**PROJET TUTOTRE**

**Université Paris 8**

**Systèmes Robotiques: génération de trajectoires, suivi de trajectoire, évitements d'obstacles**

**2015/2016**

**Abdelbasset MARZAQ Encadrée par : Mme Lynda SEDDIKI**

**Abdelhakim MOURIK**

**TABLES DES MATIERES**

[1. Introduction 3](#_Toc440461063)

[1.1 Robotique : 3](#_Toc440461064)

[1.2 Un bref aperçu historique 3](#_Toc440461065)

[1.3 Robots mobiles autonome 5](#_Toc440461066)

[1.4 Application des robots mobiles 5](#_Toc440461067)

[1.5 Problèmes en robotique mobile 6](#_Toc440461068)

[1.6 Composants d'un robot mobile 6](#_Toc440461069)

[1.7 L’évolution du robot 6](#_Toc440461070)

[1.8 Les robots d’aujourd’hui 7](#_Toc440461071)

[2. Méthodes de génération de trajectoire 7](#_Toc440461072)

[2.1 Problématique générale 7](#_Toc440461073)

[2.2 Les méthodes Probabilistes 7](#_Toc440461074)

[2.3 Les méthodes déterministes : 8](#_Toc440461075)

[3. Suivi de trajectoire 9](#_Toc440461076)

[4. Bibliographie 9](#_Toc440461077)

**Table de figures**

[Figure 1.Schéma des interactions d’un robot avec son environnement 4](#_Toc440875739)

[Figure 2. La tortue de Grey Walter : ELSIE 4](#_Toc440875740)

[Figure 3. John Hopkins University : BEAST 5](#_Toc440875741)

[Figure 4. Shakey Stanford Research Institute 5](#_Toc440875742)

[Figure 5.CartStanford 5](#_Toc440875743)

[Figure 6. GenghisRobotique Réactive 6](#_Toc440875744)

[Figure 7. 6](#_Toc440875745)

[Figure 8. Robot humanoïde ‘ASIMO’ 8](#_Toc440875746)

[Figure 9. Robot humanoïde ‘NAO’ 9](#_Toc440875747)

[Figure 10. Principe de fonctionnement des méthodes de l’échantillonnage 10](#_Toc440875748)

[Figure 11. Evolution d'un arbre couvrant l'espace libre par la méthode RRT. 10](#_Toc440875749)

[Figure 12. Arbre de diffusion aléatoire, opération d’extension 11](#_Toc440875750)

[Figure 13. Calcul d’un chemin entre deux configurations « méthode de champs de potentiels » 12](#_Toc440875751)

[Figure 14. Diagramme de Voronoï 13](#_Toc440875752)

[Figure 15.. Algorithme DIJKSTRA 14](#_Toc440875753)

# 1. Introduction

## 1.1 Robotique :

La robotique mobile autonome vise plus spécifiquement à concevoir des systèmes capables de se déplacer de façon autonome. Les applications directes se situent notamment dans les domaines de l’automobile, de l’exploration planétaire ou de la robotique de service par exemple. 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 qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a. Ainsi, le robot devrait être capable d'effectuer des tâches diverses de plusieurs manières et accomplir correctement sa tâche même s'il rencontre de nouvelle situation inattendues.

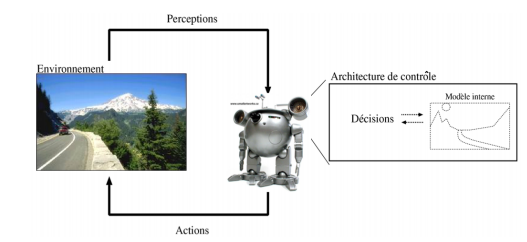


Figure 1.Schéma des interactions d’un robot avec son environnement

Cette définition s’illustre par un schéma classique des interactions d’un robot avec son environnement (Figure 1.1). Ces différentes interactions correspondent au cycle Perception / Décision / Action. La manière dont un robot gère ces différents éléments est définie par son architecture de contrôle, qui peut éventuellement faire appel à un modèle interne de l’environnement pour lui permettre alors de planifier ses actions à long terme.

## 1.2 Un bref aperçu historique

Le terme de robot apparaît pour la première fois dans une pièce de Karel Capek en 1920 : Rossum's Universal Robots. Il vient du tchèque 'robota' (servitude) et présente unevision des robots comme serviteurs dociles et efficaces pour réaliser les taches pénibles mais qui déjà vont se rebeller contre leurs créateurs.



Figure 2. La tortue de Grey Walter : ELSIE

La Tortue construite par GreyWalter dans les années 1950, est l’un des tout premiers robots mobiles autonomes.Grey Walter n'utilise que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours sujets de recherche pour les rendre de plus en plus génériques.

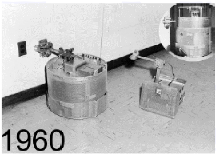


Figure 3. John Hopkins University : BEAST

Dans les années 60, les recherches en électronique vont conduire, avec l’apparition du transistor, à des robots plus complexes mais qui vont réaliser des tâches similaires. Ainsi le robot "Beast" (Figure 1.3) de l’université John Hopkins est capable de se déplacer au centre des couloirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blancs) en utilisant des photo-diodes et de s’y recharger.

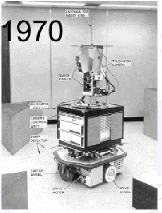


Figure 4. Shakey Stanford Research Institute

Les premier liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey. Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle, qui à l’époque travaille essentiellement sur des approches symboliques de la planification. La perception de l’environnement, qui à l’époque est considérée comme un problème séparé, voire secondaire, se révèle particulièrement complexe et conduit là aussi à de fortes contraintes sur l’environnement.



Figure 5.CartStanford

Ces développements de poursuivent avec le StanfordCart dans les années 1980, avec notamment les premières utilisations de la stéréo-vision pour la détection d’obstacles et la modélisationde l’environnement.



Figure 6. GenghisRobotique Réactive

Une étape importante est à signaler au début des années 1990 avec l’apparition de la robotique réactive, représentée notamment par Rodney Brooks. Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots, beaucoup plus réactifs et adaptés à leur environnement. Ces robots n’utilisent pas ou peu de modélisation du monde, problématique qui s’est avérée être extrêmement complexe. Ces développements ont continué depuis et l’arrivée sur le marché à partir des années 1990 de plates-formes intégrées a permis à de très nombreux laboratoires de travailler sur la robotique mobile et à conduit à une explosion de la diversité des thèmes de recherche. Ainsi, même si les problèmes de déplacement dans l’espace restent difficiles et cruciaux, des laboratoires ont pu par exemple travailler sur des approches multi-robot, la problématique de l’apprentissage ou sur les problèmes d’interactions entre les hommes et les robots.

## 1.3 Robots mobiles autonome

Il existe différents type de systèmes:

* **Machine télécommandé** Seul l'opérateur commande et assure la réalisation de l'opération.



* **Machine télé opéré,** l'opérateur assure la décision en utilisant les perceptions provenant de la machine.

****

* **Robot mobile,** autonome ou semi autonome, l'operateur peut intervenir dans la décision

****

Figure 7.

## 1.4Application des robots mobiles

Sous la pression des forces économiques, il y a 3 grands domaines dans lesquels les robots sont utiles, voire indispensable.

* **La production**

Ses critères essentiels sont : l’automatisation, la rapidité de reconfiguration, la flexibilité, l’apprentissage. L’environnement peut agir sur la gestuelle des robots ou être contraint pour faciliter la commande.

* **L’exploration**

Dans le sens le plus large, il s’agit de faire exécuter au robot des tâches dans les zones auxquelles l’homme ne peut pas accéder en raison du danger comme « les incendies, le nucléaire et déminage » ou de l’éloignement comme « les fonds marins, spatial ».

* **L’aide individuelle**

Le robot est un outil, un assistant pour les taches pénibles, ennuyeuses, dangereuses, il décuple la force, augmente la précision, agit à distance comme en chirurgie. Des systèmes exosquelettes, prothèses, bras sur fauteuil roulant sont les aides au handicap.

## 1.5Problèmes en robotique mobile

On distingue un certain nombre de problèmes en robotique mobile. Bien évidemment, l’aspect matériel, qui consiste a choisir et dimensionner aussi bien la structure mécanique du système que sa motorisation, son alimentation et l’architecture informatique de son système de contrôle-commande apparait comme le premier point a traiter. Le choix de la structure est souvent effectue parmi un panel de solutions connues et pour lesquelles on a déjà résolu les problèmes de modélisation, planification et commande. Le choix des actionneurs et de leur alimentation est généralement assez traditionnel. La plupart des robots mobiles sont ainsi actionnes par des moteurs électriques a courant continu avec ou sans collecteur, alimentes par des convertisseurs de puissance fonctionnant sur batterie. De la même fac ̧on, les architectures de contrôle-commande des robots mobiles ne sont pas différentes de celles des systèmes automatiques ou robotiques plus classiques. On y distingue cependant, dans le cas général, deux niveau de spécialisation, propres aux systèmes autonomes : une couche décisionnelle, qui a en charge la planification et la gestion (séquentielle, temporelle) des évènements et une couche fonctionnelle, chargée de la génération en temps réel des commandes des actionneurs. Bien évidemment, l’architecture du robot dépend fortement de l’offre et des choix technologiques du moment. Pour plus de renseignements sur la technologie des robots mobiles, on pourra avec profit examiner l’ouvrage de Jones, Flynn et Seiger Jones 99], qui est à la fois un manuel élémentaire de robotique et un guide pratique de l’apprenti bricoleur.

## 1.6 Composants d'un robot mobile

Un robot mobile est constitué de composantes matérielles et logicielles. Parmi les composantes matérielles, on retrouve une plateforme mobile à laquelle sont rattachées toutes les autres composantes comme les capteurs, les actionneurs et une source d’énergie.

## 1.7 L’évolution du robot

Le degré d’évolution d’un robot est directement lié à l’information introduite dans son cerveau artificiel. Cette introduction constitue la phase d’apprentissage. A partir de Cela on peut diviser les robots en 2 groupes :

* Ceux qui, une fois la phase d’apprentissage terminée, accomplissent les tâches sans avoir recours à des informations extérieures. Ils sont aveugles et ont un comportement en boucle ouverte par rapport à leur environnement. Tout est connu d’avance, les robots industriels apprennent une suite de gestes ou trajectoires qu’ils reproduisent toujours dans le même ordre. Les seuls capteurs d’environnement sont ceux liés à la sécurité ou à la synchronisation avec d’autres machines. Ces systèmes fonctionnent d’une manière à ce qu’ils excluent la moindre adaptation aux modifications de l’environnement. Ce sont des manipulateurs dépourvu de tout sens.
* Ceux qui, après la phase d’apprentissage tiennent compte de l’environnement et s’adaptent. Les tâches sont effectuées en mode interactif entre le robot et son environnement. Le rebot doit extraire à chaque instant les paramètres réels de la tâche, les comparer aux paramètres désirés et se piloter avec les valeurs issues de cette comparaison. Ce sont ces machines que l’on peut nommer robots. C’est le début de l’intelligence artificielle.

## 1.8 Les robots d’aujourd’hui

Ils courent, marchent, volent, nagent, parlent, nous imitent, et tentent de nous comprendre. Ils sont minuscules, gigantesques, anthropoïdes ou informes, et parfois mous. Les robots sont de plus en plus présents dans les sociétés et commencent à intégrer la plupart des secteurs d’activités.

**Les robots humanoïdes**

Un robot humanoïde ou [androïde](https://fr.wikipedia.org/wiki/Andro%C3%AFde) est un [robot](https://fr.wikipedia.org/wiki/Robot) dont l'apparence générale rappelle celle d'un corps humain[1](https://fr.wikipedia.org/wiki/Humano%C3%AFde#cite_note-1). Généralement, les robots humanoïdes ont un torse avec une tête, deux bras et deux jambes, bien que certains modèles ne représentent qu'une partie du corps, par exemple à partir de la taille. Certains robots humanoïdes peuvent avoir un « visage », avec des « yeux » et une « bouche ».

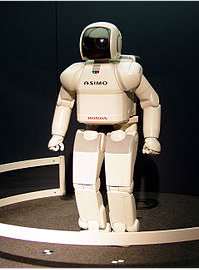


Figure 8. Robot humanoïde ‘ASIMO’

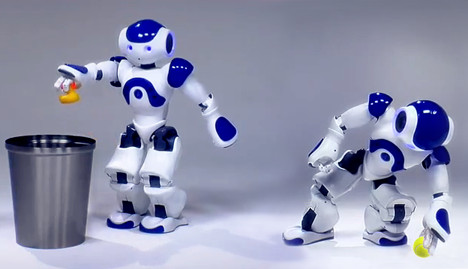
[](javascript:)

Figure 9. Robot humanoïde ‘NAO’

# 2. Méthodes de génération de trajectoire

## 2.1 Problématique générale

La génération de trajectoire est un important problème de recherche en intelligence artificielle qui consiste à calculer une trajectoire géométrique d’un état initial à une configuration finale, pour un objet mobile donné.

Le problème de la planification de trajectoire est généralement formulé de la manière suivante : on considère un robot mobile A se déplaçant dans un espace de travail W, l’objectif est de trouver les chemins qui relient la position du départ du robot Pi à sa position finale Pf.

Comment peut-on programmer des mouvements précis sur un robot ? Comment peut-on planifier un mouvement et suivre une trajectoire ? Comment un robot peut-il éviter des obstacles ?

Pour répondre à toutes ces questions on doit effectuer des recherches sur les méthodes qui permettent de planifier un mouvement ou suivre une trajectoire. Ces méthodes sont des algorithmes précis basés sur les mathématiques et l’informatique.

On peut distinguer 2 catégories différentes de ces méthodes :

## 2.2 Les méthodes Probabilistes

Ce sont des méthodes qui permettent toujours de trouver un chemin s’il en existe mais ne trouve pas toujours le même chemin à chaque exécution contrairement aux méthodes déterministes.

Elles disposent pour leur recherche d’une représentation globale du monde. Ces méthodes nécessitent la construction d’un espace de recherche qui permet l’obtention implicite d’un graphe valué. A l’aide de ce graphe, une recherche heuristique permet de trouver le trajet qui minimise les fonctions de coût utilisées.

Les méthodes de planification probabiliste. Résoudre un problème de planification de mouvement consiste à explorer l’espace de recherche afin de trouver une solution. La spécificité des méthodes probabilistes peut se résumer à un parcours aléatoire de l’espace de recherche, réduisant ainsi la complexité de la résolution.

On distingue 2 méthodes principales dans les méthodes Probabilistes :

* **les méthodes de l’échantillonnage**

Ce sont des méthodes basées sur l’approche le Probabilisticroadmap « PRM ». Elles consistent à capturer la connexité de l’espace libre par un graphe (ou un arbre) appelé réseau construit à partir de configurations choisies aléatoirement dans l’espace des configurations. Une méthode locale permet de tester l’existence d’un chemin faisable entre deux nœuds donnés du graphe et ainsi créer des arcs entre les nœuds. Elle définit un chemin d’un robot entre ces deux configurations.

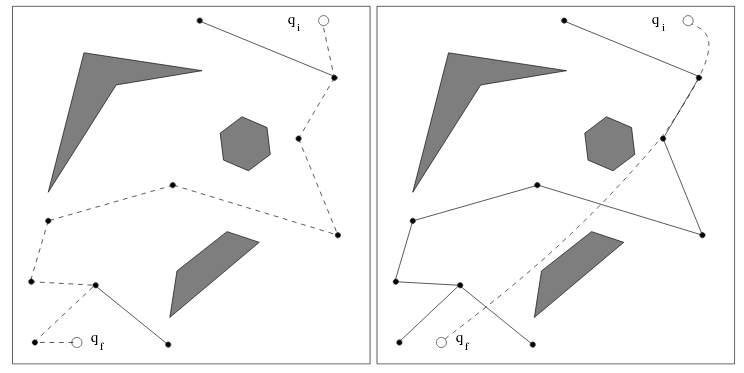


Figure 10. Principe de fonctionnement des méthodes de l’échantillonnage

La figure de gauche représente deux configurations « Qi et Qf » qu’on veut relier par un chemin faisable s’il existe. L’algorithme réussit à connecter ces configurations au réseau et trouver ainsi un chemin faisable les reliant. Une procédure d’optimisation permet de lisser le chemin obtenu, figure de droite.

* **Les méthodes de diffusion**

Ces méthodes qui sont devenues très populaires dernièrement, utilisent une stratégie de diffusion basée sur des algorithmes tels que « RRT » (Rapidly Exploring Random Tree) qui se base sur la densité d'échantillonnage. Le nœud à étendre est choisi aléatoirement avec une probabilité inversement proportionnelle à la densité du graphe (nombre de nœuds situés dans un voisinage).

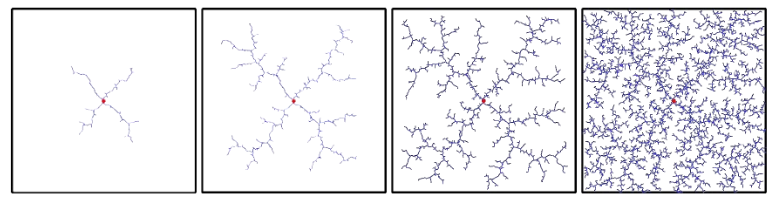


Figure 11. Evolution d'un arbre couvrant l'espace libre par la méthode RRT.

Les algorithmes RRT se composent d'une phase de construction et d'une phase de connexion. Une configuration est aléatoirement échantillonnée. La droite passant par cette configuration et le nœud de l'arbre le plus proche déterminant le nœud et la direction d'extension de l'arbre (voir Figure10.)

Pour comprendre le principe des méthodes de diffusion aléatoire, On suppose un arbre de racine « Q start » dont tous les nœuds sont dans l'espace libre de l’espace configuration « *CS* » (c'est-à-dire CS privé des volumes représentant les obstacles).

- On tire tout d'abord un point « Q rand » de *CS* au hasard suivant une certaine loi de probabilité.

- On cherche parmi les nœuds de l'arbre le point le plus proche de q rand pour la distance dont on a munit *CS* ; on note ce point « Q near » .

- On calcule alors le plus court chemin entre « Q near » et « Q rand » grâce à une méthode appelée «Steering Method».

- On effectue l'opération d'extension : on parcourt ce chemin en partant de « Qnear » et on s'arrête dès que l'on rencontre un obstacle. Ce point limite est noté« Q new » (voirefigure ci-dessous).

- On ajoute « Q new » à l'arbre en créant une arête entre « Qnear » et « Q new »

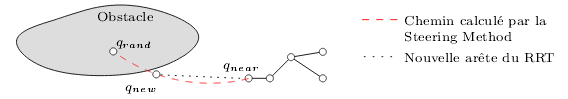


Figure 12. Arbre de diffusion aléatoire, opération d’extension

On parcourt le chemin de « Qnear » à « Q rand ». Si on rencontre un obstacle, « Q new » est le point limite ne créant pas de collision. Sinon, « Q new » est « q rand ».

## 2.3 Les méthodes déterministes :

Les méthodes précédentes se basent toutes sur la structuration de l’espace des configurations et la modélisation a priori de sa connectivité.

D'autre méthodes existent leur principe consiste à déterminer les déplacements du robot en ne considérant qu’une représentation locale de l’environnement et à percevoir la planification de mouvement comme un problème d’optimisation.

Parmi ces méthodes on trouve :

* **champs de potentiel**

Ce sont des méthodes basées sur des stratégies réactives en suivant le principe « Perception / Action ». Le robot est soumis à un champ de forces répulsives et attractives.

Un obstacle génère un champ de potentiel répulsif tandis que l'objectif à atteindre génère un champ de potentiel attractif. L'algorithme calcul donc un vecteur résultant qui indiquera au robot comment effectuer son déplacement.

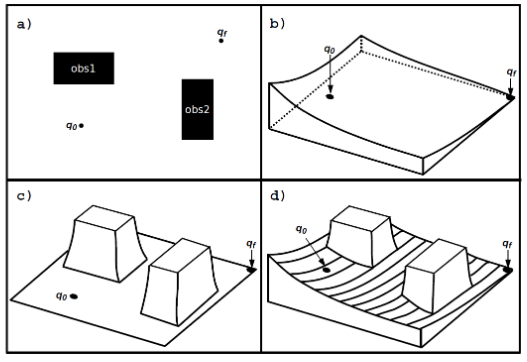


Figure 13. Calcul d’un chemin entre deux configurations « méthode de champs de potentiels »

(a) Espace de configuration du robot et représentation des obstacles obs 1 et obs 2 dans celui-ci.

(b) Champ attractif généré par la position finale.

(c) Champ répulsif exercé par les obstacles.

(d) Combinaison des deux.

* **Méthode des contraintes**

A la différence de la méthode du potentiel, les obstacles n’agissent sur le robot pendant le processus de minimisation que quand il en est très proche et a tendance à y pénétrer.

* **Décomposition cellulaire**

L’espace de configuration libre est décomposé en un ensemble de cellules dont la connexité est représentée par un graphe d’adjacence. Les cellules sont en contact entre elles mais ne sechevauchent pas.

A l’étape de requête et après avoir identifié les cellules contenant q i « position initiale » et q f « position finale », une recherche dans le graphe permet de trouver, si elle existe, une séquence de cellules adjacentes connectant la cellule initiale à la cellule finale. Si une telle séquence est produite elle est alors transformée en un chemin.

Elle consiste à partitionner l’espace des configurations libres du robot en un ensemble de régions connexes adjacentes. La description de la décomposition obtenue est alors capturée dans un graphe de connectivité dont les nœuds correspondent aux différentes cellules et les arcs aux relations d’adjacence entre elles. Le problème de planifier un mouvement entre deux configurations situées initialement dans deux cellules différentes est résolu en deux étapes :

a. exploration du graphe de connectivité et détermination d’un chemin reliant les cellules contenant les deux configurations initiales du graphe ;

b. recherche de la solution au problème de planification à partir de l’enveloppe définie par la liste de cellules adjacentes trouvée en a.

Le diagramme de **Voronoï** est l’un des exemples de la décomposition cellulaire, il s’agit d’une décomposition d’un espace ou d’un plan à des cellules à partir d’un ensemble de points appelés « germes », Il permet de construire un réseau comportant des polygones convexes. Il est utilisé dans les calculs de trajectoires des robots mobiles. Le calcul du diagramme de Voronoï est un des problèmes célèbres de la géométrie algorithmique. Son intérêt s’explique par la remarquable diversité de ses propriétés. Il permet notamment de résoudre en temps optimal d’autres problèmes comme le calcul des plus proches voisins. Pour un ensemble de points, le diagramme de Voronoï est formé de segments de droite. Pour un ensemble d’obstacles, le diagramme de Voronoï est formé de segments de droites et d’arcs de paraboles.

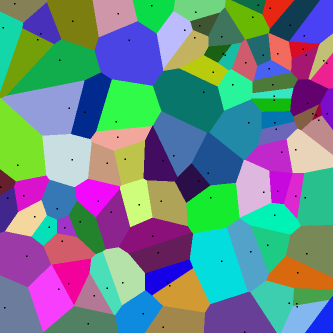


Figure 14. Diagramme de Voronoï

La figure 2.1 représente les cellules « surfaces colorées » et chaque cellule représente la « zone d'influence » d'un germe « les points noirs ».

Le diagramme de Voronoi permet de capturer la connectivité de l’espace libre et générer des solutions éloignées le plus des obstacles. Une telle méthode, bien qu’elle soit générale en théorie, reste limitée à des espaces de dimension peu élevée. Une autre variante de cette méthode est développée par Brooks pour des espaces de travail polygonaux et se base sur la représentation de ceux-ci par des cylindres généralisés.

Les chemins à suivre par le robot sont alors déterminés en considérant la connectivité entre les axes de ces cylindres.

# 3. Suivi de trajectoire

Le suivi de trajectoire consiste à asservir la configuration du robot sur une trajectoire de référence. L’évitement réactif d’obstacles, doit assurer que la trajectoire est sans collision et la déformer sur un intervalle quand une collision est détectée.L’évitement d’obstacles peut amener le suivi de trajectoire à ralentir, voire même à s’arrêter sur la trajectoire si la collision ne peut être éliminée.

Le problème de suivi d’une trajectoire de référence pour un robot mobile est apparu comme un problème de premier ordre pour la communauté roboticienne dans ces dernières années. En effet, la forte utilisation des robots mobiles dans les domaines où l’être humain ne peut pas être présent, notamment dans les sites nucléaires à haut risque ou dans le cas de l’exploration spatiale, nécessite la mise en œuvre de lois de commande autonomes et performantes pour assurer les tâches assignées aux robots.

## 3.1 Algorithme de Dijkstra

Le principe de cet algorithme est de diffuser le long des nœuds du graphe une fonction de coût qui évalue le coût du déplacement d’un nœud vers le suivant. Si cette fonction est, par exemple, la distance entre deux nœuds et que la diffusion commence à partir du nœud de départ, le chemin le plus court pour atteindre tous les nœuds du graphe peut être déterminé.

L’algorithme de Dijkstra[P.Chrétienne, 1994] n’est valable que pour les graphes à valuation positive ou nulle, qui ne contiennent pas de circuits négatifs. A chaque itération, un sommet x reçoit son étiquette définitive, on dit qu’il est fixé. L’itération principale sélectionne le sommet x d’étiquette minimale parmi ceux déjà atteints par un chemin provisoire d’origine s. Pour tout successeur y de x, on regarde si le chemin passant par x améliore le chemin déjà trouvé de s à y : si oui on remplace V [y] par min(V [y], V [x] + W(x, y)) et on mémorise qu’on parvient en y via x en posant P[y] := x. Si tout les sommets sont accessibles au départ de s, l’algorithme se déroule en N itérations. En pratique, des sommets peuvent ne pas être accessibles. Si on veut calculer seulement un plus court chemin de s vers un 11 autre sommet t (problème B), il suffit d’arrêter l’algorithme dés que t est fermé, par exemple en modifiant le test de fin en "jusqu’à (V M in = +∞) ou x = t".

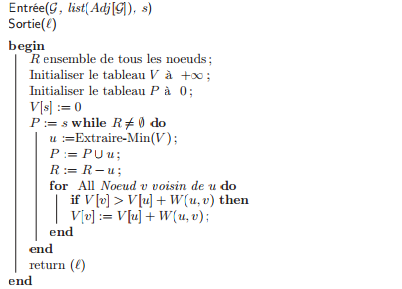


Figure .. Algorithme DIJKSTRA

L’inconvénient majeur de l’algorithme de Dijkstra est qu’il est insensible à la densité du graphe. Le nombre d’itération de la boucle While, au plus N, ne peut pas être amélioré par construction de l’algorithme. En revanche l’essentiel du travail est dû à la boucle interne trouvant le prochain sommet i à fixer.

## 3.2 Algorithme de Sedgewick et Vitter

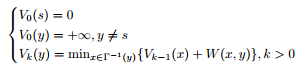
Cet algorithme a été conçu pour les graphes non orientés dont les sommets sont des points d’un espace euclidien, et les arrêtes des segments entre les points valués par la longueur euclidienne de segment (distances entre les points). Il est conçu pour le problème A, c’est-à-dire le calcule d’un plus court chemin entre deux sommets s et t. Sa structure générale est similaire à celle de l’algorithme de Dijkstra.

## 3.3 Algorithmes à buckets

Les algorithmes à buckets sont des variantes de l’algorithme de Dijkstra intéressantes quand les coûts des arcs sont entiers et leurs maximum U n’est pas trop grand. On partitionne l’intervalle des valeurs des étiquettes en B intervalles de largeur commune L, numérotés de 0 à B − 1, et on associe à chacun d’eux un ensemble de sommets appelé bucket. Ce système est codable comme un tableau de B listes. Pour trouver le sommet x à fixer dans l’algorithme de Dijkstra à buckets, on cherche d’abord le bucket non vide de plus petit indice k. On balaie ensuite ce dernier pour localiser et extraire le sommet x d’étiquette minimale. Pour chaque successeur y dont on peut améliorer l’étiquette : on cherche le bucket de y, on le balaie pour localiser et enlever y, on modifie l’étiquette de y avec (V [y] := V [x] + W(x, y)), on localise le nouveau bucket de y et enfin, on insère y en tête de ce bucket. Les sommets en pratique sont bien répartis et les listes buckets sont courtes, ce qui donne de très bonnes performances moyennes.

## 3.4 Algorithme de Bellman

Cet algorithme à correction d’etiquettes a été conçu dans les années 50[P.Chrétienne, 1994] par Bellman et Moore. Il est prévu pour des valuations quelconques et peut être adapté pour détecter un circuit de coût négatif. Il s’agit d’une méthode de programmation dynamique, c’est-à-dire d’optimisation récursive, décrite par la relation suivante :



Vk(x) désigne la valeur des plus courts chemins d’au plus k arcs entre le sommet s et le sommet x. Les deux premières relations servent à stopper la récursion. Le sommet s peut être considéré comme un chemin de 0 arc et de coût nul. La troisième relation signifie qu’un chemin optimal de k arcs de s à y s’obtient à partir des chemins optimaux de k − 1 arcs de s vers tout prédécesseur x de y. En effet, tout chemin optimal est formé de portions optimales, sinon on pourrait améliorer le chemin tout entier en remplaçant une portion non optimale par une portion plus courte. La formulation récursive étant peu efficace, on calcule en pratique le tableau V itérativement, pour les valeurs croissantes de k. Nous donnons ci-après un algorithme simple. Les étiquettes en fin d’étape k sont calculées dans un nouveau tableau à partir du V des étiquettes disponibles en début d’étape. Pour tout sommet y, on regarde si V [y] est améliorable en venant d’un prédécesseur de y. En fin d’étape on écrase V par le nouveau tableau et on passe à l’étape suivante. En l’absence de circuit absorbant, on peut se restreindre aux chemin élémentaires pour trouver un plus court chemin de s vers tout autre sommet. Or, un tel chemin n’a pas plus de N − 1 arcs. Les étiquettes sont donc stabilisées en au plus N − 1 itérations. En pratique, elles peuvent se stabiliser plus tôt, et un meilleur test de fin est quand Vk = Vk−1. La complexité est en O(N.M) : il y a au plus N − 1 itérations principales, consistant à consulter les prédécesseurs de tous les sommets, c’est-à-dire les M arcs.

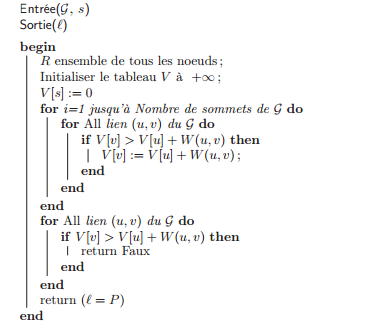


Figure 16. Algorithme de Bellman

## 3.5 Algorithme FIFO

Cet algorithme simple à correction d’étiquettes examine les sommets dans l’ordre FIFO grace à une file Q de sommets. Au début, seul s est dans Q. Une iteration principale traite tous les sommets présents dans Q au début de l’itération. Ensuite l’algorithme balaye les successeurs des sommets et les placé, ceux dont l’étiquettes améliorées, à la fin de Q. L’algorithme se termine quand Q est vide. En fait, l’algorithme FIFO est un dérivé de l’algorithme de Bellman, très intéressant car n’utilisant pas les prédécesseurs.

## 3.6 Algorithme de D’Esopo et Pape

Cet algorithme utilise une pile-file Next, il s’agit d’une file où on peut ajouter un élément à la fin (En Queue) ou en tête (Push). Comme dans FIFO, un sommet atteint la première fois et mis en fin de file. Par contre si on le revisite, on l’insère en tête de file (à condition qu’il ne soit pas déjà en file). Ce critère heuristique de gestion peut s’expliquer intuitivement. Quand on atteint pour la fois un sommet, il n’est pas urgent de développer ses successeurs car les chemins obtenus risquent d’être mauvais dans un premier temps. En revanche un sommet déjà visité et dont l’étiquette vient de diminuer doit être développée en priorité pour propager l’amélioration.

## 3.7 Algorithme de Floyd

L’algorithme de Floyd calcule un distancier N × N donnant les valeurs des plus courts chemins entre tout couple de sommets (problème C). Pour cet algorithme le tableau V des étiquettes devient une matrice N × N, V [i, j] désignant le coût des plus courts chemins de i à j.

## 3.8 Algorithme A∗

L'algorithme A\* ne fait qu’ajouter une fonction heuristique pour n’explorer qu’une partie du graphe utile à la recherche de la solution optimale : classiquement, l’éloignement du nœud exploré à l’objectif sert d’heuristique minorante qui permet de garantir l’optimale au moment de l’arrêt de l’algorithme.

# 4. Bibliographie

thèse histoire robotique

Wiki

Livre : Traité de Robotique 1 les architectures de Charles BOP

Ci-dessous on retrouve les algorithmes qui permettent de modéliser les mouvements du robot selon les méthodes probabilistes :

* Algo du fil d’Ariane : « échantillonnage »