'un chromosome où le premier gène correspond au point de départ (numéro 1) et le dernier gène au point d'arrivée (numéroté N+2). Les gènes intermédiaires l-2 représentent les sommets des obstacles de l'environnement.

FIGURE 3.7 – structure d'un chromosome

3.5.2 structure de l'algorithme

L'organigramme est représenté sur les figures 3.8 et le pseudo-code de l'algorithme de planification statique est montré ci-dessous, tel que :
Pgc, Pgm, Pgr,Pgi : sont les probabilités d'application des opérateurs de croisement, mutation, réparation et inversion respectivement dans chaque génération.
Ppc, Ppm, Ppr, Ppi : sont les probabilités d'application des opérateurs de croisement, mutation, réparation et inversion respectivement dans chaque population.

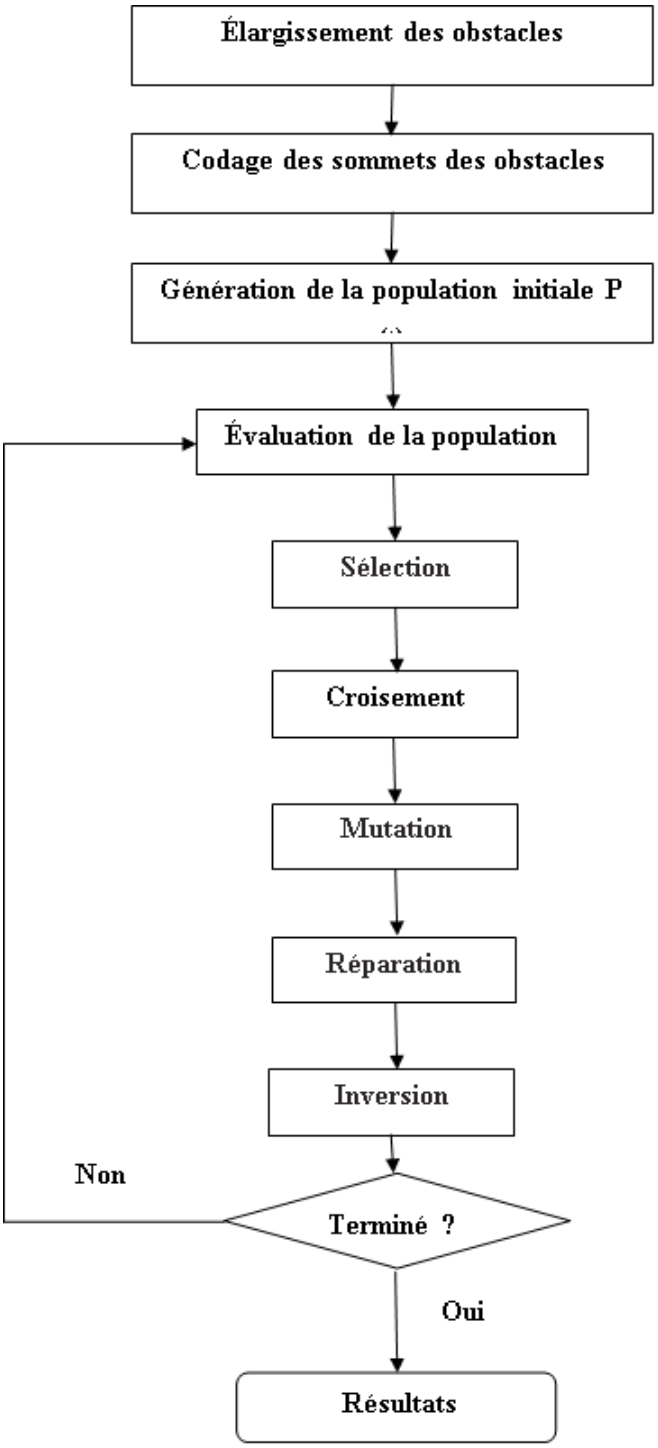


FIGURE 3.8 – l'organigramme de la planification statique

Procédure de planification avec GAs
Début
t $\leftarrow$ 0
Élargissement des obstacles
Codage de sommets des obstacles
Génération de la population initiale P (t) de N individus
Évaluation de P(t)
o Générer la probabilité d'application de chaque opérateur
(Pgc, Pgm, Pgr,Pgi) dans chaque génération.
o Générer la probabilité d'application de chaque opérateur
(Ppc, Ppm, Ppr, Ppi) dans chaque population.
Tant que (la condition d'arrêt n'est pas atteinte) faire
t $\leftarrow t + 1$
o Sélectionner K= Ppc* N individus à partir de P(t) pour le croisement
• croisement
o Sélectionner M= Ppm* N individus à partir de P(t) pour la mutation
• mutation
o Sélectionner L= Ppr* N individus à partir de P(t) pour la réparation
• réparation
o Sélectionner Q= Ppi* N individus à partir de P(t) pour l'inversion
• inversion
Évaluation de P(t)
Sélectionner les meilleurs N individus parmi la population
de N+ K individus (parents et enfants)
Fin de Tant que
Choisir le meilleur individu parmi les meilleurs de chaque génération
Fin.

-les étapes de l'exécution de l'algorithme sont décrites ci-dessous :

Élargissement des obstacles

L'environnement de navigation du robot se compose d'un ensemble d'obstacles stationnaires dont la forme est irrégulière. Ces obstacles peuvent être modélisés en les enfermant dans des rectangles ou des cercles, ce type d'approximation est standard dans la littérature d'évitement d'obstacles [56]. Mathématiquement, les enveloppes circulaires peuvent être représentées par des inégalités du second degré tandis que des enveloppes polygonales peuvent être décrites par des inégalités linéaires de premier ordre [57]. Dans notre

approche, nous avons utilisé la modélisation rectangulaire où chaque obstacle est enfermé par un rectangle ou un carré. Il est ensuite amplifié par une valeur déterminée, à partir de la distance minimale avec laquelle le robot peut approcher les obstacles sans collision, tenant compte de ses dimensions physiques, comme représenté dans la figure 3.9.

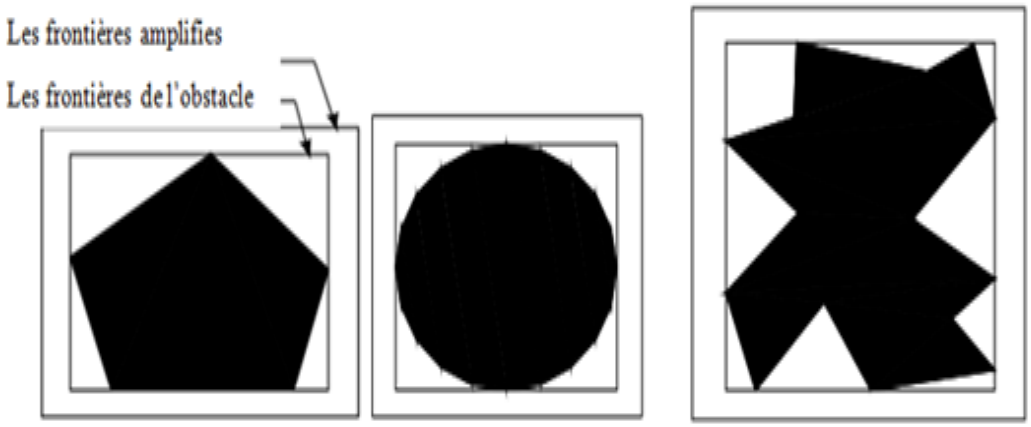


FIGURE 3.9 – élargissement des obstacles [27]

codage des sommets des obstacles

Les sommets des obstacles agrandis sont numérotés aléatoirement comme il est représenté dans la figure 3.10. Ces sommets sont utilisés en tant que gènes potentiels pour un chromosome dont le phénotype présente un chemin entre deux endroits donnés.

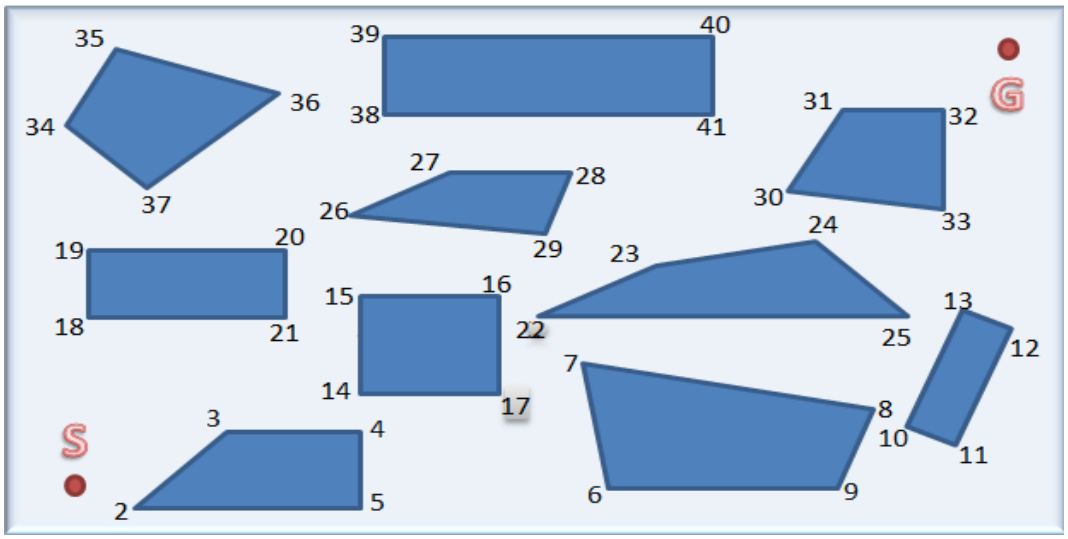


FIGURE 3.10 – numérotation des sommets

Génération de la population initiale

Le point clé dans la manipulation des algorithmes génétiques pour trouver des solutions à un problème donné est la syntaxe appropriée des chromosomes dans la population ainsi que leur interprétation. Dans notre approche, les chromosomes choisis dans la population sont des chromosomes de longueur variable où chaque chromosome représente un ensemble de sommets choisis à partir du graphe. Chaque gène dans le chromosome correspond à un sommet spécifique dans le graphe, de tel sorte que le premier gène dans le chromosome correspond au point de départ, alors que le dernier correspond au point de destination. L'ensemble des sommets choisis représente un chemin potentiel pour le robot entre le début et le sommet de destination, où chaque sommet peut appartenir une fois, au plus, à ce chemin. On s'intéresse aux chemins faisables et infaisables afin de diversifier la population initiale.

Par exemple, pour un graphe de 'n' sommets, le point de départ aura l'étiquette 1, le point de destination l'étiquette n+2 et les autres sommets intermédiaires seront numérotés dans l'intervalle [2... n+1]. Dans la population initiale, les chromosomes sont produits comme suit :

- (i) Choisir aléatoirement un nombre entier k dans l'intervalle [2... n+1], où k représente la taille du chromosome.
- (ii) Choisir aléatoirement K nombres entiers dans l'intervalle [2... n+1], ces nombres entiers correspondent aux gènes du chromosome. La duplication n'est pas autorisée.
- (iii) Pour assurer qu'il n'y a pas des sommets répétés dans le même chromosome, on procède de la manière suivante : pour chaque sommet choisi, on met dans un tableau de 'n' cases, rempli initialement par des zéros, le nombre '1' à la position qui correspond au sommet choisi pour indiquer que ce sommet a déjà été choisi dans ce chromosome. Lors du remplissage du chromosome, on choisit un sommet parmi l'ensemble qui a des étiquettes égales à « 0 » dans le tableau d'existence.

Par exemple, si le nombre total des sommets est égal à 10, le tableau d'existence au début sera :

0	0	0	0	0	0	0	0	0	0

Si on veut terminer le remplissage d'un chromosome de 5 gènes sans prendre en considération les sommets de début et de fin, il a la représentation suivante :

9	4	7	3	?

Son tableau d'existence sera :

0	0	1	1	0	0	1	0	1	0

Donc le dernier gène sera obligatoirement un parmi l'ensemble 1, 2, 5, 6, 8, 10, ce qui veut dire qu'il a l'étiquète '0' dans le tableau d'existence.

Évaluation de la population

Cette étape est le deuxième aspect important dans la manipulation des algorithmes génétiques. Dans cette étape, on évalue les individus composant la population par la fonction d'évaluation qui nous donne deux valeurs pour chaque chromosome. La première indique si le chemin est faisable ou non (elle prend deux valeurs : 0 si le chemin est faisable et 1 si le chemin est infaisable), et la deuxième donne la fonction fitness de chaque chromosome qu'il soit faisable ou non.

• La fonction fitness des chemins faisables :

La fonction d'évaluation utilisée pour évaluer les chemins faisables est simplement la longueur de chemin produit, donnée par :

$$E_f = \sum_{i=0}^{i+1} d(\mathrm{V}_i, V_{i+1}) \tag{3.3}$$

Où d(V_i,V_{i+1}) représente la distance directe entre le sommet V_i et V_{i+1}.

• La fonction fitness des chemins infaisables est donnée par :

$$E_i = \mathrm{s} + \mathrm{z} \tag{3.4}$$

Où s dénote le nombre d'obstacles intersectés le long du chemin et z le nombre moyen d'obstacles intersectés par segment. Nous avons utilisé cette fonction afin de bien différencier entre les chemins infaisables.

Étant donné les deux fonctions d'évaluation, le but du processus d'optimisation dans notre approche est de réduire au minimum les valeurs de « E_f» et de « E_i » dans les populations respectives. Dans le cas où la population contient les chemins faisables et infaisables, on considère que les chemins infaisables sont plus mauvais que le plus mauvais des chemins faisables. Une constante suffisamment grande C est ajoutée aux fonctions fitness des chemins infaisables pour assurer que les valeurs de la fonction fitness de n'importe quel chemin infaisable donné est plus mauvaise que celles des chemins faisables. 'C' est définie par :

C=(N+2)*D

(3.5)

Où N+2 indique le nombre maximum des gènes dans un individu, et D dénote la longueur maximale d'un segment de chemin (par exemple, ce serait la diagonale de l'environnement rectangulaire).

Les opérateurs génétiques utilisés

Cinq opérateurs ont été utilisés dans l'algorithme proposé, trois opérateurs génétiques standards : croisement, mutation et sélection, et deux opérateurs additionnels : réparation et inversion. Chaque opérateur est appliqué avec deux probabilités, une dans les générations et l'autre pour les individus de chaque population.

Croisement Cet opérateur combine deux individus (parents) choisis aléatoirement pour produire deux nouveaux individus (fils) comme suit : Pour chaque individu père, on choisit un point de croisement aléatoirement de telle sorte que la partie qui suit les points de croisement des deux parents soit permutée. C'est à dire le premier fils est généré en combinant la première partie du premier parent avec la deuxième partie du deuxième parent, et le second fils est généré en combinant la première partie du deuxième parent avec la deuxième partie du premier parent. Les fils ainsi obtenus doivent avoir des caractéristiques de chacun des parents. La figure 3.11 montre l'opération

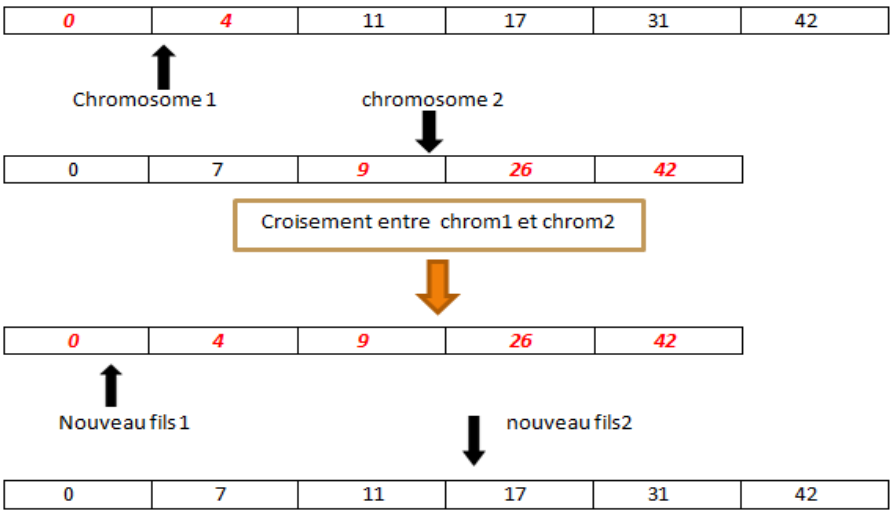


FIGURE 3.11 – opérateur de croisement

taux de croisement Dans notre cas, on a utilisé deux taux de croisement :
- l'un pour les générations : Indique la probabilité avec laquelle l'opérateur de croisement est appliqué aux générations (appliqué à combien de générations).

- Et l'autre pour la population : indique la probabilité avec laquelle l'opérateur de croisement est appliqué à la population (appliqué à combien de chromosomes dans la population).

Un taux trop bas de croisement peut introduire peu de nouveaux individus dans la population et peut mener à la stagnation du processus de la reproduction. Un taux trop élevé mène à une exploration très rapide de l'espace de recherche mais l'exécution de l'algorithme peut être dégradée et un meilleur individu (chromosome) peut être jeté très rapidement avant de reproduire sa structure.

mutation La mutation introduit un changement de gêne aléatoirement choisi d'un chromosome choisi aussi aléatoirement. Cette opération permet de diversifier la population et d'explorer l'espace de recherche (voir figure 3.12) :

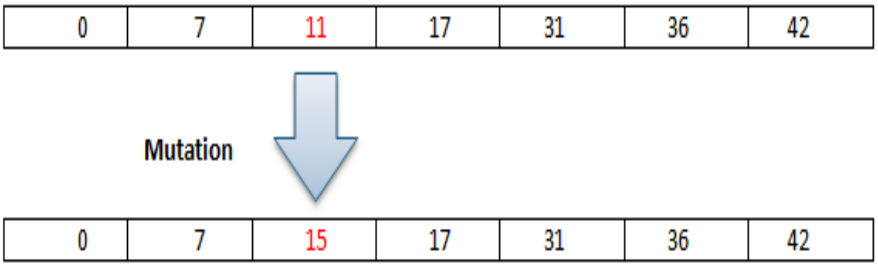


FIGURE 3.12 – opérateur de mutation

Taux de mutation Pour cet opérateur, on a aussi utilisé deux taux de mutation ; l'un pour les générations et l'autre pour les populations comme le cas du croisement. La mutation introduit de nouveaux secteurs encore inconnus dans la recherche. Un taux élevé de mutation augmente la diversité dans la population mais présente l'aspect aléatoire excessif dans la recherche. Réciproquement, un taux trop bas de mutation réduit la diversité et mène à un optimum local.

Inversion des gènes L'inversion contribue à l'ajustement et la mise en ordre de gênes (peut être désordonné) du même chromosome. Comme les autres opérateurs, elle est utilisée avec deux probabilités, pour les générations et pour la population, mais un taux trop élevé amène à l'infaisabilité de chemins c'est-à-dire la destruction de chemins faisable. On a utilisé deux types d'inversion :

Inversion de deux gênes Cet opérateur consiste à permuter deux gênes du même chromosome (on met l'un à la position de l'autre), comme montré à la figure 3.13.

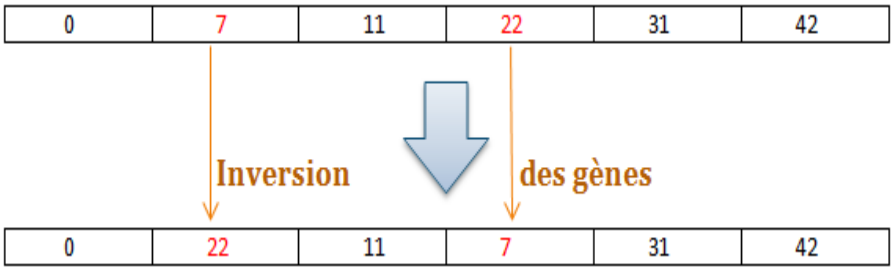


FIGURE 3.13 – inversion de deux gènes

Inversion de bloc de gênes Cet opérateur consiste à inverser l'ordre d'un bloc de gênes dans le même chromosome comme il est montré dans la figure 3.14.

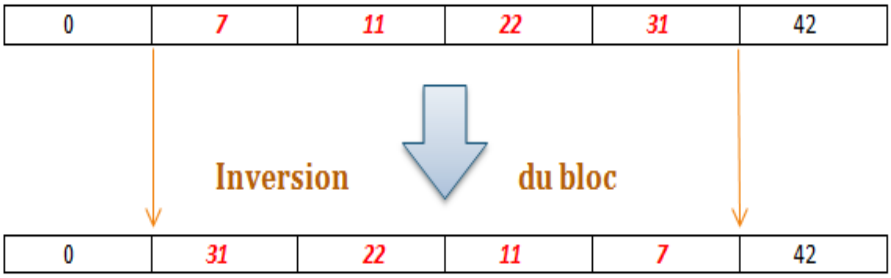


FIGURE 3.14 – inversion de bloc de gènes

Réparation Comme son nom l'indique, l'opérateur de réparation ajuste (répare) un segment infaisable aléatoirement choisi d'un chemin infaisable. Il est appliqué juste pour les chemins infaisables.

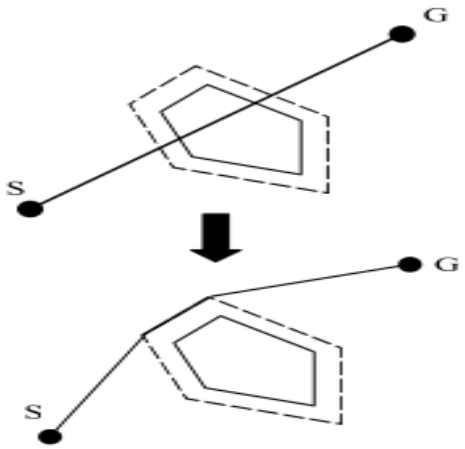


FIGURE 3.15 – opérateur de réparation

Une illustration de l'application de l'opérateur de réparation est donnée par la figure 3.15. Cet opérateur utilise les sommets de l'obstacle agrandi (montré par des tirets) pour déterminer un chemin faisable autour de l'obstacle intersecté. La figure 3.16 montre qu'il

y a plusieurs possibilités pour amener le robot du point S au point G, sans heurter cet obstacle. On a considéré ce problème comme un petit problème d'optimisation et on a appliqué l'algorithme de Dijkstra (défini au chapitre 1 paragraphe (1.4.1)) pour trouver le petit chemin optimal local qui contourne l'obstacle intersecté. Dans l'exemple illustré par la figure ci-dessous, le chemin optimal donné par l'algorithme de Dijkstra c'est celui du milieu.

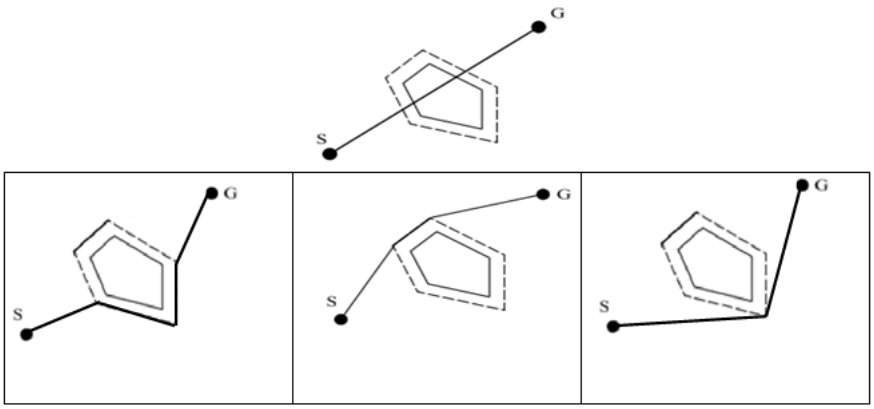


FIGURE 3.16 – réparation avec l'algorithme de Dijkstra

Cet opérateur est utilisé comme les autres opérateurs avec deux probabilités, pour les générations et pour la population. Dans notre approche, nous avons introduit cet opérateur avec une grande probabilité à cause de son importance et son efficacité. Quand la longueur du chemin est le critère d'évaluation, réparer aléatoirement des segments infaisables pourrait contribuer à l'amélioration rapide de la solution. Par conséquent, la probabilité de choisir l'opérateur de réparation prend une grande valeur. Après avoir produit une nouvelle solution en réparant des segments, la nouvelle solution sera évaluée en utilisant la fonction d'évaluation.

Sélection Cet opérateur est utilisé pour deux cas, la sélection des individus (parents) qui seront croisés pour avoir des nouveaux fils et la sélection des individus (chromosomes) pour une nouvelle génération parmi l'ensemble qui contient les individus de l'ancienne population et les nouveaux fils résultants de l'application de l'opérateur de croisement.
La sélection est basée sur la roulette où chaque individu est associé à une portion dont la longueur est proportionnelle à sa fonction fitness. Un individu qui possède une grande valeur de fitness, possède une grande chance d'être choisi. Dans notre cas, cela revient à associer la population à un intervalle réel [0, n] (n étant la somme des fitness de tous les individus). La largeur de l'intervalle associé à un individu est proportionnelle à sa fitness.

La sélection se fera d'une manière aléatoire en suivant la loi uniforme dans l'intervalle [0, n].
Un exemple avec cinq gênes est représenté sur la figure suivante :

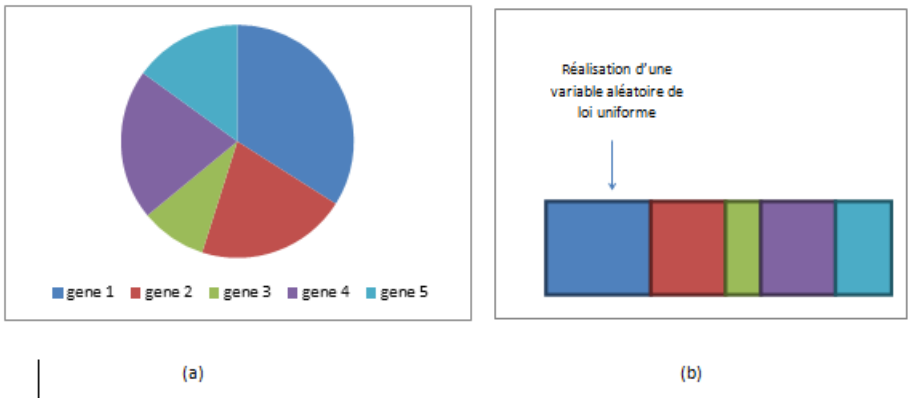


FIGURE 3.17 – (a) Roulette de loterie biaisée, (b) sa projection sur un intervalle donné. L'individu 1 est sélectionné après un tirage aléatoire.

3.6 planification dynamique

Comme indiqué dans le début du chapitre, le robot utilise le chemin généré par le planificateur statique pour se déplacer à travers les obstacles, le planificateur dynamique est automatiquement déclenché pour gérer la situation quand un obstacle dynamique est détecté. La zone de perception du robot peut être représentée par une valeur fixe, qui correspond à la portée du capteur embarqué. Les distances entre le robot et chacun des sommets de l'obstacle mobile peut être mise à jour périodiquement. Si la distance entre le robot et n'importe quel sommet de l'obstacle mobile est plus courte que cette valeur fixe, on peut dire que l'obstacle mobile est entré dans la zone de perception, et ainsi il peut être détecté par le capteur du robot.

on suppose qu'il y a un module dans le robot qui peut donner la forme de l'obstacle mobile et sa trajectoire, aussi bien que sa vitesse et sa direction. Le robot exploite ces informations (position, vitesse et direction des obstacles) ainsi que ses propres paramètres (position, vitesse et direction), afin de prévoir la possibilité de collision avec ces obstacles. Ce processus se fait selon plusieurs étapes :

Étape 1 : calcul du point d'intersection imaginaire entre les deux trajectoires du robot et de l'obstacle mobile.
Étape 2 : Calcul du temps nécessaire par le robot pour atteindre ce point d'intersection.
Étape 3 : à cet instant (le temps calculé à l'étape 2), trouver la position de l'obstacle et par conséquent ses sommets.

Étape 4 : tout dépend de la position de l'obstacle mobile au moment de l'intersection. Deux cas sont possibles, soit il y aura collision entre le robot et l'obstacle soit il n'y aura pas.
Pour identifier ces deux cas, la condition suivante doit être satisfaite, quand le robot atteint le point imaginaire d'intersection :
Si les positions des sommets de l'obstacle ne se coupent pas avec le chemin du robot, le robot peut continuer de suivre ce chemin.
Étape 5 : dans le cas contraire, il y a une intersection avec l'obstacle et le chemin suivi par le robot, ce qui implique une collision entre le robot et l'obstacle. Ce problème peut être résolu par l'algorithme du CTA. Cet algorithme trouve la position d'attente du robot et la durée exigée pour cela, jusqu'à ce que l'obstacle mobile croise le robot.

La survie du robot est assurée en choisissant sa position d'attente. Cependant, si l'obstacle emprisonne le robot [58], le robot réagit de la manière suivante :

1. Le premier cas est de rester en attente jusqu'à ce que l'obstacle passe. Ainsi, la méthode d'attente est fréquemment adoptée (comme il est expliqué dans notre travail).
2. Le deuxième cas où l'obstacle a atteint son objectif qui est le point d'intersection. Le robot reste en attente pendant une durée prédéfinie, si la position de l'obstacle est encore inchangée et le robot est encore emprisonné, comme montré dans la figure 3.18, alors le robot doit prendre une autre décision. Des méthodes conventionnelles de reprise telles que« "wallfollowing method" [59] » peuvent être utilisées. La méthode considère chaque situation dynamique instantanée en tant que situation statique. Dans ce cas, le robot utilise les sommets de l'obstacle mobile pour contourner cet obstacle et revenir à son chemin de départ, comme le montre la figure 3.19.

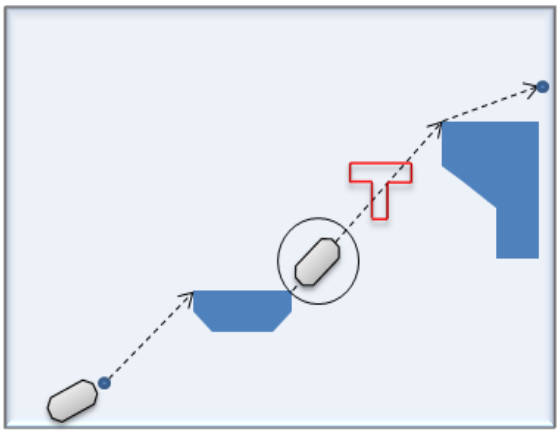


FIGURE 3.18 – l'obstacle a atteint son objectif qui est le point d'intersection

Dans le cas réel, le temps nécessaire pour que le robot réagisse et évite la collision devrait être assez court. Par conséquent, un algorithme efficace est un algorithme qui a un

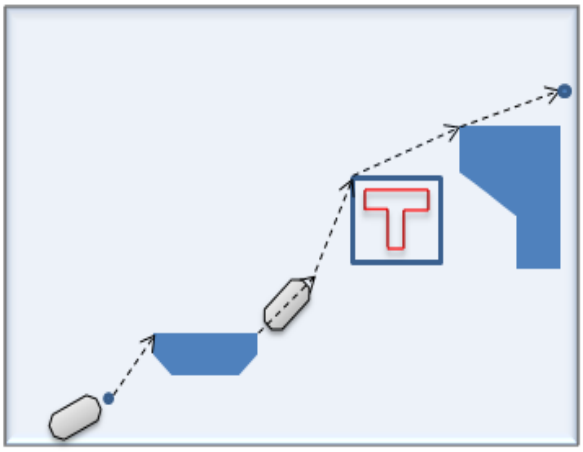


FIGURE 3.19 – le robot contourne l'obstacle

temps de planification relativement court pour permettre au robot de changer sa direction et éviter la collision avec l'obstacle, qui est un des buts principal de notre travail.

3.7 Conclusion

Dans ce chapitre nous avons proposé un algorithme de planification de trajectoire et d'évitement d'obstacles. Cet algorithme profite des avantages des algorithmes génétiques qui ont été utilisés récemment avec succès dans le problème de planification de chemins, pour trouver le chemin optimal à travers les obstacles statiques. Puis en suivant ce chemin, le planificateur dynamique basé sur l'algorithme du CTA est automatiquement déclenché pour gérer la situation quand un obstacle dynamique est détecté. L'algorithme peut être appliqué donc à des obstacles ayant des formes irrégulières, dans des environnements dynamiques ou partiellement dynamiques de même pour les environnements statiques.

L'algorithme présenté dans ce chapitre vise à surmonter les principaux inconvénients des approches existantes décrites dans le chapitre II. Il utilise une simple représentation de l'environnement utilisant la méthode du graphe de visibilité, généré initialement à partir des sommets des obstacles statiques, limitant de ce fait l'espace de recherche, ce qui nous amène à diminuer d'une manière significative le temps de calcul. Il utilise également une représentation variable de la longueur du chromosome, où chaque gène représente un simple sommet d'obstacle afin de réduire au minimum l'utilisation de l'espace mémoire du calculateur.

Chapitre 4

Simulations, tests et résultats

4.1 introduction

Dans ce chapitre nous présentons les résultats de simulation qu'on a faits en utilisant le logiciel MATLAB. Avec ces résultats, les performances de l'approche basée sur les algorithmes génétiques et l'algorithme du CTA sont évaluées. Quatre environnements sont examinés en utilisant l'approche développée. Les résultats d'évaluations illustrent clairement la première contribution principale de notre mémoire qui est sa capacité de générer pour le robot, une trajectoire libre sans collision avec les obstacles dans un environnement dynamique.
Toutes les simulations sont réalisées sur un ordinateurs de type Intel(R) Core(TM) i3 cpu M 380 @ 2.53 GHz possédant une RAM de 2 Go. Les simulation sont faite sur un PC (Sony vaio) core deux duo , ram 4 Gb

4.2 Les environnements de simulation

L'approche est testée dans quatre environnements différents, chaque environnement contient des obstacles statiques et dynamiques. Le critère d'optimisation du chemin donné par le planificateur génétique est la longueur. La solution trouvée par l'algorithme est optimale ou proche de l'optimale. Les nombres d'obstacles statiques et dynamiques dans les quatre environnements de simulation sont présentés dans le tableau 4.1. Les nombres de sommets des obstacles statiques pour la planification avec les algorithmes génétiques sont également donnés dans le tableau 4.1. Les obstacles statiques et dynamiques dans chacun des quatre cas ont des formes aléatoires (irrégulières). Les deux premiers environnements simulent des scénarios simples où les obstacles dynamiques apparaissent au robot simultanément et ils se déplacent simplement en avant dans la même direction. Les deux derniers environnements sont des scénarios plus compliqués où les obstacles dynamiques n'apparaissent pas au robot simultanément mais chacun apparaît à un temps différent et il se déplace en avant ou vers l'arrière. De plus, les nombres des obstacles dynamiques et statiques sont plus grands que ceux des deux environnements simples. Toutes les informations sur les positions des obstacles statiques dans la carte de l'environnement sont connues par le robot avant qu'il commence à se déplacer.

environnement	Nombre d'obstacles statiques	Nombre d'obstacles dynamiques	Nombre des sommets des obstacles statiques
1	3	2	12
2	5	3	20
3	8	4	32
4	20	4	90

TABLE 4.1 – nombre des obstacles dans les quatre environnements

4.3 Les paramètres de contrôle de l'algorithme

Cette partie concerne le planification avec les algorithmes génétiques .Généralement, le dernier aspect critique en utilisant les algorithmes génétiques est le choix approprié des paramètres du réglage de ces algorithmes. Malheureusement, il n'y a aucune manière formelle de définir ces paramètres appropriés. Traditionnellement, ceci est réalisé expérimentalement. Pour nous, après plusieurs tests, nous avons fixé les paramètres de contrôle de l'algorithme comme montré dans le tableau 4.2 avec la taille de la population égale à 30 et le nombre de génération égal à 100.

opérateur	Probabilité dans les générations	Probabilité dans les population
croisement	0.7	0.6
mutation	0.4	0.3
réparation	0.85	0.8
Inversion de gènes	0.1	0.05
Inversion de bloc des gènes	0.05	0.5

TABLE 4.2 – Les paramètres de contrôle de l'algorithme

4.4 Résultat de simulation

4.4.1 Environnement 1

(3 obstacles statiques et deux dynamiques)

Premièrement, nous avons testé notre algorithme avec un environnement simple, qui est constitué de trois obstacles statiques de couleur noire, avec l'élargissement en bleu, et deux dynamiques colorés en rouge dont les trajectoires sont représentées par deux simples lignes vertes. Le nombre total de sommets de l'environnement statique est égal à 12. La figure 4.1 (a) montre que le robot utilise premièrement le planificateur statique pour déterminer le chemin optimal basé sur les sommets des obstacles statiques. La figure 4.1 (b) montre que si l'obstacle mobile est détecté avec une possibilité de collision, une position et un temps d'attente sont calculés par le planificateur dynamique qui utilise l'algorithme du CTA (Concept de Temps d'Attente).

Dans la figure 4.1 (b) on voit bien que la trajectoire du robot est intersectée par les deux obstacles dynamiques de l'environnement. Le vecteur d'attente donné par le planificateur dynamique est [0 0 1.5861](seconde), pour le premier obstacle ,il n'y aura donc pas de collision et le robot n'a pas besoin de calculer ni la position d'attente, ni le temps d'attente. En revanche, pour le deuxième obstacle, il y aura une collision ce qui amène le robot à calculer une position d'attente (représentée par le cercle en bleu) et un temps d'attente

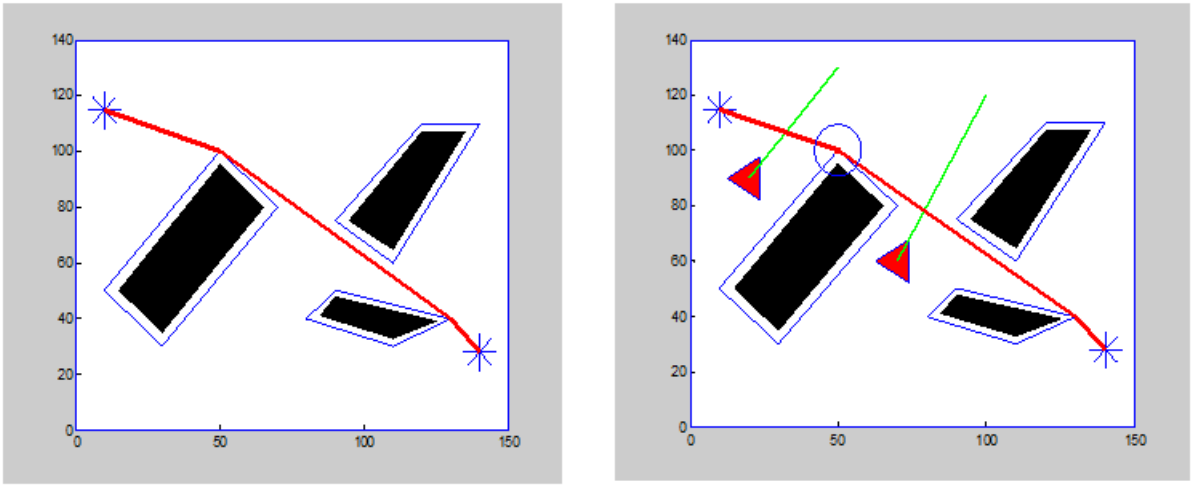


FIGURE 4.1 – (a)planificateur génétique , (b)planificateur de CTA

égal à 1.5861 secondes (ce temps représente le temps nécessaire à l'obstacle pour dépasser le point d'intersection).

La figure 4.2 montre la longueur du chemin optimal donné par le planificateur statique, ainsi que le temps nécessaire pour trouver ce chemin et la période de calcul du planificateur dynamique. La simulation est exécutée dix fois, les résultats dans le tableau sont les résultats moyens des 10 simulations.

Espace de recherche (nombre de sommets)	**12**
Nombre des obstacles dynamiques	2
Nombre des obstacles statiques	3
Meilleur chromosome	[13 11 2 14]
Longueur du chemin donné par le planificateur statique (cm)	158.3405
Vecteur d'attente (seconde)	[0 0 1.5861]
Temps de calcul du planificateur statique (seconde)	0.523856
Temps de calcul du planificateur dynamique (seconde)	0.077183

FIGURE 4.2 – résultats de l'environnement 1

4.4.2 Environnement 2

(5 obstacles statiques et 3 dynamiques)

Le deuxième environnement est représenté par la figure 4.3 , il contient cinq obstacles statiques et deux obstacles dynamiques. L'ensemble des 20 sommets statiques représentent l'espace de recherche. La même stratégie que l'environnement 1 est appliquée. Le robot utilise, en premier temps, le planificateur statique pour obtenir la trajectoire optimale basée sur des obstacles statiques (voir la figure 4.3 (a)), et le planificateur dynamique basé sur l'algorithme du CTA pour réagir aux obstacles dynamiques (figure 4.3 (b)).

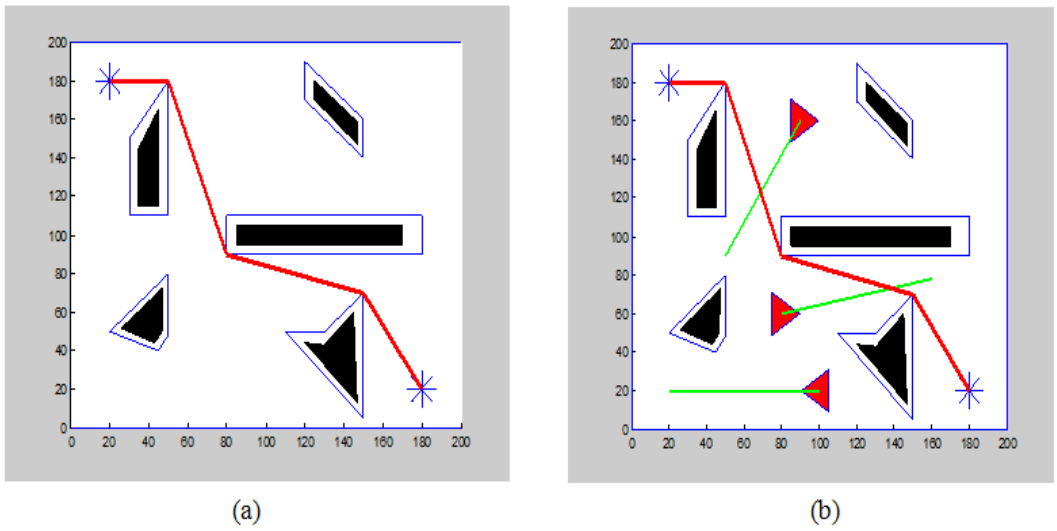


FIGURE 4.3 – (a)planificateur statique , (b)planificateur dynamique

La trajectoire optimale statique est donnée par le chromosome [21 12 13 7 22] (la ligne en rouge), avec 21 représente le point de départ, 22 représente le point d'arrivée et les sommets 12, 13 et 7 représentent les points intermédiaires. Concernant les obstacles dynamiques, on voit que la trajectoire du robot est intersecté juste par les trajectoires des obstacles 2 et 3. Le planificateur dynamique nous donne le vecteur de temps d'attente suivant :[0 0 0 0],ce qui indique qu'il n'y aura pas une collision entre le robot et ces deux obstacles. Si on analyse les résultats donnés par l'algorithme, sachant que le vecteur de vitesse du robot dans chaque segment est [0.3 0.6 0.7 0.3] m/s, et les vitesses des deux obstacles 0.6 et 0.7 m/s respectivement, on voit que le temps nécessaire au robot pour atteindre le premier point d'intersection est 13.0103 secondes. Cette valeur n'appartient pas à l'intervalle du temps [332.8881 340.8881](seconde),qui est nécessaire pour le passage de l'obstacle par le point d'intersection , donc il n'y aura pas de collision. Le même raisonnement est effectué pour le deuxième obstacle, le temps nécessaire pour atteindre le deuxième point d'intersection entre le robot et l'obstacle est 38.835 secondes et l'intervalle de temps pour le passage d'obstacle de ce point est [162.7008 170.7008].

Les résultats sur la figure 4.4 montrent que dû à l'augmentation de l'espace de recherche, l'algorithme a besoin de plus de temps pour trouver le chemin optimal.

Espace de recherche (nombre de sommets)	**20**
Nombre des obstacles dynamiques	3
Nombre des obstacles statiques	5
Meilleur chromosome	[21 12 13 7 22]
Longueur du chemin donné par le planificateur statique (cm)	255.6729
Vecteur d'attente (seconde)	[0 0 0 0]
Temps de calcul de planificateur statique (seconde)	1.095054
Temps de calcul du planificateur dynamique (seconde)	0.100285

FIGURE 4.4 – résultats de l'environnement 2

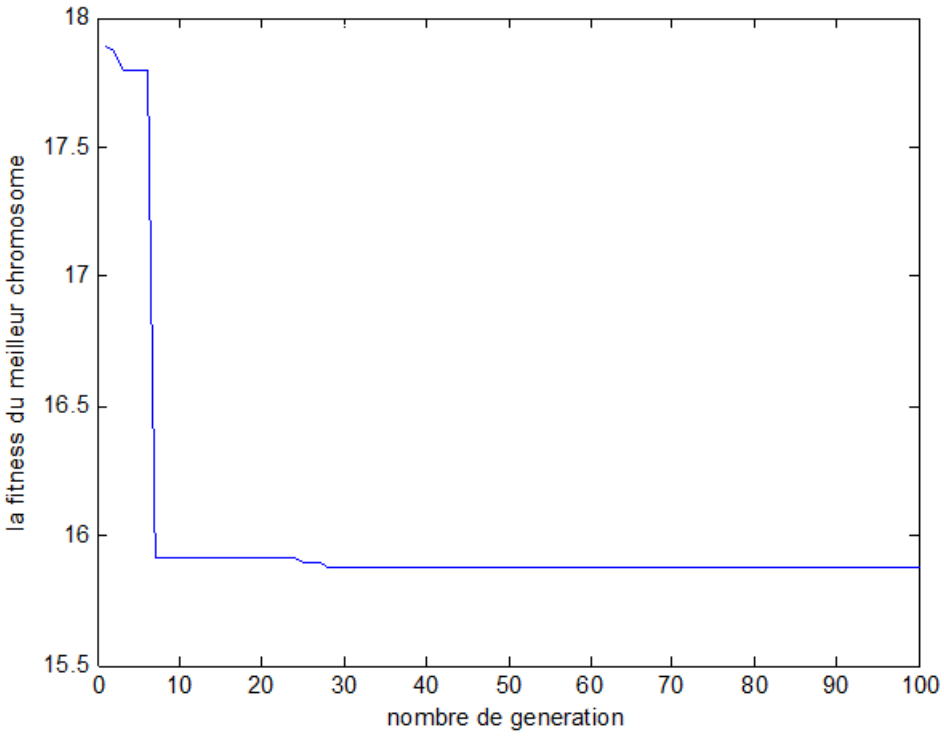


FIGURE 4.5 – l'évolution de la fonction fitness du meilleur chromosome

La figure 4.5 représente l'évolution de la fonction fitness du meilleur chromosome (chemin). D'après cette figure, on remarque que le premier chemin faisable a été trouvé dès la première génération, et la meilleure solution est obtenue dans la 28^{ieme} génération.

4.4.3 Environnement 3

(8 obstacles statiques et 4 dynamiques)

Dans le troisième cas, un environnement plus compliqué est testé. Trois obstacles statiques et un dynamique ont été ajoutés dans l'environnement. Les trajectoires des obstacles dynamiques ne sont pas des lignes droites, l'obstacle dynamique peut changer sa direction pendant son déplacement tel qu'il est représenté dans la figure 4.6(b).

Le planificateur doit trouver une chaine des sommets ordonnés qui amène le robot de la position initiale [16, 2] cm vers une position finale [2, 17]cm sans couper les obstacles statique et qui assurent une langueur minimale. La figure 4.6(a) représente la trajectoire trouvée par les algorithmes génétiques. La figure 4.6 (b) montre que le robot utilise le planificateur dynamique pour trouver une position et un temps d'attente pour éviter la collision avec l'obstacle 1.

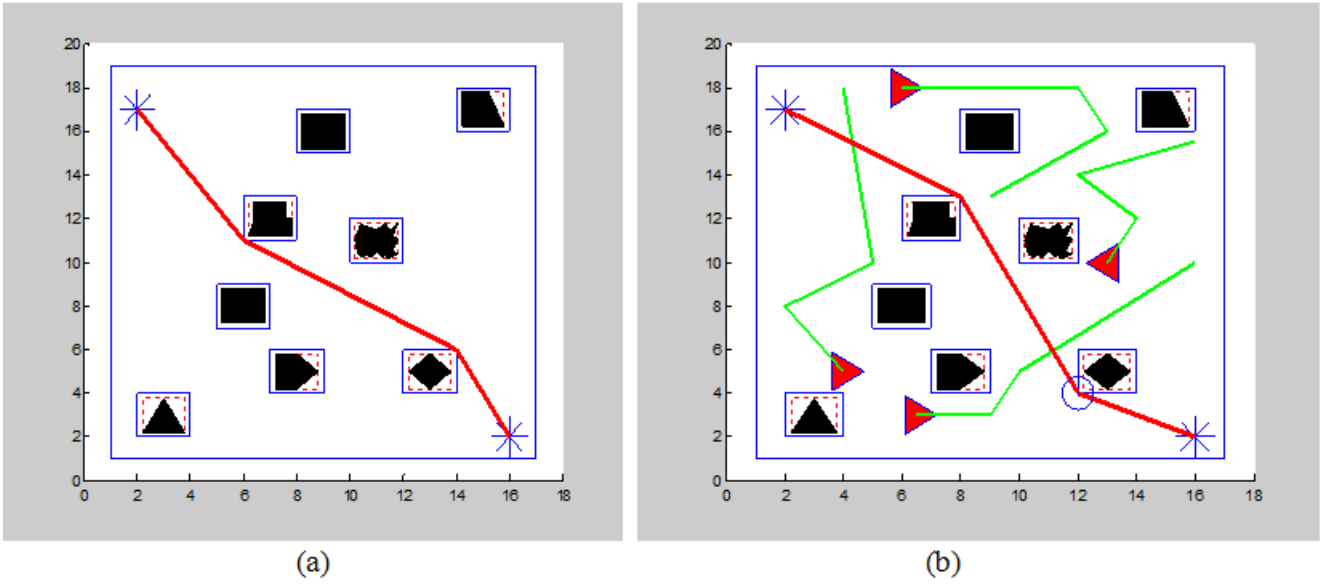


FIGURE 4.6 – (a)planificateur statique , (b)planificateur dynamique

Puisque l'environnement est plus compliqué, les résultats donnés par le planificateur statique, qui est basé sur les algorithmes génétiques ne sont pas identiques à chaque fois. Cependant, tous les résultats donnés par ce planificateur sont des résultats optimaux ou proche-optimaux. La figure 4.7 montre les résultats moyens de dix simulations.

Espace de recherche (nombre de sommets)	32
Nombre des obstacles dynamiques	4
Nombre des obstacles statiques	8
Meilleur chromosome	[33 9 27 34]
Longueur du chemin donné par le planificateur statique (cm)	20.9028
Vecteur d'attente (seconde)	[0 3.5690 0]
Temps de calcul de planificateur statique (seconde)	3.455583
Temps de calcul de planificateur de dynamique (seconde)	0.08457

FIGURE 4.7 – résultats de l'environnement 3

4.4.4 Environnement 4

(20 obstacles statiques et 4 dynamiques)

L'environnement quatre est le plus compliqué des quatre environnements présentés. Il contient 20 obstacles statiques avec 90 sommets représentant l'espace de recherche. Comme il est montré dans la figure 4.8 (a), cet environnement est très compliqué du point de vue recherche d'une trajectoire faisable et optimale. La plupart des techniques qui sont développées ont trouvé leurs limites dans des environnements encombrants comme celui-ci [45][3] [46][47], que ce soit pour donner une solution optimale ou pour minimiser le coût de planification (le temps de calcul, ...).

Concernant les obstacles dynamiques, cet environnement contient quatre obstacles qui ont différentes trajectoires (simple et compliquée) et ces obstacles peuvent changer leurs vitesses pendant le mouvement (figure 4.8 (b)).

L'augmentation des obstacles dans l'environnement de navigation augmente considérablement la complexité de la tâche de planification, en termes de temps de calcul ou d'optimisation de la langueur. Cette complexité sera plus significative si on passe à la pratique, parce que l'augmentation des obstacles implique la réduction de l'espace libre où le robot va se mouvoir et avec la considération des dimensions physique du robot la tâche sera plus compliquée à réaliser.

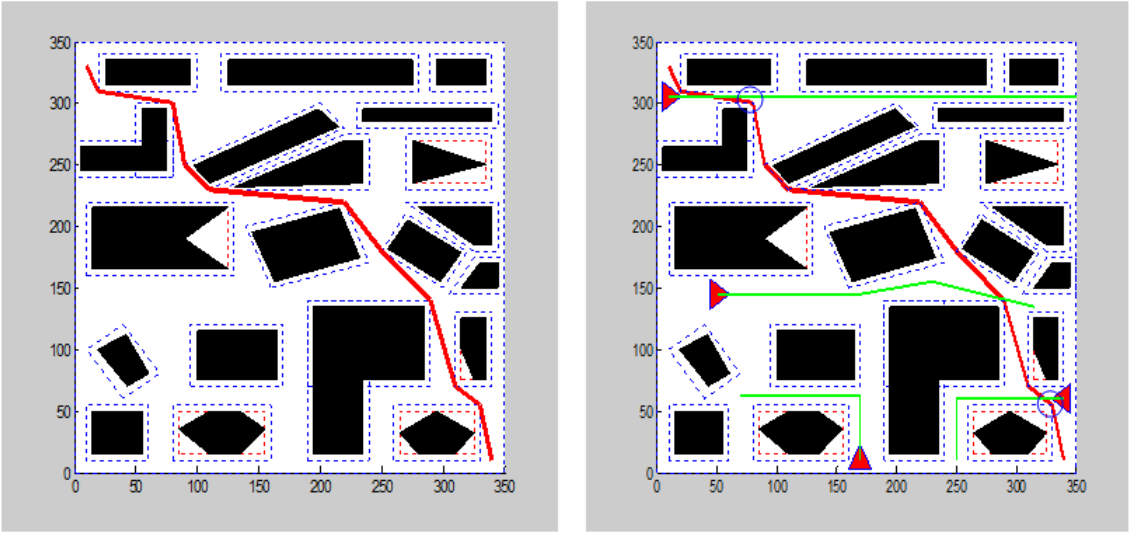


FIGURE 4.8 – (a)planificateur statique , (b)planificateur dynamique

La figure suivant donne les résultats de simulation de cet environnement.

Espace de recherche (nombre de sommets)	90
Nombre des obstacles dynamiques	4
Nombre des obstacles statiques	20
Meilleur chromosome	[91 19 29 15 41 39 61 62 59 77 92]
Longueur du chemin optimal donné par le planificateur statique (cm)	532.6458
Vecteur d'attente (seconde)	[0 0.3152 0 0 0 0 0 0 5.8450 0]
Temps de calcul du planificateur statique (seconde)	19.018568
Temps de calcul du planificateur dynamique (seconde)	0.188072

FIGURE 4.9 – résultats de l'environnement 4

D'après la figure 4.9 le robot va attendre le passage d'obstacle mobile au début du 2^{ime} segment (la position d'attente est montré par une cercle bleu sur la figure, cette position est correspond au sommet numéro 19,) durant 0.315 s. et au début du dernier segment (sommet numéro 77) durant 5.8450 s.

Les deux figures 4.10(a) et 4.10(b) représentent l'évolution du minimum et de la

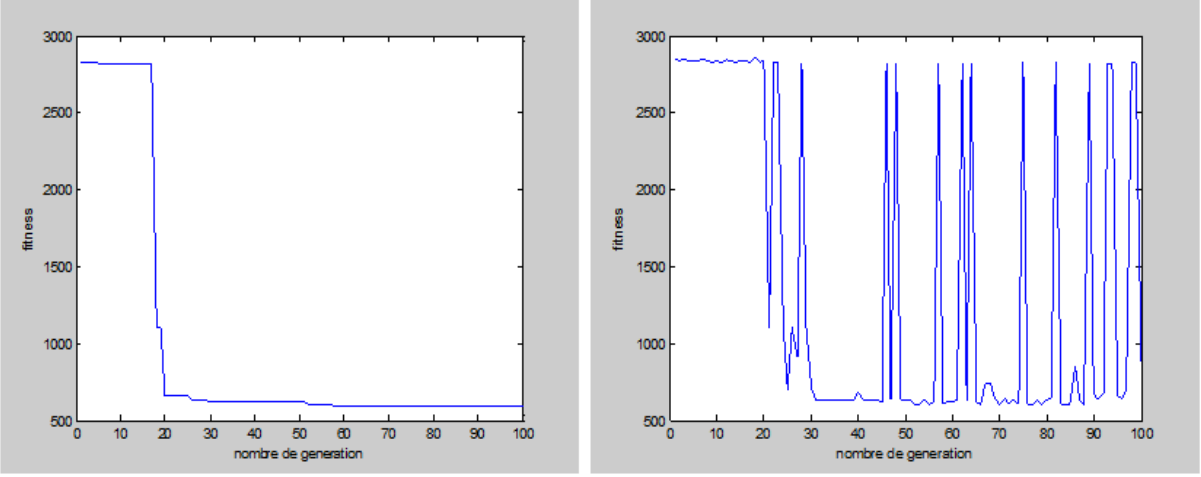


FIGURE 4.10 – (a)l'évolution du meilleur chromosome, (b)la moyenne des fitness

moyenne des fitness dans chaque génération respectivement. Initialement, l'approche ne trouve aucun chemin faisable. De la première figure, on constate que l'algorithme a été déroulé 20 fois pour arriver à trouver le premier chemin faisable. L'obtention de la trajectoire optimale est trouvée à la génération 57.

4.5 Évaluation de l'algorithme

Pour évaluer l'efficacité du planificateur statique, nous avons fait les tests suivants :
(a) le temps d'exécution pour trouver le premier chemin faisable ;
(b) le nombre de générations pour déterminer le premier chemin faisable ;
(c) le temps d'exécution pour trouver le chemin final ;
(d) le nombre de générations requises pour trouver le chemin final ;
(e) la longueur du chemin final.
Nous avons effectué plus de 20 tests, avec un nombre de générations égal à 100. Les résultats moyens de tous ces tests sont représentés dans les deux tableaux 4.3 et 4.4 respectivement.

environnement	le temps d'exécution pour trouver le premier chemin faisable (s)	le nombre de générations pour déterminer le premier chemin faisable
1	0.8142	1
2	0.9456	1
3	1.0681	4
4	16.8765	43

TABLE 4.3 – statistique des résultats

Dans le tableau 4.3, il parait clairement que l'augmentation de nombre d'obstacle statique et dynamique (environnement 4) influe considérablement sur le temps de calcul et la recherche du chemin optimal.

environnement	le temps d'exécution pour trouver le chemin optimal (s)	le nombre de générations pour déterminer le chemin optimal	la longueur du chemin optimal
1	1.1320	4	158.3405
2	2.4412	18	255.6729
3	3.6365	39	20.9028
4	52.8546	87	541.32

TABLE 4.4 – statistique des résultats

4.6 Influence et ajustement des paramètres

L'ajustement des paramètres des algorithmes génétiques est parmi les objectifs atteints par la simulation. La multitude d'expérimentations de simulation que nous avons effectué à l'aide de notre simulateur, nous a permis de savoir l'influence de chaque paramètre sur les résultats obtenus et ainsi de choisir les valeurs qui ont permis de donner de bons résultats.

4.6.1 influence d'operateur de mutation

On va donner deux taux (bas et élevé) pour cet opérateur pour voir son influence sur les résultats :

opérateur	Probabilité dans les générations	Probabilité dans les populations
croisement	0.8	0.7
mutation	0.1	0.1
réparation	0.8	0.8
Inversion de gènes	0.05	0.1
Inversion de bloc des gènes	0.05	0.1
La taille de la population est : 30	Nombre de génération est : 100	

TABLE 4.5 – taux des opérateurs avec mutation faible

Un taux très bas de mutation égale à 0.1 donne le résultat présenté dans la figure 4.11. Malgré que Le croisement est appliqué avec un taux égal à (0.8 pour les générations /0.7

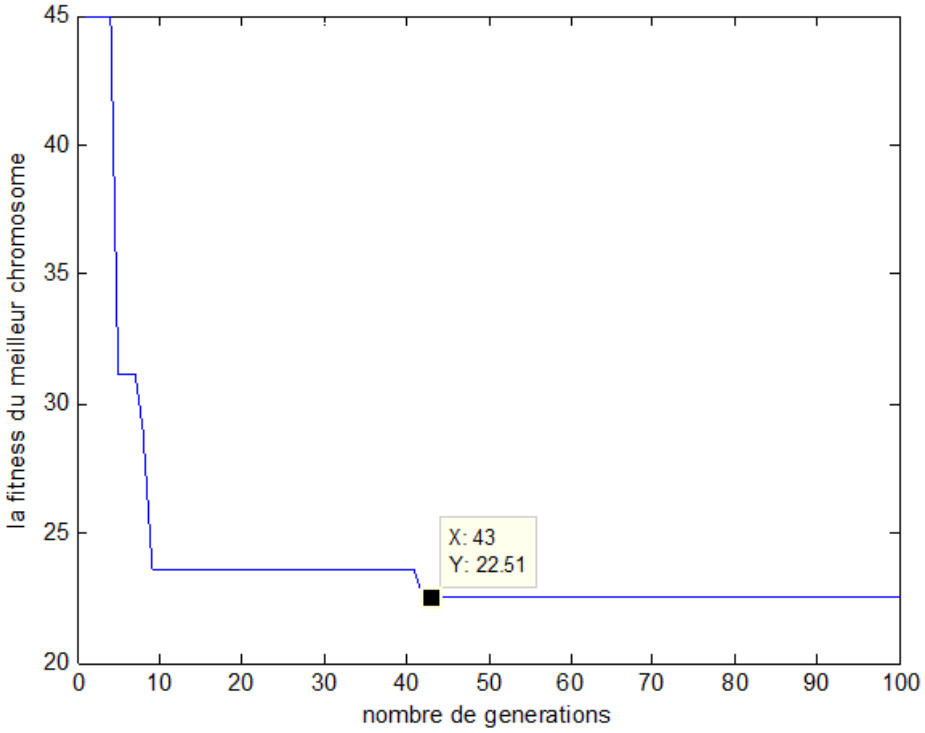


FIGURE 4.11 – l'évolution de fitness du meilleur chromosome avec un taux de mutation faible

pour les populations), le résultat nous indique que la recherche avec un taux faible de mutation se fait dans le même secteur, ce qui conduit à la convergence de l'approche vers un minimum local (voir figure 4.11) où sa fonction fitness est égale à 22.51 (cm).

opérateur	Probabilité dans les générations	Probabilité dans les populations
croisement	0.1	0.01
mutation	0.9	0.9
réparation	0.1	0.01
Inversion de gènes	0.05	0.1
Inversion de bloc des gènes	0.05	0.1
La taille de la population est : 30	Nombre de génération est : 100	

TABLE 4.6 – taux des opérateurs avec mutation élevé

Un taux très élevé de mutation augmente considérablement le temps de calcul, et introduit l'aspect aléatoire excessif, et peut également détruire certains chromosomes de bonne fitness, ce qui apparait clairement dans l'évolution de la moyenne des fitness des générations sur la figure 4.12.

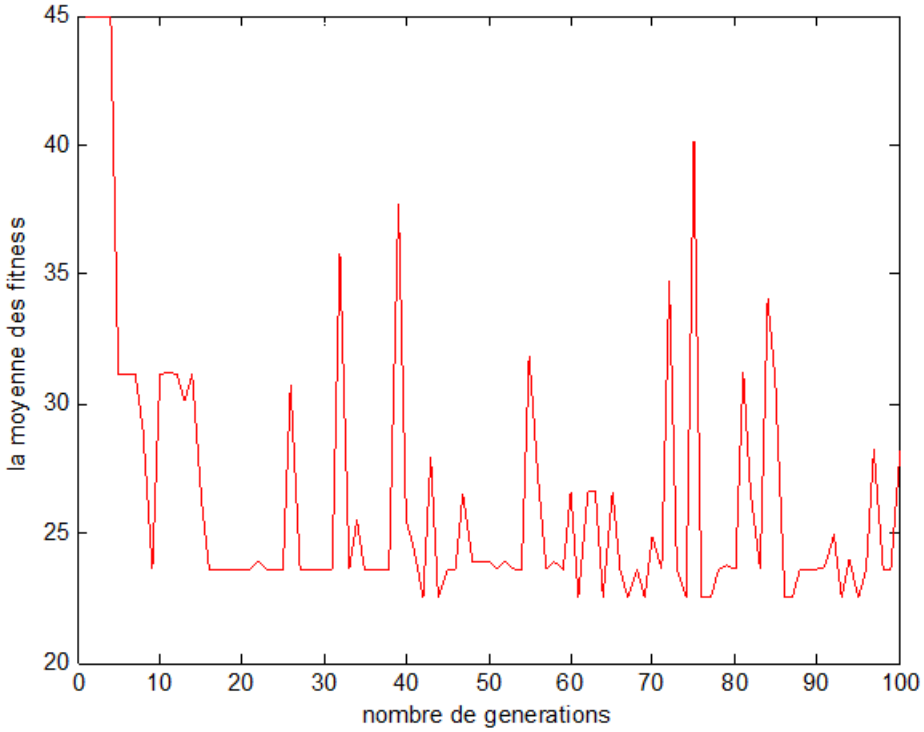


FIGURE 4.12 – l'évolution de fitness du meilleur chromosome avec un taux de mutation élevé

4.6.2 influence d'opérateur de croisement

Pour voir l'influence de cet opérateur on l'a introduit aussi avec deux taux, l'un taux bas (0.01), et l'autre élevé .

opérateur	Probabilité dans les générations	Probabilité dans les populations
croisement	0.01	0.01
mutation	0.5	0.3
réparation	0.8	0.8
Inversion de gènes	0.05	0.1
Inversion de bloc des gènes	0.05	0.1
La taille de la population est : 30	Nombre de génération est : 100	

TABLE 4.7 – taux des opérateurs avec croisement faible

Avec un taux de croisement très bas, on remarque qu'il n'y a pas une exploitation de l'espace recherche (après la seizième génération, on ne voit pas une amélioration dans l'algorithme, malgré que le chromosome de fitness 21.12 n'est pas un minimum global), à cause de la stagnation de la reproduction, l'optimum local (de fitness 21.12) est obtenu à la fin de la recherche (voir figure 4.13).
Un taux très élevé de croisement va augmenter le temps de calcul.

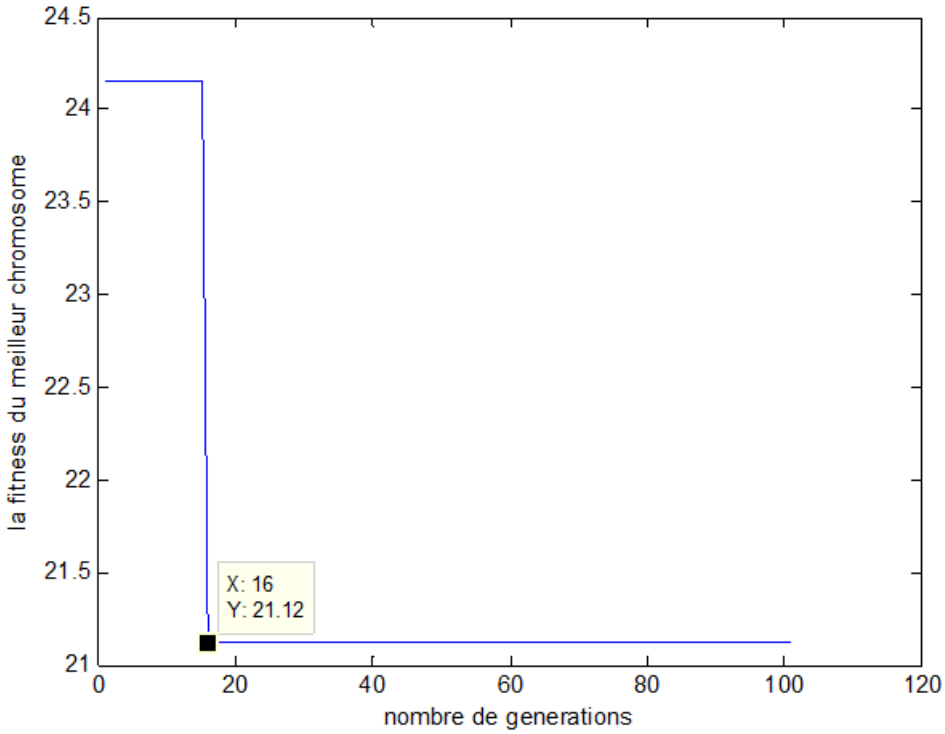


FIGURE 4.13 – l'évolution de fitness du meilleur chromosome avec un taux faible de croisement

4.6.3 influence de l'opérateur de réparation

Cet opérateur a une grande importance, parce qu'il fait la réparation des segments intersectés avec les obstacles, et sert à l'apparition des trajectoires faisables pour accélérer le processus de recherche. Dans la figure 4.14, la première trajectoire faisable est apparue à la génération 55, ce qui montre le manque du rôle de la réparation.

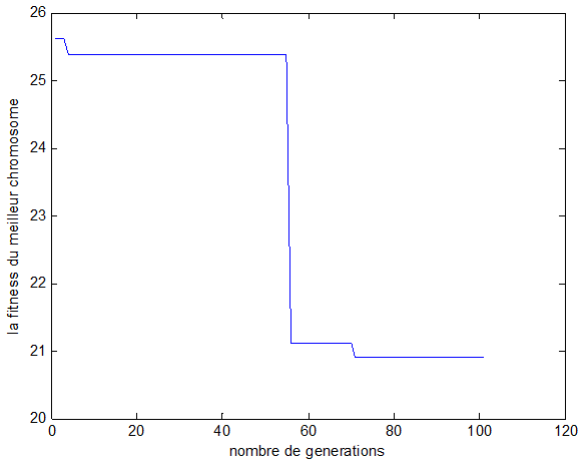


FIGURE 4.14 – l'évolution de fitness du meilleur chromosome avec un taux de réparation faible

La réparation n'a aucun mauvais effet sur la population parce qu'elle est appliquée seulement à des trajectoires infaisables, en revanche elle augmente le temps de calcul.

4.6.4 influence de Nombre de générations

Dans notre approche, le nombre des générations représente la condition d'arrêt. Un nombre petit, peut ne pas permettre d'atteindre l'optimum global, mais également un nombre très grand augmente considérablement le temps de calcul. On peut donc conclure que tous les opérateurs doivent être utilisés avec un taux adéquat pour aboutir à une bonne optimisation soit en termes de longueur de trajectoire ou en termes de temps de calcul.

4.7 Conclusion

Dans ce chapitre on a testé notre approche dans quatre environnements de simulation, la complexité de ces environnements en terme de nombre des obstacles statiques ou dynamique augmente du premier jusqu' à le quatrième. L'objectif de ces tests est de simuler différentes situations pour valider notre approche. De plus on a fait certains tests dans l'environnement 3 avec différents taux des opérateurs utilisés afin de visualiser leurs influences sur les résultats obtenus.

conclusion générale

La tâche principale destinée à un robot mobile autonome est de naviguer d'une manière autonome dans son environnement pour rejoindre un but bien défini, tout en évitant les obstacles statiques ou dynamique rencontrés dans son chemin. La planification de la trajectoire représente une des phases les plus importantes pour une tâche complète de navigation. Nous nous sommes intéressés, au cours de ce travail, aux concepts et outils algorithmiques permettant la planification de la trajectoire complète d'un point de départ vers un point de destination. L'approche développée est basé sur les algorithmes génétiques et l'algorithme du concept de temps d'attente (CTA) dans un environnement encombré d'obstacles qui ont des formes, des tailles et des endroits arbitraires. L'algorithme proposé est applicable à des environnements statiques, partiellement dynamiques aussi bien que les environnements dynamiques.

L'approche est divisée en deux parties, statique et dynamique, dans la partie statique, elle utilise les sommets des obstacles stationnaires comme un espace de recherche. En utilisant les algorithmes génétiques, elle trouve le chemin optimal pour le robot. Quand un obstacle mobile est détecté avec la possibilité de collision avec le robot, la partie dynamique est activée, dans cette partie, l'approche applique le concept de temps d'attente (CTA) pour calculer la position et le temps d'attente afin éviter cet obstacle.

La simulation nous a permis de tester et valider notre approche de planification, Ajuster les paramètres de cette dernière et y découvrir certaines limites et en conséquence de l'améliorer en adoptant des solutions aux problèmes découverts. Les résultats obtenus en simulant les trajectoires données par notre approche dans des environnements complexes et dynamiques sont, dans la majorité des cas, suffisants (cible atteinte, temps d'exécution faible et des trajectoires plus ou moins optimaux). Ces résultats illustrent clairement la première contribution principale de notre mémoire, celle que l'approche est capable de générer pour le robot, une trajectoire libre sans collision avec les obstacles dans un environnement dynamique.

Dans ce travail, on a introduit quelques nouvelles contributions qui sont :

1. L'introduction de l'algorithme de Dijikstra (définie au chapitre I paragraphe 1.1.4) dans l'opérateur de réparation, pour trouver le chemin optimal local qui contourne l'obstacle intersecté.
2. Dans le cas où l'algorithme de Dijikstra ne trouve aucune solution, l'approche sé-

lectionne un sommet aléatoire de l'obstacle intersecté pour contourner cet obstacle.

3. La combinaison de l'algorithme de CTA avec les algorithmes génétiques pour la planification de la trajectoire dans les environnements dynamiques.

Si ces quelques résultats sont encourageants, il faut aussi considérer les limitations de la recherche dans cette expérience. Le premier problème auquel nous avons dû faire face est que les dimensions du robot sont ignorées dans ce travail bien que les dimensions des obstacles sont prises en considération dans le calcul. C'est un sujet qui n'a pas été systématiquement étudiée dans la planification de la trajectoire dans les environnements dynamiques pour les robots mobiles. Par conséquent, ce sujet peut être considéré comme une perspective pour notre travail.
Le deuxième problème, est que dans notre travail, aucune marque ou configuration particulière des capteurs n'est indiquée. On considère que les capteurs peuvent acquérir les informations sur le mouvement des obstacles mobiles telles que la vitesse et la direction de l'obstacle une fois que cet obstacle est entré dans la gamme de conception du robot. Pour un modèle plus réaliste, on doit discuter comment les capteurs obtient les informations sur le mouvement de l'obstacle dynamique.

En termes de perspectives, l'amélioration de notre méthode se situe principalement au niveau de l'implémentation de notre approche sur un robot réel, tel que le « Robucar » ou « B21r » du centre de recherche CDTA. Dans un but global de développer un projet qui traite une tâche complète amenant à la navigation, cette tâche comporte la phase de perception, la localisation et la cartographie, puis la planification et l'exécution de mouvement.
Mais aussi on peut prévoir :

- L'extension de la phase d'exécution de la trajectoire après la planification en utilisant des lois de commandes robustes (modes glissants, logique floue. etc.)
- Pour des modèles plus réalistes, des environnement plus complexe et plus réalistes seront utilisés et examinés en utilisant cette approche. Les environnements dans ce travail sont décrits dans une surface bidimensionnelle et la trajectoire des obstacles dynamiques est constituée par une série de segments des lignes. Pour modeler des environnements plus réalistes pour le robot, des modèles tridimensionnels sont prévus et les trajectoires des obstacles dynamiques peuvent être des courbes au lieu des segments des lignes droites. L'environnement tridimensionnel utilise toujours les sommets comme un espace de recherche ; la seule différence est que dans le calcul on doit ajouter la coordonnée Z.
- De plus, ce travail peut être prolongé à des environnements dynamiques inconnus avec multiples robots.

Bibliographie

[1] A. B. Michael Affenzeller, Stephan Winkler, Stefan Wagner. *" Genetic Algorithms and Genetic Programming (Modern Concepts and Practical Applications)"*.Editorial : Taylor Francis Group, LLC, 2009, pp.

[2] ActivMedia. P3-DX. *"World's Most Popular Intelligent Wheeled Robot"*, 2004,http : //www.activrobots.com , le 06/05/2012.

[3] A. Nearchou. *"Path planning of a mobile robot using genetic heuristics"*,Robotica, vol. 16, pages 575–588, Sept 1998.

[4] Bernard BAYLE. *"robotique mobile"*,livre,Ecole Nationale Supérieure de Physique de Strasbourg Université de Strasbourg, 2005.

[5] B. H. V. Topping, J. Sziveri, A. Bahreinejad, J. P. B. Leite, and B. Cheng. *"Parallel processing,neural networks and genetic algorithms"*, Elsevier Science, vol. 29, no. 10, pp. 763-786, 1998.

[6] C. Gregg. *"Genetic Algorithms in Autonomous Embedded Systems"*, ECE 687 : Embedded Systems , pp. 1-7, 2009.

[7] C. Urmson, J. Anhalt, " High Speed Navigation of Unrehearsed Terrain. *"Red Team Technology for Grand Challenge 2004."*,Rapport technique, Université de Carnegie Mellon, Pittsburg, Etats-Unis, 2004.

[8] David FILLIAT . *"robotique mobile"*,cours, École Nationale Supérieure de Techniques Avancées' ParisTech 2011.

[9] D.E. Goldberg. *"Genetic Algorithms in Search, Optimization, and Machine Learning"*, (Addison-Wesley, Reading, Mass., 1989).

[10] F. G. Lobo, C. F. Lima, and Z. Michalewicz. *" Parameter Setting in Evolutionary Algorithms"*, Springer-Verlag Berlin Heidelberg, 2007.

[11] Frédéric LARGE. *"Navigation Autonome D 'un Robot Mobile en Environnement dynamique incertain"*, thèse de doctorat, université de Savoie France 2003.

[12] G. Campion, G. Bastin et B. D'Andréa-Novel. *" Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots"*,IEEE Transactions on Robotics and Automation,vol. 12, no. 1, pages 47–62, 1996.

[13] Hui Miao. *" Robot Path Planning in Dynamic Environments using a Simulated Annealing Based Approach"*, these de master, Faculty of Science and Technology Queensland University of Technology, mars 2009.

[14] J. Cai, M. Peng and S. Ma. *"RL-ART2 Neural Network Based Mobile Robot Path Planning"*, Proceedings of the IEEE Sixth International Conference on Intelligent System Design and Applications, vol. 2, pages 581-585, Oct 2006 .

[15] J. Chestnutt and M. Lau,. *"Footstep Planning for the Honda ASIMO Humanoid"*, IEEE International Conference on Robotics and Automation, pages 629-634, Apr 2005.

[16] Jean-Claude Latombe. *"Robot motion planning"*, Stanford University,1990.

[17] J.H. Holland. P3-DX. *"Adaptation in Natural and Artificial Systems"*, University of Michigan Press, Ann Arbor, 1975.

[18] J. P. Laumond. *"La robotique mobile"*, Hermès Sience Publications, 2001, Paris, ISBN 2-7462-0246-8.

[19] Lin, H., Xiao, J., Michalewicz, Z. *"Evolutionary navigator for a mobile robot."*, IEEE Conference on Robotics and Automation, vol. 3, pp. 2199–2204. San Diego, USA, May 1994.

[20] M. L. Lamali. *"Approches évolutives pour la classification non supervisée de données"*,memoire de fin d'études ,ESI, Alger 2009.

[21] M. Park and M. Lee. *"Real-time path planning in unknown environment and a virtual hill concept to escape local minima"*, IEEE Conference on Industrial Electronics Society, vol 3, pages 2223-2228, Nov 2004.

[22] M. Tarokh. *"Hybird Intelligent Path Planning for Articulated Rovers in Rough Terrain"*,Fuzzy Sets and Systems, vol 159, pages 2927 – 2937, Nov 2008.

[23] M. T. Jones. *"AI Application Programming"*, David Pallai editor, 2003.

[24] N. Ying and L. Eicher. *"Real-time 3D path planning for sensor-based underwater robotics vehicles in unknown environment"*, OCEANS 2000 MTS/IEEE Conference and Exhibition, vol 3, pages 2051-2058, Sept 2000.

[25] O. KAZAR and L. FRECON. *"MANUEL D'INTELLIGENCE ARTIFICIELLE"*, METIS Lyon Tech, 2009, p. 709.

[26] Osamu Takahashi, R.J. Schilling. *"Motion planning in plane using generalized Voronoi diagrams"*, IEEE Robotic and Automation,Avril 1989.

[27] P .Raja, S ,Pugazhenthi. *"Path planning for a mobile robot in dynamic environments"*, International Journal of the Physical Sciences, vol 6, 4721-4731,2011.

[28] R. Francois-Gérard and A. Souquet. *"Algorithmes genetiques"*, 2004.

[29] R. Leardi. *"Nature-inspired Methods in Chemometrics : Genetic Algorithms and Artificial Neural Networks"*, ELSEVIER B.V., 2003, p. 402.

[30] R. L. Haupt and S. E. Haupt. *"PRACTICAL GENETIC ALGORITHMS"*, Published by John Wiley Sons, Inc., Hoboken, New Jersey, 2004, p. 251.

[31] R. Smierzchalski and Z. Michalewicz. *"Adaptive modeling of a ship trajectory in collision situations at sea"*, 1998 IEEE Conf. Evolutionary Computation, pp. 342-347, 1998.

[32] R. Smierzchalski, and Z. Michalewicz. *"Modeling of ship trajectory in collision situations by an evolutionary algorithm,"*,IEEE Trans Evolutionary Computation, vol. 4, pp. 227–241, 2000.

[33] R. Smierzchalski and Z. Michalewicz. *"Path planning in dynamic environments"*in Innovations in Machine Intelligence and Robot Perception, S. Patnaik, L. C. Jain, G. Tzafestas and V. Bannore, Eds, Berlin : Springer-Verlag, 2005.

[34] S.N.Sivanandam and S.N.Deepa. *"Introduction to Genetic Algorithms"*,Springer-Verlag Berlin Heidelberg, 2008, p. 453.

[35] Thierry CHATROUX. *"Algorithmes génétiques parallèles pour la planification de trajectoires de robots en environnement dynamique"*, l'institut d'Informatique et de Mathématiques Appliqu.es de Grenoble (I.M.A.G.) 1993.

[36] T. Kanade, C. Thorpe et W. Whittaker. *"Autonomous Land Vehicle Project at CMU"*,In ACM Annual Computer Science Conference, pages 71–80. ACM Press, 1986.

[37] Trojanowski.K Michalewicz. Z Xiao. J. *"Adding memory to the evolutionary planner/navigator."*, IEEE Conference on Evolutionary Computation, pp. 483–487. Indianapolis, USA, April 1997.

[38] T. Shibata and T. Fukuda. *"Intelligent motion planning by genetic algorithm with fuzzy critic"*, Proc. of the 8th IEEE Int. Symp. on Intelligent Control, Chicago, Illinois (August, 1993) pp. 565–569.

[39] T. Zheng and X. Zhao. *"A Novel Approach for Multiple Mobile Robot Path Planning in Dynamic Unknown Environment"*, IEEE Conference on Robotics, Automation and Mechatronics, pages 1-5, Dec 2006 .

[40] Xiao, J., et al. *" Adaptive evolutionary planner/navigator for mobile robots"*, IEEE Trans. Evol. Comput. 1, 18–28 (1997).

[41] Xiao, J., Michalewicz, Z, Zhang, L. *" Evolutionary planner/navigator : operator performance and self- tuning"*, IEEE Conference on Evolutionary Computation, pp. 366–371. Nagoya, Japan, May 1996.

[42] Y. Chang, Y. Yamamoto. *"Online Path Planning Strategy Integrated with Collision and Dead-lock Avoidance Schemes for Wheeled Mobile Robot in Indoor Environment"*,Industrial Robot-An International Journal, vol 35, pages 421 – 434, 2008.

[43] Y. Davidor. *"Genetic algorithms and robotics : a heuristic strategy for optimization"*, Singapore : World Scientific Publishing, vol. 1, pages 220-225, 1991.

[44] Yongxing Hao. *"A Practical Framework for Formation Planning and Control Of Multiple Unmanned Ground Vehicles"*, Doctrat of Philosophy in Mechanical Engineering, University of Delaware, 2004.

[45] Y.Wang, D .Mulvaney, I.Sillitoe. (2006) *"Genetic-based Mobile Robot Path Planning using Vertex Heuristics"*, IEEE International Conference on Robotics and Automation 2006.

[46] Y.Wang, D .Mulvaney, I.Sillitoe., Swere, E. (2008). *"Robot Navigation by Waypoints"*, Springer Science + Business Media B.V. 2008, pp. 175-207.

[47] Y.Wang, D .Mulvaney, I.Sillitoe. *"Mobile robot path planning in dynamic environments"*, in Proceedings of the International Conference on Robotics and Automation, vol. 44. Roma : IEEE, Apr. 10–14, 2007, pp. 71–76.

[48] Z. Cai and Z. Peng. *"The application of a novel encoding mechanism in path planning for a mobile robot"*, Robot, vol. 23, pp. 230–233, 1997.

[49] C. Qixin, H. Yanwen, and Z. Jingliang. *"The application of a novel encoding mechanism in path planning for a mobile robot"*,IEEE International Conference on Intelligent Robots and Systems, 2006.

[50] D. Gallardo, O. Colomina, F. Flórez, and R. Rizo. *"A Genetic Algorithm for Robust Motion Planning"*, IEEE, pp. 1-8, 1998.

[51] H. Miao. *"A Multi-Operator Based Simulated Annealing Approach For Robot Navigation in Uncertain Environments"*, Journal of Computer Science, no. 4, pp. 50-61, 2009.

[52] P. Tang, Q. Zhang, Y. Yang. *"Studying on path planning and dynamic obstacle avoiding of soccer robot"*, Proceedings of the 3rd World Congress on Intelligent Control and Automation, vol. 2, pages 1244- 1247, July 2000.

[53] P. Zhang, T. Lu. *"Soccer Robot Path Planning based on the Artificial Potential Field Approach with Simulated Annealing"*, Robotica, vol 22, pages 563 – 566, Oct 2004.

[54] Azariadis PN, Aspragathosc NA . *"Obstacle representation by Bump-surfaces for optimal motion-planning"*, Robot. Auton. Syst., 51 : 129–150 (2005).

[55] Chakravarthy A, Ghose D . *"Obstacle avoidance in a dynamic environment : a collision cone approach"*, IEEE T. Syst. Man Cy. A., 28(5) : 562-574 (1998).

[56] Chazelle B. *"Approximation and decomposition of shapes"*, Adv. Robotics. I : Algorithm. Geom. Aspects Robotics. Schwartz JT, Yap CK, Eds. Hillsdale, NJ : Lawrence Erlbaum, pp. 145–185 (1987).

[57] Luh GC, Liu WW . *"Motion planning for mobile robots in dynamic environments using a potential field immune network"*, IMechE J. Syst. Control Eng., 221 : 1033-1046 (2007).

[58] Ge SS, Cui YJ . *" Dynamic motion planning for mobile robots using potential field method"*, Auton. Robot, 13 : 207-222 (2002).

[59] Borenstein J, Koren Y . *" Real-time obstacle avoidance for fast mobile robots"*, IEEE T. Syst. Man Cy., 19(5) : 1179-1187 (1989).