PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Tesis de Maestría
en Informática

Planificación de movimientos

aplicada en robótica autónoma móvil

Facundo Benavides

2012

Facundo Benavides
Planificación de movimientos aplicada en
Robótica autónoma móvil
ISSN 0797-6410
Tesis de Maestría en Informática
Reporte Técnico RT 12-11
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República.
Montevideo, Uruguay, noviembre, 2012

Planificación de movimientos aplicada en robótica autónoma móvil

Tesis de Maestría

Facundo Benavides
fbenavid@fing.edu.uy

Supervisor
Eduardo Grampín

PEDECIBA Informática
Universidad de la República
Montevideo, Uruguay

27 de noviembre de 2012

Resumen

La robótica autónoma móvil se ha desarrollado principalmente para hacer posible la realización de tareas de alto nivel por parte de máquinas sin la necesidad del control humano. A tales efectos, los robots deberían tener la habilidad de moverse apropiadamente en un mundo real. Consecuentemente, planificar sus propios movimientos se vuelve una de los problemas más importantes a ser resuelto en el diseño de robots autónomos móviles.

El presente trabajo sistematiza un relevamiento del estado del arte sobre planificación de movimientos de sistemas robóticos móviles y describe la implementación de un planificador como propuesta de solución a una instancia particular del problema de planificación de movimientos.

El relevamiento incluye la descripción, en orden cronológico, de los métodos más reconocidos y utilizados por la comunidad científica.

En el desarrollo del planificador se emplearon enfoques basados en construcción de *roadmaps*, descomposición aproximada en celdas (descomposición rectangular), muestreo probabilístico de configuraciones y búsqueda basada en algoritmos de naturaleza estocástica.

Particularmente, la propuesta se concentra en la planificación de movimientos para robots móviles -birrodados con control diferencial- que se desempeñan en entornos estáticos bidimensionales a partir de la construcción de un *roadmap* empleando diagramas de *Voronoi*, una grilla regular cuya granularidad preserva la relación entre el tamaño del robot, los obstáculos y área total, el muestreo aleatorio de configuraciones pertenecientes al *roadmap* y algoritmos genéticos para computar caminos "óptimos" libres de colisiones que cumplen con ciertos atributos de calidad.

Palabras clave: planificación de movimientos, robótica autónoma móvil, diagramas de Voronoi, algoritmos genéticos.

Agradecimientos

A Eduardo, Gonzalo y Alfredo por su orientación y aportes.

Al PEDECIBA.

A los amigos.

A mi pequeña Cata, quien ha sido una fuente de alegría, inspiración y energía para concluir este trabajo.

A todos, muchas gracias y **Salud!**

Índice general

Índice de figuras

Índice de cuadros

Parte I

Introducción

Introducción

La robótica autónoma móvil se ha desarrollado principalmente para hacer posible la realización de tareas de alto nivel por parte de máquinas sin la necesidad del control humano. A tales efectos, los robots deberían tener la habilidad de moverse apropiadamente en un mundo real. Consecuentemente, planificar sus propios movimientos se vuelve uno de los problemas más importantes a ser resuelto en el diseño de robots autónomos móviles. Dicha planificación suele involucrar el modelado del entorno, tratamiento de información incompleta, incertidumbre en la lectura de los sensores, evitar obstáculos -posiblemente imprevisibles en cuanto a sus movimientos-, conocer la cinemática de los cuerpos involucrados y manejo de múltiples robots y objetivos. Sin embargo, a pesar de presentar tantos inconvenientes, la planificación de movimientos es una área muy activa de investigación debido a que una correcta resolución puede redundar en ahorro de tiempo, inversiones y la necesidad de supervisión humana, además de servir usualmente como base para el desarrollo de otras habilidades.

Generalmente, la tarea consiste en encontrar un camino óptimo o subóptimo, libre de colisiones, entre una configuración o estado inicial y uno final, siguiendo ciertos criterios para evaluar la optimalidad de los caminos examinados. Algunos de los criterios más utilizados son: largo, seguridad y suavidad del recorrido, ahorro de energía. En tal sentido, y a los efectos de aportar soluciones robustas, flexibles, confiables y computacionalmente eficientes se han propuesto una amplia gama de enfoques que incluyen el uso de métodos tradicionales (*roadmaps*, *cell decomposition* o campos artificiales de potencial), probabilísticos y basados en técnicas de Inteligencia Artificial como: redes neuronales artificiales, aprendizaje por refuerzo, métodos bioinspirados, algoritmos evolutivos, entre otros.

Antecedentes

En Uruguay, el Instituto de Computación de la Facultad de Ingeniería de la UdelaR ha promovido varios emprendimientos en las áreas de Inteligencia Artificial y Robótica. Desde el año 2003 a la fecha, se han dirigido más de diez proyectos de grado. En 2006 se diseñó el primero de cinco curso electivos de grado que se dictan hasta la actualidad. Simultáneamente, desde 2004 se organiza el evento de robótica sumo.uy con el afán de difundir el conocimiento generado hacia la sociedad, dando a conocer el trabajo realizado y brindando la posibilidad de participación en varias competencias robóticas. Con estos antecedentes, en el año 2010 se inició el trabajo de relevamiento del estado del arte en planificación de movimientos de sistemas robóticos móviles. Durante el 2011 se desarrolló el proyecto de investigación que contempló la implementación de un planificador como propuesta de solución a una instancia particular del problema de planificación de movimientos. Ambos constituyen los contenidos principales de la presente tesis de maestría.

Organización

El presente trabajo resume el relevamiento del estado del arte sobre planificación de movimientos aplicada en sistemas robóticos autónomos y describe una propuesta de planificador de movimientos desarrollada por el autor.

El documento está organizado en tres partes: *Conocimientos previos*, *Hacia un sistema de navegación autónomo* y un apéndice *Relevamiento del estado del arte* donde se registra el relevamiento realizado.

La primera parte está dedicada a exponer los principales conocimientos teóricos utilizados para el desarrollo del sistema propuesto. La segunda parte motiva y describe la propuesta desarrollada, los

experimentos y pruebas realizadas, los resultados y principales conclusiones extraídas y, finalmente, aquellos aspectos que se entienden necesarios o interesantes para continuar trabajando.

Finalmente, el apéndice contiene una breve introducción a la Robótica autónoma móvil, donde se mencionan los antecedentes más relevantes y se describen las características más salientes de los agentes robóticos inteligentes. También se plantea formalmente el problema de Planificación de movimientos, donde se describe los principales métodos de resolución -tradicionales, probabilísticos y basados en técnicas de Inteligencia Artificial- desarrollados desde que se comenzó a abordar el problema hasta la actualidad.

Asimismo, en la Bibliografía se presenta una selección de libros y artículos conteniendo las principales referencias bibliográficas del área.

Parte II

Conocimientos previos

En esta parte del documento, con el propósito de brindar un marco teórico que colabore en la comprensión del trabajo realizado, se describen los métodos y técnicas utilizados para el desarrollo del sistema de planificación que se describe en el capítulo 3. Por el contrario, la justificación sobre las elecciones realizadas se encuentra al comienzo del referido capítulo.

Capítulo 1

Planificación de movimientos

1.1. Motivación

La robótica autónoma móvil se ha desarrollado principalmente para hacer posible la realización de tareas de alto nivel por parte de máquinas sin la necesidad del control humano. A tales efectos, los robots deberían tener la habilidad de moverse apropiadamente en un mundo real. Consecuentemente, planificar sus propios movimientos se vuelve uno de los problemas más importantes a ser resuelto en el diseño de robots autónomos móviles.

1.2. Descripción del problema

La planificación de movimientos es uno de los mayores problemas a resolver durante la creación de un robot autónomo. No es el único e interactúa con otros también muy importantes como: control en tiempo real, sensado y planificación de tareas. En tal sentido, crear robots autónomos requiere, definitivamente, dotarlos del poder de decidir automáticamente qué movimientos realizar para lograr sus objetivos o de otro modo, dotarlos de la habilidad de planificar automáticamente sus movimientos.

El objetivo de la planificación de movimientos, en términos generales, es que dada una especificación de una tarea utilizando un lenguaje de alto nivel, el robot sea capaz de descomponerla automáticamente en un conjunto de movimientos primitivos que le permitan cumplir con lo solicitado. Típicamente, esta tarea suele ser la búsqueda de caminos óptimos o cuasi-óptimos, libres de colisiones, entre un estado o configuración inicial y uno final, respetando ciertos criterios de evaluación. Los más frecuentemente utilizados son la distancia recorrida, el ahorro de energía, la suavidad y seguridad de las trayectorias.

1.2.1. Problema básico

El principal motivo para definir una formulación básica, al problema de planificación de movimientos, es lograr aislar algunos aspectos considerados centrales e investigarlos en profundidad, dejando algunas dificultades adicionales para un análisis posterior.

En el problema básico se asume que el robot es el único objeto móvil del entorno y se ignoran, también, los aspectos dinámicos[1] de su movimiento. También se desprecian las interacciones entre los objetos físicos, transformando un problema de planificación de movimientos físicos en un problema

[1] Principalmente, la velocidad y aceleración.

de planificación de caminos puramente geométrico. Además, se reduce la complejidad del aspecto geométrico asumiendo que el robot es un cuerpo rígido no articulado.

De todas estas simplificaciones surge que la formulación para el problema básico de planificación de movimientos es la siguiente:

Sea A un objeto rígido (*"el robot"*) moviéndose en un espacio euclidiano W, llamado área de trabajo o entorno, representado por R^n (n=2 o n=3). Sean O_1, ..., O_q un conjunto de objetos rígidos distribuidos en W, llamados obstáculos. Si se asume que tanto la geometría de A y O_i como la ubicación de O_i en W son conocidas y que no existen restricciones cinemáticas[2] que limiten el movimiento de A, el problema básico de planificación de movimientos es:

Dada una posición y orientación inicial y una posición y orientación objetivo de A en W, generar un camino especificando una secuencia continua de posiciones y orientaciones para A evitando colisionar con O_i, comenzando desde la posición y orientación inicial y terminando en la posición y orientación objetivo.

1.3. Métodos de resolución

Existe un gran número de métodos para resolver el problema de planificación de movimientos. Sin embargo, a pesar de sus diferencias, estos métodos se basan en un número reducido de enfoques que pueden clasificarse en: *roadmap*, *cell decomposition* y *potential field*. Básicamente, estos tres enfoques difieren en los grafos de conectividad que construyen y las representaciones empleadas.

En esta sección se describen, sólamente, los métodos utilizados para el desarrollo del planificador que se describe en el capítulo 3. La figura 1.1 presenta un entorno de trabajo. El mismo servirá para describir el funcionamiento de cada uno de los enfoques utilizados y, luego, para la realización de pruebas del sistema.

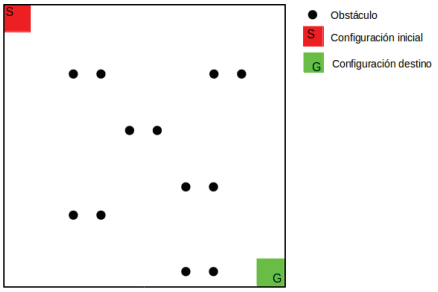


Figura 1.1: Entorno de trabajo.

1.3.1. Roadmaps

Este enfoque consiste en capturar la conectividad del espacio de configuraciones libres y reflejarla en una red de curvas unidimensionales llamada *roadmap*.

El problema de planificación de movimientos se reduce pues a conectar la configuración inicial y final a la red y encontrar un camino libre de colisiones que conecte ambos puntos. El camino resultante, si existe, se compone por tres tramos o subcaminos: un tramo que conecta la configuración inicial al *roadmap*, otro contenido en el *roadmap* y por último uno que conecta el *roadmap* con la configuración destino.

[2] En B.3.3.2 se describen las restricciones que limitan el movimiento de un robot.

1.3.1.1. Espacio de configuración

Para crear planes de movimiento para robots se debe, primero, conocer una especificación completa de la localización de cada punto del robot de modo de asegurar que no se produzcan colisiones.

Configuración q es la especificación de un punto del sistema.

Configuración del robot *A(q) o R(q)* es la especificación completa de la posición de todos los puntos de un sistema robótico.

Espacio de configuraciones C, también conocido como *C-Space*, es el espacio de todas las posibles configuraciones del sistema.

Espacio de configuraciones de un obstáculo CO_i es el conjunto de configuraciones en las cuales el robot intersecta con el obstáculo O_i en el entorno de trabajo W.

$$CO_i = \{q \epsilon C \mid A(q) \cap O_i \neq \varnothing\}$$

Espacio de configuraciones libres C_{free} es el conjunto de las configuraciones en las cuales el robot no intersecta ningún obstáculo presente en el entorno de trabajo.

$$C_{free} = C \setminus \left(\bigcup_i CO_i \right) = \left\{ q \epsilon C \mid A(q) \cap \bigcup_i CO_i = \varnothing \right\}$$

Roadmap Una unión de curvas unidimensionales es un *roadmap RM* si para todo q_{init} y q_{goal} en C_{free} que pueden conectarse por un camino, se cumplen las siguientes condiciones:

1. Accesibilidad: existe un camino desde $q_{init} \epsilon C_{free}$ hacia algún $q'_{init} \epsilon RM$.
2. Salida: existe un camino desde algún $q'_{goal} \epsilon RM$ hacia $q_{goal} \epsilon C_{free}$.
3. Conectividad: existe un camino en RM entre q'_{init} y q'_{goal}.

A partir de estas definiciones se puede reformular el problema de planificación de movimientos como el problema de determinar una correspondencia continua $c : [0,1] \rightarrow C$, o camino, desde una configuración inicial $c(0)=q_{init}$ hasta una configuración destino $c(1)=q_{goal}$, tal que ninguna configuración q_i perteneciente al camino causa una colisión entre el robot y un obstáculo.

Finalmente, se puede definir formalmente un *camino libre de colisiones* como la correspondencia continua $c : [0,1] \rightarrow C_{free}$ y un *camino semi-libre* como la correspondencia continua $c : [0,1] \rightarrow cl(C_{free})$ en la cual $cl(C_{free})$ denota la clausura de C_{free}.

1.3.1.2. Diagramas de Voronoi

Además de ser una de las principales construcciones dentro de la Geometría Computacional, al igual que los grafos de visibilidad y las redes de autopistas, los diagramas de Voronoi son una de las técnicas más usuales para la construcción de *roadmaps*. [Roq96, FGLM01, RD05, WvH07, BG08]

Sea S un conjunto de entidades geométricas en $\mathcal{R}^d$ y una distancia métrica $\|.\|$, el diagrama de Voronoi correspondiente a S es la partición de $\mathcal{R}^d$ en la máxima cantidad de regiones, tales que los puntos pertenecientes a cada región de Voronoi cumplen la condición de estar más cerca de una sola entidad de S que de ninguna otra. [WvH07]

En robótica móvil han sido aplicados considerando, usualmente, el espacio correspondiente a $\mathcal{R}^2$, la distancia euclidiana entre puntos y el principio del vecino más próximo. Así pues, si se considera a $P = \{p_1, p_2, ..., p_n\}$ un conjunto de puntos no alineados en el plano y sea $d(p_i, p_j)$ la distancia

euclidiana entre los puntos p_i y p_j, la región de Voronoi $R(p_i)$ generada por el punto p_i queda definida por la ecuación 1.1:

$$R(p_i) = \left\{ p \in \mathcal{R}^2; d(p, p_i) \leq d(p, p_j), \forall j \neq i \right\} \tag{1.1}$$

En este contexto, a los puntos p_i se les denomina *generadores de Voronoi* y al conjunto de todas las regiones de Voronoi $R(p_1), R(p_2), ..., R(p_n)$, *diagrama de Voronoi* de P. Además, a una frontera común a dos regiones se le denomina *arista de Voronoi* y a un punto de corte entre tres o más aristas, *vértice de Voronoi*.

La figura 1.2 muestra el diagrama de Voronoi correspondiente al entorno de la figura 1.1. En el se pueden apreciar los puntos generadores del diagrama -doce (12) obstáculos (círculos negros)-, las regiones generadas (en diferentes tonos de gris), las aristas que las delimitan (líneas negras) y los vértices.

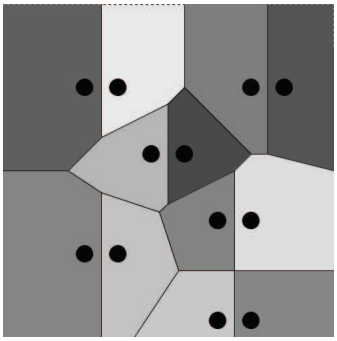


Figura 1.2: Diagrama de Voronoi.

Los diagramas de Voronoi presentan dos propiedades de interés en el contexto de la planificación de movimientos para robots móviles:

1. Toda arista de Voronoi pertenece a la mediatriz del segmento de recta formado por los dos puntos generadores de las regiones que determinan la arista.

2. Todo vértice de Voronoi está ubicado exactamente en el circuncentro del polígono definido por los puntos generadores de las regiones que determinan el vértice.

Estas propiedades señalan la pertinencia de utilizar diagramas de Voronoi para generar *roadmaps* donde se pueden computar caminos de máxima seguridad (entendiendo esta como la máxima distancia hacia los obstáculos). A tales efectos deben resolverse, además de computar un diagrama de Voronoi [3] a partir de la ubicación de los obstáculos, las siguientes tareas:

1. computar el camino más corto sobre *roadmap* generado.

2. elegir puntos de entrada y de salida, desde la configuración origen al *roadmap* y desde el *roadmap* hasta la configuración final, respectivamente.

En la figura 1.3 se resaltan las regiones de Voronoi (correspondientes al diagrama de la figura 1.2) que contienen las configuraciones inicial y destino y un camino posible a recorrer.

[3] Existen algoritmos que lo realizan en tiempo $O(n \lg n)$, siendo n el número de puntos generadores, convirtiéndolos en serios candidatos para su uso en sistemas robóticos. [GbHL05]

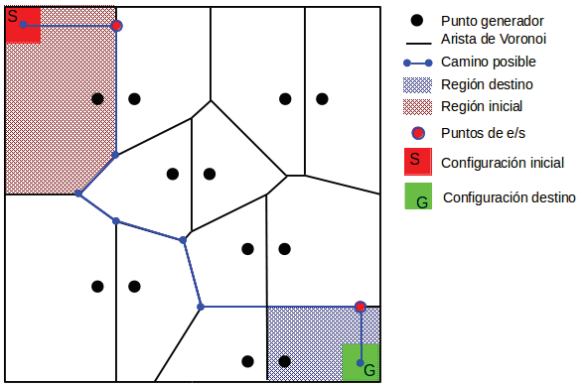


Figura 1.3: Camino de máxima seguridad.

1.3.2. *Cell Decomposition*

Quizás el más vastamente estudiado, este enfoque consiste en descomponer el espacio de configuraciones libres en regiones que no se solapan, llamadas celdas, de forma tal que un camino entre dos configuraciones que pertenezcan a la misma región se pueda computar fácilmente. Luego, se construye un grafo no dirigido donde estén representadas las relaciones de adyacencia entre celdas. Finalmente, se realiza un búsqueda sobre el grafo que da como resultado una secuencia de celdas o *canal* -que conecta las celdas que contienen las configuraciones inicial y final-, a partir del cual se puede computar un camino libre de colisiones.

A su vez, los métodos que realizan descomposición del espacio de configuración libre en regiones pueden clasificarse en exactos[4] o aproximados. Los métodos exactos de descomposición tienen la propiedad de ser completos[5]. Por el contrario, los métodos aproximados no son completos pero son más eficientes y posibilitan el ajuste de la precisión en la descomposición para aproximarse a la completitud. Suelen denominarse *resolution-complete methods*.

1.3.2.1. Métodos aproximados

Los métodos aproximados descomponen el espacio de configuraciones libres en celdas. Estas deben tener una forma simple predefinida y la unión de todas las celdas no coincide exactamente con el espacio descompuesto sino que está incluida estrictamente en él. Luego, se construye un grafo de conectividad que represente las relaciones de adyacencia entre celdas y se realiza una búsqueda de caminos sobre el mismo.

Las razones para estandarizar la forma de las celdas son:

1. realizar la descomposición de forma iterativa aplicando reiteradas veces la misma fórmula
2. tener un algoritmo relativamente insensible a errores numéricos

Generalmente, estos aspectos redundan en planificadores más simples de implementar que los basados en descomposición exacta. Además, con los métodos aproximados es posible mantener controlado el tamaño del espacio de configuración libre en torno a los caminos generados. Sin embargo, los métodos aproximados son incompletos y pueden fallar en la búsqueda de un camino, aunque este exista.

[4] En B.4.2.1 se describen con más detalle.
[5] En B.2 se describen las propiedades de interés en los los algoritmos de resolución.

La mayoría de los métodos aproximados manejan el tamaño de las celdas de forma jerárquica permitiendo adaptar el valor a la geometría de la región, evitando predefinir un tamaño que genere dificultades: de ser demasiado grande podría impedir el hallazgo de caminos libres de colisiones y de ser demasiado pequeño implicaría un incremento del esfuerzo computacional.

En un sentido práctico y debido a que el consumo de espacio de memoria y tiempo de ejecución de los métodos que implementan este enfoque crece rápidamente cuando aumenta la dimensión del espacio, los métodos aproximados son aplicables solamente cuando la dimensión del espacio de configuración es menor o igual a cuatro.

Rectanguloide Designa una región con las siguientes características en el espacio Cartesiano R^n:

$$\left\{(x_{1,\cdots},x_n) \mid x_1 \epsilon \left[x_1^{'}, x_1^{''}\right], \cdots, x_n \epsilon \left[x_n^{'}, x_n^{''}\right]\right\}$$

donde las diferencias $x_i^{'} - x_i^{"}$, $i = 1, \cdots, n$ son las dimensiones del rectanguloide.

Este enfoque asume que el conjunto de posiciones posibles para un robot A, están contenidas en un rectanguloide $D \subset R^n$. El espacio de configuraciones libres C_{free} se representa como: $C_{free} = R \setminus O$, donde O es el espacio de configuraciones de los obstáculos y $R = int\,(D)$. En este caso, $\Omega = cl\,(R)$ es un rectanguloide en R^n.

Descomposición rectangular Una descomposición rectangular P de una región rectanguloide Ω es una colección de rectanguloides $\{k_i\}_{i=1,\cdots,r}$ tal que $\Omega = \cup_{i=1}^{r} k_i$ y $\forall i_1, i_2 \epsilon [1, r], i_1 \neq i_2 : int\,(k_{i_1}) \cap int\,(k_{i_2}) = \oslash$

Cada rectanguloide k_i es una celda de la descomposición P de Ω. Además, toda celda k_i puede ser clasificada en:

- Vacía: *si y solo si* el interior no intersecta el espacio de configuración de obstáculos. $int\,(k_i) \cap O = \oslash$
- Llena: *si y solo si* k_i está totalmente contenida en el espacio de configuración de obstáculos. $k_i \subseteq O$
- Parcial: en otro caso.

En la figura 1.4 se muestran, las configuraciones inicial y final y una descomposición rectangular del entorno correspondiente al entorno de la figura 1.1.

Grafo de conectividad El grafo de conectividad asociado a una descomposición rectangular P de Ω es un grafo no-dirigido G definido de la siguiente forma:

- Los nodos de G son celdas *Vacías* y *Parciales* de P.
- Dos nodos de G están conectados *si y solo si* las celdas asociadas son adyacentes.

Canal Dada una descomposición rectangular P de Ω, un canal es definido como una secuencia $(k_{\alpha j})_{j=1,\cdots,p}$ de celdas *Vacías* o *Parciales* tales que cualquier pareja de celdas consecutivas $k_{\alpha j}$ y $k_{\alpha j+1}$, $j \epsilon [1, p-1]$, son adyacentes.

En un canal en el que todas sus celdas son *Vacías*, cualquier camino que conecte cualquier configuración en $k_{\alpha 1}$ con otra cualquiera en $k_{\alpha p}$ y que pertenezca al interior del canal $(int\left(\cup_{j=1}^{p} k_{\alpha j}\right))$, es un camino libre de colisiones.

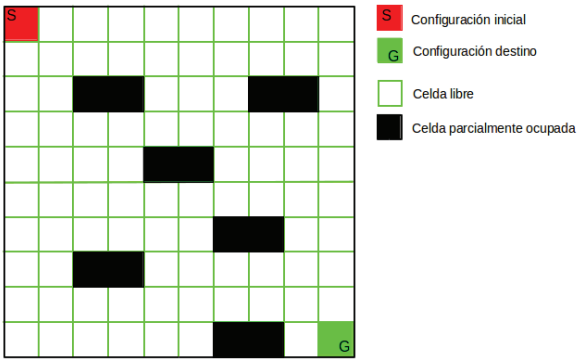


Figura 1.4: Descomposición aproximada. Descomposición rectangular.

En un canal en el que existe al menos una celda de tipo Parcial, puede existir un camino libre de colisiones pero no se puede garantizar su existencia.

La figura 1.5 muestra un camino libre de colisiones extraído de un canal computado a partir de la descomposición rectangular de la figura 1.4. Para el caso de requerirse que el camino posea algún atributo de calidad en particular, deben computarse todos los canales posibles de forma exhaustiva seleccionándose el óptimo según los criterios establecidos.

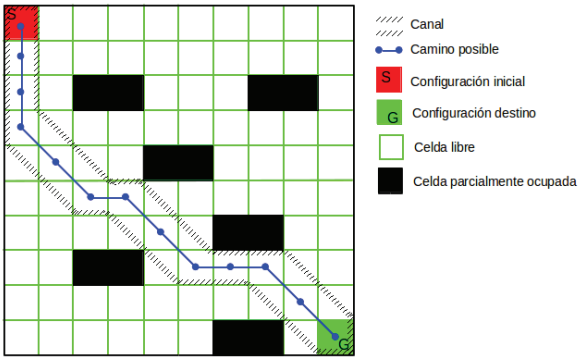


Figura 1.5: Descomposición aproximada. Camino libre de colisiones.

Capítulo 2

Optimización de caminos

Con el propósito de abordar algunas de las extensiones al problema básico de planificación de movimientos o mejorar la eficiencia computacional de los métodos tradicionales o probabilísticos (en ambos casos, basados en una representación completa del entorno), se han realizado variadas propuestas que se basan en métodos o técnicas provenientes del campo de la Inteligencia Artificial.

En tal sentido, en los últimos quince años se han presentado resultados auspiciosos que dan cuenta de soluciones robustas, flexibles, tolerantes a los cambios en el entorno, confiables y computacionalmente eficientes.

2.1. Algoritmos evolutivos

Durante la década de 1950 los sistemas evolutivos comenzaron a despertar interés en un conjunto de investigadores dedicados a las ciencias de la computación que pensaban que la evolución natural podría utilizarse como una herramienta efectiva para resolver problemas de ingeniería.

Los algoritmos genéticos *(AG)* fueron concebidos originalmente por *John Holland* en la década de 1960 como técnicas probabilísticas de búsqueda que imitaran la evolución natural de las especies basándose en el principio de supervivencia de los individuos más aptos [Gol89, Mit98]. El objetivo principal fue estudiar el fenómeno de la adaptación de las especies y desarrollar formas para que el mecanismo natural de la adaptación pudiera introducirse en un sistema computacional.

En la actualidad, los conceptos centrales, entre las muy diversas implementaciones de la original idea de *Holland*, son: cromosoma, población, función de *fitness* y operadores genéticos.

2.1.1. Proceso evolutivo

El funcionamiento del algoritmo se basa en que inicialmente, algunas potenciales soluciones al problema son codificadas como cromosomas que forman una población de individuos. Cada individuo es evaluado a través de una función de *fitness*. Esta función entrega una medida de la aptitud de cada individuo para sobrevivir en su entorno y permite accionar un mecanismo de selección o disputa por la supervivencia, que al igual que la selección natural, elige o privilegia a los más aptos. Estos individuos, a su vez, son elegidos para reproducir nuevas generaciones de individuos a partir de la aplicación de los operadores genéticos.

Este proceso es repetido generación tras generación hasta que la población de individuos converge a representaciones de soluciones de buena calidad, donde el mejor individuo tiene buenas chances de representar una solución óptima o cercana al óptimo.

2.1.2. Operadores genéticos

Los operadores más comunmente utilizados son:

Recombinación/Cruzamiento Se utiliza para explotar la información contenida en las soluciones (intermedias) generando nuevos individuos a partir de dos o más existentes. Este operador puede consistir en dividir dos individuos en una parte elegida en forma aleatoria y combinar las partes resultantes entre sí para formar dos nuevos individuos.

Mutación Se utiliza para explorar el espacio de búsqueda. Este operador cambia alguna porción (elegida al azar) del individuo de forma de generar más diversidad y, potencialmente, mejores individuos.

Selección Se utiliza para implementar el principio de supervivencia de los individuos más adaptados (semejante al esquema de selección natural). Este operador selecciona a los individuos que conformarán la siguiente generación de soluciones.

2.1.3. Aplicación al problema

Algunas características de los Algoritmos Evolutivos los han convertido en serios candidatos para su aplicación al problema de planificación de movimientos. Las más notorias y usualmente resaltadas son:

- buen desempeño en espacios de búsqueda de dimensión grande
- tolerancia a diferentes tipos de funciones a optimizar (no requieren linealidad, continuidad ni diferenciabilidad, p.ej.)
- procesamiento paralelo

Las soluciones al problema de planificación de movimientos basadas en el uso de Algoritmos evolutivos suelen formularse reinterpretando el problema de planificación como un problema de aprendizaje por refuerzos[1].

Este tipo de problemas toma como hipótesis, que el comportamiento inteligente surge de un agente que intenta maximizar la esperanza, a largo plazo, de la suma de refuerzos recibidos -usualmente en un ambiente desconocido-. La meta es que el agente aprenda a comportarse de manera (cuasi-)óptima solamente guiados por su afán de maximizar una señal de refuerzo, pero sin la presencia de un experto que le indique qué acciones tomar en cada momento. [Diu10, SB98a]

En tal sentido, en un problema de AR pueden identificarse cuatro elementos principales [SB98b]:

- la política: o función del agente, que determina el comportamiento del agente en cada momento.
- la función de refuerzo: que define el objetivo en un problema de AR, definiendo una correspondencia entre cada estado percibido por el agente y un refuerzo (recompensa o penalización) que indica o mide que tan deseable de alcanzar es dicho estado.
- la función de valor: mientras la función de refuerzo indica qué es bueno -considerando solamente el presente-, esta indica qué es bueno en el largo plazo. Puede verse también como la cantidad de refuerzos que un agente puede acumular en el futuro a partir del presente estado. Aunque varios

[1] En la década de 1980 y 1990, a partir de ideas provenientes de la psicología, teoría de control, optimización y microeconomía, nació como subcampo de la Inteligencia Artificial y el Aprendizaje Automático, el Aprendizaje por Refuerzo (AR). [Diu10]

métodos de *AR* utilizan una función de valor, esta no es estrictamente necesaria. Otros métodos de búsqueda como Algoritmos genéticos, Programación genética, Recocido simulado (*Simulated Annealing*) y bioinspirados (*Ant Colony Systems*) son utilizados para resolver problemas de *AR*. Estos métodos realizan un búsqueda directamente en el espacio de políticas sin requerir nunca de una función de valor.

- el modelo del entorno: busca imitar el comportamiento del entorno permitiendo la predicción tanto de estados como recompensas futuras. La incorporación de modelos y planificación en *AR* es relativamente nueva. Actualmente, son utilizados tanto métodos basados en modelo como libres de modelo.

Finalmente, el esquema de interacción entre un agente y su entorno en un problema de *AR* se puede apreciar en la figura 2.1.

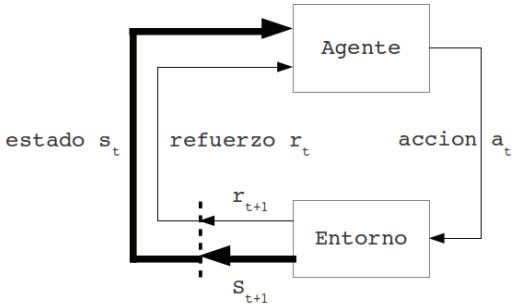


Figura 2.1: Interacción agente-entorno en *AR*.

En dicho modelo, el agente y el entorno interactúan en cada paso de una secuencia discreta de pasos de tiempo *t=1,2,3,....* A cada paso de tiempo t, el agente percibe su entorno recibiendo una representación del estado $s_t \epsilon S$ -donde S es el conjunto de los posibles estados-, basándose en esa información elige una acción $a_t \epsilon A(s_t)$ -donde $A(s_t)$ es el conjunto de acciones disponibles en el estado s_t-. En el siguiente paso de tiempo, *t+1*, el agente recibe, como consecuencia de la acción elegida, una recompensa $r_{t+1} \epsilon R$, y percibe al entorno en un nuevo estado s_{t+1}. [KLM96]

En tal sentido, se relevaron diversas aplicaciones al problema de planificación de movimientos. Los aspectos que fueron objeto de su aplicación se resumen a continuación:

- En el contexto de sistemas robóticos autónomos móviles de carga, computar caminos libres de colisión minimizando el tiempo de traslado y maximizando la carga transportada.
- En el contexto de sistemas robóticos industriales, computar caminos libres de colisión para brazos robóticos poliarticulados.
- Computar caminos libres de colisión en entornos simulados y realeas, estáticos y dinámicos, considerando: largo, seguridad y suavidad de los mismos.
- En el contexto de seguimiento de trayectorias, mejorar la precisión del robot.

Parte III

Hacia un sistema autónomo de navegación

Esta parte del documento presenta una propuesta de solución al problema de planificación de movimientos en entornos controlados. La misma fue concebida con el doble propósito de adquirir experiencia en la implementación de soluciones basadas en algunos de los enfoques, métodos o técnicas relevadas y, simultáneamente, ser la base para la construcción posterior de un sistema autónomo de navegación.

Asimismo, cabe resaltar que, para el desarrollo de la propuesta mencionada, se utilizan sólo algunos de los conocimientos relevados. La selección se realizó tomando en consideración los siguientes aspectos:

- la experiencia previa del autor en su uso,
- las características más salientes de cada una (ventajas/desventajas) y
- la existencia de evidencia empírica de su aplicación al caso de estudio en el que se trabajaría.

Particularmente, la propuesta se concentra en la planificación de movimientos para robots móviles -birrodados con control diferencial- que se desempeñan en entornos estáticos y parcialmente observables[2].

A tales efectos, se empleó un enfoque basado en la construcción de un *roadmap* y la optimización de caminos libres de colisiones. Para la construcción del *roadmap* se utilizó un diagrama de Voronoi y para la optimización de los caminos se utilizó un Algoritmo Genético sobre una representación basada en una descomposición aproximada del entorno. Además, se consideró la posibilidad de computar caminos libres de colisiones que satisficieran más de un criterio de optimalidad.

Asimismo, se consideraron trabajos previamente desarrollados que implementan algunos de los enfoques elegidos. Concretamente, y con la idea de combinarlas, se consideraron las propuestas presentadas por *Roque* y *Doering* en [RD05], donde se utilizan diagramas de *Voronoi* para representar la conectividad del espacio de configuraciones libres del entorno y sobre los cuales realizar luego la búsqueda del camino más corto, y las presentadas por *Zhang et al* en [ZZZ08], donde se utiliza un algoritmo genético, que opera sobre un modelo del entorno basado en grillas regulares, para calcular caminos "óptimos" que cumplen ciertos atributos de calidad (largo, seguridad y suavidad).

El capítulo 3 describe la implementación del planificador de movimientos y cómo se realizó la combinación entre las propuestas mencionadas. La sección 3.1 describe cómo se ha modelado el entorno en el que se mueve el robot, en tanto la aplicación del algoritmo genético para la resolución del problema de optimización se describe en la sección 3.2.

Finalmente, el capítulo 4 detalla los experimentos y pruebas realizadas, mientras que las principales conclusiones y trabajos a futuro se desarrollan en el capítulo 6.

[2] Debido a la incertidumbre que genera el ruido presente en las medidas del sensor global utilizado.

Capítulo 3

Descripción del sistema

3.1. Modelado del entorno

Para el modelado del entorno se deben considerar sus propiedades características. En el presente trabajo el entorno que se utiliza es:

- mono-agente: sólo hay un robot moviéndose en el escenario.
- parcialmente observable: si bien se utiliza un sistema de visión global, el mismo brinda información con un nivel no despreciable de incertidumbre y este ruido en la información brindada por el sensor puede afectar la medida de rendimiento del robot.
- estático: dado que el robot es el único objeto móvil, el entorno no cambia mientras se elige la siguiente acción.

A partir de ello, para la representación del espacio de configuraciones libres (x, y, θ)[1] se utilizó una combinación de métodos que son de aplicación en entornos con esas características. Por un lado, se realizó una descomposición rectangular aproximada basada en una grilla regular, donde cada celda puede representar una zona libre u ocupada (las celdas parcialmente ocupadas son consideradas como totalmente ocupadas).

Por otro lado, se computó el diagrama de *Voronoi* generado a partir del conjunto de puntos ubicados en el centro de cada celda ocupada. De este modo, el centro de cada celda ocupada -que representa total o parcialmente un obstáculo- será también un punto generador de *Voronoi*.

Esta representación presenta dos características de interés:

1. Facilita la combinación de las propuestas de base ([RD05, ZZZ08]) ya que unifica en una misma representación las características predominantes de las representaciones utilizadas, separadamente, en ambas.
2. Al permitir representar realidades o escenarios equivalentes, facilita la realización de estudios comparativos significativos entre las propuestas involucradas.

En la figura 3.1 se puede apreciar un escenario de ejemplo modelado según esta representación.

[1] El sistema posee tres grados de libertad que están asociados a la ubicación del robot en R^2 y su orientación respecto a un sistema de referencia.

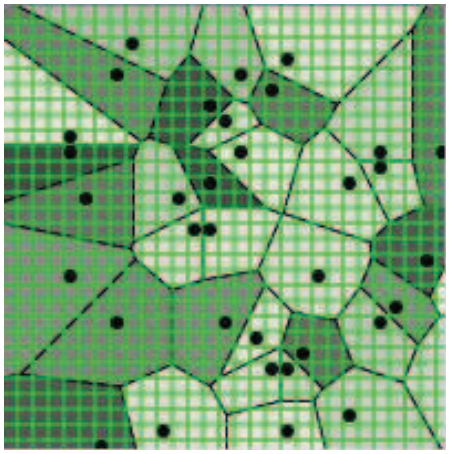


Figura 3.1: Representación del espacio de configuraciones.

3.2. Aplicación de un Algoritmo Genético

Tomando en consideración algunos de los atributos que caracterizan a los algoritmos genéticos (en adelante AG), a saber: buen desempeño en espacios de búsqueda de gran dimensión, tolerancia a diferentes tipos de funciones (no requieren linealidad, continuidad ni diferenciabilidad) y procesamiento paralelo; y que existe una amplia gama de ejemplos de su aplicación al problema de planificación de movimientos, se plantea la utilización de un AG para mejorar la calidad de soluciones que aportan otros métodos.

El objetivo principal que se persigue es realizar una búsqueda en el espacio de configuraciones libres, determinando un camino libre de colisiones, donde se valorarán especialmente las siguientes propiedades: largo, suavidad y seguridad. En este caso, el largo de un camino está determinado por la suma de las distancias euclidias de las aristas que lo componen; la suavidad atiende a la dimensionalidad del camino y la relación entre las direcciones de los tramos que lo componen; la seguridad se determina sumando las distancias euclidias de cada arista del camino al obstáculo más cercano.

La búsqueda de caminos óptimos se realiza a partir de una población inicial integrada por caminos que fueron computados considerando todos los canales de la descomposición aproximada del entorno que contienen los caminos del *roadmap* generado a partir del diagrama de Voronoi.

Estos caminos poseen ciertos atributos:

1. son caminos seguros (debido a que se basan en un diagrama de *Voronoi* correspondiente al entorno).

2. son perfectibles en cuanto al largo y seguridad (debido a que sólo conectan celdas adyacentes).

A tales efectos, el AG debe ser capaz de, en las primeras etapas, explorar adecuadamente el dominio de búsqueda (espacio de configuraciones libres) garantizando la consideración global de las restricciones impuestas por el entorno y, evitando caer en convergencias prematuras, mientras que en las etapas finales debe explotar el conocimiento adquirido durante el proceso de búsqueda para mejorar el resultado final lo más posible aproximándolo al óptimo.

3.2.1. Representación

La representación elegida para cada individuo (camino) o cromosoma está dada por una secuencia de celdas libres, de largo variable, que debe necesariamente comenzar con la celda donde está ubicado el robot (contiene la configuración inicial) y tener como último elemento a la celda definida como destino de la trayectoria (contiene la configuración final).

Es importante notar que esta representación habilita, al igual que en [ZZZ08], tanto la existencia de individuos que representan soluciones al problema (caminos factibles), como de individuos que no representan caminos libres de colisiones (caminos infactibles). Este aspecto es relevante ya que, para evaluar correctamente todos los caminos que son explorados por el algoritmo durante la evolución, la función de *fitness* debe ser consistente y coherente en la evaluación del conjunto de individuos.

Aunque esto supone un mayor esfuerzo computacional, en contrapartida, permite trabajar sobre poblaciones genéticamente más ricas, y contribuye a no sesgar el proceso de búsqueda, sobretodo, en sus etapas más tempranas. La explicación reside en comprender que caminos infactibles están conformados por tramos infactibles y también, eventualmente, por tramos factibles. Estos últimos son de interés porque pueden, al menos potencialmente, contener la representación parcial de buenas soluciones y, por tanto, es deseable que dicha información no se pierda. En resumen, se intenta evitar que el algoritmo caiga en mínimos locales a costo de realizar un mayor esfuerzo de exploración o búsqueda.

3.2.2. Población inicial

La población inicial se genera siguiendo los siguientes pasos:

1. Generar el diagrama de *Voronoi* a partir del conjunto de puntos correspondiente a los centros de las celdas ocupadas presentes en el entorno (obstáculos).

2. Identificar las regiones R_s y R_e que contienen las configuraciones inicial y objetivo de la trayectoria a planificar, respectivamente.

3. Computar una búsqueda de caminos[2] sobre el *roadmap* o grafo subyacente (vértices y aristas del diagrama de *Voronoi*) para cada pareja de vértices (v_s, v_e), donde v_s y v_e pertenecen al conjunto de vértices de las regiones R_s y R_e, respectivamente.

4. Sortear un número aleatorio de caminos pertenecientes al conjunto de caminos generado en el paso 3.

5. Para cada camino seleccionado en el paso 4, calcular los canales (secuencias de celdas que lo representa en la grilla regular subyacente).

6. Tomando en cuenta la representación basada en secuencias de celdas calculada en 5, eliminar aleatoriamente un número variable de celdas de cada camino del conjunto. Aunque, objetivamente, este paso implica la pérdida de información genética, se espera que este efecto sea compensado durante la evolución mediante la aplicación del operador de cruzamiento. Por el contrario, el propósito de su aplicación es que contribuya a la eliminación de celdas redundantes y genere una población inicial con individuos que representen caminos más cortos y más suaves, a riesgo de que también puedan ser menos seguros, incluso infactibles (como se mencionó en 3.2.1).

En la figura 3.2 pueden apreciarse gráficamente los elementos considerados en la generación de la población inicial y un camino factible como ejemplo.

[2] Se utilizó una implementación del algoritmo de *Dijkstra* [Dij59] contenido en la biblioteca utilizada para la implementación de *AG*.

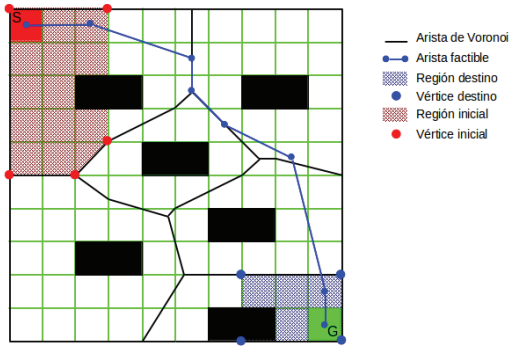


Figura 3.2: Generación de caminos.
Se destacan las regiones inicial y destino (R_s y R_e), los vértices de dichas regiones (v_s, v_e), el diagrama de Voronoi y las aristas de un camino factible.

3.2.3. Función de fitness

Entrega una medida sobre la bondad de un camino. Recorre valores reales del intervalo $[0, 1]$. Además, debe garantizar que los caminos infactibles siempre sean evaluados en menor medida que los factibles.

3.2.3.1. Caminos factibles

Para evaluar los caminos factibles se consideran tres aspectos: largo, seguridad y suavidad. Estos se expresan a través de términos normalizados al intervalo $[0, 1]$ y son ponderados con diferentes pesos[3] (w_i), como se puede apreciar en la ecuación 3.5. A su vez, las ecuaciones [3.1, 3.2 y 3.3] describen cómo se calcula cada uno de ellos.

$$d_{term} = \frac{maxPathLength - pathLength}{maxPathLength - minPathLength} \tag{3.1}$$

$$sa_{term} = \frac{pathSafety}{maxPathSafety} \tag{3.2}$$

$$sm_{term} = \frac{avePathSmooth}{pathSmooth + avePathSmooth} \tag{3.3}$$

$$w_d + w_{sa} + w_{sm} = 1 \tag{3.4}$$

$$f_f \quad = \quad w_d * d_{term} + w_{sa} * sa_{term} + w_{sm} * sm_{term} \tag{3.5}$$

- d_{term} representa la componente de distancia y se calcula a partir de la suma de las longitudes de todos los segmentos de recta que componen un camino. Para normalizarlo se toman como referencia los valores de largo máximo ($maxPathLength$: longitud del más largo de los caminos del diagrama de Voronoi) y mínimo ($minPathLength$: distancia euclidiana entre la configuración inicial y objetivo). De esta forma, los individuos cuyos largos se asemejen más $maxPathLength$ obtendrán un d_{term} cercano a 0, mientras que los que de largo se asemejen más a $minPathLength$ serán evaluados más cerca de 1.

[3] Los valores utilizados durante las etapas de calibración y evaluación del AG se registran en los cuadros 3.1 y 4.3, respectivamente.

- sa_{term} representa la componente de seguridad y se calcula a partir de la suma de las distancias de cada segmento de un camino al obstáculo más cercano. Para normalizarlo se toma como referencia el valor de seguridad máxima ($maxPathSafety$: seguridad del más seguro de los caminos del diagrama de Voronoi). De este modo, los individuos cuyo nivel de seguridad se asemeje a $maxPathSafety$ obtendrán un sa_{term} cercano a 1, mientras que los caminos menos seguros serán evaluados más cerca de 0.

- sm_{term} representa la componente de suavidad y se calcula a partir de la suma de ángulos entre segmentos contiguos de un camino. Para normalizarlo se toma como referencia el valor de suavidad promedio ($avePathSmooth$: promedio de los valores de suavidad de los caminos del diagrama de Voronoi). De este modo, los caminos cuyo nivel de suavidad se asemeje a 0 obtendrán un sm_{term} cercano a 1, mientras que los que estén cerca del promedio o incluso por encima serán evaluados más cerca de 0.

3.2.3.2. Caminos infactibles

Para evaluar los caminos infactibles se consideran tres aspectos: largo, tasa de infactibilidad y grado de infactibilidad. Estos se expresan a través de términos normalizados al intervalo $[0, 1]$, como se puede apreciar en la ecuación 3.9. La figura 3.3 presenta un camino infactible donde se destacan aristas y tramos infactibles.

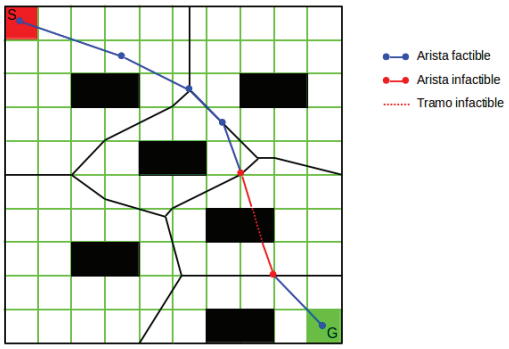


Figura 3.3: Camino infactible.

A su vez, las ecuaciones [3.1, 3.6 y 3.7] describen cómo se calcula cada uno de ellos.

$$ur_{term} = 1 - pathUR \tag{3.6}$$

$$cdw_{term} = \frac{maxPathCrossDepth - pathCrossDepth}{maxPathCrossDepth} \tag{3.7}$$

$$w_d + w_{ur} + w_{cdw} = 1 \tag{3.8}$$

$$f_u = w_d * d_{term} + w_{ur} * ur_{term} + w_{cdw} * cdw_{term} \tag{3.9}$$

- ur_{term} representa la componente de tasa de infactibilidad y se calcula dividiendo la cantidad de aristas infactibles de un camino entre el número de aristas totales. Para normalizarlo se considera el valor de referencia 1, que se obtienen cuando todas las aristas son infactibles. De este modo, los caminos totalmente infactibles tienen una tasa de infactibilidad ($pathUR$) igual a 1 y obtienen un ur_{term} igual a 0, mientras que los que son parcialmente infactibles serán evaluados con valores pertenecientes al intervalo (0,1).

- cdw_{term} representa la componente de grado de infactibilidad y se calcula dividiendo la suma de las longitudes de todos los tramos infactibles de un camino entre el largo total del mismo. Para normalizarlo se toma como referencia una cota máxima para el grado de infactibilidad (*maxPathCrossDepth*: cantidad total de celdas ocupadas multiplicada por la diagonal de una celda). De este modo, cuanto menos tramos infactibles o más cortos sean, el cdw_{term} de un camino será más cercano a 1.

3.2.4. Operadores genéticos

En esta sección se describe el funcionamiento de los operadores genéticos que se aplican a los individuos durante la evolución.

3.2.4.1. Cruzamiento

Se aplica a cada individuo (pw_{f1}), con probabilidad p_c y consiste en:

1. sortear otro camino de la población (pw_{f2}).
2. sortear una celda o posición de cruzamiento (i).
3. generar dos nuevos caminos *hijos*, conformados a partir del cruzamiento de la información genética de los *padres* (pw_{f1} y pw_{f2}), como lo indican las ecuaciones 3.10 y 3.11:

$$pw_{s1} = \{[pw_{f1,1}, ..., pw_{f1,i}], [pw_{f2,i+1}, ..., pw_{f2,n}]\} \tag{3.10}$$

$$pw_{s2} = \{[pw_{f2,1}, ..., pw_{f2,i}], [pw_{f1,i+1}, ..., pw_{f1,m}]\} \tag{3.11}$$

En la figura 3.4a se puede apreciar gráficamente el resultado de su aplicación.

3.2.4.2. Mutación

Se aplica a cada individuo, con probabilidad p_m y consiste en sortear una celda del camino, pw_i, y reemplazarla por otra. Las celdas candidatas son todas las que contengan algún tramo de las aristas (pw_{i-1}, pw_i) y (pw_i, pw_{i+1}).

En la figura 3.4b se puede apreciar gráficamente el resultado de su aplicación. El conjunto de celdas candidatas se resaltan con círculos de color celeste.

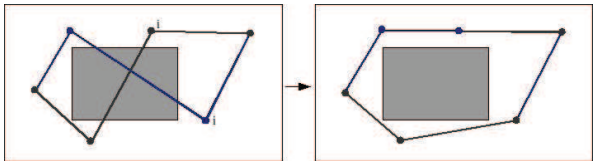
3.2.4.3. Suavizar

Es un operador *ad-hoc*. Su definición busca brindar un mecanismo por el cual puedan eliminarse celdas. La intención es dotar al algoritmo de una herramienta que le permita construir caminos más cortos y suaves a la vez que puede contribuir a eliminar infactibilidades. Otro efecto positivo, aunque secundario, es el de la eliminación de celdas redundantes[4].

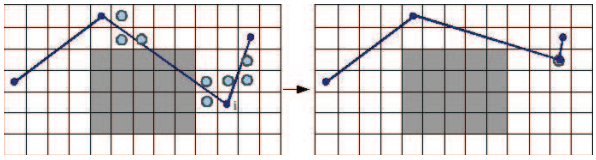
Se aplica a cada individuo, con probabilidad p_s y consiste en sortear una celda del camino, pw_i, y eliminarla.

En la figura 3.4c se puede apreciar gráficamente el resultado de su aplicación.

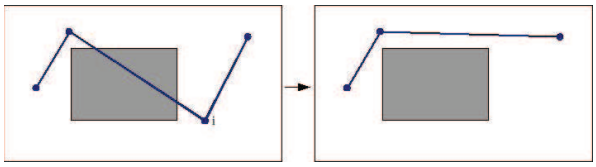
[4]Celdas pertenecientes al segmento de recta determinado por las celdas antecesora y sucesora inmediatas.



(a) Operador de Cruzamiento.



(b) Operador de Mutación.



(c) Operador *ad-hoc* Suavizar.

Figura 3.4: Aplicación de operadores genéticos.

3.2.4.4. Selección

A los efectos de ser coherente con las decisiones previas, que apostaron a preservar la diversidad o riqueza genética en las etapas tempranas del proceso de evolución, se utilizó un operador de selección moderadamente elitista. Aunque se garantiza que los mejores individuos sean seleccionados para integrar la siguiente población (5 %), los restantes son elegidos por sorteo en igualdad de condiciones, sin tomar en cuenta el valor de *fitness* asociado y por ende, habilitando la elección de individuos factibles e infactibles por igual.

3.2.5. Condiciones de terminación

A los efectos de terminar la ejecución del algoritmo se definieron los siguientes criterios de parada:

Convergencia Este criterio determina cuando la evolución ya no provee mejores individuos (soluciones) o las mejoras obtenidas no son suficientemente significativas como para seguir invirtiendo recursos en ella. Se determina contando la cantidad de pasos de evolución consecutivos donde no se registraron cambios en el mejor individuo ni en el *fitness* promedio de la población.

Cuando este criterio se cumple se puede asumir que:

1. la población convergió a un conjunto de individuos genéticamente similares, de cuyo cruzamiento no es probable esperar mejores soluciones.
2. el potencial de explotación se redujo a valores que hacen inviable la inversión de más esfuerzos por mejorar ya que el mismo no se vería recompensado por la ganancia obtenida.

Con tal propósito se definieron los siguientes parámetros:

- mínimo número de pasos de evolución: *5ev.*
- máximo número de pasos sin cambios en el mejor individuo: *5ev.*
- mínimo porcentaje de mejoría en el fitness promedio de la población: *0,08 %.*

Número máximo de pasos Este criterio es utilizado como mecanismo preventivo que garantiza la terminación del algoritmo. El máximo número de pasos utilizado fue *155ev.* Este valor fue determinado en forma empírica luego de verificar que todas las ejecuciones realizadas culminaron al cumplirse las condiciones del primer criterio de parada.

3.2.6. Calibración

Se calibraron los siguientes parámetros para la ejecución del AG: tamaño de la población inicial (ps) y probabilidad de aplicación de los operadores de mutación (p_m), cruzamiento (p_c) y suavizar (p_s).

3.2.6.1. Entornos

A tales efectos, y con el propósito de lograr cierto nivel de flexibilidad en las soluciones encontradas por el AG para posibilitar su uso en escenarios con diferentes configuraciones, se definieron tres (3) tamaños de entornos: 10x10, 20x20 y 50x50 celdas[5]. Los tamaños se determinaron en función de dos objetivos: viabilizar la comparación de resultados con los trabajos previos considerados (10x10) y probar la capacidad de adaptación del algoritmo a entornos de diferente dimensión (20x20 y 50x50).

A su vez, para minimizar la dependencia entre los valores calibrados y los entornos de prueba utilizados, para cada uno de los tamaños escogidos se utilizaron seis (6) instancias diferentes (escenarios) donde en cada una se dispusieron aleatoriamente[6] los obstáculos y se mantuvieron fijas las configuraciones inicial y destino (celdas superior izquierda e inferior derecha, respectivamente). La cantidad de los mismos es variable pero se mantuvo acotada entre un número mayor al 5 % y menor al 15 % del total de celdas de cada tamaño de grilla.

El cuadro 3.1 presenta un resumen de los parámetros generales utilizados en el proceso de entrenamiento.

Cantidad de Casos (escenarios)	18
Repetición de cada caso	10
Peso del objetivo *largo*	60 %
Peso del objetivo *seguridad*	15 %
Peso del objetivo *suavidad*	25 %
Porcentaje mínimo de celdas ocupadas	5 %
Porcentaje máximo de celdas ocupadas	15 %

Cuadro 3.1: Parámetros generales de entrenamiento.

[5] Entre los diferentes tamaños de entorno se consideran celdas de igual dimensión (unidades de distancia), de modo que el aumento de celdas no implica y no debe confundirse con un aumento de la resolución de la descomposición aproximada.

[6] La investigación sobre *benchmarking* en el área plantea que no existe un consenso sobre entornos que puedan utilizarse en forma estandarizada para medir el desempeño de sistemas de planificación de movimientos aplicados en robótica móvil.

Teniendo en cuenta que se realizarían diez (10) ejecuciones de prueba sobre diesiocho (18) entornos diferentes, se procuró sortear, al menos parcialmente, los efectos directos de la explosión combinatoria que supondría la evaluación de los rangos completos de valores para cada uno de los parámetros, dividiendo la calibración en dos etapas: *aproximación* y *refinamiento*.

3.2.6.2. Aproximación

La etapa de aproximación consistió en la ejecución de una serie de corridas del algoritmo donde se variaron los valores de todos los parámetros considerando incrementos medianos.

De este modo, se redujo el tamaño de las series (cantidad de ejecuciones) y se pudo obtener un conjunto de valores aproximados para los valores de los parámetros (ps, p_m, p_c y p_s).

En el cuadro 3.2 se presentan los valores utilizados en la etapa de aproximación.

	Mín	Máx	Inc
ps	40	160	30
p_m	0.01	0.02	0.002
p_c	0.5	0.6	0.02
p_s	0.25	0.50	0.125

Cuadro 3.2: Calibración I. Valores utilizados en la etapa de aproximación.

El criterio utilizado para la selección de los valores, tanto en la etapa de aproximación como en la de refinamiento, fue el de significancia estadística, donde dados dos algoritmos, A y B, se considera que el algoritmo A es mejor que B si la diferencia entre los *fitness* promedio de las poblaciones finales alcanzadas por cada algoritmo es estrictamente mayor que la mayor de las desviaciones estándar de cada población. En la ecuación 3.12 se expresa dicha desigualdad:

$$|f_{AVG}(A) - f_{AVG}(B)| > \text{máx}(std(f_A), std(f_B)) \tag{3.12}$$

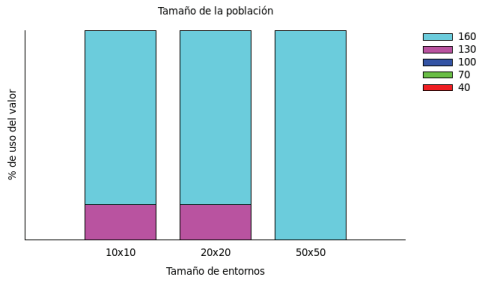
En el cuadro 3.3 se muestran los resultados obtenidos.

En el cuadro 3.3a se presentan los resultados para el tamaño de la población. La gráfica sugiere que, en cualquiera de los tres tamaños de escenario, el algoritmo logró converger a mejores soluciones cuando el tamaño de la población era 160 (exceptuando unos pocos casos donde también lo logró utilizando poblaciones más pequeñas de 130 individuos).

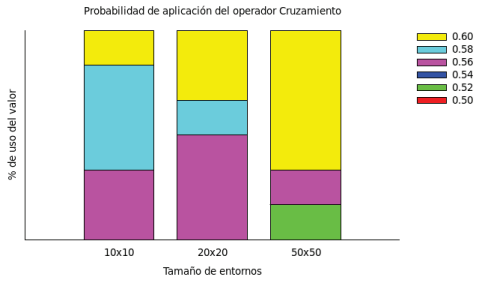
En el cuadro 3.3b se presentan los resultados para la probabilidad de aplicación del operador de cruzamiento. La gráfica sugiere que, en cualquiera de los tres tamaños de escenario, los mejores resultados se obtuvieron utilizando probabilidades mayores o iguales a 0,56.

En el cuadro 3.3c se presentan los resultados para la probabilidad de aplicación del operador de mutación. La gráfica sugiere que, en cualquiera de los tres tamaños de escenario, los mejores resultados se obtuvieron utilizando probabilidades entre 0,0125 y 0,0167.

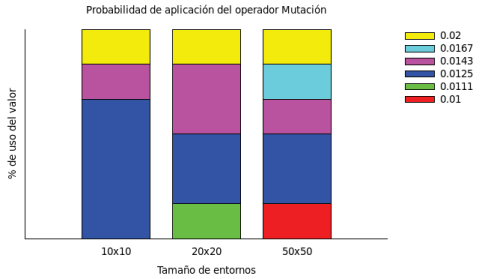
En el cuadro 3.3d se presentan los resultados para la probabilidad de aplicación del operador "suavizar". La gráfica sugiere que, dependiendo del tamaños de escenario, los mejores resultados se obtuvieron utilizando alguno de los tres valores de probabilidad considerados. Por tanto, para este parámetro, la etapa de aproximación no ha proporcionado evidencia que permita restringir el rango de valores a considerar.



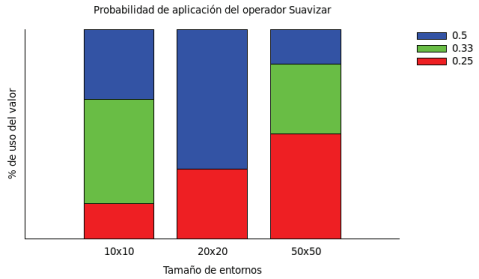
(a) Tamaño de la población.



(b) Probabilidad de cruzamiento.



(c) Probabilidad de mutación.



(d) Probabilidad de "Suavizar".

Cuadro 3.3: Calibración I. Resultados obtenidos.

3.2.6.3. Refinamiento

A partir del análisis realizado sobre los resultados obtenidos en la etapa de aproximación, se definieron los valores para la etapa de refinamiento (ver cuadro 3.4).

La etapa de refinamiento consistió en la ejecución de una nueva serie de corridas del algoritmo, tomando en cuenta, para cada parámetro, sólo los valores pertenecientes a un entorno con centro en el valor hallado en la etapa previa de aproximación (ej.: [ps_{aprox}-∂, ps_{aprox}+∂]) y considerando incrementos más pequeños entre valores.

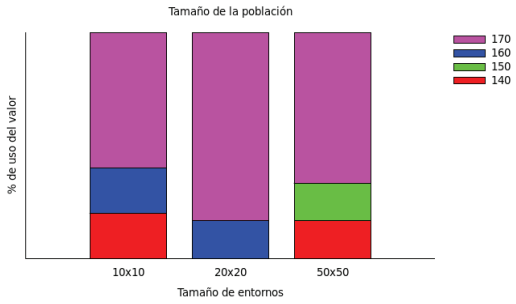
	Mín	Máx	Inc
ps	140	170	10
p_m	0.0105	0.0167	0.0009
p_c	0.56	0.62	0.02
p_s	0.25	0.50	0.125

Cuadro 3.4: Calibración II. Valores utilizados en la etapa de refinamiento.

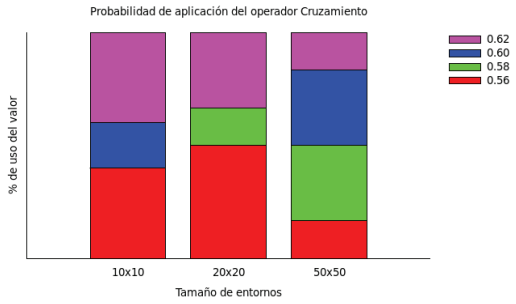
En el cuadro 3.5 se muestran los resultados obtenidos en esta etapa de calibración. Los mismos representan, para cada tamaño de escenario utilizado en las pruebas de calibración (grillas de 10x10, 20x20 y 50x50 celdas), como se distribuyó el uso de los diferentes valores considerados para cada parámetro en los mejores algoritmos (según el criterio de significancia estadística mencionado antes).

Vale decir, y a modo de ejemplo, si se considera el cuadro 3.5a, para el parámetro *tamaño de la población*, en el caso de escenarios *s10* (grillas 10x10) los valores 140, 160 y 170 fueron utilizados en la configuración del mejor algoritmo en el 20 %, 20 % y 60 % de las ejecuciones realizadas, respectivamente.

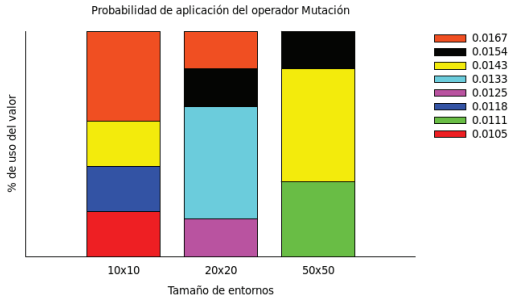
De todos los parámetros calibrados, los resultados más claros refieren al tamaño de la población y a la probabilidad de aplicación del operador *ad-hoc* suavizar. En cuanto al primero, el cuadro 3.5a muestra como el valor *170* predomina en los tres tipos de escenarios. Además sugiere cierta independencia entre el tamaño de la población y el tamaño de los escenarios utilizados. En cuanto al segundo, en cambio, si bien existe un valor -*0.25*- que predomina en los tres casos, la importancia de su aplicación parece aumentar a medida que aumenta el tamaño de los escenarios.



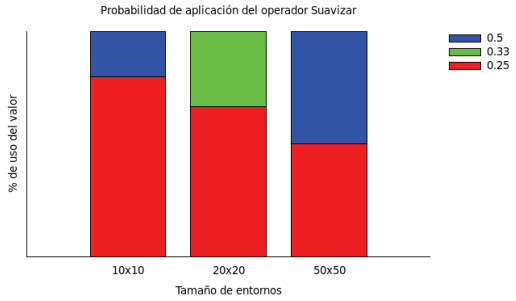
(a) Tamaño de la Población.



(b) Probabilidad de Cruzamiento.



(c) Probabilidad de Mutación.



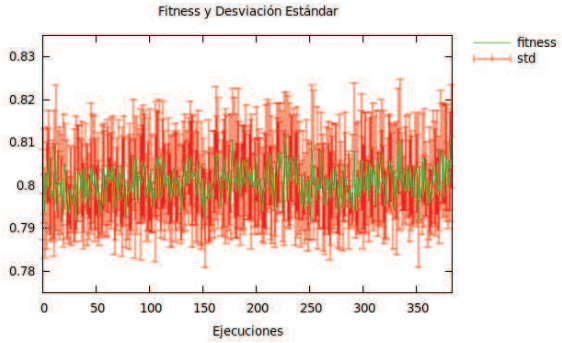
(d) Probabilidad de Suavizar.

Cuadro 3.5: Calibración II. Resultados obtenidos.

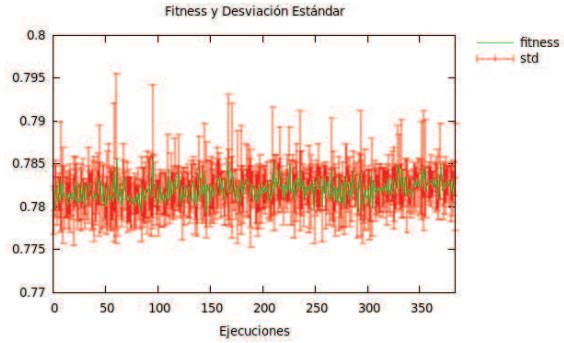
Para el caso del tamaño de la población, estos resultados tienen sentido si son analizados desde el siguiente ángulo: en el caso de escenarios de mayor tamaño valores grandes de este parámetros se explican en la necesidad que demuestra el algoritmo de contar inicialmente con suficiente diversidad de información genética para lograr converger a individuos de buena calidad cercanos al óptimo.

Sin embargo, para el caso de escenarios de menor tamaño la explicación podría surgir al considerar los valores de *fitness* promedio y desviación estándar alcanzados con cada configuración.

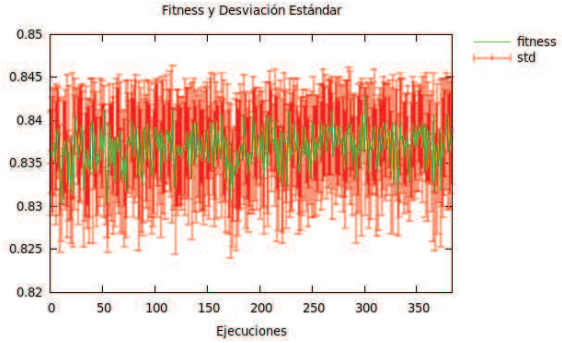
En el cuadro 3.6 se puede apreciar como dichos valores se mantienen acotados en un rango muy pequeño. Esto indicaría que casi todos los valores de configuración permiten al algoritmo converger a buenos individuos pero son las configuraciones con poblaciones grandes las que estadísticamente logran mejorar, aunque marginalmente, la solución.



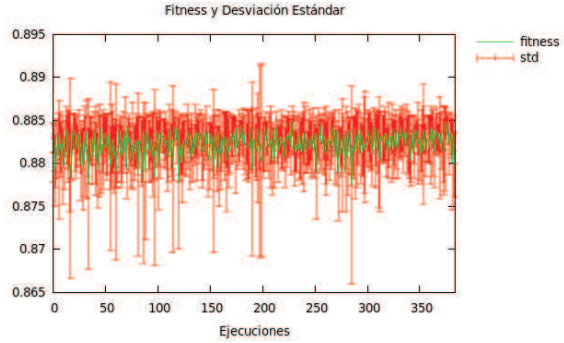
(a) Instancia #1.



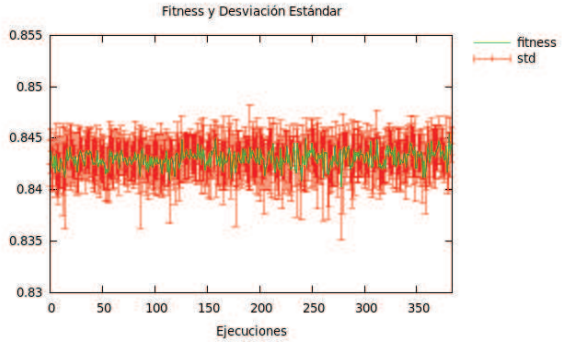
(b) Instancia #2.



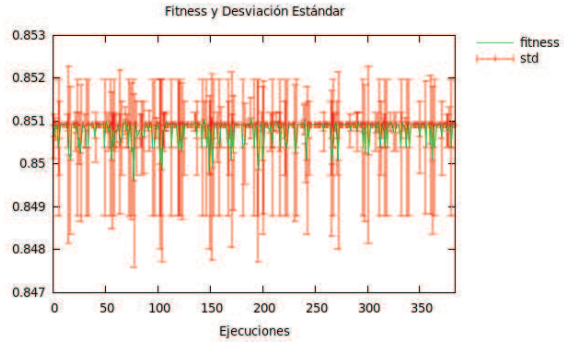
(c) Instancia #3.



(d) Instancia #4.



(e) Instancia #5.



(f) Instancia #6.

Cuadro 3.6: *Fitness* promedio y *Std* para escenarios pequeños.

Para el caso de la probabilidad de aplicación del operador suavizar el resultado es más intuitivo ya que la oportunidad de la aplicación del mismo aumenta en la medida que los escenarios sean más grandes, contengan más celdas, obstáculos y, finalmente, los caminos sean más largos y sinuosos. Por otro lado, y considerando que es un operador que está más emparentado con la explotación de resultados previos que con la exploración de nuevas experiencias, también es razonable el resultado obtenido ya

que en escenarios más pequeños la experiencia a explotar es menor y menos rica que en entornos más vastos.

Finalmente, los resultados obtenidos sobre la probabilidad de aplicación del operador de cruzamiento no son determinantes en torno a un único valor, pero si lo son en cuanto a que, independientemente del tamaño del escenario, sería razonable utilizar el valor promedio (0.58). Por el contrario, en el caso de los resultados obtenidos para la probabilidad de mutación, lo más saliente es que predomina el uso de las probabilidades más altas (entre 0.0133 y 0.0167), pero no se distingue ninguna relación fuerte entre el tamaño de los escenarios y el valor de este parámetro.

Capítulo 4

Experimentos y resultados

4.1. Evaluación del AG

A los efectos de poder evaluar el rendimiento del algoritmo y la calidad de las soluciones obtenidas, una vez definido el conjunto de valores de la configuración, se realizaron un serie más de ejecuciones. Para su realización se consideraron los mismos tamaños de escenario (10x10, 20x20 y 50x50 celdas) y configuración inicial y destino empleados durante el proceso de calibración.

4.1.1. Entorno simulado

Los cuadros 4.1 y 4.2 resumen las características más importantes de las plataformas utilizadas para la ejecución de las pruebas.

CPU	Intel Pentium Dual Core E2140 - 1.60GHz
RAM	2GB DDR

Cuadro 4.1: Plataforma de hardware.

OS	GNU/Linux - Xubuntu 2.6.31-22
IDE	Eclipse Helios 3.6
Lenguaje	Java 1.6
AG Library	JGAP 3.3.3
Graph Library	JGrapht 0.8.1
Voronoi Library	Quickhull3d 1.4
Visor	Processing 1.0.7

Cuadro 4.2: Plataforma de software.

4.1.2. Parámetros

En el cuadro 4.3 se muestran los valores utilizados en la configuración del *AG*.

Tamaño de la población (individuos)	170
Probabilidad de uso del operador de mutación (p_m)	0.0139
Probabilidad de uso del operador de cruzamiento (p_c)	0.59
Probabilidad de uso del operador "suavizar" (p_s)	0.33

(a) Parámetros calibrados.

Porcentaje de elitismo en la selección	5 %
Cantidad máxima de generaciones	155
Cantidad mínima de generaciones	5
Máxima cantidad de generaciones sin variar el mejor individuo	5
Mínimo porcentaje de mejoría del *fitness* promedio de la población	8 %

(b) Otros parámetros.

Cuadro 4.3: Parámetros de ejecución del *AG*.

Por otro lado, en el cuadro 4.4 se muestran los parámetros generales aplicados durante la evaluación del *AG*. Debe advertirse que, si bien cada instancia de escenario se genera respetando los parámetros generales, la cantidad y ubicación de los obstáculos en cada escenario se determina en forma independiente y aleatoria.

Cantidad de Casos (escenarios)	100
Repetición de cada caso	50
Peso del objetivo *largo*	60 %
Peso del objetivo *seguridad*	15 %
Peso del objetivo *suavidad*	25 %
Porcentaje mínimo de celdas ocupadas	10 %
Porcentaje máximo de celdas ocupadas	20 %

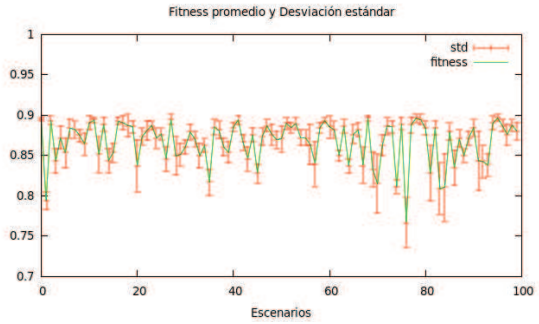
Cuadro 4.4: Parámetros generales de evaluación.

4.1.3. Resultados

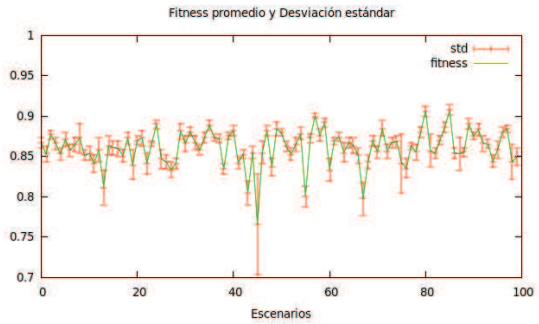
El cuadro 4.5 presenta el *fitness* promedio y desviación estándar obtenidos al ejecutar el planificador en cien (100) instancias diferentes de cada tamaño de escenarios considerado. Las principales conclusiones que arrojan se describen a continuación:

1. considerando que la generación de los escenarios se realizó en forma aleatoria, puede concluirse que para un gran número de situaciones se alcanzan soluciones de muy buena calidad [1].

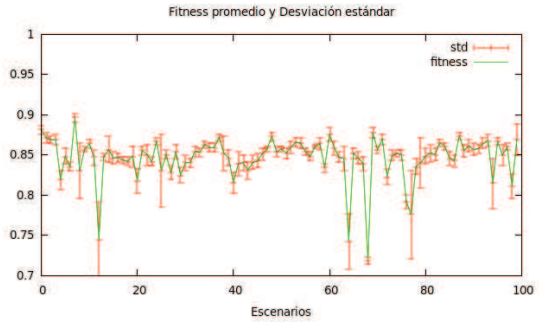
2. teniendo en cuenta que cada caso se ejecutó cincuenta (50) veces y que la desviación estándar, en casi todos los casos, se mantuvo acotada por debajo de valores bajos, el algoritmo muestra una muy buena estabilidad en lo que refiere a converger globalmente a soluciones de calidad.



(a) 10x10.

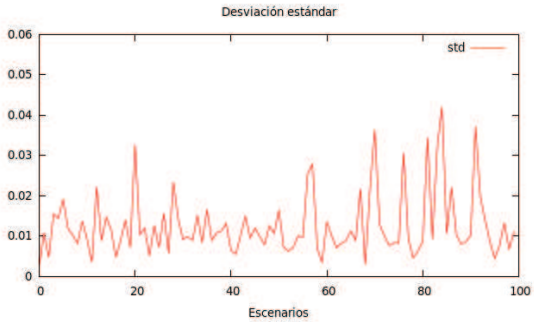


(b) 20x20.

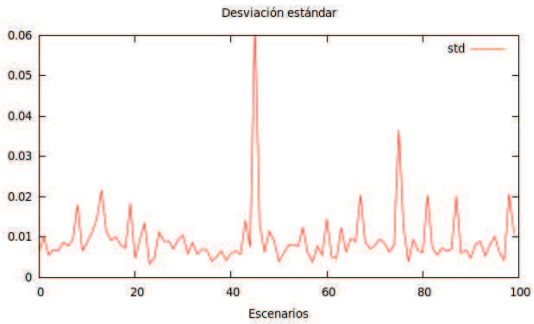


(c) 50x50.

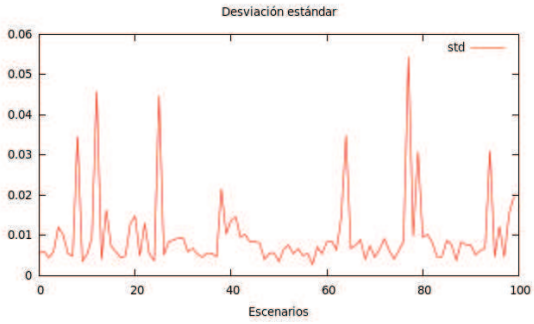
Cuadro 4.5: Evaluación. Fitness promedio y Desviación estándar.



(a) 10x10.



(b) 20x20.



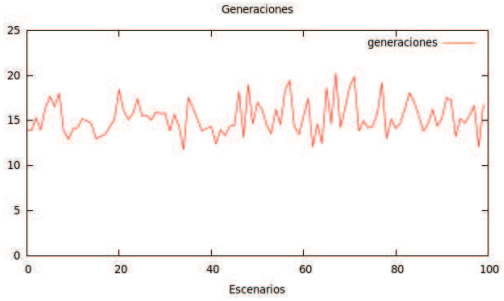
(c) 50x50.

Cuadro 4.6: Evaluación. Desviación estándar.

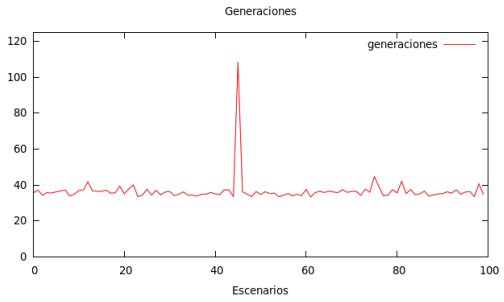
[1] Considerando la medida de calidad que ofrece la función de *fitness*.

3. en el cuadro 4.6 se presentan los gráficos de desviación estándar exclusivamente donde también puede apreciarse la buena estabilidad referida anteriormente.

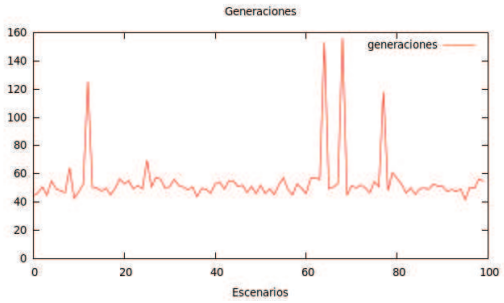
El cuadro 4.7 muestra la cantidad de generaciones requeridas en cada caso para alcanzar la condición de parada y en el cuadro 4.8 el tiempo insumido por el algoritmo en la resolución de cada entorno. Observando los resultados obtenidos pueden extraerse varias conclusiones:



(a) 10x10.

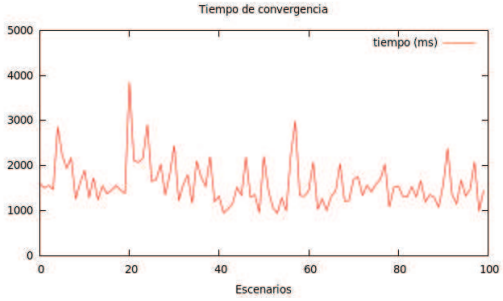


(b) 20x20.

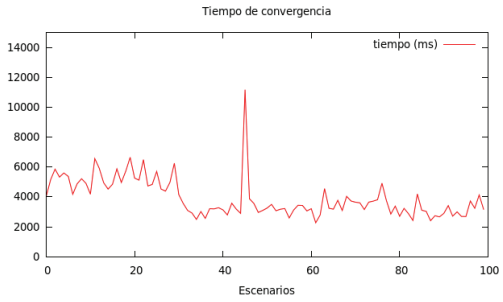


(c) 50x50.

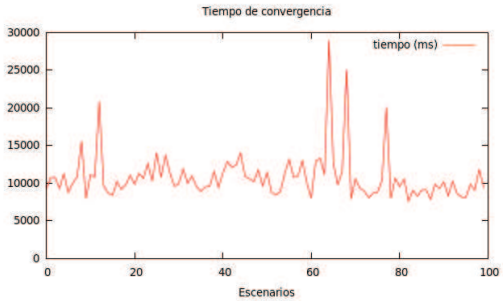
Cuadro 4.7: Evaluación. Generaciones.



(a) 10x10.



(b) 20x20.



(c) 50x50.

Cuadro 4.8: Evaluación. Tiempo.

1. el algoritmo es sumamente estable, tanto en la cantidad de generaciones que requiere evolucionar como en el tiempo insumido en la resolución de cada generación.

2. comparando los gráficos de generaciones correspondientes a entornos de diferente tamaño, puede apreciarse como aumenta la cantidad de generaciones requeridas para alcanzar la condición de

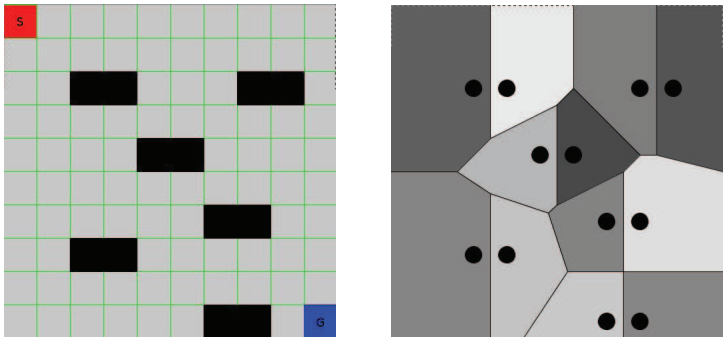
parada a medida que aumenta la dimensión del entorno y con ella el tamaño de los individuos (largo de camino), la cantidad de combinaciones a realizar entre los mismos, la proporción de individuos factibles respecto del total y dentro de ese subconjunto también aumenta la cantidad de individuos factibles candidatos a solución (elevado valor de *fitness*). Podría decirse que de la mano de estos aspectos, aumenta la competitividad y el algoritmo necesita más generaciones para poder seleccionar un camino que se destaque como mejor solución por encima de los restantes.

3. por otro lado, comparando los gráficos de tiempo correspondientes a entornos de diferente tamaño, se aprecia como el tiempo requerido por entorno es mayor para entornos más grandes, aunque la forma de la gráfica sugiere que existan otros factores.

4.2. Estudio comparativo

En esta sección interesa realizar un análisis comparativo entre la presente propuesta y la presentada en [ZZZ08]. A tales efectos y como forma de viabilizar una comparación equitativa, se realizaron ejecuciones de ambas propuestas sobre el mismo escenario utilizado en [ZZZ08].

La figura 4.1 muestra una representación gráfica del entorno. Mientras en la figura 4.1a se presenta una vista de la descomposición aproximada del entorno donde se destacan las celdas origen, destino y los obstáculos (celdas parcialmente ocupadas), en la figura 4.1b se puede apreciar el diagrama de Voronoi generado a partir de los obstáculos.



(a) Descomposición aproximada. (b) Diagrama de Voronoi.

Figura 4.1: Entorno de prueba.
(4.1a) La configuración inicial pertenece a la celda roja. La configuración destino pertenece a la celda azul. Las celdas ocupadas se destacan en negro.
(4.1b) Los puntos generadores y aristas de Voronoi se destacan en negro (puntos y líneas, respectivamente). Las regiones se distinguen con tonalidades de grises.

4.2.1. Comparativo numérico

Se realizaron comparaciones tanto a nivel de la calidad de las soluciones como del desempeño computacional de los diferentes enfoques. En tal sentido, se comparon indicadores de la propuesta desarrollada contra los obtenidos al ejecutar:

1. una implementación del trabajo previo [ZZZ08].
   Zhang et al aplican un AG para optimizar caminos según largo, suavidad y seguridad. En tal

sentido, construyen la población inicial basándose en una representación que denominan *Digital Potential Fields* (*DPF* en adelante) que permite computar caminos entre dos configuraciones preestablecidas. La figura 4.2a muestra el mapa resultante de aplicar *DPF* a la descomposición aproximada rectangular de la figura 4.1a. Sobre el mapa se computan todos los canales que unen (0,18) a partir de celdas adyacentes donde se cumple que el dígito de una celda origen es menor que el de la celda destino en a lo sumo 2 unidades. La figura 4.2b muestra un camino computado a partir de la dicha representación.

(a) Mapa DPF. (b) Camino sobre *DPF*.

Figura 4.2: Representación *Digital Potential Fields*.

2. una búsqueda exhaustiva realizada sobre la descomposición aproximada rectangular de la figura 4.1a.
   A tales efectos, se implementó una búsqueda en profundidad (*DFS*) sobre las celdas libres del entorno. El cuadro 4.9 presenta los resultados obtenidos al realizar la búsqueda *DFS* para el cálculo del camino libre de colisiones.

Tiempo de ejecución	Fitness	Largo	Seguridad	Suavidad
649049.0(ms)	0.81985	665.68	247.16	0.58905

Cuadro 4.9: Indicadores de la búsqueda *DFS*.

El cuadro 4.10 presenta los resultados obtenidos al aplicar ambas propuestas (*DPF* refiere a la implementación propia de la propuesta realizada por *Zhang et al* y *DV* refiere a la presente propuesta debido a que es una aplicación de *AG* sobre una representación basada en Diagramas de Voronoi) al caso de prueba que se muestra en la figura 4.1.

En base a los indicadores presentados en ambos cuadros (4.9 y 4.10), se pueden extraer las siguientes conclusiones:

1. Realizar una búsqueda exhaustiva, aún en entornos pequeños y considerando un espacio de búsqueda más reducido (sólo se conectaron celdas adyacentes), tiene un costo computacional elevado que lo transforma en una mala estrategia de planificación si se piensa aplicar en sistemas robóticos autónomos en entornos dinámicos.

2. Las dos propuestas basadas en *AG* logran encontrar caminos de mejor calidad que los obtenidos mediante la búsqueda exhaustiva implementada. Esto se debe, principalmente, a que avanzando

de a una celda por vez no se logra explorar todo el espacio de búsqueda y, por ende, las soluciones que conectan celdas distantes no son consideradas. La solución obtenida se considera una cota inferior debido a que, como se mencionó anteriormente, los atributos de largo y suavidad pueden ser mejorados al considerar tramos conformados por celdas no adyacentes. Por el contrario, el tiempo de cómputo es considerado como una cota superior a mejorar.

	AG sobre *DPF*	*AG* sobre *DV*
Generación del entorno (ms)	1632	1217
Evolución del AG (ms)	275.79	1299
Generaciones promedio	12.51	17.09
Fitness promedio	0.82990	0.84669
Desviación estándar	0.01067	0.00567
Mejor *Fitness*	0.84583	0.85459
Desviación al mejor	0.01267	0.00790
Largo promedio	656.27	657.63
Seguridad promedio	253.58	280.46
Suavidad promedio	0.29111	0.21318

Cuadro 4.10: Comparativo numérico I.
NOTA: El largo y seguridad promedio se miden en unidades de distancia mientras que la suavidad promedio se mide en radianes por vértice.

3. Con respecto a la eficiencia computacional de los algoritmos basados en *AG*, surge que la propuesta basada en *DPF* demanda más tiempo en la fase de generación del entorno pero la evolución del algoritmo es casi cinco veces menor que la propuesta basada en *DV*. También el número de generaciones requeridas para alcanzar soluciones de buena calidad es menor en el caso de la propuesta basada en *DPF*.

4. Por otro lado, en cuanto a la convergencia del *AG* se aprecia que la propuesta basada en *DV* presenta mejores niveles de estabilidad en el sentido de converger a poblaciones donde sus individuos son más parecidos en calidad.

5. Finalmente, al considerar la calidad de las soluciones obtenidas, surge que la propuesta basada en *DV* logra alcanzar mejores soluciones. En general, de largos similares pero con mejores atributos en cuanto a seguridad y suavidad.

En resumen, la propuesta basada en *DV* demuestra ser robusta (siempre retornó soluciones factibles), flexible (se ha evaluado sobre un conjunto variado de entornos -de diferentes dimensiones y espacio de configuraciones de los obstáculos) y eficaz (encontrando soluciones de calidad).

Sin embargo, y considerando que en robótica móvil el tiempo de computo invertido en los procesos deliberativos es un factor de suma importancia, a veces tanto o más que la propia calidad de los resultados que se obtienen[2], podrían caber las siguientes interrogantes: ¿a qué puede deberse la diferencia en la eficiencia computacional entre ambas propuestas? ¿para el caso de estudio analizado, se justifica

[2] Existen numerosas aplicaciones donde no alcanza con calcular la mejor decisión sino que es necesario calcularla a tiempo.

tal diferencia por el logro de mejores resultados? ¿sería posible reducir el tiempo de cómputo de la propuesta basada en DV sin afectar sensiblemente o manteniendo el nivel de calidad de los resultados obtenidos?

Algunas explicaciones surgen al observar que la representación utilizada en la propuesta basada en DV ocasiona que en la generación de la población inicial los individuos contengan una cantidad de puntos intermedios sensiblemente menor. De algún modo, se paga tributo por una generación del entorno menos costosa pero con una inicialización menos rica, genéticamente hablando. Evidentemente, la diversidad genética a pesar de ser menor es suficiente, lo que queda demostrado al observar la calidad de las soluciones que se logran, sin embargo, existe un costo computacional asociado a compensar esta desventaja mediante la realización de más generaciones. Otro factor relacionado con este sobre costo es el tamaño de las poblaciones utilizadas, aunque no lo asociamos exclusivamente a este hecho. En forma bastante intuitiva se podría suponer que el tamaño de población, casi tres veces mayor al utilizado en la propuesta basada en DPF, es una forma también, de compensar la menor riqueza genética que proporciona la representación de los individuos en la propuesta basada en DV. Sin embargo, preferimos asociarlo al hecho de que ese valor surgió de un proceso de calibración realizado sobre escenarios con diferentes tamaños y cantidad de obstáculos, incluso, variedad de situaciones que se producían dependiendo de la ubicación de dichos obstáculos en el entorno y relativa entre si. Ergo, este valor está más emparentado con proporcionar generalidad y la posibilidad concreta de que el algoritmo, así configurado, logre entregar soluciones de buena calidad en una variedad muy importante de entornos.

Por tanto, con el propósito de intentar responder las dos últimas interrogantes, se realizaron dos nuevas ejecuciones del algoritmo, modificándole los valores de sus parámetros de configuración. De este modo, se buscó confirmar que recalibrando el algoritmo para la instancia específica del problema, sería posible reducir sensiblemente el tiempo de cómputo manteniendo la calidad de las soluciones alcanzadas con los valores de configuración surgidos de la etapa de calibración. A tales efectos se consideraron dos juegos de valores: la configuración utilizada en la propuesta basada en DPF y una configuración *ad-hoc* donde se hizo variar el tamaño de la población hasta provocar que los tiempos de evolución fueran similares a los tiempos medios de la propuesta basada en DPF.

El cuadro 4.11 presenta los resultados obtenidos al ejecutar la propuesta basada en DV con estas configuraciones.

Los mismos son elocuentes y confirman las hipótesis sugeridas anteriormente. Los indicadores de la primera ejecución (cuadro 4.11:Columna 1) demuestran que era posible mantener la calidad de las soluciones obtenidas disminuyendo sensiblemente el tiempo de cómputo, incluso por debajo de los obtenidos con la propuesta basada en DPF. Los indicadores de las soluciones encontradas en comparación con los indicadores presentados en el cuadro 4.10 son similares en largo y suavidad pero mejores en cuanto a seguridad.

Asimismo, en el caso de la segunda ejecución (cuadro 4.11:Columna 2) puede apreciarse que el aumento en el tamaño de la población requerido para emparejar los tiempos de ejecución no trae aparejado ninguna mejora significativa en cuanto a la calidad de los resultados.

En suma, puede concluirse que si se cuenta con cierta información extra del entorno (cantidad y área total ocupada por los obstáculos) es posible optimizar el proceso de planificación, disminuyendo el tiempo de cómputo invertido en l la ejecución del AG sin comprometer la calidad de las soluciones.

4.2.2. Comparativo gráfico

El cuadro 4.12 presenta gráficamente las soluciones obtenidas con cada una de las variantes ejecutadas.

El cuadro 4.12a muestra la solución alcanzada al utilizarse el enfoque de *Zhang et al.* El cuadro 4.12b muestra la solución alcanzada con el enfoque desarrollado en el presente trabajo (utilizando el juego de

	Parámetros de *DPF*	Parámetros *ad-hoc*
Generación del entorno (ms)	1305	1136
Evolución del AG (ms)	223	265
Generaciones promedio	15	15.45
Fitness promedio	0.83437	0.83720
Desviación estándar	0.01376	0.01169
Mejor *Fitness*	0.85433	0.85433
Desviación al mejor	0.01996	0.01714
Largo promedio	662.44	661.07
Seguridad promedio	291.19	287.55
Suavidad promedio	0.29058	0.27583

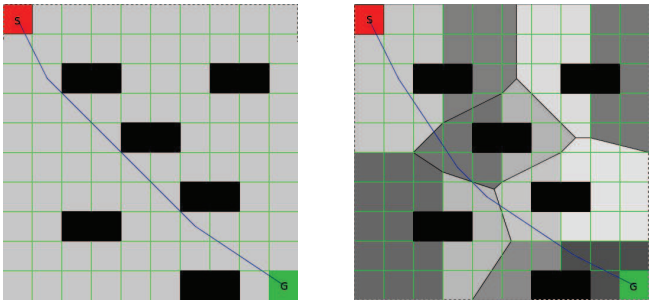
Cuadro 4.11: *AG* sobre *DV* con otros parámetros.

parámetros calibrados como se describe en 3.2.6). El cuadro 4.12c muestra la solución encontrada con el mismo enfoque que 4.12b pero utilizando los valores de parámetros de 4.12a. Finalmente, el cuadro 4.12d muesta una solución alcanzada con el mismo enfoque que 4.12b pero utilizando un tamaño de población ajustado manualmente. Como puede apreciarse, el camino resultante en el cuadro 4.12a es razonablemente suave y directo pero, en comparación con cualquiera de los tres resultados restantes (basados en *DV*), es menos seguro. Por el contrario, los caminos hallados a partir del uso de *DV* son más seguros y de largo y suavidad similares al del cuadro 4.12a.

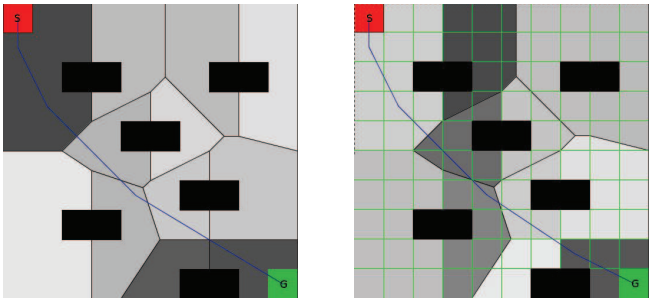
4.2.3. Múltiples tareas

En este punto, una vez realizado el abordaje de las interrogantes planteadas antes y en observancia de la aplicación práctica de ambos enfoques en el control de un robot real capacitado para la realización de múltiples tareas en un entorno, podría corresponder una cuarta interrogante: ¿cuál es el efecto en cada caso (aplicando cada enfoque) de la realización secuencial de múltiples tareas en diferentes lugares del entorno?

En la generación del entorno, y más específicamente, en entender las implicaciones de cada una de las formas de representación utilizada se encuentra la respuesta. En un caso, *DPF*, la generación del entorno debe realizarse cada vez que, habiendo el robot variado su posición inicial, deba planificarse un camino. De modo que si el robot debiera realizar una secuencia de n tareas en k lugares diferentes del entorno, para codificar correctamente los caminos que conducen a dichos destinos, tendrían que generarse en total k diferentes *DPF's*. Por el contrario, en el caso de la propuesta basada en *DV* la codificación de los caminos es independiente de la posición inicial del robot. Resultando en que la tarea computacionalmente más intensa, la generación del entorno, se realiza una única vez.



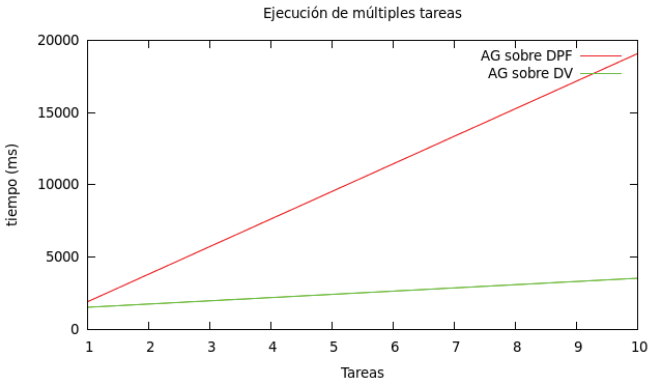
(a) AG sobre DPF. (b) AG sobre DV.



(c) AG sobre DV configurado según DPF. (d) AG sobre DV configurado *ad-hoc*.

Cuadro 4.12: Comparativo gráfico.

En suma, si se consideran entornos estáticos donde el robot deba realizar múltiples tareas, la diferencia entre las dos propuestas (en torno al aspecto de la eficiencia computacional) crece, siendo cada vez más notoria con la realización de más tareas. En el cuadro 4.13 puede apreciarse gráficamente, para un conjunto de diez (10) tareas realizadas secuencialmente, cuál es el peso de la planificación -medido en tiempo de ejecución- cuando se utilizan ambas propuestas.



Cuadro 4.13: Tiempo de ejecución de múltiples tareas.

Capítulo 5

Prueba de concepto

Finalmente, se realizaron pruebas del sistema en un entorno real. El principal objetivo fue comprobar la factibilidad práctica del enfoque utilizado. En tal sentido, se sometió al sistema a cierta incertidumbre en la información de entrada (proveniente de los sensores) y también en la ejecución de las órdenes enviadas al robot.

5.1. Entorno real

Para poder considerar al robot como una partícula (puntual), tanto a nivel de su representación en los modelos como en los algoritmos, se tomó en cuenta la propuesta realizada en [WvH07] que permite tratar al robot como un punto conservando la proporcionalidad entre las áreas ocupadas por los diferentes objetos y el área total del entorno.

De este modo, y a los efectos que la planificación se realice sobre el mismo modelo del entorno que durante las pruebas simuladas, se determinó la resolución de la descomposición aproximada en diez (10) filas y columnas y, a partir del diámetro del robot, se calcularon el tamaño del área total del espacio de configuraciones y de los obstáculos.

El cuadro 5.1 resume las características de la plataforma robótica y del entorno de trabajo utilizados.

Arena	Área rectangular de 1680x1680 milímetros.
Sistema de visión	Doraemon. [AB02]
Robot	Khepera III + KoreBot II. [KT]

Cuadro 5.1: Características del entorno real.

5.2. Arquitectura

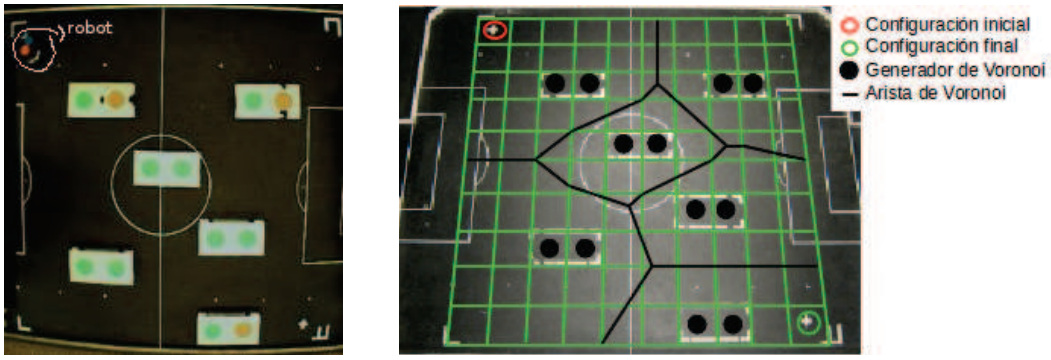
El sistema de visión se eligió en base a la existencia de experiencia previa en su uso para la realización de tareas similares (reconocimiento de objetos móviles y estáticos). La plataforma robótica fue elegida en base a tener experiencia previa en su uso y considerando además que fuera posible a

futuro, por sus características técnicas, la implantación completa en el propio robot de todas las partes componentes del sistema (interface con la visión, planificador, toma de decisiones).

Corresponde señalar entonces que, en esta etapa del desarrollo, el procesamiento de la información se distribuye de la siguiente manera:

- sistema de visión *Doraemon* en una *PC* que oficia de servidor de visión.
- procesamiento de los datos provenientes del servidor de visión y planificación de movimientos en otra *PC* que sirve de servidor de comandos.
- ejecución de comandos en el robot.

En la figura 5.1a se puede apreciar la imagen procesada por el sistema de visión, mientras que en la figura 5.1b se aprecia el modelo del entorno construido a partir de la ubicación del robot y los obstáculos (información enviada desde el sistema de visión).



(a) Imagen procesada por Doraemon. (b) Modelo del entorno.

Figura 5.1: Modelado del entorno real.

Las pruebas se realizaron en condiciones de luminosidad controlada, siendo este un factor clave debido a que de él dependen la correcta identificación de los objetos y la posterior construcción del modelo. La planificación de movimientos se realiza una única vez, exclusivamente, a partir de la primera imagen asumiendo que los obstáculos son estáticos. Las imágenes siguientes sólo se procesan para obtener la ubicación del robot. La infraestructura de red utilizada para posibilitar la comunicación entre los servidores de visión y estrategia consistió en una red tradicional (cableada), mientras que el servidor de comandos y el robot se comunicaban mendiante *WIFI*.

5.3. Control de movimientos

Para realizar el seguimiento de los caminos planificados se utilizó un conjunto de acciónes primitivas. Al considerar las características de los caminos a seguir -representados mediante secuencias de celdas- e intentando simplicar al máximo el control, se determinó el conjunto de acciones de bajo nivel:

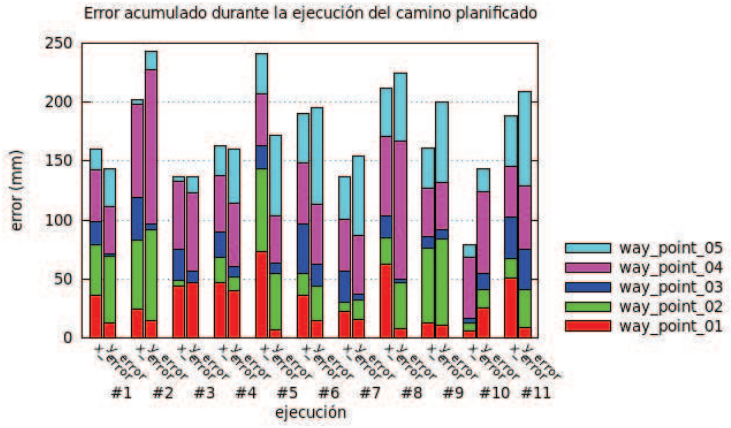
- girar: permite controlar la orientación del robot.

- avanzar: permite controlar el desplazamiento en línea recta del robot.
- detener: permite detener el movimiento del robot.

5.4. Resultados

En total, se realizaron catorce (14) ejecuciones independientes de las cuales once (11) de ellas resultaron exitosas (el robot llegó a la celda donde se encontraba la configuración destino siguiendo el camino libre de colisiones calculado previamente). En el resto de los casos el robot, o bien colisionó con alguno de los obstáculos y se interrumpió la ejecución, o bien recorrió el camino planificado pero no se detuvo en la celda donde se encontraba la configuración destino.

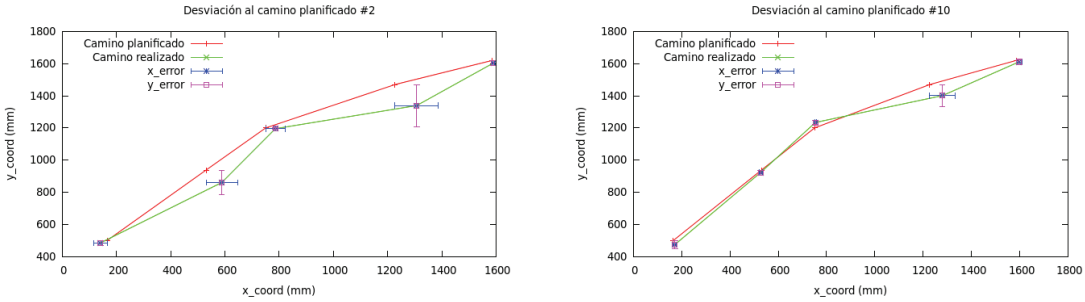
Sobre las corridas exitosas resulta de interés realizar un análisis más detallado. En primer término, se evaluó la desviación entre los caminos planificados y el recorrido realizado finalmente por el robot. En tal sentido, en el cuadro 5.2 se puede ver el error cometido en x e y para cada punto intermedio (*waypoint*) de los caminos planificados. Asimismo, el cuadro 5.3 muestra los caminos planificados y el recorrido final correspondientes a la segunda (mayor acumulación de error) y décima (menor acumulación de error) ejecucion de prueba, respectivamente.



Cuadro 5.2: Error en las ejecuciones.

Los resultados muestran que, en todas las ejecuciones, existe una desviación respecto de lo planificado. La misma se debe a la suma de errores cometidos durante el sensado y la actuación. En el caso del sistema de visión, a pesar de los esfuerzos realizados por establecer un nivel de luminosidad homogeneo e invariante y una buena calibración de colores, ante la presencia de un objeto estático (un obstáculo o el robot detenido) la información resultante contenía errores:

- identificación de objetos: 0 %
- posición del objeto: $\pm 5cm$
- rotación del objeto: $\pm 3^{\circ}$



(a) Ejecución 2.

(b) Ejecución 10.

Cuadro 5.3: Desviación durante la ejecución.

Por el contrario, y aunque el sistema de actuación presentó una presición mayor, el hecho de ejecutar el sistema en una arquitectura distribuida (procesamiento de imágenes, toma de decisiones y control de actuadores en diferentes máquinas) introdujo un retardo entre la elección de una acción y su ejecución que, en los hechos, significó que el robot no se detuviera inmediatamente al elegirse la acción *detener*.

Otro análisis refiere a la seguridad de los recorridos mediante el estudio de las distancias entre el robot y los obstáculos durante las ejecuciones. Particularmente, se podría utilizar el anillo de sensores (distancia) dispuestos alrededor del robot para medir distancias durante la ejecución de los caminos planificados, sin embargo, esta tarea ha quedado para realizarse a futuro. En la figura 5.2 se muestra la disposición de los sensores infrarojos (vistos desde abajo).

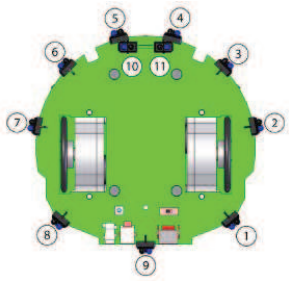
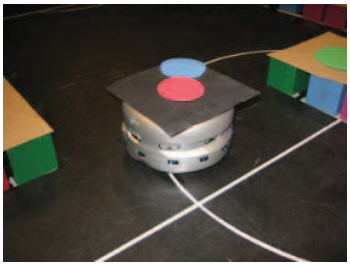
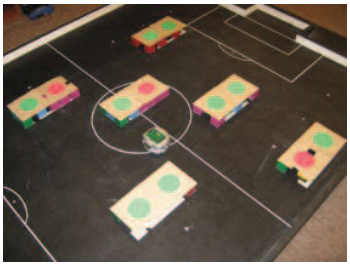


Figura 5.2: Anillo de sensores IR.

Finalmente, en la figura 5.3.a se puede apreciar al robot siguiendo un camino entre los obstáculos, en 5.3.b una vista panorámica de la arena, y en 5.3.c la trayectoria realizada en verde. A su vez, en [Ben10] se puede acceder a videos donde se aprecia al robot durante una ejecución de prueba.



(a) Robot siguiendo un camino.



(b) Vista aérea del escenario.

(c) Trayectoria realizada.

Figura 5.3: Pruebas en el entorno real.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Primeramente, la propuesta desarrollada en el presente documento ha servido para profundizar y afianzar el conocimiento y dominio de determinadas técnicas, métodos y enfoques para la realización de planificación de movimientos con robots móviles.

Del relevamiento realizado surge que, a pesar que ya se ha investigado durante casi tres décadas y se han logrado muchos avances, persisten un conjunto de entornos para los cuales la planificación de movimientos sigue siendo un problema a resolver.

En tal sentido, por su capacidad de atacar problemas de dimensiones grandes en tiempos razonables, los métodos probabilísticos han sido de amplia aplicación a pesar de no ser completos. De este modo, en la práctica, por razón de los buenos desempeños en una amplia variedad de entornos, estos métodos han ido ganándole terreno a los métodos tradicionales aún cuando algunos de estos últimos presentan la propiedad de completitud.

Podría decirse que la necesidad de contar con sistemas de planificación con gran capacidad de cómputo y capaces de responder bajo restricciones fuertes en cuanto al tiempo de respuesta (sistemas de tiempo real) aún a expensas de cierta posibilidad de fallo, en un conjunto muy amplio de escenarios, se ha impuesto a la necesidad de completitud, exactitud y optimalidad.

Por otro lado, respecto a la propuesta desarrollada de un planificador de movimientos para un conjunto particular de entornos, se entiende que el enfoque utilizado es metodológicamente factible y aplicable en sistemas reales de robots móviles.

Estas afirmaciones surgen de verificar la factibilidad de combinar diferentes enfoques para la construcción de una solución y que el tiempo consumido en la realización de los cálculos permite considerar su aplicación en robots reales condicionados por las exigencias de desempeño que presentan los entornos dinámicos.

Esencialmente, la propuesta se basa en la representación del espacio de configuraciones libres a través de la construcción de *roadmaps,* muestreo aleatorio de configuraciones en torno a las aristas del *roadmap* y búsqueda de caminos óptimos con un algoritmo probabilístico. De este modo, se rescatan bondades que presentan algunos de los métodos tradicionales combinándolos con métodos probabilísticos, superando por esa vías algunas de sus deficiencias.

En concreto y para demostrar la factibilidad de la propuesta, la combinación de enfoques se realizó en base a la combinación de dos propuestas previamente desarrolladas. En tal sentido, en comparación con la propuesta presentada en [ZZZ08], los experimentos realizados demuestran que la propuesta desarrollada mejora tanto la calidad de las soluciones alcanzadas como el desempeño en relación al

tiempo invertido para el cómputo de dichas soluciones.

Este último aspecto, a su vez, presenta dos caras. Por un lado, la eficiencia computacional de ejecuciones independientes de cada planificador, y por otro, la eficiencia computacional acumulada a lo largo de una serie de tareas realizadas por un robot real en un entorno dado. Los resultados indican que el planificador desarrollado mejora en ambos casos al desempeño de sus antecesores, siendo más notoria la mejora en el segundo caso. Allí, cuándo un robot se encuentra realizando múltiples tareas que requieren planificación de movimientos, se logran reducir sensiblemente los tiempos de espera por planificación.

Por otro lado, en relación con la propuesta de *Caminos de Visibilidad* presentada en [RD05], no se hizo una implementación de dicha propuesta que permitiera realizar comparaciones respecto al desempeño computacional, sin embargo, la propuesta desarrollada aquí computa caminos que además de la seguridad, pondera positivamente la disminución del largo y el aumento de la suavidad de los mismos. Ambos atributos no son tenidos en cuenta en la propuesta de *Caminos de Visibilidad*, significando esto una clara mejoría.

6.2. Trabajo futuro

El presente trabajo ha servido para estudiar el estado del arte de planificación de movimientos en sistemas robóticos móviles e implementar un planificador aplicable a ciertos entornos. Sin embargo, durante el proceso de desarrollo han surgido un conjunto de aspectos sobre los que se pueden realizar mejoras.

En tal sentido, sería importante extender esta investigación desarrollando las siguientes líneas de trabajo:

- Algoritmos genéticos:
  - Representar mejor el espacio de configuraciones libres utilizando una representación más granular del entorno (sustituyendo la representación con celdas por una representación entera).
  - Explorar mejor el espacio de búsqueda definiendo nuevos operadores de mutación (p.ej. operadores que agreguen nodos a un camino).
  - Utilizar un enfoque multi-objetivos para representar mejor la relación entre las diferentes funciones a optimizar (minimizar la distancia, y maximizar la seguridad y suavidad de los caminos). De este modo se puede estudiar de mejor forma un conjunto más amplio de soluciones.
- Software embebido: contar con un sistema que se ejecute totalmente en el robot (*on-board*). En este sentido, una posibilidad, que permitiría reutilizar el planificador y el software que resuelve la interacción con el sistema de visión utilizado (ambos desarrollados en Java), sería probar la versión de la máquina virtual desarrollada específicamente para sistemas embebidos. Este aspecto, además, contribuiría a disminuir los retardos existentes entre la elección de acciones y su ejecución.
- Entornos: trabajar en entornos más realistas.
  A tales efectos sería conveniente considerar:
  - indicadores de complejidad de entornos: para ponderar adecuadamente los resultados alcanzados por la planificación.

- modelar mejor los obstáculos pertenecientes al entorno (paredes, muebles, etc) utilizando diagramas generalizados de Voronoi que permitan definir figuras no puntuales como generadores del diagrama (líneas y polígonos).
- incorporar el aspecto dinámico y el conocimiento parcial al sistema de planificación.
  - migrar el sistema a una arquitectura híbrida[1]. Para detectar obstáculos móviles o desconocidos se debe incorporar sensado local permitiendo al sistema reaccionar ante eventos no planificados.
  - realizar planificación multi-capas que permita el cálculo de planes de largo plazo y correcciones basadas en planes de corto plazo (se relaciona con el ítem anterior).
  - a los efectos de tratar el conocimiento parcial sobre el entorno, integrar capacidades de construcción de mapas y localización en simultáneo.

[1] A principios de la década de 1990 se propuso una nueva arquitectura de control deliberativo/reactivo (paradigma híbrido) con el propósito de incorporar funciones cognitivas que permitiesen a los robots planificar y deliberar pero sin perder las ventajas demostradas por el paradigma reactivo. [Mur02]

Bibliografía

[AB02] J. Anderson and J. Baltes, *Doraemon user manual*, 2002, Visitada: Jul/12.

[Ark98] R. C. Arkin, *Behavior-Based Robotics*, MIT Press, 1998.

[ATBM93] J. M. Ahuactzin, E. Talbi, P. Bessiere, and E. Mazer, *Using genetic algorithms for robot motion planning*, Lecture Notes in Computer Science, vol. 708, pp. 84–93, Springer Berlin / Heidelberg, 1993.

[Bek05] G. A. Bekey, *Autonomous Robots: From Biological Inspiration to Implementation and Control*, MIT Press, 2005.

[Ben10] F. Benavides, *Home Page*, 2010, Visitada: Jul/12.

[BG08] P. Bhattacharya and M. L. Gavrilova, *Roadmap-Based Path Planning: Using the Voronoi Diagram for a Clearance-Based Shortest Path*, IEEE Robotics & Automation Magazine **15** (2008), no. 2, 58–66.

[BYW08] Z. Bi, Y. Yimin, and Y. Wei, *Hierarchical Path Planning Approach for Mobile Robot Navigation under the Dynamic Environment*, The IEEE International Conference on Industrial Informatics, July 2008, pp. 372–376.

[Car06] S. Carpin, *Algorithmic Motion Planning: The Randomized Approach*, Lecture Notes in Computer Science, vol. 4123, pp. 740–768, Springer Berlin / Heidelberg, December 2006.

[CFR10] R. Calvo, M. Figueiredo, and R. A. F. Romero, *Autonomous cognition and reinforcement learning for mobile robots*, The 2010 International Joint Conference on Neural Networks, October 2010, pp. 1–8.

[CLH+05] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion-Theory, Algorithms, and Implementation*, MIT Press, 2005.

[CY07] Y. Chen and S. Yasunobu, *Soft Target Based Obstacle Avoidance for Car-like Mobile Robot in Dynamic Environment*, IEEE International Fuzzy Systems Conference, July 2007, pp. 1–6.

[DCXG08] Z. Dehuai, X. Cunxi, L. Xuemei, and X. Gang, *Adaptive Niche Genetic Algorithm Based Path Planning and Dynamic Obstacle Avoidance of Mobile Robots*, Proceedings of the IEEE International Conference on Automation and Logistics, 2008.

[DCY08] Y. Duan, B. Cui, and H. Yang, *Robot Navigation Based on Fuzzy RL Algorithm*, Lecture Notes in Computer Science, vol. 5263, pp. 391–399, Springer Berlin / Heidelberg, September 2008.

[Dij59] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik **1** (1959), 269–271.

[Diu10] C. Diuk, *Aprendizaje por Refuerzos: Teoría y Aplicaciones en Robótica, Psicología y Neurociencias*, 2010.

[DNYS10] X. Dai, X. Ning, Z. Yao, and H. Shao, *Integrating Cloud Model in Evolutionary Algorithm for Path Planning of Mobile Robots*, Proceedings of the 2010 IEEE International Conference on Information and Automation, June 2010, pp. 2352–2356.

[FGLM01] M. Foskey, M. Garber, M. C. Lin, and D. Manocha, *A Voronoi-based hybrid motion planner*, Proceedings of the 2001 International Conference on Intelligent Robots and Systems, vol. 1, 2001, pp. 55–60.

[FL98] R. Fierro and F. L. Lewis, *Control of a nonholonomic mobile robot using neural networks*, IEEE Transaction on Neural Networks **9** (1998), no. 4, 589–600.

[GbHL05] H. González-baños, D. Hsu, and J. C. Latombe, *Motion planning: Recent developments*, Tech. report, 2005.

[Gol89] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley Professional, Upper Saddle River,NJ, USA, 1989.

[GXT08] M. Gao, J. Xu, and J. Tian, *Mobile Robot Global Path Planning Based on Improved Aumented Ant Colony Algorithm*, Second International Conference on Genetic and Evolutionary Computing, September 2008, pp. 273–276.

[GXTW08] M. Gao, J. Xu, J. Tian, and H. Wu, *Path Planning for Mobile Robot Based on Chaos Genetic Algorithm*, Fourth International Conference on Natural Computation, 2008, pp. 409–413.

[HGD08] I. Hassanzadeh, H. Ghadiri, and R. Dalayimilan, *Design and Implementation of a Simple Fuzzy Algorithm for Obstacle Avoidance Navigation of a Mobile Robot in Dynamic Environment*, Proceedings of the 5th International Symposium on Mechatronics and its Applications, May 2008.

[HX07] X. Hu and C. Xie, *Niche Genetic Algorithm for Robot Path Planning*, Third International Conference on Natural Computation, vol. 2, IEEE, August 2007, pp. 774–778.

[IMS07] R. Iraji and M. T. Manzuri-Shalmani, *A New Fuzzy-Based Spatial Model for Robot Navigation among Dynamic Obstacles*, 2007 IEEE International Conference on Control and Automation, May 2007.

[Jan04] D. Janglova, *Neural Networks in Mobile Robot Motion*, International Journal of Advanced Robotic Systems **1** (2004), no. 1, 15–22.

[JQ10] H. Jun and Z. Qingbao, *Multi-objective Mobile Robot Path Planning Base on Improved Genetic Algorithm*, IEEE International Conference on Intelligent Computation Technology and Automation (2010), 752–756.

[KL08] L. E. Kavraki and S. M. LaValle, *Motion Planning*, ch. 5, pp. 109–131, Springer, 2008.

[KLM96] L. P. Kaelbling, M. L. Littman, and A. W. Moore, *Reinforcement Learning: A Survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[KMRB10] S. Khanmohammadi, M. K. Mirnia, K. Rezvani, and M. A. Badamchizadeh, *Multi AGV hybrid path planning using fuzzy inference systems*, The 2nd International Conference on Computer and Automation Engineering, vol. 1, April 2010, pp. 789–792.

[KSLO96] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, IEEE Transaction on Robotics and Automation **12** (1996), no. 4, 566–588.

[KT] K-Team, *Khephera III*, Visitada: Jul/12.

[Lat91] J. C. Latombe, *Robot Motion Plannig*, Kluwer Academic Publishers, 1991.

[LK97] S. Lee and G. Kardaras, *Collision-Free Path Planning with Neural Networks*, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, April 1997, pp. 3565–3570.

[LLC06a] Q. Liu, Y. G. Lu, and X. Cunxi, *Fuzzy Obstacle-avoiding Controller of Autonomous Mobile Robot Optimized by Genetic Algorithm under Multi-obstacles Environment*, Proceedings of the 6th World Congress on Intelligent Control and Automation, June 2006, pp. 3255–3259.

[LLC06b] ———, *Optimal Genetic Fuzzy Obstacle Avoidance Controller of Autonomous Mobile Robot Based on Ultrasonic Sensors*, Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics, December 2006, pp. 125–129.

[LLP+05] G. Liu, T. Li, Y. Peng, et al., *The ant algorithm for solving robot path planning problem*, Proceedings of the Third International Conference on Information Technology and Applications, vol. 2, IEEE, July 2005, pp. 25–27.

[LTXZ06] Q. Li, X. Tong, S. Xie, and Y. Zhang, *Optimun Path Planning for Mobile Robots Based on a Hybrid Genetic Algorithm*, Proceedings of the Sixth International Conference on Hybrid Intelligent Systems, 2006.

[LWW06] L. Lei, H. Wang, and Q. Wu, *Improved Genetic Algorithms Based Path planning of Mobile Robot Under Dynamic Unknown Environment*, Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation, IEEE, June 2006, pp. 1728–1832.

[LZY+06] Q. Li, W. Zhang, Y. Yin, Z. Wang, and G. Liu, *An Improved Genetic Algorithm of Optimun Path Planning for Mobile Robots*, Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications, 2006.

[MABS08] M. Majdi, H. S. Anvar, R. Barzamini, and S. Soleimanpour, *Multi AGV Path Planning in Unknown Environment Using Fuzzy Inference Systems*, 3rd International Symposium on Communications, Control and Signal Processing, June 2008, pp. 172–177.

[MAT08] M. Mansouri, M. Aliyari Shoorehdeli, and M. Teshnehlab, *Integer GA for Mobile Robot Path Planning with using another GA as repairing function*, Proceedings of the IEEE International Conference on Automation and Logistics, IEEE, September 2008, pp. 135–140.

[Mit98] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, USA, 1998.

[MS10] R. Maurya and A. Shukla, *Generalized and modified ant algorithm for solving robot path planning problem*, 3rd IEEE International Conference on Computer Science and Information Technology **1** (2010), 643–646.

[Mur02] R. R. Murphy, *Introduction to AI Robotics*, MIT Press, 2002.

[Ove02] M. H. Overmars, *Recent Developments in Motion Planning*, Lecture Notes in Computer Science, vol. 2331, pp. 3–13, Springer Berlin / Heidelberg, January 2002.

[Par05] D. R. Parhi, *Navigation of Mobile Robots Using a Fuzzy Logic Controller*, Journal of Intelligent and Robotic Systems **42** (2005), no. 3, 253–273.

[PPP06] S. K. Pradhan, D. R. Parhi, and A. K. Panda, *Neuro-fuzzy technique for navigation of multiple mobile robots*, Fuzzy Optimization and Decision Making **5** (2006), no. 3, 255–288.

[QFHR09] J. Qiao, R. Fan, H. Han, and X. Ruan, *Q-Learning Based on Dynamical Structure Neural Network for Robot Navigation in Unknown Environment*, Lecture Notes in Computer Science, vol. 5553, pp. 188–196, Springer Berlin / Heidelberg, May 2009.

[RD05] W. L. Roque and D. Doering, *Trajectory planning for lab robots based on global vision and Voronoi roadmaps*, Robotica **23** (2005), no. 4, 467–477.

[RN10] S. Russell and P. Norvig, *Artificial Intelligence: A modern approach*, Pearson, 2010.

[Roq96] W. L. Roque, *Introducao a Técnicas de Planejamento de Trajetórias de Robôs Móveis*, vol. 1, pp. 125–150, -, 1996.

[Sah06] M. Saha, *Motion Planning with Probabilistic Roadmaps*, Ph.D. thesis, Stanford University, August 2006.

[SB98a] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, Journal of Cognitive Neuroscience **11** (1998), 126–134.

[SB98b] ———, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[SF93] T. Shibata and T. Fukuda, *Intelligent Motion Planning by Genetic Algorithm with Fuzzy Critic*, Proceedings of the 1993 International Symposium on Intelligent Control, August 1993, pp. 565–570.

[SFM06] M. Shimizu, M. Fujita, and H. Miyamoto, *Trajectory Generation for a Mobile Robot by Reinforcement Learning*, Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment, Springer Berlin / Heidelberg, 2006.

[SN04] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Intelligent Robotics and Autonomous Agents, MIT Press, 2004.

[SS83] J. T. Schwartz and M. Sharir, *On the piano movers' problem: II General techniques for computing topological properties of real algebraic manifolds*, Advances in applied Mathematics **4** (1983), no. 3, 298–351.

[SWI06] R. Syam, K. Watanabe, and K. Izumi, *An Adaptive Actor-critic Algorithm with Multi-step Simulated Experiences for Controlling Nonholonomic Mobile Robots*, Soft Computing - A Fusion of Foundations, Methodologies and Applications **11** (2006), no. 1, 81–89.

[TM06] G. Tan and D. Mamady, *Real-Time Global Optimal Path Planning of Mobile Robots Based on Modified Ant System Algorithm*, Lecture Notes in Computer Science, vol. 4222, pp. 204–214, Springer Berlin / Heidelberg, September 2006.

[TTL+02] K. C. Tan, K. K. Tan, T. H. Lee, et al., *Autonomous robot navigation based on fuzzy sensor fusion and reinforcement learning*, Proceedings of the 2002 IEEE International Symposium on Intelligent Control, IEEE, October 2002, pp. 182–187.

[vdB07] J. P. van der Berg, *Path Planning in Dynamic Environments*, Ph.D. thesis, Utrecht University, 2007.

[VVLC07] N. A. Vien, N. H. Viet, S. G. Lee, and T. C. Chung, *Obstacle Avoidance Path Planning for Mobile Robot Based on Ant-Q Reinforcement Learning Algorithm*, Lecture Notes in Computer Science, vol. 4491, pp. 704–713, Springer Berlin / Heidelberg, July 2007.

[WGA06] A. M. Wahdan, W. Gharieb, and M. M. Attiah, *A Practical Genetic Algorithm for Obstacles Avoidance Robot Motion Planning*, Tech. report, Computer and Systems Department, Ain Shams University, 2006.

[WMSS08] Y. Wang, D. Mulvaney, I. Sillitoe, and E. Swere, *Robot Navigation by Waypoints*, Journal of Intelligent and Robotic Systems **52** (2008), no. 2, 175–207.

[WQY06] J. F. Wu, D. X. Quin, and H. P. Yu, *Nonholonomic Motion Planning of Mobile Robot with Ameliorated Genetic Algorithm*, Proceedings of the 2006 IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2006.

[WvH07] R. Wein, J. P. van den Berg, and D. Halperin, *The visibility–Voronoi complex and its applications*, Computational Geometry: Theory and Applications **36** (2007), no. 1, 66–87.

[YWY+07] X. Yan, Q. Wu, J. Yan, et al., *A Fast Evolutionary Algorithm for Robot Path Planning*, 2007 IEEE International Conference on Control and Automation, June 2007, pp. 84–87.

[YYY+08] L. Yanju, C. Yundong, Y. Yu, D. Wang, W. Xin, Y. Jun, et al., *A path planning study of autonomous mobile robot in dynamic environment*, 3rd IEEE International Conference on Industrial Electronics and Applications (2008), 1040–1043.

[ZXLY09] P. Zhang, X. Xu, C. Liu, and Q. Yuan, *Reinforcement Learning Control of a Real Mobile Robot Using Approximate Policy Iteration*, Lecture Notes in Computer Science, vol. 5553, pp. 278–288, Springer Berlin / Heidelberg, May 2009.

[ZY06] D. Zhao and J. Yi, *Robot Planning with Artificial Potential Field Guided Ant Colony Optimization Algorithm*, Lecture Notes in Computer Science, vol. 4222, pp. 222–231, Springer Berlin / Heidelberg, September 2006.

[ZZZ08] Y. Zhang, L. Zhang, and X. Zhang, *Mobile Robot Path Planning base on the Hybrid Genetic Algorithm in Unknown Environment*, ISDA '08: Proceedings of the 2008 Eighth International Conference on Intelligent Systems Design and Applications (Washington, DC, USA), vol. 2, IEEE Computer Society, November 2008, pp. 661–665.

Parte IV

Relevamiento del estado del arte

Apéndice A

Robótica autónoma móvil

A.1. Antecedentes: camino hacia la autonomía

En el contexto de la segunda guerra mundial y posteriormente bajo la presión de una carrera por la construcción de armas nucleares liderada por la Unión Soviética y los Estados Unidos, se desarrollaron los primeros Telemanipuladores. Los logros obtenidos en la industria nuclear promovieron la inclusión de manipuladores industriales o brazos robóticos en otras áreas de aplicación, especialmente en la industria manufacturera. Estos robots fueron desarrollados principalmente para realizar en forma repetitiva tareas con un alto grado de precisión y velocidad.

Por otro lado, en la década de 1960 y bajo la influencia de la carrera espacial, se comenzaron a desarrollar robots móviles basados en Inteligencia Artificial. El enfoque ingenieril aplicado hasta el momento para la robótica industrial no era suficiente para resolver la evidente necesidad de dotar a los robots exploradores de autonomía.

De este modo, y en forma creciente hasta nuestros días, la robótica autónoma móvil ha ido concentrando la atención de los principales centros de investigación dedicados a la robótica e inteligencia artificial. [Mur02]

A.2. Agentes Robóticos Inteligentes

¿Cómo deberían ser? ¿Qué forma deberían tener? ¿Qué tarea deberían ser capaces de resolver?

Estas son algunas de las interrogantes que comúnmente surgen al pensar en agentes robóticos inteligentes. De igual modo, es frecuente asociar las capacidades y forma de un robot a la tarea para la cual fue desarrollado. Sin embargo, a estos aspectos debe sumársele un tercero que refiere a las condiciones en las cuáles debe realizarse la tarea. Es decir, en qué ambiente o entorno deberá desenvolverse el robot durante su vida es también un factor determinante. [Bek05]

En lo sucesivo, este enfoque ecológico, donde las metas del robot y las características del entorno condicionan o influencias fuertemente su diseño, será el considerado para el desarrollo de los diferentes temas.

A.2.1. Autonomía e inteligencia

Un robot autónomo es una máquina inteligente capaz de realizar tareas en un entorno por si mismo, sin que un humano controle explícitamente sus movimientos.[Bek05]

A.2.1.1. Racionalidad y omnisciencia

Un agente racional es aquel que hace lo correcto. En tal sentido, se puede decir que lo correcto es aquello que permite al agente obtener un mejor resultado. De modo que es necesario determinar una forma de medir el éxito. [RN10]

La racionalidad en un momento dado depende de cuatro factores:

- la medida de rendimiento
- el conocimiento acumulado por el agente sobre el medio
- las acciones que puede realizar el agente
- la secuencia de percepciones del agente hasta ese momento

Agente racional En cada posible secuencia de percepciones, un agente racional deberá emprender aquella acción que supuestamente maximice su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en el conocimiento que el agente mantiene almacenado. [RN10]

Agente omnisciente Es aquel que conoce el resultado de sus acciones y actúa de acuerdo con ello. [RN10]

Lamentablemente, la omnisciencia no es posible ya que los resultados no dependen exclusivamente de la "voluntad" del agente. Es importante pues, remarcar la diferencia entre racionalidad y perfección. Mientras la primera maximiza el rendimiento esperado, la segunda maximiza el resultado real.

A.2.1.2. Aprendizaje

En tal sentido, considerar al agente robótico como agente racional no implica solamente que sea capaz de recopilar información de interés, sino, que aprenda lo máximo posible a partir de dicha información. Esta afirmación se debe, básicamente, al hecho que son excepcionales los casos en los que se conoce totalmente el entorno a priori. Para el resto de los casos, resulta de gran valía que el agente sea capaz de aprender a reconocer un entorno potencialmente desconocido y cambiante. [Mur02]

A.2.1.3. Autonomía

Autonomía refiere a la capacidad de realizar tareas en un mundo real sin ninguna forma de control externo por períodos prolongados de tiempo.

Por el contrario, se dice que un agente carece de autonomía cuando, para tomar sus decisiones, se apoya más en el conocimiento previo que le fue proporcionado que en su propia experiencia.

Un agente racional debe ser autónomo puesto que debe aprender a determinar como compensar el conocimiento inicial incompleto. [Mur02]

A.2.2. Entornos de trabajo

Los entornos de trabajo son esencialmente los "problemas" para los cuales los agentes racionales son la "solución". En el diseño de un agente robótico el primer paso debe ser especificar el entorno de trabajo de la forma más completa posible. A tales efectos, los elementos que deben determinarse son: las medidas de rendimiento, el entorno y los actuadores y sensores del agente. [RN10]

A.2.2.1. Clasificación

El rango de entornos de trabajo para los que pueden diseñarse agentes robóticos es muy grande y variado. Sin embargo, pueden identificarse algunas dimensiones que permiten categorizar dichos entornos. [RN10]

Esta categorización resulta de utilidad ya que contribuye con la determinación del diseño más apropiado para el agente y sugiere qué subconjunto de técnicas son de aplicación en la implementación del mismo.

Totalmente observable vs parcialmente observable Un entorno es totalmente observable si a través de los sensores que posee el agente se logra acceder a toda la información relevante para la toma de decisiones. La relevancia a su vez está en relación directa con las medidas de rendimiento.

Determinista vs estocástico Si el próximo estado del entorno puede determinarse a partir del estado actual y la acción que elige el agente, el entorno es determinista. En caso contrario, estocástico. A su vez, si es determinista exceptuando las acciones de otros agentes, también suele denominársele *estratégico*.

Episódico vs secuencial En entornos de trabajo episódicos la experiencia del agente se divide en episodios. Es decir, las percepciones y acciones del agente se restringen a cada episodio de modo que no afectan las percepciones y acciones en el siguiente episodio. Por el contrario, en un entorno secuencial, la decisión presente puede afectar decisiones futuras.

Estático vs dinámico Si el entorno puede cambiar mientras el agente toma una decisión, el entorno es dinámico para el agente. Los entornos estáticos son más fáciles de tratar ya que el agente no debe preocuparse por el paso del tiempo mientras decide. Cuando el entorno no cambia mientras el agente delibera pero su rendimiento decrece con el tiempo, el entorno se denomina *semi-dinámico*.

Discreto vs continuo La distinción entre discreto y continuo puede aplicarse tanto al estado del medio, el manejo del tiempo y a percepciones y acciones del agente. Dependiendo cómo sean esos conjuntos de valores es como se clasifica el entorno.

Agente individual vs multiagente La distinción entre el entorno de un agente individual y uno multiagente suele parecer simple. Claramente, existen entornos donde solo actúa un agente. Sin embargo, otros entornos donde actúan varios agentes pueden o no ser considerados como multiagentes. Básicamente, la diferencia depende de cuales entidades deben considerarse como agentes. Si el comportamiento de una entidad está mejor descripta por la maximización de una medida de rendimiento, entonces sería conveniente considerarla como un agente. Por el contrario, podría considerársele simplemente como un objeto cuyo comportamiento es estocástico. En consecuencia, pueden considerarse entornos multiagente competitivos -donde la maximización de la medida de rendimiento de unos minimizan la medida de rendimiento de los otros- o cooperativos -donde maximizar la medida de un agente contribuye a maximizar la medida de rendimiento del resto-.

A.2.3. Agente robótico

Un agente es cualquier cosa capaz de percibir su medio ambiente con la ayuda de sensores y actuar en ese medio utilizando actuadores. [RN10]

Un robot inteligente es una máquina capaz de extraer información del entorno y utilizar dicho conocimiento para moverse en forma segura mientras realiza una tarea significativa persiguiendo un propósito. [Ark98]

Un robot se distingue de un agente de software por el hecho de estar situados en un mundo real, tiene dimensiones físicas y puede ejercer fuerza sobre otros objetos. [Bek05]

A.2.3.1. Sensores

Un agente robótico necesita sensores para dos tareas esenciales: percibir su entorno y monitorear su estado interno. En tal sentido, los mismos pueden clasificarse en dos categorías: extero y propioceptivos, distinguiéndolos entre los que posibilitan el sensado del medio exterior o estado interno, respectivamente.

A su vez, frecuentemente el diseño o principio en el que se basan los sensores está inspirado en los sistemas sensoriales de especies animales. [Bek05]

A.2.3.2. Actuadores

Un agente robótico debe estar equipado para interactuar con el entorno. Más precisamente, debe poder modificarlo o moverse en el.

Algunos de los tipos de actuadores más comunes son:

- músculos artificiales: aún muy lejos de ser buenas aproximaciones a los músculos biológicos. Típicamente, pueden ser aleaciones con memoria de forma, polímeros electroactivos o de tipo *McKibben*[1].
- motores eléctricos: comúnmente utilizados en robots móviles tanto para locomoción o para manipulación. Típicamente, pueden ser de corriente continua, paso a paso o servomotores.
- actuadores neumáticos o hidráulicos: muy utilizados en el industria para tareas de manipulación y muy raramente en robótica móvil.

A su vez, dependiendo del tipo de movimiento que generan, pueden clasificarse en: prismáticos (o telescópicos) y rotacionales (o de revoluta). [Bek05]

[1] Pertenecientes a la categoría de músculos artificiales neumáticos, fueron desarrollados por primera vez en la década de 1950 para uso ortopédico. Treinta años más tarde fueron comercializados bajo el nombre de *Rubbertuators*.

Apéndice B

Planificación de movimientos

[1]

B.1. Descripción del problema

La planificación de movimientos es uno de los mayores problemas a resolver durante la creación de un robot autónomo. No es el único e interactúa con otros también muy importantes como: control en tiempo real, sensado y planificación de tareas. En tal sentido, crear robots autónomos requiere, definitivamente, dotarlos del poder de decidir automáticamente qué movimientos realizar para lograr sus objetivos o de otro modo, dotarlos de la habilidad de planificar automáticamente sus movimientos.

El objetivo de la planificación de movimientos, en términos generales, es que dada una especificación de una tarea utilizando un lenguaje de alto nivel, el robot sea capaz de descomponerla automáticamente en un conjunto de movimientos primitivos que le permitan cumplir con lo solicitado. Típicamente, esta tarea suele ser la búsqueda de caminos óptimos o cuasi-óptimos, libres de colisiones, entre un estado o configuración inicial y uno final, respetando ciertos criterios de evaluación. Los más frecuentemente utilizados son la distancia recorrida, el ahorro de energía, la suavidad y seguridad de las trayectorias.

Desde su primera versión, del paradigmático problema de mover un piano [SS83], la planificación de movimientos ha evolucionado aplicándose tanto a brazos robóticos como a robots móviles, en un conjunto variado de problemas: animación digital, cirugía, verificación automática de diseño de fábricas, construcción de mapas en entornos desconocidos, navegación en entornos dinámicos, ensamblaje, diseño de medicamentos, entre otras.

B.2. Elementos de interés

Una forma simple, pero muy utilizada, de pensar el desarrollo de planificadores de movimientos toma en cuenta la tarea a realizar, las propiedades del robot y del algoritmo de planificación.

Tareas Uno de los criterios más fuertes es considerar el problema de más alto nivel para el que será utilizado el planificador. A modo de ejemplo, navegación, cubrimiento, localización, cartografía, manipulación, ensamblaje y coordinación, son una lista ilustrativa de las tantas tareas en las que se requiere planificar el movimiento.

[1] Para la elaboración del presente capítulo [Lat91, SN04, CLH+05] fueron las principales referencias bibliográficas consultadas. Las restantes se señalan puntualmente en el texto.

Propiedades del robot La forma del robot y la disposición de los actuadores condicionan fuertemente la planificación a realizar. De ellas se desprenden los grados de libertad y restricciones del sistema (holonómico o no), así como la conveniencia de utilizar modelos cinemáticos o dinámicos para los controles.

Propiedades del algoritmo Una vez definida la tarea a resolver y el robot a utilizar, debe elegirse el algoritmo. En este sentido, son de interés tres conceptos: optimalidad, completitud y complejidad computacional. La primera refiere a la calidad de la salida que provee el algoritmo. La segunda refiere a la capacidad de encontrar una respuesta siempre que exista. Finalmente, la tercera puede medirse teniendo en cuenta si el algoritmo es intensivo en el uso de la memoria o los cálculos. Frecuentemente, es necesario encontrar un punto de equilibrio donde se logren resultados de calidad aceptable con un uso razonable de los recursos computacionales. A tales efectos, la propiedad de completitud suele relajarse considerándose dos variantes: completitud dependiente de la resolución[2] y completitud estadística o probabilística[3].

B.3. Formalización

B.3.1. Problema básico

El principal motivo para definir una formulación básica, al problema de planificación de movimientos, es lograr aislar algunos aspectos considerados centrales e investigarlos en profundidad, dejando algunas dificultades adicionales para un análisis posterior.

En el problema básico se asume que el robot es el único objeto móvil del entorno y se ignoran, también, los aspectos dinámicos de su movimiento. También se desprecian las interacciones entre los objetos físicos, transformando un problema de planificación de movimientos físicos en un problema de planificación de caminos puramente geométrico. Además, se reduce la complejidad del aspecto geométrico asumiendo que el robot es un cuerpo rígido no articulado.

De todas estas simplificaciones surge que la formulación para el problema básico de planificación de movimientos es la siguiente:

Sea A un objeto rígido (*"el robot"*) moviéndose en un espacio euclidiano W, llamado área de trabajo o entorno, representado por R^n ($n=2$ o $n=3$). Sean O_1, ..., O_q un conjunto de objetos rígidos distribuidos en W, llamados obstáculos. Si se asume que tanto la geometría de A y O_i como la ubicación de O_i en W son conocidas y que no existen restricciones cinemáticas que limiten el movimiento de A, el problema básico de planificación de movimientos es:

Dada una posición y orientación inicial y una posición y orientación objetivo de A en W, generar un camino especificando una secuencia continua de posiciones y orientaciones para A evitando colisionar con O_i, comenzando desde la posición y orientación inicial y terminando en la posición y orientación objetivo.

B.3.2. Espacio de configuración

Para crear planes de movimiento para robots se debe, primero, conocer una especificación completa de la localización de cada punto del robot de modo de asegurar que no se produzcan colisiones. A tales efectos, sobrevienen las siguientes preguntas: ¿cuánta información es necesaria para especificar completamente la posición de todos y cada uno de los puntos de un robot? ¿cómo debe representarse dicha información? ¿cuáles son las propiedades matemáticas de dicha representación? ¿cómo pueden

[2] Si existe solución para un cierto nivel de resolución con el que se representa el entorno, el algoritmo la encontrará.
[3] La probabilidad de encontrar una solución tiende a 1 a medida que el tiempo tiende a infinito

tenerse en cuenta los obstáculos presentes en el entorno del robot para realizar la planificación del movimiento?

B.3.2.1. Configuración de un robot

Con el propósito de comenzar a responder estas interrogantes de forma precisa serán necesarias las siguientes definiciones.

Configuración q es la especificación de un punto del sistema.

Configuración del robot *A(q) o R(q)* es la especificación completa de la posición de todos los puntos de un sistema robótico.

Espacio de configuraciones Q es el espacio de todas las posibles configuraciones del sistema. También se conoce como *C-Space* denotado por C[4].

Grados de libertad La cantidad de grados de libertad de un sistema robótico es igual a la dimensión del espacio de configuraciones o, de otro modo, el número mínimo de parámetros requerido para especificar una configuración.

A partir de estas definiciones se puede reformular el problema de planificación de movimientos como el problema de determinar una correspondencia continua $c : [0,1] \rightarrow Q$ o camino desde una configuración inicial $c(0)=q_{init}$ hasta una configuración destino $c(1)=q_{goal}$, tal que ninguna configuración q_i perteneciente al camino causa una colisión entre el robot y un obstáculo.

B.3.2.2. Obstáculos y espacio de configuraciones

Espacio de configuraciones de un obstáculo CO_i es el conjunto de configuraciones en las cuales el robot intersecta con el obstáculo O_i en el entorno de trabajo W.

$$CO_i = \{q \epsilon C \mid A(q) \cap O_i \neq \varnothing\}$$

Espacio de configuraciones libres C_{free} es el conjunto de las configuraciones en las cuales el robot no intersecta ningún obstáculo presente en el entorno de trabajo.

$$C_{free} = C \setminus \left(\bigcup_i CO_i \right) = \left\{ q \epsilon C \mid A(q) \cap \bigcup_i CO_i = \varnothing \right\}$$

Finalmente, a partir de estas nuevas definiciones, se puede definir formalmente un camino libre de colisiones como la correspondencia continua $c : [0,1] \rightarrow C_{free}$ y un camino semi-libre como la correspondencia continua $c : [0,1] \rightarrow cl(C_{free})$ en la cual $cl(C_{free})$ denota la clausura de C_{free}.

B.3.3. Extensiones al problema básico[5]

B.3.3.1. Múltiples objetos móviles

Existen tres extensiones relacionadas con la movilidad de diferentes objetos del entorno.

[4] Es la notación particular de Q dentro de la comunidad de planificación de movimientos.
[5] Para la redacción de esta sección se ha consultado además [vdB07] como referencia bibliográfica.

Obstáculos móviles

Primeramente, se considera que uno o todos los obstáculos en el entorno se están moviendo. El problema resultante se conoce como *Problema dinámico de planificación de movimientos*. A diferencia del problema básico (estático), el problema dinámico no puede resolverse en base a la construcción de una camino geométrico. En cambio, debe generarse una función continua que retorne la configuración del robot para cada instante de tiempo. El enfoque más general suele ser agregar la dimensión temporal al espacio de configuraciones del robot. El robot se transforma, en este nuevo espacio de configuración temporal, en un punto moviéndose entre obstáculos estacionarios.

Cabe observar que la dimensión temporal, a diferencia de las otras dimensiones del espacio, es irreversible. En tal sentido, varios de los métodos o enfoques tradicionales (ver B.4) son aplicables al problema dinámico pero deben realizarse modificaciones o extensiones que tomen en cuenta las especificidades de la dimensión temporal. A su vez, el Problema dinámico de planificación puede hacerse más realista introduciendo restricciones a la velocidad y aceleración del robot. Sin embargo, mayormente suelen abordarse sólo dos situaciones: velocidad ilimitada y velocidad limitada.

Espacio temporal de configuraciones

Sea A un objeto rígido moviéndose en un entorno $W{=}R^n$ ($n{=}2$ o $n{=}3$), poblado con obstáculos O_i (objetos rígidos potencialmente en movimiento) con $i{=}1, \ldots, q$. $O_i(t)$ determina la región de W ocupada por el objeto O_i en el tiempo t. Si además se puede asumir que el movimiento realizado por los O_i es conocido de antemano y el mismo no está influenciado por el movimiento de A, considerando que C sea el espacio de configuraciones de A (definido como en B.3.2.1), en todo instante de tiempo t, los obstáculos $O_i(t)$ determinan un conjunto de configuraciones en C, definido según la ecuación B.1:

$$CO_i(t) = \{q \in C \mid A(q) \cap O_i(t) \neq 0\} \tag{B.1}$$

Es importante notar que la forma de $O_i(t)$ no depende de t, pero su ubicación en C si lo hace.

Problema dinámico de planificación de movimientos Es planificar una secuencia de movimientos -libres de colisión- para A desde la configuración inicial q_{init} en $t{=}0$ hasta la configuración objetivo q_{goal} en el instante de tiempo $T{\geq}0$. T se conoce comúnmente como tiempo de llegada.

Este problema requiere un función que especifique las configuraciones que debe tomar A en todos los instantes de tiempo del intervalo $[0, T]$. Formalmente, una solución al problema es una función continua de la forma presentada en la ecuación B.2:

$$t \in [0, T] \mapsto q(t) \in C \setminus \bigcup_{i=1}^{q} CO_i(t) \tag{B.2}$$

con t representando el tiempo, $q(0){=}q_{init}$ y $q(T){=}q_{goal}$. A esta función también se le conoce como *Trayectoria* de A. De aquí surge que el problema dinámico de planificación de movimientos también sea conocido como *Planificación de trayectorias*.

A pesar que la ubicación de los obstáculos puede variar con t, no se puede reflejar esta situación en C de una forma que permita reducir los movimientos de A a los movimientos de un punto entre un número conocido de restricciones. Sin embargo, esta dificultad se puede resolver agregando la dimensión temporal al espacio de configuraciones. Se obtiene de ese modo un nuevo espacio, $CT = C \times [0, +\infty)$, llamado *espacio temporal de configuraciones*, *espacio de configuraciones tiempo-espacio* o *espacio de configuraciones espacio-tiempo*. CT es un espacio continuo de dimensión $m{+}1$, donde m es la dimensión de C. Un camino en un espacio de configuraciones tiempo-espacio -o trayectoria- es una correspondencia continua $\xi : [0, 1] \to CT$, también conocido como *CT-path*.

En función de estas definiciones, es correcto afirmar que un robot A se corresponde en CT según una pareja *configuración-tiempo* (q,t), que significa que la configuración de A en el instante t, es q.

Por otro lado, todo obstáculo O_i se corresponde en C con una región estacionaria CTO_i, llamada *CT-obstacle*, definida según la ecuación B.3:

$$CTO_i = \{(q,t) \mid A(q) \cap O_i(t) \neq 0\} \tag{B.3}$$

El espacio de configuraciones libres en CT se define según la ecuación B.4:

$$CT_{free} = CT \setminus \bigcup_{i=1}^{q} CTO_i \tag{B.4}$$

La clausura de CT_{free}, por consiguiente, es conocida como espacio semi-libre de configuraciones en CT.

Finalmente, un camino libre de colisiones en un espacio de configuraciones tiempo-espacio -o trayectoria libre- (*free CT-path*) es una correspondencia continua $\xi : [0,1] \rightarrow CT_{free}$. Análogamente, un camino semi-libre es una correspondencia continua $\xi : [0,1] \rightarrow cl(CT_{free})$.

En función de estas formalizaciones, el problema de planificación de trayectorias puede redefinirse de la siguiente forma:

Problema dinámico de planificación de movimientos Computar una trayectoria ξ *free CT-path* entre dos pares configuración-tiempo (q_{init}, 0) y (q_{goal}, T).

En relación con el problema básico de planificación existe una diferencia importante. Dado que ningún objeto puede volver en el tiempo, una trayectoria *CT-path* ξ debe ser monótona creciente en el parámetro tiempo, satisfaciendo: $\forall s, s' \in [0,1]$, $con\, \xi(s) = (q,t)\ y\, \xi(s') = (q',t')$; $s < s' \Rightarrow t < t'$. Entonces, ξ determina una trayectoria $q(t) \mid \forall s \in [0,1] : \xi(s) = (q(t), t)$.

En la práctica, esta última condición no es suficiente ya que podrían calcularse velocidades o aceleraciones infinitamente grandes. Consecuentemente, el Problema dinámico de planificación puede hacerse más realista introduciendo restricciones a la velocidad y aceleración del robot. Esto significaría introducir restricciones a la pendiente y curvatura de la función ξ en relación a la dimensión temporal. Sin embargo, estas restricciones aumentan considerablemente la dificultad de resolución del problema, de modo que suelen abordarse, mayormente, sólo dos situaciones: velocidad ilimitada y velocidad limitada. Otra variante al problema dinámico de planificación es liberar el tiempo de llegada al objetivo, implicando que lo importante es poder alcanzar la configuración objetivo en algún instante de tiempo $T \geq 0$.

Múltiples Robots

Una segunda extensión es considerar que en el entorno, como partes de un mismo sistema, conviven múltiples robots. El problema resultante se conoce como *Problema de planificación de caminos multi-robot*. En este caso, múltiples robots se mueven independientemente en el mismo entorno entre obstáculos estacionarios. Dos enfoques utilizados para su resolución son: Planificación centralizada y Planificación desacoplada. Con el primer enfoque se considera a todos los robots como partes de un único robot. Como contrapartida a semejante simplificación, la dimensión del espacio de configuración del robot resultante es más grande con el consiguiente perjuicio en tiempo de cómputo. El segundo enfoque, en tanto, plantea abordar la planificación de caminos en forma independiente para cada robot y luego tratar las potenciales interacciones entre caminos. En este caso, la complejidad del cómputo se mantiene al mismo nivel pero se pierde en completitud.

Robots articulados

Otra extensión posible es considerar que los robots son objetos articulables en lugar de considerarlos como objetos rígidos conformados por una única pieza monolítica. En estos casos los robots pueden estar compuestos por múltiples objetos rígidos unidos entre ellos por articulaciones que restringen sus movimientos relativos. Esta extensión suele estar asociada fuertemente con la planificación de movimientos de manipuladores automáticos o brazos robóticos.

B.3.3.2. Restricciones cinemáticas

Existen dos tipos de restricciones cinemáticas que pueden afectar el movimiento de un robot.
A modo de ejemplo:

- en un robot articulado compuesto por múltiples partes (objetos rígidos) conectadas por intermedio de articulaciones, estas restringen el movimiento relativo entre las partes conectadas imponiendo restricciones cinemáticas holonómicas.
- un robot móvil -birrodado con control diferencial- puede moverse libremente hacia adelante y atrás y realizar giros. Sin embargo, no puede trasladarse de costado y sus giros suelen estar limitados por aspectos mecánicos relacionados con la capacidad de orientar las ruedas.

Restricciones holonómicas

Esencialmente, no afectan el problema básico de planificación de movimientos. Comúnmente, permiten la disminución de la dimensión del espacio de configuraciones.

Asumiendo que en cualquier instante de tiempo t, existe una restricción escalar adicional a la configuración del robot, que se expresa según la ecuación B.5:

$$F(q,t) = F(q_1, \ldots, q_m, t) = 0 \tag{B.5}$$

donde F es una función continua con derivada distinta de cero. Dicha restricción determina un subconjunto de configuraciones de C donde el robot puede estar. Utilizando el *Teorema de la Función Implícita*, se puede expresar una de las coordenadas, q_m por ejemplo, en función de las *m-1* restantes y el tiempo: $g(q_1, ..., q_{m-1}, t)$. La función g es continua, de modo que la ecuación B.5 define un subespacio continuo de C de dimensión *m-1*. De hecho, dicho subespacio es el espacio de configuraciones del robot y las *m-1* parámetros son las coordenadas de las configuraciones en dicho espacio.

Igualdad holonómica Una restricción escalar de la forma *F(q,t)=0*, donde F es una función continua con derivada distinta de cero, es una restricción de igualdad holonómica.

Típicamente, si se considera un brazo robótico articulado cuyos movimientos están restringidos a un plano y que está compuesto por dos objetos móviles (A_1 y A_2) unidos por dos articulaciones rotacionales como se muestra en la figura B.1.

Sin considerar las articulaciones, el espacio compuesto de las configuraciones de ambos objetos móviles tiene dimensión seis (R^2x S^1x R^2x S^1, cada objeto se puede ubicar en un punto del plano con determinada orientación). Sin embargo, cada articulación aplica dos restricciones escalares de la forma *F(q,t)=0*. La primer articulación impone que un punto de A_1 (el centro de rotación) esté fijo respecto a todo el espacio de configuraciones. La segunda articulación impone que un punto de A_2 (el centro de rotación) esté fijo respecto de A_1.

Finalmente, tomando en cuenta estas cuatro restricciones, el espacio de configuraciones del robot tiene dimensión dos (S^1x S^1).

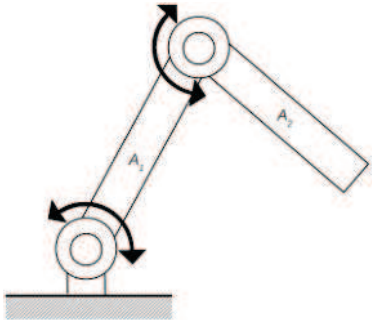


Figura B.1: Restricción holonómica. Brazo robótico articulado.

Desigualdad holonómica Una restricción escalar de la forma $F(q,t)<0$ o $F(q,t)\leq 0$, determina un subconjunto de C que, generalmente, tiene la misma dimensión y es una restricción de desigualdad holonómica.

Típicamente, este tipo de restricciones se corresponden con limitaciones mecánicas o la presencia de un obstáculo. Las mismas determinan un subconjunto de C que posee, generalmente, la misma dimensión que C.

Restricciones no holonómicas

Son restricciones más difíciles de tratar y, por ello, representan una extensión al problema básico de planificación. Resumidamente, plantean tres dificultades principales:

- ¿Cómo estar seguro que una restricción, posiblemente, no-holonómica, realmente lo es?
  Resultados provenientes de la Geometría Diferencial pueden utilizarse a los efectos de caracterizar más precisamente este tipo de restricciones.
- Determinar si una restricción no-holonómica restringe el conjunto de configuraciones alcanzables desde una configuración dada.
  Aplicando herramientas de Teoría de Control se puede establecer condiciones simples para determinar cuáles restricciones no-holonómicas no tienen efecto sobre el rango de configuraciones alcanzables.
- Construir un planificador efectivo, capaz de generar caminos confiables para un robot sujeto a ciertas restricciones de este tipo. La planificación de movimientos con restricciones no-holonómicas se transformó en un área de investigación atractiva luego de las contribuciones realizas por *Laumond* en la década de 1980 (~1986) y su factibilidad de aplicación en robótica móvil.

Si consideramos un robot A mientras se está moviendo. Su configuración q es diferenciable en función del tiempo t. Si, además, se impone que el moviendo de A está sujeto a una restricción escalar que se expresa según la ecuación B.6:

$$G\left(q, \dot{q}, t\right) = G\left(q_1, \ldots, q_m, \dot{q_1}, \ldots, \dot{q_m}, t\right) = 0 \tag{B.6}$$

donde G es una función continua y $\dot{q_i} = \frac{dq_i}{dt}$ para todo i=1, ..., m. El vector velocidad $\dot{q} = (\dot{q_1}, \ldots, \dot{q_m})$ es un vector del espacio tangente a C en q ($T_q(C)$)[6].

[6]Notar que, en ausencia de restricciones de la forma B.6, el espacio tangente es el espacio de velocidades de A.

Una restricción cinemática de la forma B.6 es holonómica si es integrable (todos los parámetros correspondientes a la velocidad $\dot{q}_1, \ldots, \dot{q}_m$ pueden eliminarse y la ecuación B.6 puede reescribirse según la forma que expresa la ecuación B.5). De lo contrario, la restricción es no-holonómica.

Igualdad no-holonómica Una restricción escalar no integrable de la forma $G\left(q_1, \ldots, q_m, \dot{q_1}, \ldots, \dot{q_m}, t\right) = 0$, donde G es una función continua, es una restricción de igualdad no-holonómica. Una restricción de este tipo restringe el espacio de velocidades alcanzable por A en cualquier configuración q a un subespacio vectorial de $T_q(C)$ de m-1 dimensiones.

Desigualdad no-holonómica Una restricción escalar de la forma $G\left(q, \dot{q}, t\right) < 0$ o $G\left(q, \dot{q}, t\right) \leq 0$, generalmente, restringe el conjunto de velocidades alcanzables por A sin cambiar la dimensión del subespacio y es una restricción de desigualdad no-holonómica.

B.3.3.3. Incertidumbre

En el problema básico de planificación de movimientos se asume que el robot es perfectamente capaz de controlar sus movimientos a los efectos de seguir los caminos generados por el planificador. Esta asunción es razonablemente realista en contextos muy particulares donde, por ejemplo, no existen muchos obstáculos ni zonas intrincadas y la configuración objetivo no debe alcanzarse, necesariamente, con precisión. En esos casos la incertidumbre puede ser mitigada haciendo crecer levemente el espacio de configuraciones de los obstáculos *C-obstacles*. Las planificaciones que se realizan con esta estrategia se conocen como *gross motions*.

Sin embargo, esta estrategia no es siempre aplicable. En entornos densamente ocupados por obstáculos, el crecimiento de *C-obstacles* puede reducir tanto al espacio de configuraciones libres C_{free}, que las configuraciones inicial y objetivo podrían dejar de estar conectadas a través de un camino libre. En estos casos es necesario incorporar, total o parcialmente, información sensada a la planificación de movimientos de forma de poder monitorear los efectos de los movimientos y comandar apropiadamente movimientos correctivos.

Una planificación de movimientos en este caso, consiste en una combinación de movimientos (comandos a los motores) y percepciones (lectura de sensores) interactuando para disminuir la incertidumbre y conducir el robot hacia su objetivo. Estos planes son conocidos como *motion strategy* y los movimientos que producen suelen llamarse *fine motions*.

Invariablemente, el sensado no es perfecto y no provee conocimiento perfecto sobre la configuración actual durante la ejecución del plan. De modo que, la incertidumbre no está presente sólo en la planificación, sino que también afecta la ejecución del plan. Por tanto, es necesario que el planificador pueda anticiparse a estas situaciones de modo de reducir la incertidumbre sobre la información más sensible intentando garantizar cierto nivel de precisión que le permita cumplir el cometido de conducir al robot hacia su objetivo.

Otro aspecto que introduce la obtención de información a través de sensado es que el reconocimiento del objetivo no debería ser una cuestión de suerte. Vale decir, el reconocimiento de cierta información del entorno suele requerir la planificación de movimientos que permitan la identificación y localización confiable por parte de los sensores.

B.3.3.4. Objetos manipulables

Hasta aquí, se han presentado variantes del problema básico de planificación que consideran que en el entorno pueden "convivir" múltiples robots y obstáculos. Estos últimos, pueden ser estáticos o estacionarios, pero se asumió que en ningún caso el movimiento de el o los robots interfiere con la configuración de los obstáculos. Precisamente, esta extensión plantea la existencia de objetos manipulables. Es decir, objetos que puedan ser afectados por los movimientos realizados por un robot.

A tales efectos, existen básicamente dos formas de lograr que un robot afecte la configuración de un objeto: empujándolo o agarrándolo. La primera, es más simple pero menos sofisticada, permitiendo manipular un menor número de objetos en un menor número de situaciones; la segunda suele ser más compleja pero brinda mayores posibilidades.

El efecto de introducir este aspecto puede estar relacionado, también, con dos situaciones: el robot puede modificar el espacio de configuraciones de los obstáculos (objetivo intermedio) con miras a encontrar un camino hacia la configuración objetivo (objetivo final) y, por otro lado, lograr cierta disposición para ciertos objetos manipulables del entorno puede ser, propiamente, su objetivo final. Surge entonces una forma más sofisticada de planificación de movimientos conocida como *manipulation planning*.

Debido a la realización de múltiples planes y la selección de objetivos intermedios, la dimensión del espacio de configuraciones para una planificación que incluya la manipulación de objetos suele estar sujeta a explosión combinatoria. En tal sentido, el diseño de planificadores multi-capas o jerárquicos surgió como una solución que intenta abordar las diferentes planificaciones en diferentes niveles.

B.4. Métodos de resolución tradicionales

Existe un gran número de métodos para resolver el problema de planificación de movimientos. Sin embargo, a pesar de sus diferencias, estos métodos se basan en un número reducido de enfoques que pueden clasificarse en: *roadmap*, *cell decomposition* y *potential field*. Básicamente, estos tres enfoques difieren en los grafos de conectividad que construyen y las representaciones empleadas. Hace algo más de una década, cuando las computadoras tenían menos poder de cómputo, estas diferencias podían afectar significativamente el costo computacional de los planificadores, incluso en espacios bidimensionales. Actualmente, con el avance del poder de cómputo de los componentes electrónicos de procesamiento, la importancia de las mismas ha disminuido considerablemente. Lo relevante, en nuestros días, es encontrar enfoques que sean capaces de escalar a problemas de más dimensiones (seis o más). Desafortunadamente, ninguno de ellos en su formato original -diseñado para atacar el problema de planificación en su versión más básica- es capaz de realizarlo. [GbHL05]

B.4.1. *Roadmap*

El uso de *Roadmaps* en Planificación de movimientos comienza a partir del trabajo de *Canny*. Este enfoque consiste en capturar la conectividad del espacio de configuraciones libres y reflejarla en una red de curvas unidimensionales llamada *roadmap*. El problema de planificación de movimientos se reduce pues a conectar la configuración inicial y final a la red y encontrar un camino libre de colisiones que conecte ambos puntos. El camino resultante, si existe, se compone por tres tramos o subcaminos: un tramo que conecta la configuración inicial al *roadmap*, otro contenido en el *roadmap* y por último uno que conecta el *roadmap* con la configuración destino.

Definición Una unión de curvas unidimensionales es un *roadmap* RM si para todo q_{init} y q_{goal} en C_{free} que pueden conectarse por un camino, se cumplen las siguientes condiciones:

1. Accesibilidad: existe un camino desde $q_{init} \epsilon C_{free}$ hacia algún $q'_{init} \epsilon RM$.
2. Salida: existe un camino desde algún $q'_{goal} \epsilon RM$ hacia $q_{goal} \epsilon C_{free}$.
3. Conectividad: existe un camino en RM entre q'_{init} y q'_{goal}.

Basados en diferentes principios se han propuesto varios métodos que se diferencian, básicamente, por el tipo de *roadmap* que construyen. Los más destacados son: Grafos de visibilidad (*Visibility graph*), Retracción, Red de autopistas (*Freeway net*) y Siluetas (*Silhouette*).

B.4.1.1. Grafos de Visibilidad

Es una de las implementaciones más antiguas y ha sido empleada ampliamente en planificadores de movimientos de robots. Se aplica a espacios de configuración de dos dimensiones y, básicamente, el principio por detrás de esta implementación es la construcción de un camino semi-libre de segmentos de recta que conectan las configuraciones inicial q_{init} y final q_{goal} pasando por los vértices del espacio de configuraciones de los obstáculos CO.

Grafo de visibilidad Es un grafo no dirigido G cuyos nodos son q_{init}, q_{goal} y los vértices del espacio de configuraciones de los obstáculos CO; y cuyas aristas son las aristas del espacio de configuraciones de los obstáculos CO y los segmentos de recta que unen dos nodos de G y pertenecen a C_{free}.

Existen algoritmos de construcción y búsqueda que permiten encontrar un camino semi-libre en tiempos $O(n^2)$ para el peor caso, siendo n la cantidad de nodos de G.

En la figura B.2 se puede apreciar un grafo de visibilidad para un espacio de configuraciones simple (donde existen tres obstáculos -polígonos en gris- disconexos que conforman el espacio de configuraciones de obstáculos) y el camino más corto (línea resaltada en negrita) en la clausura del espacio de configuraciones libres C_{free}.

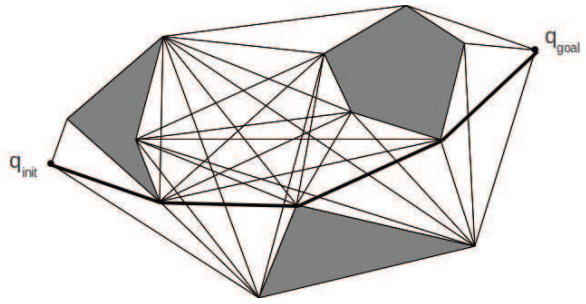


Figura B.2: Grafo de visibilidad.

B.4.1.2. Retracción

Este método consiste en definir una correspondencia continua entre el espacio de configuraciones libres del robot C_{free} y una red unidimensional de curvas R incluida en C_{free}.

Retracción Sea X un espacio topológico e Y un subconjunto de X. Una correspondencia sobreyectiva A de los elementos de X en Y es una retracción de X sobre Y *si y solo si* A es continua y su restricción a Y es la Identidad.

Preservar la conectividad Sea ρ una retracción del espacio topológico X sobre Y. Se dice que ρ preserva la conectividad de X *si y solo si*, para todo $x \epsilon X$, x y $\rho(x)$ pertenecen a la misma componente del camino en X.

Preposición Sea $\rho : C_{free} \rightarrow R$ una retracción que preserva la conectividad, donde $R \subset C_{free}$ es una red unidimensional de curvas. Existe un camino libre entre dos configuraciones q_{init} y q_{goal} *si y solo si* existe un camino en R entre $\rho(q_{init})$ y $\rho(q_{goal})$.

Los métodos de retracción se basan en la proposición anterior y son determinados por la elección de la correspondencia ρ. Cuando $C = R^2$ y C_{free} es el interior de una región poligonal, consisten en retraer C_{free} sobre su diagrama de Voronoi. Dicho de otro modo, en este caso, el diagrama de Voronoi correspondiente a C_{free} es el *roadmap* R utilizado por el método de retracción.

B.4.1.3. Diagramas de Voronoi

Además de ser una de las principales construcciones dentro de la Geometría Computacional, al igual que los grafos de visibilidad y las redes de autopistas, los diagramas de Voronoi son una de las técnicas más usuales para la construcción de *roadmaps*. [Roq96, FGLM01, RD05, WvH07, BG08]

Sea S un conjunto de entidades geométricas en $\mathcal{R}^d$ y una distancia métrica $\|.\|$, el diagrama de Voronoi correspondiente a S es la partición de $\mathcal{R}^d$ en la máxima cantidad de regiones, tales que los puntos pertenecientes a cada región de Voronoi cumplen la condición de estar más cerca de una sola entidad de S que de ninguna otra. [WvH07]

En robótica móvil han sido aplicados considerando, usualmente, el espacio correspondiente a $\mathcal{R}^2$, la distancia euclidiana entre puntos y el principio del vecino más próximo. Así pues, si se considera a $P = \{p_1, p_2, ..., p_n\}$ un conjunto de puntos no alineados en el plano y sea $d(p_i, p_j)$ la distancia euclidiana entre los puntos p_i y p_j, la región de Voronoi $R(p_i)$ generada por el punto p_i queda definida por la ecuación B.7:

$$R(p_i) = \left\{p \in \mathcal{R}^2; d(p, p_i) \leq d(p, p_j), \forall j \neq i\right\} \tag{B.7}$$

En este contexto, a los puntos p_i se les denomina *generadores de Voronoi* y al conjunto de todas las regiones de Voronoi $R(p_1), R(p_2), ..., R(p_n)$, *diagrama de Voronoi* de P. Además, a una frontera común a dos regiones se le denomina *arista de Voronoi* y a un punto de corte entre tres o más aristas, *vértice de Voronoi*.

Los diagramas de Voronoi presentan dos propiedades de interés en el contexto de la planificación de movimientos para robots móviles:

1. Toda arista de Voronoi pertenece a la mediatriz del segmento de recta formado por los dos puntos generadores de las regiones que determinan la arista.

2. Todo vértice de Voronoi está ubicado exactamente en el circuncentro del polígono definido por los puntos generadores de las regiones que determinan el vértice.

Estas propiedades permiten el uso de diagramas de Voronoi para generar *roadmaps* donde se pueden computar caminos de máxima seguridad entre obstáculos. Otra ventaja estriba en la posibilidad de generalizar dichos diagramas, pudiéndose definir generadores (no puntuales) con formas geométricas más representativas de obstáculos reales. Así pues, sería posible representar una pared definiendo un generador con forma de segmento de recta o utilizar un polígono para representar la proyección ortogonal sobre el plano de otro robot, muebles, entre otros tipos de obstáculos que suelen presentarse en entornos reales. La figura B.3 muestra un diagrama de Voronoi generado a partir de tres cuadrados (estos podrían representar tres obstáculos presentes en el entorno).

Región de Voronoi generalizada Una región generalizada de Voronoi R_i está definida por la clausura del conjunto de puntos más cercanos al obstáculo CO_i. La ecuación B.8 formaliza esta noción.

$$R_i \quad = \quad R(CO_i) = \{q \in C_{free} \mid d_i(q) \leq d_h(q), \forall h \neq i\} \tag{B.8}$$

$$d_i(q) = min_{c \in CO_i} d(q, c)$$

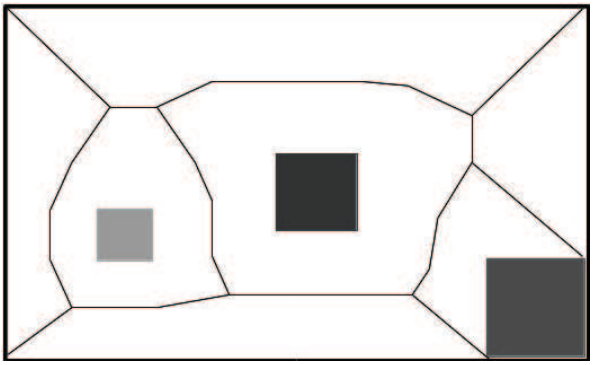


Figura B.3: Diagrama de Voronoi.

Arista de Voronoi generalizada Una arista generalizada de Voronoi F_{ij} es el conjunto de puntos que equidistan a dos obstáculos CO_i y CO_j y tiene como obstáculos más cercanos a CO_i y CO_j. La ecuación B.9 formaliza esta noción.

$$F_{ij} = \{q \in S_{ij} \mid d_i(q) \leq d_h(q), \forall h\} \tag{B.9}$$

$$S_{ij} = \{x \in Q \mid d_i(q) - d_j(q) = 0\}$$

Diagrama de Voronoi generalizado Un diagrama de Voronoi generalizado (*GVD - del inglés Generalized Voronoi Diagram*) puede definirse como la unión de todas las aristas de Voronoi generalizadas. La ecuación B.10 formaliza esta noción.

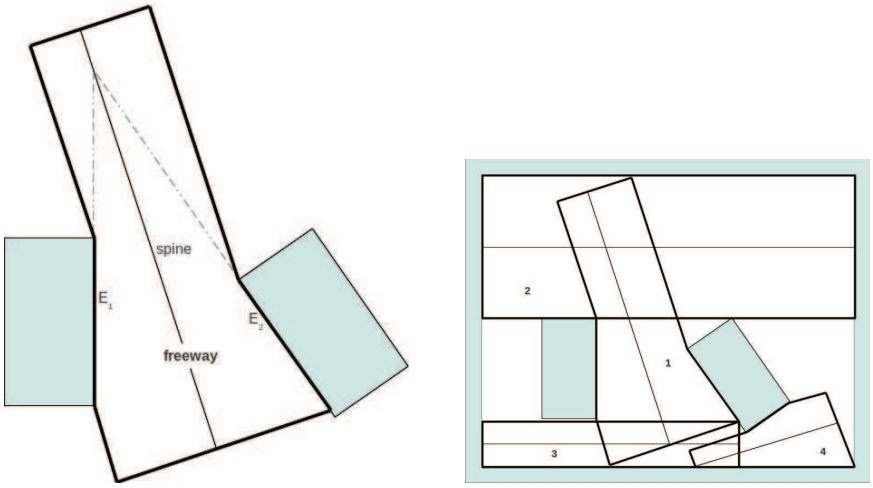
$$GVD = \bigcup_i \bigcup_j F_{ij} \tag{B.10}$$

Existen algoritmos que permiten computar un diagrama de Voronoi en tiempo $O(n \lg n)$, siendo n el número de puntos generadores (u óbstáculos en la noción generalizada). [GbHL05]

B.4.1.4. Red de autopistas

Al igual que los métodos basados en retracciones, este método intenta mantener al robot lo más lejos posible de los obstáculos. Fue propuesto por *Brooks* en 1983 con la intención de capturar los movimientos de traslación y rotación de un objeto en una representación más intuitiva que los espacios de configuración. Consiste, básicamente, en extraer determinadas figuras geométricas -llamadas autopistas o *freeways*- del espacio de configuración, conectarlas en un grafo llamado *freeway net* y realizar una búsqueda en dicho grafo.

Una *freeway* puede verse, genéricamente, como un cilindro a través de cuyo eje central se desplaza el robot. La figura B.4a presenta un caso general donde se puede observar que el eje o *spine* es en efecto, la bisectríz del ángulo formado por los lados de los obstáculos que delimitan la autopista. Un robot sólo puede navegar a través de una autopista si existe en cada punto del eje al menos una orientación en la que sea factible su posicionamiento. Del mismo modo, solo puede cambiar de autopista si la intersección del conjunto de orientaciones factibles de cada eje en el punto de intersección no es vacía. En la figura B.4b se presenta un diagrama incompleto con múltiples autopistas. En resumen, la red de autopistas es una representación de las opciones de movimiento del robot sobre cada autopista y entre autopistas.



(a) Freeway. (b) Múltiples Freeway.

Figura B.4: *Freeways*.

Extracción de autopistas

Las autopistas son extraídas del conjunto $\varepsilon = W \setminus O$, donde W representa el entorno de trabajo y O el conjunto de obstáculos, considerando todos los pares de aristas de O. Un par de aristas (E_1 y E_2) producen una autopista *si y solo si* satisfacen las siguientes condiciones:

1. Sean $i, j \epsilon \{1, 2\}$, $i \neq j$, ambos extremos de E_i pertenecen al semiplano opuesto al de E_j.
2. El producto interno de los vectores normales a E_1 y E_2 es negativo.

La satisfacción de ambas condiciones implica que los lados en cuestión se encuentran "enfrentados" (ver figura B.4a). De este modo se delimitan las autopistas por los lados. Los extremos son delimitados intersectando cada autopista con el espacio de obstáculos O y proyectando los puntos de intersección sobre el eje. El tiempo total de este proceso es $O\left(n_O^3\right)$, donde n_O es el número total de aristas de O.

Armado del grafo

Una vez que se cuenta con un conjunto de autopistas, el método consiste en crear un grafo no dirigido que determine todas los posibles movimientos del robot sobre las autopistas. A tales efectos se determinan todos los puntos de intersección de ejes. Luego se crean los nodos del grafo considerando, para cada autopista, todos los puntos intersección y las orientaciones factibles en dichos puntos. Finalmente, las aristas se crean conectando nodos que se corresponden con puntos pertenecientes a una misma autopista o se corresponden con un punto perteneciente a dos autopistas.

Las experiencias realizadas con este método muestran que funciona bien y rápido en entornos relativamente poco simples sin muchos obstáculos. Sin embargo, el método no es completo y por tanto, aunque existiera, puede no encontrar un camino libre de colisiones.

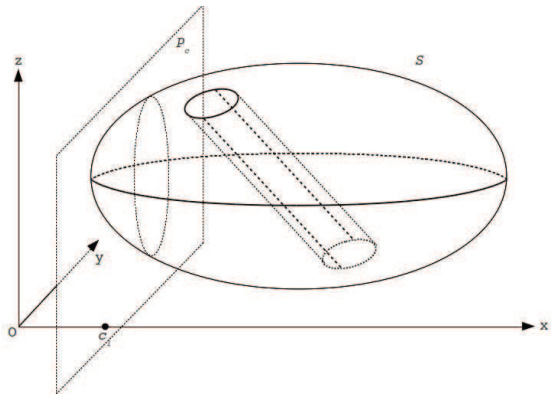
B.4.1.5. Siluetas

Propuesto por *Canny* en 1998, a diferencia de los métodos anteriores, es un método general basado en *roadmaps*. Este método genera caminos semi-libres, es completo y, por tanto, garantiza el hallazgo de un camino si existe. *Canny* también demostró que en un entorno poblado con obstáculos cuyas fronteras puedan ser representadas por p polinomios de grado w (a lo sumo), cualquier problema de planificación de movimientos puede ser resuelto utilizando este método en $O\left(p^n \left(\log p\right) w^{\left(O\left(n^4\right)\right)}\right)$, donde n son los grados de libertad del robot (la dimensión del espacio de configuración).

El método consiste en la construcción de un *roadmap* mediante un algoritmo que reduce recursivamente el problema en subproblemas para espacios de dimensión más chica, y luego realizando una búsqueda sobre un grafo que represente dicho *roadmap*.

Construcción del *roadmap*

En el nivel cero de la recursión, se desplaza un hiperplano de dimensión *n-1* a través de S -la clausura del espacio de configuración de dimensión *n*-, en dirección perpendicular a alguno de los ejes del sistema de coordenadas en el cual S es representado. Este proceso de "barrido" comienza en cierta abscisa $c{=}c_0$. El plano P_c, se desplaza aumentando continuamente el valor de c. De este modo, el *roadmap* se construye incorporando todos los puntos extremos (mínimos y máximos) de la intersección $S \cap P_c$. La sucesión de puntos extremos va conformando tramos de curvas algebraicas denominadas *curvas de silueta* que finalmente conformarán el *roadmap*. Para la mayoría de los valores de c, las curvas que se van generando están conectadas entre sí, sin embargo, para un conjunto finito de valores de c, llamados *valores críticos*, la conectividad del *roadmap* se ve alterada. Estos nuevos puntos que aparecen se llaman *puntos críticos*. En la figura B.5 se puede observar el sistema de coordenadas de referencia, el hiperplano P_c y el conjunto S.



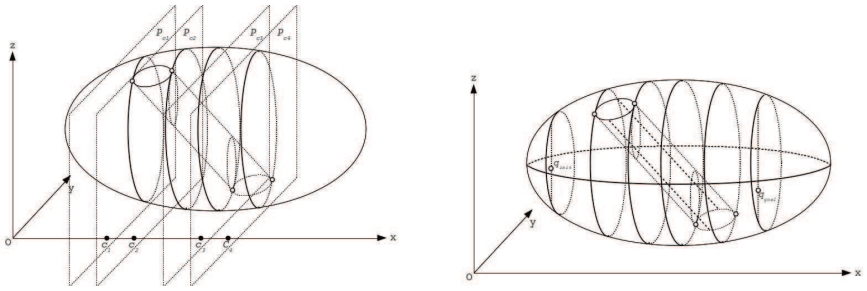
(a) Conjunto tridimensional S. Plano Pc en $x{=}c$. Curvas de silueta en negrita, extraídas a partir de puntos extremos de $S \cap P_c$.

Figura B.5: Siluetas. Elementos principales.

Durante el proceso de barrido, cuando el hiperplano alcanza un punto crítico se realiza una llamada recursiva para conectar dicho punto con las curvas de silueta previamente generadas. Esta llamada genera el lanzamiento de un nuevo proceso de barrido con un nuevo hiperplano de dimensión n-2. Las

llamadas recursivas terminan cuando, durante el barrido, no se encuentran puntos críticos, o cuando la dimensión del conjunto que se está procesando es igual a dos. En la figura B.6a se pueden observar los diferentes valores y puntos críticos así como las curvas de silueta que se generan a partir del proceso recursivo.

La figura B.6b muestra el roadmap resultante del proceso completo, incluyendo los puntos inicial y final de la trayectoria buscada.



(a) Valores y puntos críticos que aparecen durante el proceso de barrido. Curvas de silueta en negrita, generadas durante el proceso recursivo.

(b) Roadmap completo.

Figura B.6: Siluetas. Proceso de construcción del mapa.

B.4.2. *Cell Decomposition*

Quizás el más vastamente estudiado, este enfoque consiste en descomponer el espacio de configuraciones libres en regiones que no se solapan, llamadas celdas, de forma tal que un camino entre dos configuraciones que pertenezcan a la misma región se pueda computar fácilmente. Luego, se construye un grafo no dirigido donde estén representadas las relaciones de adyacencia entre celdas. Finalmente, se realiza un búsqueda sobre el grafo que da como resultado una secuencia de celdas o canal -que conecta las celdas que contienen las configuraciones inicial y final-, a partir del cual se puede computar un camino libre de colisiones.

A su vez, los métodos que realizan descomposición del espacio de configuración libre en regiones pueden clasificarse en exactos o aproximados. Los métodos exactos de descomposición tienen la propiedad de ser completos, esto es, encuentran una solución siempre que exista. Por el contrario, los métodos aproximados no son completos aunque, si la precisión en la descomposición puede ajustarse tanto como sea necesario (celdas arbitrariamente chicas), puede lograrse la completitud. Suelen denominarse *resolution-complete methods*. En contrapartida a la ventaja de ser completos, los métodos exactos son, excepto para problemas muy simples, más intensivos en cálculos matemáticos y por tanto más complejos y menos eficientes que los aproximados.

B.4.2.1. Métodos exactos

Estos métodos descomponen el espacio de configuraciones libres en celdas de manera tal que la unión de todas las celdas coincide exactamente con el espacio descompuesto. Sin embargo, no todas

las descomposiciones exactas son apropiadas. En general, se espera que la descomposición cumpla con las siguientes condiciones:

1. La geometría de las celdas debe ser suficientemente simple como para facilitar el cálculo de caminos entre configuraciones pertenecientes a una misma celda.

2. Determinar la adyacencia de dos celdas cualesquiera no debe representar un problema de gran dificultad ni calcular un camino que permita navegar entre dos celdas adyacentes.

A partir de estas condiciones debe interpretarse que las celdas son regiones donde todos los caminos entre distintas configuraciones que pertenecen a una misma celda son cualitativamente equivalentes y, contrariamente, las fronteras entre regiones adyacentes son regiones "críticas" donde se producen cambios significativos.

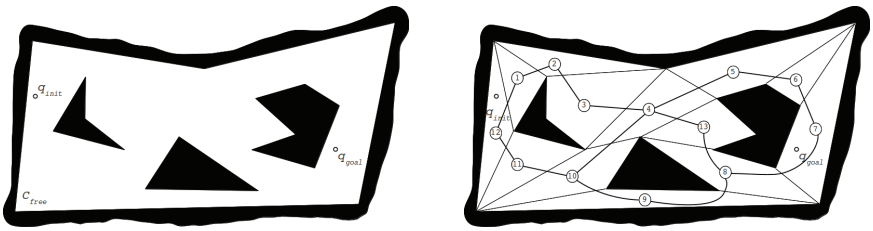
Espacio de configuración poligonal

Descomposición poligonal convexa Una descomposición poligonal convexa K de C_{free}, es una colección finita de polígonos convexos, llamados celdas, tal que la intersección de dos celdas cualesquiera es vacía o coincide con la frontera y la unión de todas las celdas es igual a la clausura de C_{free}. Dos celdas k y k' en K son adyacentes *si y sólo si* $k \cap k'$ es un segmento de línea de largo > 0.

Grafo de conectividad Un grafo de conectividad G asociado con una descomposición poligonal convexa K de C_{free} es un grafo no dirigido especificado de la siguiente forma:

- Los nodos de G son celdas de K.
- Dos nodos en G son conectados *si y sólo si* las correspondientes celdas son adyacentes en K.

La figura B.7a presenta un espacio de configuración bidimensional, mientras la figura B.7b ilustra el resultado obtenido luego de realizar la descomposición poligonal convexa y el grafo de conectividad asociado.



(a) Espacio de configuración bidimensional.

(b) Descomposición poligonal convexa. Grafo de conectividad.

Figura B.7: Descomposición exacta de celdas. Descomposición poligonal convexa.

El algoritmo de planificación de un camino libre de colisiones entre dos configuraciones es el siguiente:

1. Generar una descomposición poligonal convexa K de C_{free}.

2. Construir un grafo de conectividad G asociado con la descomposición K.

3. Buscar en G un canal entre q_{init} y q_{goal}.

4. Si la búsqueda es exitosa retorna el camino, error en otro caso.

La salida del algoritmo es una secuencia k_1,...,k_p de celdas tal que $q_{init} \epsilon k_1$, $q_{goal} \epsilon k_p$ y $\forall j \epsilon [1, p-1]$, k_j y k_{j+1} son adyacentes. El interior de ese canal, $int\left(\cup_{j=1}^{p} k_j\right)$, está incluido totalmente en C_{free}. Sea $\beta_j = \delta k_j \cap \delta k_{j+1}$, donde δk_j denota la frontera de la celda k_j, un mecanismo simple para generar un camino libre de colisiones (contenido en el interior del canal) es considerar los puntos medios Q_j de todo segmento β_j, conectando q_{init} y q_{goal} a través de segmentos de línea cuyos sucesivos vértices son Q_1, ..., $Q_{p\text{-}1}$.

La figura B.8 muestra un camino libre de colisiones calculado a partir de la construcción de un canal y utilizando los puntos medios de las fronteras.

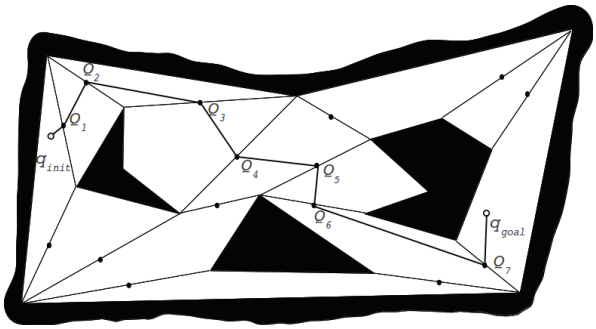


Figura B.8: Descomposición poligonal convexa. Camino libre de colisiones.

Un espacio de configuración poligonal puede descomponerse en forma óptima en tiempo polinomial en relación al número de vértices. Sin embargo, la presencia de zonas cóncavas (hoyos, *p. ej.*) transforma la descomposición en un problema *NP*-duro.

Alternativamente, se han desarrollado métodos no óptimos de descomposición poligonal convexa que se basan en un proceso de barrido con una línea recta vertical. Este proceso se detiene en cada vértice de un obstáculo, generando dos segmentos de recta vertical que unen el vértice con los obstáculos que estén por encima y debajo del mismo. De este modo, las fronteras del espacio de configuración de los obstáculos y los segmentos de recta generados durante el barrido, determinan una descomposición trapezoidal del espacio libre de configuración (también llamada *descomposición vertical*). Dicha descomposición genera celdas de forma triangular o trapezoidal. En la figura B.9 puede observarse una descomposición vertical realizada con una línea paralela al eje de ordenadas que barre el espacio de configuración en el sentido creciente de las abscisas.

Con esta alternativa es posible resolver concurrentemente la descomposición vertical, la generación del grafo de conectividad e identificar las celdas que contienen las configuraciones inicial y final. Todo el proceso toma un tiempo $O\left(n \log n\right)$, donde n es el número total de vértices en el espacio de configuración de obstáculos.

La figura B.10 muestra las diferentes etapas de un planificador basado en este enfoque.

Este método puede generalizarse a $C{=}R^3$, con obstáculos poliédricos. En ese caso, el barrido se realiza con un plano, descomponiendo el espacio de configuración en paralelepípedos.

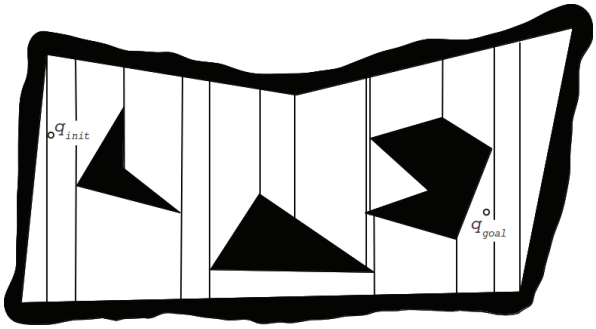
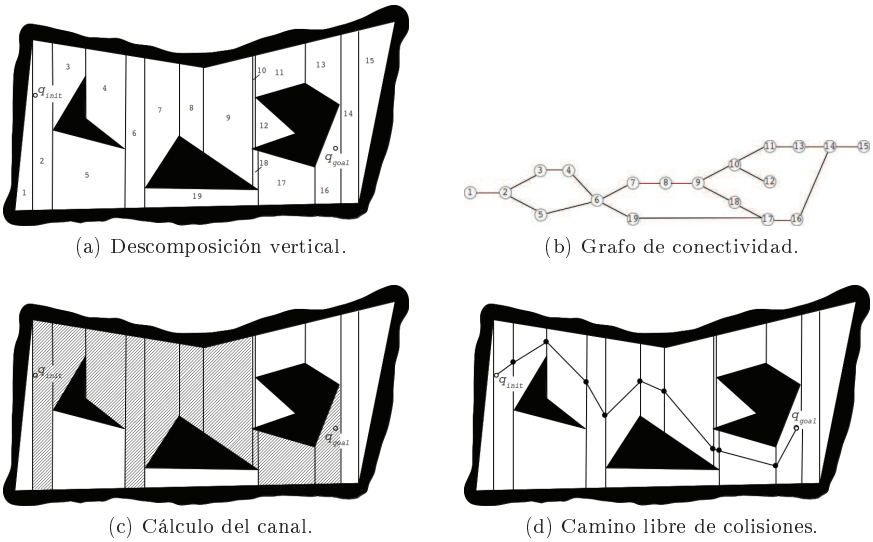


Figura B.9: Descomposición exacta en celdas. Descomposición vertical.



(a) Descomposición vertical.

(b) Grafo de conectividad.

(c) Cálculo del canal.

(d) Camino libre de colisiones.

Figura B.10: Descomposición vertical. Proceso de planificación.

B.4.2.2. Métodos aproximados

Al igual que los métodos exactos, los métodos aproximados descomponen el espacio de configuraciones libres en celdas, pero a diferencia de los primeros, las celdas deben tener una forma simple predefinida y la unión de todas las celdas no coincide exactamente con el espacio descompuesto sino que está incluida estrictamente en él. Luego, como en los métodos exactos, se construye un grafo de conectividad que represente las relaciones de adyacencia entre celdas y se realiza una búsqueda de caminos sobre el mismo.

Las razones para estandarizar la forma de las celdas son:

1. realizar la descomposición de forma iterativa aplicando reiteradas veces la misma fórmula
2. tener un algoritmo relativamente insensible a errores numéricos

Generalmente, estos aspectos redundan en planificadores más simples de implementar que los basados

en descomposición exacta. Además, con los métodos aproximados es posible mantener controlado el tamaño del espacio de configuración libre en torno a los caminos generados. Por el contrario, los métodos aproximados son incompletos y pueden fallar en la búsqueda de un camino, aunque este exista.

La mayoría de los métodos aproximados manejan el tamaño de las celdas de forma jerárquica permitiendo adaptar el valor a la geometría de la región, evitando predefinir un tamaño que genere dificultades: de ser demasiado grande podría impedir el hallazgo de caminos libres de colisiones y de ser demasiado pequeño implicaría un incremento del esfuerzo computacional.

En un sentido práctico y debido a que la complejidad en espacio de memoria y tiempo de ejecución de los métodos que implementan este enfoque crece rápidamente con la dimensión del espacio, los métodos aproximados son aplicables solamente cuando la dimensión del espacio de configuración es menor o igual a cuatro.

Rectanguloide Designa una región con las siguientes características en el espacio Cartesiano R^n:

$$\left\{(x_{1,\cdots},x_n) \mid x_1 \epsilon \left[x_1^{'}, x_1^{''}\right], \cdots, x_n \epsilon \left[x_n^{'}, x_n^{''}\right]\right\}$$

donde las diferencias $x_i^{'} - x_i^{"}$, $i = 1, \cdots, n$ son las dimensiones del rectanguloide.

Este enfoque asume que el conjunto de posiciones posibles para un robot A, están contenidas en un rectanguloide $D \subset R^n$. El espacio de configuraciones libres C_{free} se representa como: $C_{free} = R \setminus O$, donde O es el espacio de configuraciones de los obstáculos y $R = int\,(D)$. En este caso, $\Omega = cl\,(R)$ es un rectanguloide en R^n.

Descomposición rectangular Una descomposición rectangular P de una región rectanguloide Ω es una colección de rectanguloides $\{k_i\}_{i=1,\cdots,r}$ tal que $\Omega = \cup_{i=1}^{r} k_i$ y $\forall i_1, i_2 \epsilon [1, r], i_1 \neq i_2 : int\,(k_{i_1}) \cap int\,(k_{i_2}) = \oslash$

Cada rectanguloide k_i es una celda de la descomposición P de Ω. Además, toda celda k_i puede ser clasificada en:

- Vacía: *si y solo si* el interior no intersecta el espacio de configuración de obstáculos. $int\,(k_i) \cap O = \oslash$
- Llena: *si y solo si* k_i está totalmente contenida en el espacio de configuración de obstáculos. $k_i \subseteq O$
- Parcial: en otro caso.

Grafo de conectividad El grafo de conectividad asociado a una descomposición rectangular P de Ω es un grafo no-dirigido G definido de la siguiente forma:

- Los nodos de G son celdas *Vacías* y *Parciales* de P.
- Dos nodos de G están conectados *si y solo si* las celdas asociadas son adyacentes.

Canal Dada una descomposición rectangular P de Ω, un canal es definido como una secuencia $(k_{\alpha j})_{j=1,\cdots,p}$ de celdas *Vacías* o *Parciales* tales que cualquier pareja de celdas consecutivas $k_{\alpha j}$ y $k_{\alpha j+1}$, $j \epsilon [1, p-1]$, son adyacentes.

En un canal en el que todas sus celdas son *Vacías*, cualquier camino que conecte cualquier configuración en $k_{\alpha 1}$ con otra cualquiera en $k_{\alpha p}$ y que pertenezca al interior del canal $(int\left(\cup_{j=1}^{p} k_{\alpha j}\right))$, es un camino libre de colisiones.

En un canal en el que existe al menos una celda de tipo Parcial, puede existir un camino libre de colisiones pero no se puede garantizar su existencia.

En la figura 4.1a se muestran, una descomposición rectangular, un canal y las configuraciones inicial y final del camino buscado, mientras la figura B.12 muestra un camino libre de colisiones extraído del canal computado en la figura 4.1a.

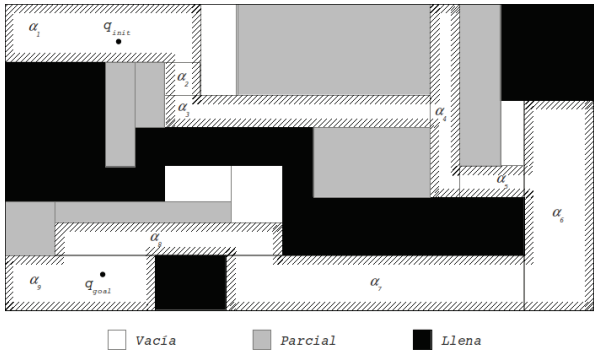


Figura B.11: Descomposición aproximada.

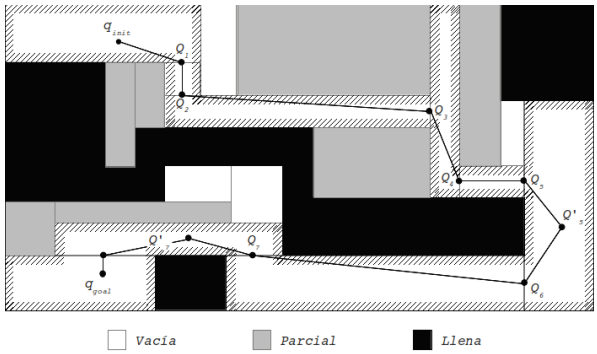


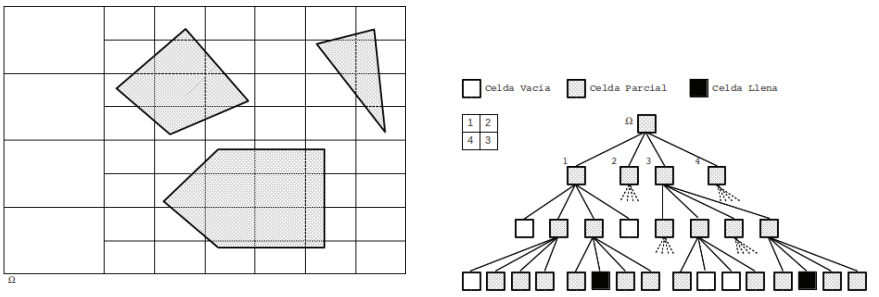
Figura B.12: Descomposición aproximada. Camino libre de colisiones.

Descomposición 2^m-*Tree*

Existen variadas formas de descomponer un espacio rectanguloide. Una de las técnicas más ampliamente utilizadas se llama 2^m-*Tree*[7]. Una descomposición 2^m-*Tree* de Ω es un árbol de grado 2^m. Cada nodo del árbol es un rectanguloide (celda) etiquetada como *Vacía*, *Parcial* o *Llena*. La raíz del árbol es Ω. Los nodos que representan celdas *Vacía* o *Llena* son hojas y los restantes nodos (*Parcial*) tienen 2^m hijos con exactamente las mismas dimensiones. Típicamente, si m=2 al árbol se le llama *quadTree* y si m=3 se le llama *octTree*.

Las figuras B.13 y B.14 muestran un ejemplo de descomposiciones *quadTree* y *octTree*.

[7] m es la dimensión del espacio de configuración.



(a) *quadTree* de profundidad 3. (b) Vista parcial del árbol correspondiente.

Figura B.13: Descomposición *quadTree*.

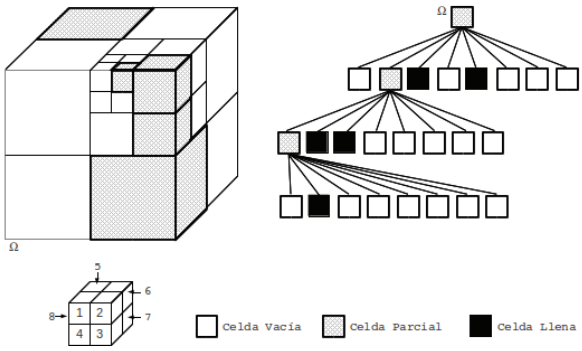


Figura B.14: Descomposición octTree.

B.4.3. *Campos artificiales de potencial*

Una aproximación más directa al problema de planificación de movimientos, a diferencia de los métodos descriptos anteriormente que se basan en capturar la conectividad global del espacio de configuraciones libres en un grafo que luego se utiliza para realizar la búsqueda de un camino, es discretizar todo el espacio de configuraciones en una grilla regular de configuraciones donde realizar la búsqueda del camino. En este caso, debido al tamaño de la grilla, se requieren heurísticas de búsqueda suficientemente potentes. Una de las más exitosas se basa en funciones que son interpretadas como campos de potencial.

La metáfora que sugiere esta terminología es considerar al robot como una partícula que se mueve en el espacio de configuraciones bajo la influencia de campos artificiales de potencial producidos por la configuración objetivo y los obstáculos. Típicamente, aunque no necesariamente, la función de potencial es definida sobre el espacio de configuraciones libres como la suma de un potencial atractor, que empuja al robot hacia la configuración objetivo, y un potencial repulsivo que lo aleja de los obstáculos.

En este enfoque, la planificación de movimientos es realizada de forma iterativa. En cada iteración la fuerza artificial $\overrightarrow{F}(q) = -\overrightarrow{\nabla} U(q)$, inducida por la función de potencial, es considerada como la dirección más prometedora para realizar el siguiente movimiento y la generación del camino se realiza a lo largo de dicha dirección de a ciertos incrementos.

Originalmente, los campos artificiales de potencial fueron concebidos como una forma de realizar evasión de obstáculos en tiempo real, aplicables en contextos donde no se dispone a priori de un modelo

global del entorno y, por el contrario, se reconoce el entorno mediante sensado durante la ejecución de los movimientos.

El énfasis, claramente, está puesto en la eficiencia para un entorno real desconocido y no tanto en el logro del objetivo. Vale decir, la función de potencial puede tener mínimos locales que generen un detenimiento del robot en otra configuración diferente a la configuración objetivo buscada. Este es el mayor inconveniente a mitigar al pensar en utilizar una planificación basada en este enfoque. Sin embargo, su eficiencia en una amplia gama de situaciones y entornos, simplicidad y robustez frente a errores en el control o sensado, en contrapartida a ser incompletos, explica el por qué de su creciente popularidad entre los planificadores desarrollados en la práctica. [GbHL05]

Estructura general

El campo de fuerzas artificiales $\overrightarrow{F}(q)$ en C es producido por una función de potencial $U : C_{free} \rightarrow R$, con: $\overrightarrow{F}(q) = -\overrightarrow{\nabla} U(q)$. En C=R^n (n=2 o n=3), q puede ser *(x, y)* o *(x, y, z)*, con lo cual:

$$\overrightarrow{\nabla} U = \begin{pmatrix} \delta U/\delta x \\ \delta U/\delta y \end{pmatrix} \text{ o } \overrightarrow{\nabla} U = \begin{pmatrix} \delta U/\delta x \\ \delta U/\delta y \\ \delta U/\delta z \end{pmatrix}.$$

En orden de lograr que el robot sea atraído hacia la configuración destino mientras es repulsado por los obstáculos, U se define como la suma de dos funciones de potencial más elementales como se indica en la ecuación B.11

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{B.11}$$

donde U_{att} es el potencial atractor asociado a la configuración objetivo q_{goal} y U_{rep} es el potencial repulsivo asociado con las configuraciones de los obstáculos O. U_{att} es independiente de las configuraciones de los obstáculos O y U_{rep} es independiente de la configuración objetivo q_{goal}. De este modo, $\overrightarrow{F}$ se define como la suma de los dos vectores de la ecuación B.12:

$$\overrightarrow{F}_{att} = -\overrightarrow{\nabla} U_{att} \;\; y \;\; \overrightarrow{F}_{rep} = -\overrightarrow{\nabla} U_{rep} \tag{B.12}$$

Potencial atractor

El potencial atractor puede definirse como una parábola, en los términos de la ecuación B.13:

$$U_{att}(q) = \frac{1}{2}\xi \rho^2_{goal}(q) \tag{B.13}$$

donde ξ es un factor de escalado y $\rho_{goal}(q)$ denota la distancia Euclidiana $\|q - q_{goal}\|$. Esta función es positiva o nula y tiene un mínimo en q_{goal}, donde $U_{att}(q_{goal})$=0. Por otro lado, la función ρ_{goal} es diferenciable en todo C. De modo que, en cualquier configuración q, la fuerza atractora artificial $\overrightarrow{F}_{att}$ derivada del potencial atractor U_{att} está determinada por la ecuación B.14:

$$\overrightarrow{F}_{att}(q) = -\overrightarrow{\nabla} U_{att}(q) = -\xi \rho_{goal}(q) \overrightarrow{\nabla} \rho_{goal}(q) = -\xi (q - q_{goal}) \tag{B.14}$$

Potencial repulsivo

La idea principal que subyace en la definición de un potencial repulsivo es crear una barrera alrededor del espacio de configuración de obstáculos. Por el contrario, es deseable que el potencial repulsor no afecte los movimientos del robot cuando esté suficientemente alejado de los obstáculos. Una forma de lograrlo es definiendo la función de potencial repulsivo como lo indica la ecuación B.15:

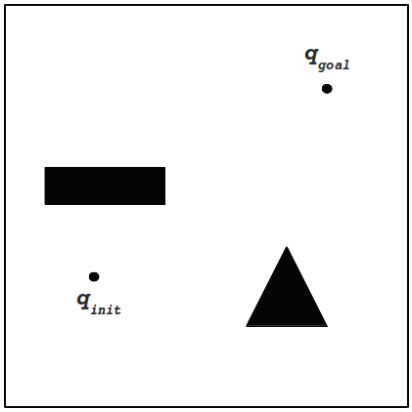
$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & si\,\rho(q) \leq \rho_0 \\ 0 & si\,\rho(q) > \rho_0 \end{cases} \tag{B.15}$$

donde η es un factor de escalado, $\rho(q)$ denota la distancia desde q hasta el espacio de configuraciones de los obstáculos $\rho(q) = \text{mín}_{q'\epsilon O}\,\|q - q'\|$ y ρ_0 es una constante positiva que representa la distancia de influencia de los obstáculos. La función U_{rep} es positiva o nula, tiende a infinito a medida que el robot se acerca a la región donde están los obstáculos y se anula cuando la distancia entre la configuración del robot y la de los obstáculos es mayor que ρ_0.

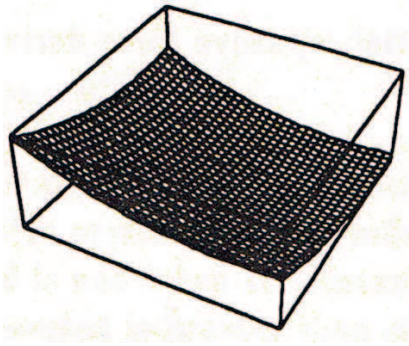
Si, además, se cumple que el espacio de configuración de los obstáculos determina una región convexa con una frontera diferenciable a trozos, ρ es diferenciable en todo C_{free}. La fuerza repulsiva derivada del potencial repulsivo U_{rep} está determinada por la ecuación B.16:

$$\overrightarrow{F}_{rep}(q) = -\overrightarrow{\nabla} U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\overrightarrow{\nabla}\rho(q) & si\,\rho(q) \leq \rho_0 \\ 0 & si\,\rho(q) > \rho_0 \end{cases} \tag{B.16}$$

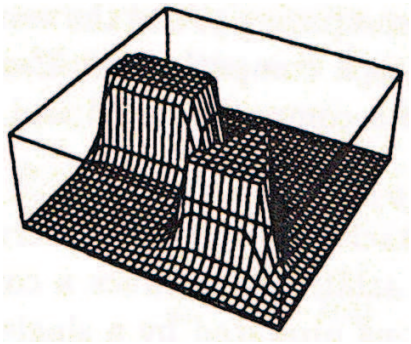
La figura B.15 muestra un espacio de configuraciones de dos dimensiones conteniendo dos obstáculos (figura B.15a). El potencial atractor tiene su mínimo en la configuración objetivo q_{goal} (figura B.15b), mientras que el potencial repulsivo se aprecia en la figura B.15c. La suma de ambos potenciales se muestra en la figura B.15d. Finalmente, en las figuras B.15e y B.15f pueden apreciarse un diagrama de curvas de nivel junto al camino encontrado y una matriz conteniendo una discretización del vector gradiente negado correspondiente al potencial total sobre el espacio de configuraciones libres, respectivamente.



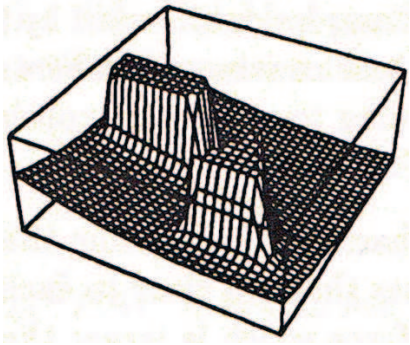
(a) Espacio de configuración bidimensional.



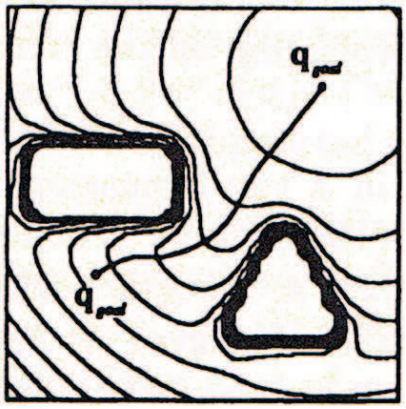
(b) Potencial atractor.



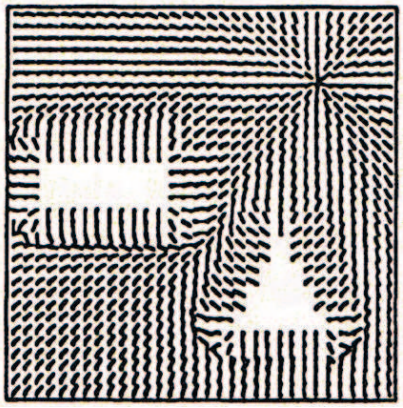
(c) Potencial repulsivo.



(d) Potencial total.



(e) Curvas de nivel y camino encontrado.



(f) Gradiente negado discretizado.

Figura B.15: Campos artificiales de potencial.

B.5. Métodos de resolución probabilísticos[8]

Sobre mediados de la década de 1990 comenzaron a proponerse métodos probabilísticos basados en *roadmaps* (*Probabilistic Roadmap Methods - PRM*). En contraposición a los métodos tradicionales basados en *roadmaps*, en este caso no se intenta representar fielmente el espacio de configuraciones libres, sino, por el contrario, construir una representación simplificada del mismo, independiente de su geometría.

Considerando que la posición y orientación de un robot móvil que se mueve en el plano puede representarse por la tupla de parámetros (x, y, z) y que a menudo los robots móviles están sometidos a restricciones dinámicas que para representarlas es necesario considerar, adicionalmente, las velocidades ($\dot{x}$, $\dot{y}$, $\dot{z}$), resultando en un espacio de configuraciones de seis dimensiones. Además, si hubieran múltiples robots cooperando en el mismo entorno la dimensión del espacio sería aún mayor.

En este contexto, y atendiendo que estos escenarios comenzaron a ser cada vez más frecuentes en los últimos años, surgieron algoritmos simples basados en muestreo o *sampling* de configuraciones, como una herramienta poderosa para realizar planificación de movimientos en espacios de gran dimensión.

Actualmente, los planificadores basados en *sampling* son capaces de resolver, en tiempos razonables, el problema de planificación de movimientos de robots con varios grados de libertad (más de una decena, p.ej.) tomando en cuenta restricciones cinemáticas y dinámicas, de estabilidad, de visibilidad, de contacto, de energía y robots reconfigurables, entre otras.

Existen dos clases principales de algoritmos. Una computa previamente un *roadmap* permitiendo que, sobre un entorno estático, múltiples planificaciones puedan consultarse y realizarse rápidamente. La otra clase no realiza ningún cálculo previo y construye un pequeño *roadmap* en simultáneo mientras procesa una sola planificación tan rápido como sea posible. Esta última es utilizada, generalmente, cuando el entorno cambia con frecuencia y, por tanto, realizar cálculos previamente no es factible.

La primera clase es conocida como *multi-query planning*, mientras que la segunda se conoce como *single-query planning*.

B.5.1. *Multi-query planning*

Estos algoritmos operan en dos etapas. La primera, o de aprendizaje, donde se computa un *roadmap* G que captura la conectividad del espacio de configuraciones libres C_{free} de la forma más precisa posible invirtiendo para ello un tiempo razonable. Para realizar ese cálculo se procede a muestrear aleatoriamente el espacio de configuraciones C y se retienen las configuraciones que pertenecen al espacio de configuraciones libres. Estas configuraciones *libres* se denominan *milestones* y son consideradas como nodos del grafo G. Las aristas del grafo serán todos los segmentos de recta libres de colisión entre dos *milestones*.

La segunda etapa, o de consulta, es donde se busca conocer un camino libre de colisiones entre dos configuraciones q_{init} y q_{goal}.

La figura B.16 muestra un grafo probabilístico -resultado de la primera etapa de un algoritmo de *multi-query planning* en un entorno bidimensional.

La clave de la construcción de mapas probabilísticos es la distribución de probabilidad utilizada para realizar el muestreo de los potenciales nodos del grafo. En tal sentido, comprendiendo que la estrategia de muestreo juega un rol significativo en el desempeño del algoritmo, se han propuesto

[8] Para la elaboración de la presente sección [KSLO96, Ove02, GbHL05, CLH+05, Car06, Sah06] se utilizaron como referencias bibliográficas principales.

varias estrategias que intentan, por distintos medios, incrementar la densidad de configuraciones en las zonas más críticas del entorno.

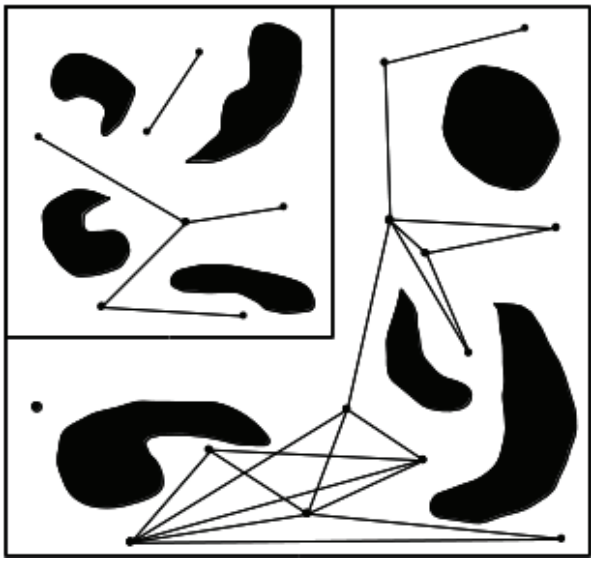


Figura B.16: *Random sampling. Multi-query planning.*

B.5.1.1. *Random sampling*

Una de las primeras estrategias de muestreo es, además, la más simple y sencilla de implementar. En entornos estratégicos presenta la gran ventaja de ser, al menos teóricamente, impredecible. Sin embargo, existen escenarios donde esta estrategia presenta desempeños muy pobres. Tal es el caso de entornos con pasajes estrechos.

B.5.1.2. *Sampling Near the Obstacles*

Las estrategias de muestreo basadas en obstáculos intentan propiciar un muestreo cercano a la frontera del espacio de configuraciones de obstáculos. Su principal objetivo es asegurar un nivel de muestreo en pasajes angostos. La motivación surge en visualizar que estos pasajes también pueden entenderse como corredores en el espacio de configuraciones libres rodeados de obstáculos.

OBPRM (*Obstacle-based PRM*) es uno de los primeros y exitosos algoritmos representativo de las estrategias de muestreo basadas en obstáculos. Otros utilizan distribuciones de probabilidad no uniformes (*Gaussian sampler*) o incluso modifican levemente el modelo del entorno dilatando el espacio de configuraciones libres para ensanchar los pasajes entre obstáculos aumentando la probabilidad de lograr capturar adecuadamente la conectividad del entorno.

B.5.1.3. *Sampling Inside Narrow Passages*

Estas estrategias intentan generar suficientes configuraciones libres dentro de pasajes angostos. Las dos más conocidas son *Bridge Planner* y *GVD* (*Generalized Voronoi Diagrams*). En el caso de los

GVD además se asegura que las configuraciones muestreadas se encuentran a máxima distancia de los obstáculos circundantes.

B.5.1.4. *Visibility-Based Sampling*

El objetivo de los *PRM* basados en visibilidad es generar *roadmaps* de visibilidad con pocos nodos a partir de estructurar el espacio de configuraciones en dominios de visibilidad. Definen un dominio de visibilidad de una configuración q como el conjunto de todas las configuraciones que pueden conectarse a q. De este modo, a diferencia de los *PRM*, en la etapa de aprendizaje no se aceptan todas las configuraciones libres sino sólo las q que cumplen:

1. q no puede estar conectada a ningún nodo existente en el *roadmap* (es un nodo nuevo)
2. q conecta al menos dos nodos existentes

B.5.1.5. *Manipulability-Based Sampling*

Estos algoritmos basan su estrategia de muestreo en sesgar, dinámicamente, la distribución uniforme. En el contexto de brazos robóticos suelen presentarse escenarios con zonas donde la manipulabilidad es restringida. Básicamente, dónde la manipulabilidad es alta se realiza muestreo aleatorio de las configuraciones y cuando es restringida se rechazan configuraciones con probabilidad proporcional al valor de manipulabilidad asociado.

Otras

Existen algoritmos de muestreo que se basan en otras estrategias: *Quasirandom Sampling*, *Grid-Based Sampling* y *Connection Sampling*.

B.5.2. *Single-query planning*

A diferencia de como lo hacen los algoritmos de *multi-query planning*, no hay ninguna etapa de precálculo en los algoritmos que adoptan el enfoque *single-query planning*. En lugar de ello, se construye un *roadmap* local más chico o parcial en simultáneo mientras se realiza una consulta. Inicialmente, se muestrean configuraciones en zonas cercanas a las configuraciones q_{init} y q_{goal}. No se desea construir un mapa de todo el entorno. De este modo, el *roadmap* consiste originalmente en dos árboles cuyas raíces son las configuraciones q_{init} y q_{goal}, que se van expandiendo con sucesivos muestreos hasta que es posible conectar un nodo de un árbol con el del otro. Ambos árboles son expandidos de manera idéntica. De este modo, por construcción, siempre existe un camino libre de colisiones entre la raíz de un árbol y cualquiera de los restantes nodos.

La figura B.17 muestra un grafo probabilístico -resultado de un algoritmo de *single-query planning* en un entorno bidimensional. Los nodos destacados con un círculo corresponden a las configuraciones q_{init} y q_{goal}.

Típicamente, las dos formas más frecuentemente utilizadas para lograr este objetivo son: *Expansive Space Tree (EST)* y *Rapidly-exploring Random Tree (RRT)*.

B.5.2.1. *Expansive Space Tree*

Asigna pesos a los nodos de forma de representar la densidad del grafo en la vecindad de cada nodo. Luego, para expandir el árbol, asigna una probabilidad a los nodos de modo que sean más probables aquellos cuyo vecindario está menos poblado.

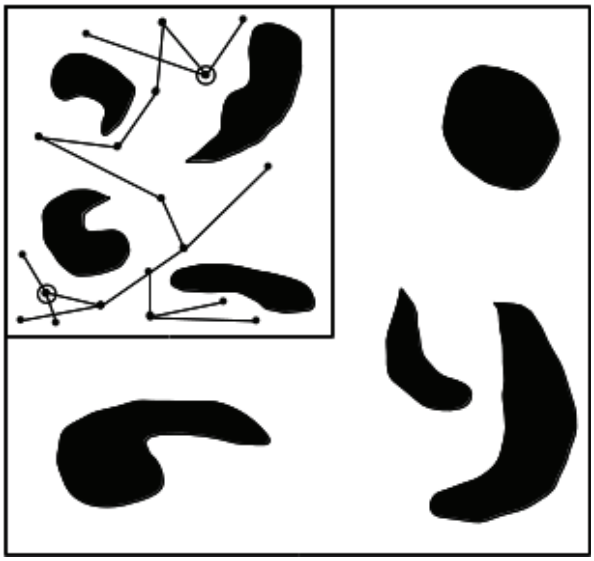


Figura B.17: *Random sampling. Single-query planning.*

B.5.2.2. *Rapidly-exploring Random Tree*

Tiene por objetivo explorar el espacio de configuraciones en forma rápida e incremental. La forma de hacerlo suele ser sesgando la exploración al asignar mayor probabilidad de elección -para ser expandido- al nodo que se encuentre más cercano a la configuración objetivo. [KL08]

B.5.3. Completitud probabilística

En general, los planificadores basados en algoritmos que implementan *random sampling* no son capaces de detectar la no existencia de un camino libre de colisiones. Esto se debe fundamentalmente al hecho de que el número de nodos a muestrear es un parámetro del algoritmo. Este valor se intenta estimar de modo que el espacio de configuraciones sea muestreado adecuadamente pero el algoritmo termina si ningún camino es encontrado. De modo que estos algoritmos no son completos. En cambio, sólo pueden garantizar completitud probabilística. Esto significa que, cuando existe un camino libre, el algoritmo es capaz de encontrarlo con alta probabilidad. Además, la completitud probabilística sólo asegura un adecuado desempeño computacional cuando existe un camino. A pesar de ello, esta débil forma de completitud permite que estos algoritmos sean capaces de resolver en la práctica problemas de más dimensiones que los algoritmos completos.

B.5.4. Ventajas

Algunas de las ventajas que presentan estos algoritmos son:

- tienen buen desempeño incluso en espacios de configuración de muchas dimensiones.
- son fáciles de implementar.
- en entornos competitivos, suponen una dificultad al adversario en virtud de su naturaleza estocástica.

B.6. Métodos de resolución basados en IA

Con el propósito de abordar algunas de las extensiones al problema básico de planificación de movimientos o mejorar la eficiencia computacional de los métodos tradicionales o probabilísticos (en ambos casos, basados en una representación completa del entorno), se han realizado variadas propuestas que se basan en métodos o técnicas provenientes del campo de la Inteligencia Artificial.

En tal sentido, en los últimos quince años se han presentado resultados auspiciosos que dan cuenta de soluciones robustas, flexibles, tolerantes a los cambios en el entorno, confiables y computacionalmente eficientes.

Los enfoques más comúnmente aplicados se basan en redes neuronales artificiales, sistemas *fuzzy* o neuro-*fuzzy*, aprendizaje por refuerzo, métodos bioinspirados y algoritmos evolutivos.

A continuación se señalan, en forma resumida, algunos de los usos u objetos de aplicación de las técnicas y métodos mencionados. Por el contrario, la utilización de algoritmos evolutivos es estudiada con mayor nivel de detalle en la sección B.6.1:

- redes neuronales artificiales
  - crear campos artificiales de potencial [LK97]
  - integrar la cinemática y la dinámica de un sistema robótico no-holonómico en un módulo controlador de movimientos [FL98]
  - cálculo del espacio de configuraciones libres y de caminos libres de colisión [Jan04]
  - sistema de navegación para entornos desconocidos basado en aprendizaje por refuerzo y redes neuronales de estructura dinámica auto-adaptativa [QFHR09]
- sistemas fuzzy y neuro-fuzzy
  - controlador de sistema multi-robot para entornos parcialmente desconocidos [Par05]
  - controlador de movimientos para evitar múltiples obstáculos conocidos [LLC06b, LLC06a]
  - controladores de sistemas multi-robot para entornos con obstáculos estáticos y estacionarios [PPP06, MABS08, KMRB10]
  - planificadores basados en campos artificiales de potencial que utilizan un sistema fuzzy para obtener el gradiente de las funciones de potencial y navegar en entornos con obstáculos estacionarios [IMS07]
  - controlador de movimientos para entornos dinámicos o parcialmente conocidos basado en la determinación de puntos intermedios y su elección mediante un sistema fuzzy de inferencias [CY07]
  - controlador de movimientos para entornos dinámicos [HGD08]
- aprendizaje por refuerzo
  - planificador de caminos para entornos dinámicos basado en grillas de ocupación como modelo del mundo [TTL+02]
  - control de movimientos para robots no-holonómicos aplicado al problema de seguimiento de trayectorias [SWI06]
  - planificador de trayectorias aplicado al problema de fútbol de robots [SFM06]
  - aprendizaje de comportamientos reactivos para navegación en entornos dinámicos y desconocidos basado en sistemas neuro-fuzzy [DCY08]

- control de movimientos para seguimiento de caminos [ZXLY09]
- sistema de navegación para entornos desconocidos basado en módulos de sistemas de inferencia fuzzy y redes neuronales artificiales [CFR10]

- *ant systems*
  - planificador de caminos para entornos estáticos y obstáculos conocidos [LLP+05]
  - planificador de caminos para entornos estáticos, multi-obstáculo y con obstáculos cóncavos basado en Campos Artificiales de Potencial [ZY06]
  - planificador de caminos para entornos estáticos basado en *roadmaps* [TM06]
  - planificador de caminos para entornos estáticos conocidos de gran tamaño basado en una combinación de aprendizaje por refuerzos y *ant colony systems* [VVLC07]
  - planificador de caminos mejorado con la aplicación de un algoritmo genético [GXT08]
  - planificador de caminos para entornos estáticos y obstáculos conocidos [MS10]

B.6.1. Algoritmos evolutivos

B.6.1.1. Introducción

Durante la década de 1950 los sistemas evolutivos comenzaron a despertar interés en un conjunto de investigadores dedicados a las ciencias de la computación que pensaban que la evolución natural podría utilizarse como una herramienta efectiva para resolver problemas de ingeniería.

Los algoritmos genéticos *(AG)* fueron concebidos originalmente por *John Holland* en la década de 1960 como técnicas probabilísticas de búsqueda que imitaran la evolución natural de las especies basándose en el principio de supervivencia de los individuos más aptos [Gol89, Mit98]. El objetivo principal fue estudiar el fenómeno de la adaptación de las especies y desarrollar formas para que el mecanismo natural de la adaptación pudiera introducirse en un sistema computacional.

En la actualidad, los conceptos centrales, entre las muy diversas implementaciones de la original idea de *Holland*, son: cromosoma, población, función de *fitness* y operadores genéticos. El funcionamiento del algoritmo se basa en que inicialmente, algunas potenciales soluciones al problema son codificadas como cromosomas que forman una población de individuos. Cada individuo es evaluado a través de una función de *fitness*. Esta función entrega una medida de la aptitud de cada individuo para sobrevivir en su entorno y permite accionar un mecanismo de selección o disputa por la supervivencia, que al igual que la selección natural, elige o privilegia a los más aptos.

Estos individuos, a su vez, son elegidos para reproducir nuevas generaciones de individuos a partir de la aplicación de los operadores genéticos (siendo el cruzamiento y la mutación genética, los más comúnmente utilizados).

Recombinación/Cruzamiento Se utiliza para explotar la información contenida en las soluciones (intermedias) generando nuevos individuos a partir de dos o más existentes. Este operador puede consistir en dividir dos individuos en una parte elegida en forma aleatoria y combinar las partes resultantes entre sí para formar dos nuevos individuos.

Mutación Se utiliza para explorar el espacio de búsqueda. Este operador cambia alguna porción (elegida al azar) del individuo de forma de generar más diversidad y, potencialmente, mejores individuos.

Selección Se utiliza para implementar el principio de supervivencia de los individuos más adaptados (semejante al esquema de selección natural). Este operador selecciona a los individuos que conformarán la siguiente generación de soluciones.

Este proceso es repetido generación tras generación hasta que la población de individuos converge a representaciones de soluciones de buena calidad, donde el mejor individuo tiene buenas chances de representar una solución óptima o cercana al óptimo.

B.6.1.2. Aplicación

Las soluciones al problema de planificación de movimientos basadas en el uso de Algoritmos evolutivos suelen formularse reinterpretando el problema de planificación como un problema de aprendizaje por refuerzos[9].

Este tipo de problemas toma como hipótesis, que el comportamiento inteligente surge de un agente que intenta maximizar la esperanza, a largo plazo, de la suma de refuerzos recibidos -usualmente en un ambiente desconocido-. La meta es que el agente aprenda a comportarse de manera (cuasi-)óptima solamente guiados por su afán de maximizar una señal de refuerzo, pero sin la presencia de un experto que le indique qué acciones tomar en cada momento. [Diu10, SB98a]

En tal sentido, en un problema de AR pueden identificarse cuatro elementos principales [SB98b]:

- la política: o función del agente, que determina el comportamiento del agente en cada momento.
- la función de refuerzo: que define el objetivo en un problema de AR, definiendo una correspondencia entre cada estado percibido por el agente y un refuerzo (recompensa o penalización) que indica o mide que tan deseable de alcanzar es dicho estado.
- la función de valor: mientras la función de refuerzo indica qué es bueno -considerando solamente el presente-, esta indica qué es bueno en el largo plazo. Puede verse también como la cantidad de refuerzos que un agente puede acumular en el futuro a partir del presente estado. Aunque varios métodos de AR utilizan una función de valor, esta no es estrictamente necesaria. Otros métodos de búsqueda como Algoritmos genéticos, Programación genética, Recocido simulado (*Simulated Annealing*) y bioinspirados (*Ant Colony Systems*) son utilizados para resolver problemas de AR. Estos métodos realizan un búsqueda directamente en el espacio de políticas sin requerir nunca de una función de valor.
- el modelo del entorno: busca imitar el comportamiento del entorno permitiendo la predicción tanto de estados como recompensas futuras. La incorporación de modelos y planificación en AR es relativamente nueva. Actualmente, son utilizados tanto métodos basados en modelo como libres de modelo.

Finalmente, el esquema de interacción entre un agente y su entorno en un problema de AR se puede apreciar en la figura B.18.

En dicho modelo, el agente y el entorno interactúan en cada paso de una secuencia discreta de pasos de tiempo $t=1,2,3,....$ A cada paso de tiempo t, el agente percibe su entorno recibiendo una representación del estado $s_t \epsilon S$ -donde S es el conjunto de los posibles estados-, basándose en esa información elige una acción $a_t \epsilon A(s_t)$ -donde $A(s_t)$ es el conjunto de acciones disponibles en el estado s_t-. En el siguiente paso de tiempo, $t+1$, el agente recibe, como consecuencia de la acción elegida, una recompensa $r_{t+1} \epsilon R$, y percibe al entorno en un nuevo estado s_{t+1}. [KLM96]

[9] En la década de 1980 y 1990, a partir de ideas provenientes de la psicología, teoría de control, optimización y microeconomía, nació como subcampo de la Inteligencia Artificial y el Aprendizaje Automático, el Aprendizaje por Refuerzo (AR). [Diu10]

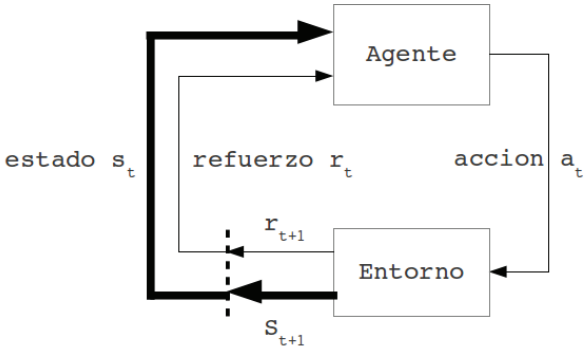


Figura B.18: Interacción agente-entorno en *AR*.

B.6.1.3. Casos de estudio

Algunas características de los Algoritmos Evolutivos los han convertido en serios candidatos para su aplicación al problema de planificación de movimientos. Las más notorias y usualmente resaltadas son:

- buen desempeño en espacios de búsqueda de dimensión grande
- tolerancia a diferentes tipos de funciones a optimizar (no requieren linealidad, continuidad ni diferenciabilidad, p.ej.)
- procesamiento paralelo

A continuación se describen, de un conjunto de aplicaciones de Algoritmos Genéticos al problema de planificación de movimientos, los aspectos más destacados:

- En el contexto de un sistema robótico de carga, en un entorno estático conocido totalmente, se plantea el uso de un algoritmo genético para computar caminos libres de colisión donde se minimice el tiempo de traslado y maximice la carga transportada por un robot móvil. La representación de los individuos (caminos) consta de secuencias de puntos -nodos de un *roadmap*- de largo variable. La función de *fitness* se implementa utilizando un sistema de inferencia difusa para facilitar la interacción con los humanos expertos en el problema. Sólo se realizan pruebas simuladas. [SF93]
- En el contexto de un sistema robótico industrial, en un entorno estático conocido totalmente, se utiliza un algoritmo genético para computar caminos libres de colisión para un brazo robótico de una articulación. [ATBM93]
- Simulaciones con brazos robóticos de hasta 3 articulaciones en entornos estáticos conocidos de tres dimensiones. [WGA06]
- Se plantea el uso de un algoritmo genético para computar caminos libres de colisión en entornos estáticos y dinámicos. La representación de los individuos (caminos) consta de secuencias de puntos -celdas de una grilla regular- de largo variable. Sólo se consideran individuos factibles y se busca explotar el conocimiento del entorno para el diseño de operadores genéticos a medida. Se utiliza un sistema de inferencia difusa para adaptar dinámicamente las probabilidades de aplicación de los operadores de mutación y cruzamiento, procurando evitar caer en mínimos locales. Sólo se realizan pruebas simuladas. [LZY+06, LTXZ06]

- Se plantea el uso de un algoritmo genético para computar caminos libres de colisión en entornos estáticos y dinámicos. La representación de los individuos (caminos) consta de secuencias de puntos de R^2 de largo variable. Se consideran tanto individuos factibles como infactibles, a los efectos de asegurar cierta diversidad genética. La función de *fitness* pondera tres criterios para evaluar a los individuos: largo, seguridad y suavidad. Cuando el entorno es dinámico y un camino en ejecución se torna infactible se adopta una estrategia de replanificación local ejecutando nuevamente el AG para evadir obstáculos y retornar al camino originalmente planificado. [LWW06]

- En el contexto de seguimiento de trayectorias, se plantea el uso de un algoritmo genético para mejorar la precisión del robot. Se utiliza una representación de individuos que incorpora las restricciones no-holonómicas del sistema. [WQY06]

- Basándose en algunas características de ciertos ecosistemas se propone un mecanismo para mejorar el desempeño y la convergencia -manteniendo la diversidad genética- de algoritmos genéticos aplicados en la resolución del problema de planificación en entornos estáticos. [HX07, DCXG08]

- Se propone el uso de un algoritmo genético para la planificación de caminos en entornos dinámicos de dos dimensiones. Además se realiza fusión de sensores como técnica para la detección dinámica del entorno circundante y se combina el algoritmo genético con la técnica de recocido simulado (*simulated annealing*) para mejorar la capacidad de búsqueda local. Se realizan pruebas en simulador. [YYY+08]

- Se propone una arquitectura en dos capas para resolver la planificación global de un camino libre de colisiones en un entorno dinámico de dos dimensiones mediante el uso combinado de un algoritmo genético con la técnica de simulado recocido, y el uso de un sistema de inferencia difusa jerárquico (en capas) para resolver localmente la evasión de obstáculos que se interpongan durante la ejecución del camino planificado. [BYW08]

- Introduce el modelado de zonas del entorno que presenten dificultades para la navegación, como ser elevaciones o rugosidades, mediante el uso de un sistema de inferencia difuso que entrega una noción de la dificultad que presenta cada zona. [MAT08]

- Se plantea una solución basada en un modelo del entorno basado en una grilla regular con información de la ubicación de los obstáculos estáticos y la posibilidad de introducir información relativa al movimiento de obstáculos estacionarios o dinámicos en base a la detección mediante los sensores del robot. El algoritmo genético, además de acceder a esta representación, pondera los caminos en base a tres criterios: largo, seguridad y suavidad. Se utilizan operadores genéticos a medida y se permite la co-existencia de individuos factibles e infactibles durante la evolución. [ZZZ08]

Otras aplicaciones relevadas presentan variantes menores respecto a las anteriormente descritas y se encuentran en [DNYS10, JQ10, GXTW08, WMSS08, YWY+07].