Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en Ciencias de la Computación

Planeación de movimientos con índices métricos

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Angello Jahir Hoyos Ibarra

Ensenada, Baja California, México

2016

Tesis defendida por

Angello Jahir Hoyos Ibarra

y aprobada por el siguiente Comité

Dr. Edgar Leonel Chávez González

Codirector del Comité

Dr. Ubaldo Ruiz López

Codirector del Comité

Dr. Miguel Ángel Alonso Arévalo

Dra. Mónica Elizabeth Tentori Espinosa

Dr. Jesús Favela Vara

Coordinador del Programa de Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez

Directora de Estudios de Posgrado

Resumen de la tesis que presenta Angello Jahir Hoyos Ibarra como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Planeación de movimientos con índices métricos

Resumen aprobado por:

Dr. Edgar Leonel Chávez González	Dr. Ubaldo Ruiz López
Codirector de Tesis	Codirector de Tesis

Los algoritmos de planeación de movimiento basados en muestras, tales como el mapa de caminos probabilísticos (PRM) y sus variantes PRM perezoso y PRM*, construyen una representación basada en un grafo del espacio libre de colisiones en el espacio de configuraciones. Para resolver problemas que involucren muchos grados de libertad (alta dimensión) o una gran cantidad de obstáculos, estos algoritmos requieren un muestreo denso del espacio de configuraciones. Cada vez que se agrega una nueva muestra al grafo, es necesario realizar el cálculo de sus k-vecinos más cercanos en el mismo, lo que resulta ser el componente más costoso del algoritmo y requiere de una estructura de datos adicional para la búsqueda. El objetivo de este trabajo es reducir el tiempo de construcción de los mapas de caminos en espacios de configuraciones de altas dimensiones. Para ello, en lugar de construir una estructura de datos adicional para efectuar la búsqueda de vecinos más cercanos, como se hace tradicionalmente en planeación de movimiento, se propone utilizar el grafo del mapa de caminos para realizar las búsquedas. Con el fin de mejorar el tiempo de consulta, se hace uso del grafo de proximidad aproximada (APG), desarrollado para realizar búsquedas en espacios métricos. En este trabajo, se propone una modificación del algoritmo de búsqueda utilizado en el APG integrándolo en la construcción del mapa de caminos y se presenta una técnica de refinamiento del grafo en dos etapas. Los resultados experimentales muestran que el tiempo de construcción del PRM y sus variantes es menor en comparación a utilizar los algoritmos de búsqueda de k-vecinos más cercano,s descritos en el estado del arte en planeación de movimiento, para problemas en altas dimensiones.

Palabras Clave: **Planeación de movimientos, mapa de caminos probabilísticos, búsqueda aproximada de vecinos más cercanos.**

Abstract of the thesis presented by Angello Jahir Hoyos Ibarra as a partial requirement to obtain the Master of Science degree in Master in Computer Science in Computer Science.

Motion Planning with metric indexes

Abstract approved by:

Dr. Edgar Leonel Chávez González	Dr. Ubaldo Ruiz López
Thesis Co-Director	Thesis Co-Director

In motion planning, sampling-based path planning algorithms, such as probabilistic roadmaps (PRM) and its variant Lazy PRM and PRM*, maintain a collision-free roadmap in the configuration space. In problems with large number of degrees of freedom (high dimension) or environments containing many obstacles, these algorithms require a large number of samples from the configuration space. Each time a new configuration is added to the roadmap, it is necessary to compute the k-nearest neighbors of that configuration in the roadmap. This procedure is usually the most expensive component of the algorithm and requires an external index for making the search. The goal of this work is to reduce the construction time of the probabilistic roadmaps in high dimensional configuration spaces. To achieve this goal, the proposal in this work is to use the same roadmap to perform the searches; Rather than constructing an additional data structure, as is traditionally done in motion planning.To improve the query time, this work makes use of the Approximate Proximity Graph (APG) that has been developed for search in general metric spaces. In this work the search algorithms used in APG have been adapted to work in roadmaps produced using the PRM algorithm and a two-stage graph refinement technique is proposed. Experimental results show that by using the APG algorithm the time to build a roadmap PRM and its variants is reduced, compared to the most popular algorithms for k-nearest neighbor search in motion planning problems in high dimensional configuration spaces.

Keywords: **Motion planning, probabilistic roadmap, approximate nearest neighbor search.**

Dedicatoria

A lo inesperado

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico a través de la beca no. 298008 para realizar mis estudios de maestría.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

A mis directores de tesis, el Dr. Edgar Chávez y el Dr. Ubaldo Ruiz quienes me guiaron pacientemente a lo largo del desarrollo de este trabajo. Les agradezco sus enseñanzas y el haber despertado en mi el interés en el área.

A los miembros de mi comité de tesis, la Dra. Mónica Elizabeth Tentori Espinosa y el Dr. Miguel Ángel Alonso Arévalo a quienes les agradezco su tiempo, sus observaciones y sugerencias para mejorar mi trabajo de investigación.

A mis padres, por haberme dado la vida. En especial a mi madre, por impulsarme a ser mejor cada día.

A mis amigos y compañeros del posgrado, con quienes compartí muchos momentos gratos a lo largo de mi maestría.

Tabla de contenido

Página

Lista de figuras

Lista de figuras (continuación)

Lista de figuras (continuación)

Capítulo 1. Introducción

Desde hace años, existe un creciente interés por contar con agentes inteligentes que sean capaces de asistir a la humanidad en tareas de la vida cotidiana. Para lograrlo, estos agentes deben ser capaces de moverse de forma autónoma.

El término planeación de movimiento, se refiere a la habilidad de un agente para calcular sus movimientos, los cuales le permitan alcanzar ciertas metas. El objetivo principal consiste en calcular una trayectoria que permita a un agente de manera autónoma ir desde un punto inicial a uno final en el ambiente.

En la Figura 1, se muestra un ejemplo clásico llamado el problema de desplazamiento del piano (Schwartz, 1983). Este problema consiste en determinar la secuencia de movimientos que se deben seguir para desplazar un piano de una habitación a otra, sin colisionar con los obstáculos que se encuentran presentes en el ambiente.

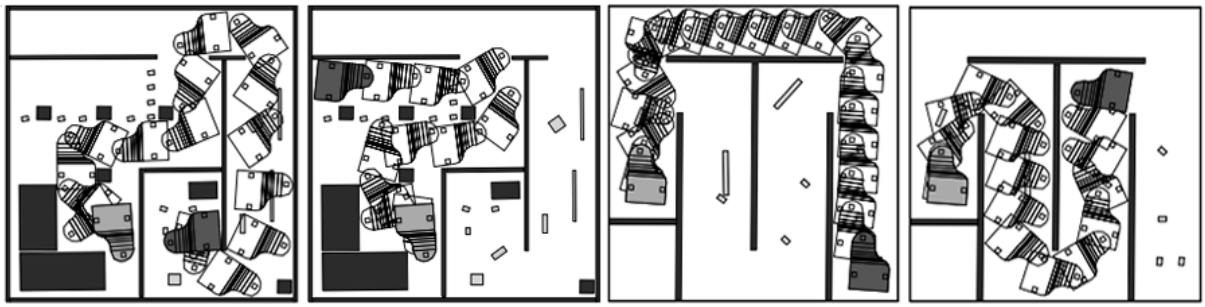


Figura 1: Problema del desplazamiento del piano en distintos ambientes.

La solución al problema de planeación de movimiento sirve como base para que un agente pueda realizar tareas adicionales. Por ejemplo; construir un mapa del ambiente, ensamblar un producto, encontrar un objeto o capturar a otro agente (LaValle, 2006). Estas tareas ayudan a dar solución a una gran cantidad de problemas que se presentan en áreas como robótica (ej. movimiento de robots humanoides, robots exploradores), inteligencia artificial (ej. solución de rompecabezas), medicina (ej. diseño de medicamentos) y entretenimiento (ej. animación de personajes) (LaValle, 2011).

La planeación de movimiento no es un problema sencillo. Se ha demostrado (Reif, 1979) que el problema de mover un agente evitando obstáculos estáticos en un espacio euclidiano de dimensión d, donde d corresponde al número de grados de libertad del agente, pertenece a la clase de problemas de decisión PSPACE-completo; es decir, se requiere de espacio polinomial para determinar si un agente puede desplazarse desde una posición a otra.

Actualmente, los algoritmos que se conocen para resolver de manera óptima problemas PSPACE-completos utilizan un tiempo de cálculo exponencial (Choset, 2005). Por lo tanto, en el área de robótica se hace uso de métodos probabilísticos para dar solución a los problemas de planeación de movimiento con un número arbitrario de estados.

El algoritmo *Mapas de caminos probabilísticos* (PRM) propuesto por (Kavraki y Latombe, 1998) es uno de los métodos probabilísticos más utilizados en planeación de movimiento. El objetivo de este algoritmo consiste en capturar la conectividad del espacio libre de colisiones en el ambiente por medio de un grafo. En la Figura 2, se puede apreciar un ejemplo del grafo PRM. La construcción de este grafo se realiza a partir de muestras aleatorias, donde cada una de ellas representa un movimiento del agente. Cada muestra se conecta con sus k-vecinos más cercanos, teniendo en cuenta los obstáculos y otras restricciones que pudieran presentarse. Una vez que se construye el grafo, el problema de planeación de movimiento se reduce a conectar la posición inicial y final del agente con sus k-vecinos más cercanos y a encontrar la trayectoria más corta en el grafo que conecte ambos puntos.

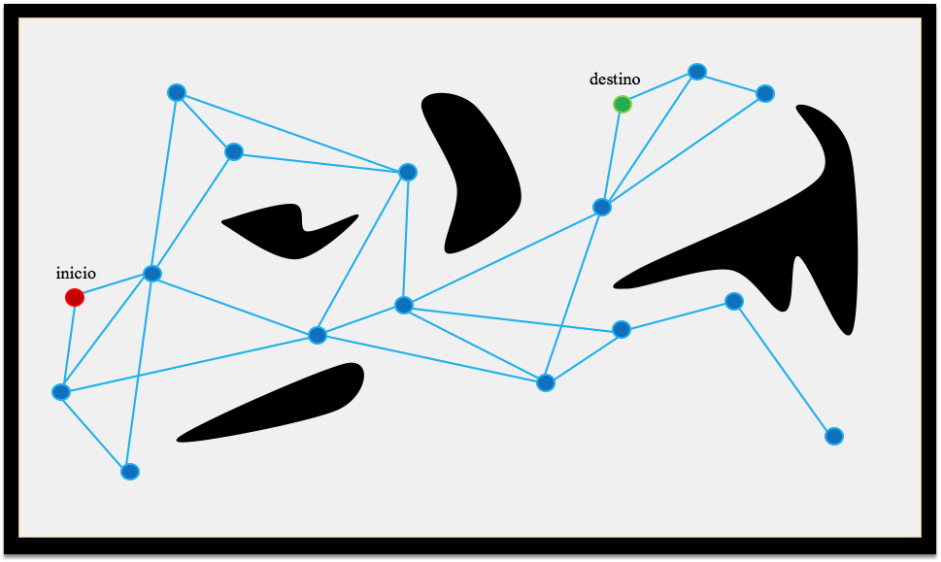


Figura 2: Ejemplo de un mapa de cambios probabilísticos (PRM).

1.1 Planteamiento del problema

De manera general, los métodos probabilísticos que se utilizan en la solución de problemas de planeación de movimiento requieren una estrategia de muestreo, un detector de colisiones y un método de búsqueda de vecinos más cercanos. El método de búsqueda es una parte fundamental para el rendimiento de los algoritmos de planeación.

En la literatura, se recomienda que para la construcción de PRMs en espacios de bajas dimensiones o con un número pequeño de vértices, la búsqueda de los k-vecinos más cercanos se realice de manera secuencial calculando la distancia a cada uno de los vértices.

Si se tiene un gran número de vértices, generalmente se hace uso de una subdivisión del espacio basada en árboles. En este caso, los árboles k-dimensionales (Atramentov y LaValle, 2002) son la técnica más popular. Los árboles k-dimensionales pueden construirse en $O(n \log(n))$ operaciones y tienen un tiempo de consulta de $O(dn^{1-\frac{1}{d}})$, donde n es el número de vértices en el árbol y d es la dimensión del espacio.

Sin embargo, se conoce que el desempeño de los algoritmos para la búsqueda de los k-vecinos se degrada a una búsqueda secuencial a medida que la dimensión intrínseca de los datos aumenta (Plaku y Kavraki, 2006).

Otro aspecto a tomar en cuenta, es que la mayoría de los algoritmos de búsqueda de vecinos más cercanos se diseñaron para realizar consultas rápidamente; pero no se considera el tiempo que toma procesar los datos para crear una estructura de búsqueda. Este es un problema para los algoritmos de planeación de movimiento, que necesitan procesar los datos lo más rápido posible.

1.2 Propuesta de solución

En esta tesis, atacamos el problema de calcular rápidamente los vecinos más cercanos en un mapa de caminos para espacios en altas dimensiones. Para lograrlo, se propone un enfoque que abandona la estrategia de utilizar una estructura de datos adicional, como los árboles k-dimensionales y se opta por utilizar el mismo mapa de caminos para realizar la búsqueda de los k-vecinos más cercanos. Con esto, se evita el costo de construir una estructura de datos adicional. Para mejorar el tiempo de búsqueda, se hace uso de un grafo de proximidad aproximada (APG) (Malkov *et al.*, 2012), el cual se diseñó para realizar la búsqueda de vecinos más cercanos en espacios métricos de altas dimensiones.

El APG ha atraído mucha atención en la comunidad especializada en búsquedas, ya que es muy sencillo de construir y tiene excelentes tiempos de búsqueda en espacios de altas dimensiones. La construcción del APG es incremental, al igual que el PRM y consiste en una simple regla. Para insertar el i-ésimo elemento al grafo, se busca de manera local a los k-vecinos más cercanos entre los i-1 elementos que ya se encuentran en el grafo, los cuales se enlazan al nuevo elemento.

Los experimentos realizados por Malkov *et al.* (2014) utilizando el APG muestran que las búsquedas presentan un excelente desempeño al incrementar el número de nodos en el grafo, incluso en altas dimensiones. Por tal motivo, se propone adaptar el APG como método de búsqueda en la construcción del mapa de caminos y así mejorar el rendimiento del PRM.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar un algoritmo de planeación de movimiento basado en muestras que mejore el rendimiento de los algoritmos actuales en problemas de altas dimensiones.

1.3.2 Objetivo específicos

- Diseñar e implementar un algoritmo de planeación de movimiento que integre el funcionamiento del algoritmo PRM y la búsqueda de vecinos más cercanos utilizando un APG.
- Evaluar el desempeño del algoritmo propuesto y los métodos de búsqueda mencionados en la literatura por medio de un conjunto de casos de prueba. Los cuales consideran distintos números de muestras y dimensión del espacio para la construcción de PRMs.
- Caracterizar las ventajas y desventajas de utilizar una búsqueda de vecinos aproximada para la solución de los problemas de planeación de movimiento.

1.4 Organización de la tesis

El resto de este trabajo de tesis se organiza de la siguiente manera. En el Capítulo 2, se resumen los enfoques clásicos existentes en el área de planeación de movimiento. En el Capítulo 3, se hace una breve descripción del problema de búsqueda de vecinos más cercanos y los métodos utilizados en planeación de movimiento. En el Capítulo 4, se propone una técnica de búsqueda de vecinos aproximada y se explica

su integración en el algoritmo de planeación de movimiento PRM. En el Capítulo 5, se describen los experimentos realizados y se presentan los resultados obtenidos. Por último, en el Capítulo 6, se presentan las conclusiones de esta investigación y el trabajo futuro.

Capítulo 2. Planeación de movimiento

En la definición de los problemas de planeación de movimiento (Latombe *et al.*, 1991) se asume un espacio euclidiano de dos o tres dimensiones, denominado espacio de trabajo $\mathcal{W}$. Además, se tiene un agente (robot) $\mathcal{A}$ y un conjunto de obstáculos $\mathcal{O}$ estáticos. Una configuración q es un vector que determina la posición y orientación de $\mathcal{A}$ con respecto a un marco de referencia fijo en $\mathcal{W}$, es decir, cada configuración q corresponde a un vector con el número de parámetros reales que determinan una posición del agente. Cada posición se puede representar como un punto en $\mathbb{R}^d$, siendo d igual al número de grados de libertad con los que cuenta el agente. Mientras que $\mathcal{A}(q)$ denota la región que ocupa $\mathcal{A}$ en $\mathcal{W}$.

Si bien los problemas de planeación de movimiento se describen en el espacio de trabajo, su solución se lleva a cabo dentro de un ambiente controlado denotado como *espacio de configuraciones* $\mathcal{C} \subseteq \mathbb{R}^d$ (Lozano-Pérez, 1983). El espacio de configuraciones comprende todas las posibles configuraciones q que se pueden aplicar en el agente. El conjunto de configuraciones libres de colisión se denota como $\mathcal{C}_{lib}$ (1), mientras que el conjunto de configuraciones donde ocurre colisión con algún obstáculo se denomina $\mathcal{C}_{obs}$ (2). Formalmente:

$$\mathcal{C}_{lib} = \{\, q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O} = \emptyset \,\} \tag{1}$$

$$\mathcal{C}_{obs} = \mathcal{C} \setminus \mathcal{C}_{lib} \tag{2}$$

La representación basada en el espacio de configuraciones permite abstraerse completamente de la geometría del espacio de trabajo y del agente, con lo que podemos representar los movimientos como puntos en el espacio. El objetivo de la planeación de movimiento consiste en encontrar una trayectoria continua $\tau : [0, 1]$ en el espacio libre de colisiones $\mathcal{C}_{lib}$, donde $\tau(0)$ corresponde a la configuración inicial q_I y $\tau(1)$ corresponde a la configuración final q_F. La trayectoria τ consiste en una secuencia continua de configuraciones q de $\mathcal{A}$ que evita el contacto con cualquier región $\mathcal{C}_{obs}$, como se muestra en la Figura 3.

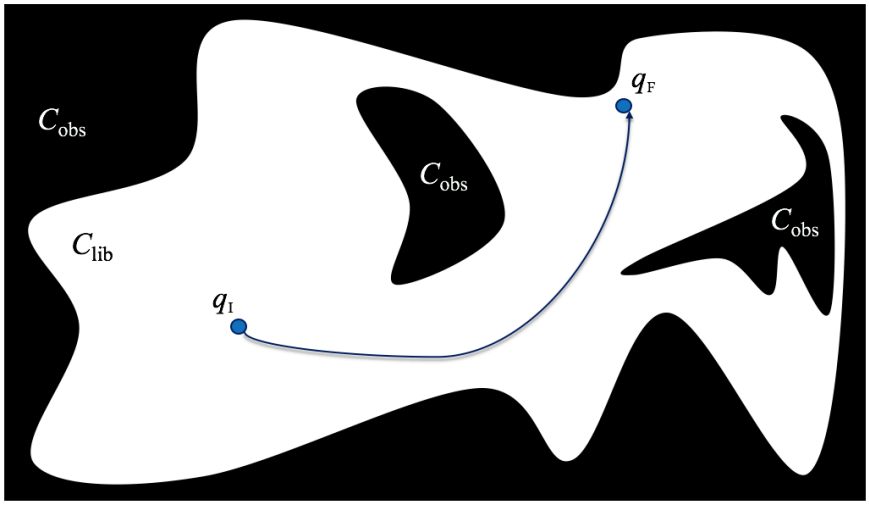


Figura 3: Ejemplo de una trayectoria de q_I hacia q_F en el espacio de configuraciones.

Los algoritmos completos[1] en el área, conocidos como métodos basados en planeación combinatoria (Craig, 1989), realizan una caracterización exacta del espacio de configuraciones y obtienen una solución, en la mayoría de los casos óptima. Sin embargo, el tiempo de cálculo requerido para realizar esta caracterización escala exponencialmente a medida que se aumenta la dimensión del espacio de configuraciones (Canny *et al.*, 1988).

2.1 Planeación basada en muestreo

La planeación basada en muestreo abandona el enfoque clásico de caraterizar explícitamente $\mathcal{C}_{lib}$ y $\mathcal{C}_{obs}$ en el espacio de configuraciones, esto permite obtener una solución rápida en problemas que se consideraban difíciles. Las soluciones que proveen estos métodos en ocasiones se consideran como subóptimas, principalmente en términos de distancia.

Los algoritmos basados en muestreo cuentan con la propiedad de ser probabilísticamente completos, es decir, encontrarán una solución, si es que esta existe, con probabilidad uno a medida que se incrementa el número de muestras en el espacio.

Es importante resaltar que en la planeación basada en muestreo cada muestra representa una configuración q del agente. La idea principal consiste en construir una representación de $\mathcal{C}_{lib}$ por medio de un grafo o un árbol utilizando muestras (configuraciones) aleatorias, las cuales se validan con un algoritmo de detección de colisiones (LaValle, 2011). El detector de colisiones es una función booleana, vista como

[1]Un algoritmo es considerado completo si este encuentra una solución, si es que existe, a un problema para el que fue diseñado y de lo contrario, informa que no es posible encontrar una solución.

una caja negra que provee el espacio de configuraciones para determinar si una muestra pertenece a $\mathcal{C}_{lib}$. A continuación, se describen los algoritmos de planeación de movimiento basados en muestreo más importantes en la literatura.

2.1.1 Mapa de caminos probabilísticos (PRM)

El mapa de caminos probabilísticos (Kavraki y Latombe, 1998), PRM por sus siglas en inglés, es un grafo $G(V, E)$ pensado para aplicaciones con múltiples consultas, siendo V el conjunto de vértices de G dado por un conjunto de muestras aleatorias en $\mathcal{C}_{lib}$. Cada una de las muestras corresponde a una configuración q. Una arista $e \in E$ conecta a un par de vértices $\{v, v'\} \in V$ siempre y cuando el segmento entre los vértices se encuentre totalmente contenido en $\mathcal{C}_{lib}$. El segmento se valida utilizando el detector de colisiones.

La construcción de un PRM se realiza en dos fases. La primera fase denominada de aprendizaje (Algoritmo 1), consiste en construir un mapa de caminos tomando muestras aleatorias q, en el espacio $\mathcal{C}$, como se muestra en la Figura 4 (a). En la Figura 4 (b), se aprecian en color amarrillo las muestras en $\mathcal{C}_{obs}$ que se desecharán y las muestras en color azul en $\mathcal{C}_{lib}$ que se tomarán como vértices del grafo.

Algoritmo 1 PRM (fase de aprendizaje)

$G = (V, E); V \leftarrow \emptyset; E \leftarrow \emptyset$
para $i = 0, ..., n$ **hacer**
    $q_R \leftarrow MuestraLibre_i$
    $K \leftarrow Vecinos(G, q_R, k)$
    $V \leftarrow V \cup \{q_R\}$
    para todo $k \in K$ **hacer**
        si $LibreDeColision(q_R, k)$ **entonces**
            $E \leftarrow E \in \{(q_R, k), (k, q_R)\}$
regresar $G = (V, E)$

Una vez que se tiene el conjunto de vértices V, cada uno de ellos se conecta a sus k vecinos más cercanos, Figura 4 (d), se comprueba que la arista que los une se encuentre en $\mathcal{C}_{lib}$, como se muestra en la Figura 4 (e). Este proceso se conoce como planeación local y consiste en discretizar una arista en diferentes puntos y verificar si cada uno de ellos se encuentra en $\mathcal{C}_{lib}$. Si todos los puntos se encuentran en $\mathcal{C}_{lib}$ la conexión se acepta, de lo contrario, la arista se desecha. Una vez realizada la verificación anterior, se repite el proceso para cada uno de los vértices, de esta forma se obtiene un grafo como se muestra en la Figura 4 (g).

La segunda fase que se denomina de consulta, consiste en agregar al PRM el par de vértices corres-

pondientes a la configuración inicial q_I y configuración final q_F, como se muestra en la Figura 4 (h). Dichos vértices se agregan al PRM siguiendo el Algoritmo 1. Finalmente, utilizando un algoritmo de búsqueda en grafos, como el algoritmo de Dijkstra (LaValle, 2006), se calcula una solución factible si es que esta solución existe como se muestra en la Figura 4 (i).

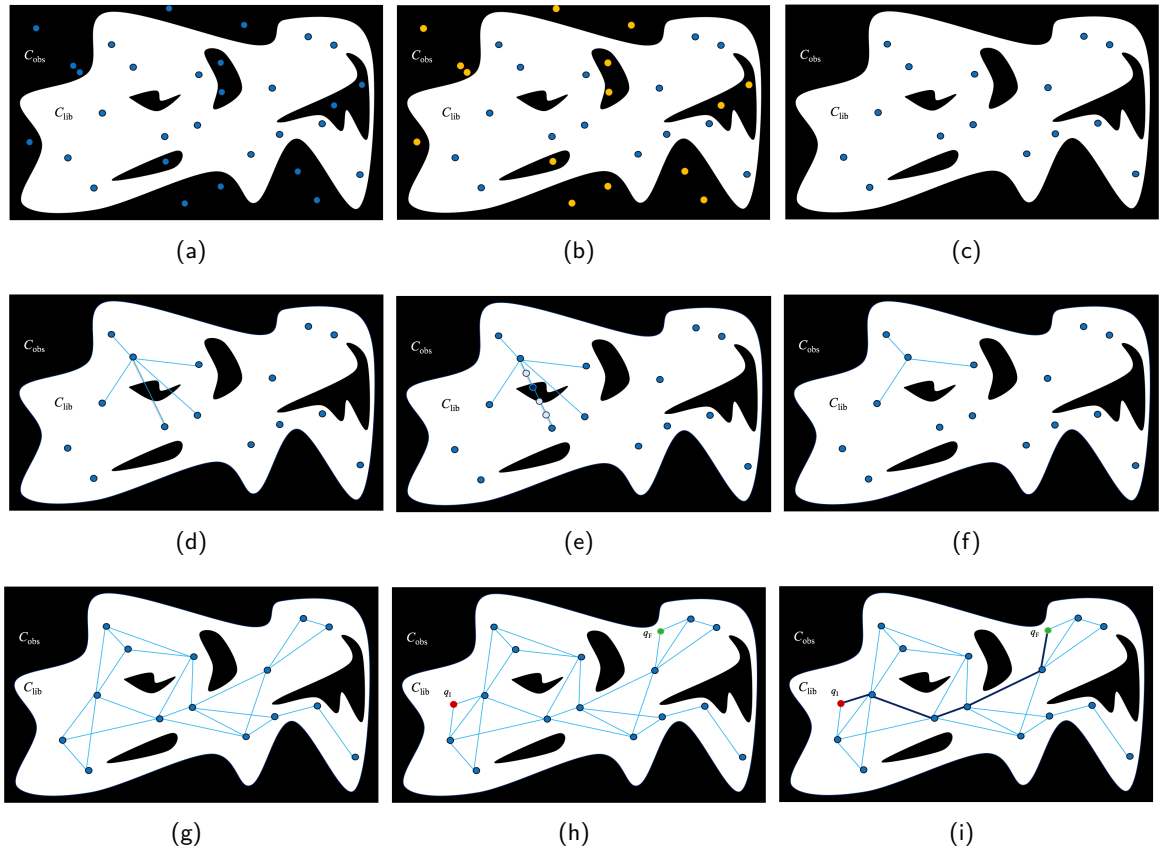
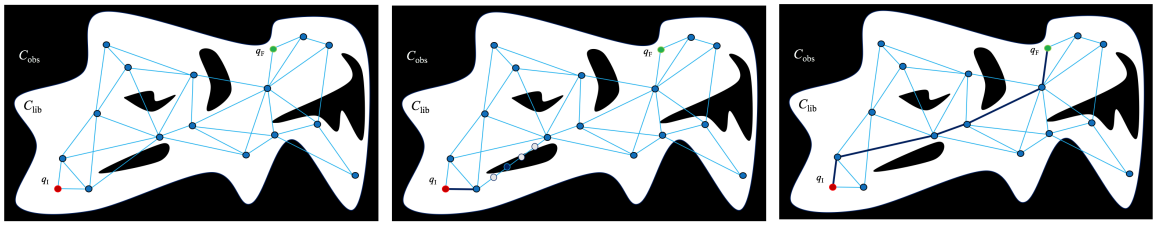


Figura 4: Funcionamiento del algoritmo PRM, en (a-f) se muestra la fase de aprendizaje, en la cual se construye el mapa de caminos, mientras que en (g-i) se muestra la etapa de consulta en la que se calcula la trayectoria a seguir desde un punto inicial q_I hasta el punto destino q_F

Al incrementar el número de muestras para la construcción del PRM se obtiene una mejor representación del espacio libre de colisiones $\mathcal{C}_{lib}$. Sin embargo, existe un compromiso, ya que cada muestra es validada para saber si esta se encuentra en colisión o no, lo que puede resultar costoso. Además es necesario verificar, con el algoritmo de planeación local, si la unión con cada uno de sus k-vecinos más cercanos se encuentra libre de colisión. Por otra parte, si se utiliza un número pequeño de muestras y de k-vecinos con la intención de mejorar el tiempo de construcción del PRM, es probable que el grafo generado no sea conexo, lo que implica que no se puede viajar entre cada uno de los vértices del mismo. Existen distintas variantes del PRM, algunas se describen a continuación:

2.1.1.1 PRM Perezoso

La propuesta de (Bohlin y Kavraki, 2000) con PRM perezoso consiste en minimizar el número de ocasiones en la que se utiliza el detector de colisiones durante el proceso de planeación y así, reducir el tiempo de ejecución del algoritmo. En este caso, los vértices V y arístas E del grafo PRM no se validan en la fase de aprendizaje. La validación de colisiones se realiza al buscar una trayectoria entre q_I y q_F. Si una colisión se detecta, el nodo y/o arista se eliminan del grafo y se busca una nueva trayectoria (ver Figura 5).



(a) (b) (c)

Figura 5: Se muestra un Lazy PRM, el cual realiza la validación de aristas al buscar una trayectoria.

Aunque esta estrategia mejora el tiempo de construcción del PRM, su desempeño continúa dependiendo de la cantidad y forma de obstáculos que se encuentran en el espacio de configuraciones. En el peor caso, se tiene que verificar cada uno de los vértices y aristas en el grafo para encontrar una trayectoría, al igual que en la implementación original del PRM.

2.1.1.2 PRM*

Recientemente, ha crecido el interés en la comunidad de robótica por diseñar algoritmos de planeación que optimizen las trayectorias bajo algún criterio, ya sea distancia, energía, etc. Al caracterizar sólo una porción del espacio de configuraciones el PRM produce soluciones que no tienen ningún tipo de garantía de optimalidad (Nechushtan *et al.*, 2010), es por eso que han aparecido distintos algoritmos y heurísticas para mejorar las trayectorias generadas por el mapa de caminos.

Actualmente, la técnica con mayor aceptación en la comunidad de robótica, es la propuesta por (Karaman y Frazzoli, 2011). Su algoritmo llamado PRM* cuenta con la propiedad de ser asintóticamente óptimo, es decir, a medida que el número de muestras tienda a infinito, la solución que se obtiene con este algoritmo converge a la solución óptima en distancia con probabilidad de uno. Lo anterior quiere

decir que las trayectorias que se encuentran en el PRM* se acercan a las trayectorias óptimas en el espacio libre $\mathcal{C}_{lib}$ a medida que el número de muestras en el grafo aumenta.

Una de las contribuciones que realiza (Karaman y Frazzoli, 2011) consiste en determinar el tamaño de la vecindad para la construcción del PRM*. Ellos demuestran que el número de vecinos más cercanos necesarios para obtener un mapa de caminos cuyas trayectorias se aproximen a una trayectoria óptima en distancia es igual a $k = 2e\log(n)$ siendo n el número de vértices en el grafo. Distintos trabajos (Jaillet y Porta, 2013), ((Lan y Schwager, 2013), (Luo y Hauser, 2014), validan este parámetro.

2.1.1.3 PRM* Perezoso

El algoritmo PRM* perezoso presentado por (Hauser, 2015), hace uso de las dos estrategias descritas anteriormente, la construcción PRM perezoso con la heurística de vecinos más cercanos del PRM*. En este algoritmo, además de minimizar el número de ocasiones en las que se utiliza el detector de colisiones, las trayectorias que se encuentran son óptimas en distancia de manera asintótica.

2.1.2 Árbol de exploración con expansión rápida (RRT)

Uno de los algoritmos más populares de planeación de movimiento es el árbol de exploración con expansión rápida (LaValle y Kuffner, 1999), RRT por sus siglas en inglés. Este algoritmo se pensó para encontrar una trayectoria entre un par de configuraciones a diferencia del PRM en el cual se pueden realizar múltiples consultas una vez que se construye.

En la construcción del árbol de exploración RRT, la raíz del mismo se define como la posición inicial del agente q_I, como se muestra en la Figura 6 (a) y utilizando vértices aleatorios denominados q_R se lleva acabo la exploración del espacio de configuraciones. Cada vértice corresponde a posibles configuraciones q del agente $\mathcal{A}$. El objetivo de los vértices q_R es sesgar la exploración a una determinada región, como se muestra en la Figura 6 (b). Inicialmente estos vértices, tienen alta probabilidad de caer en zonas inexploradas, permitiendo que el árbol tienda a expandirse rápidamente.

Cada vez que se genera un vértice aleatorio q_R se búsca el vecino más cercano en el árbol y se denomina temporalmente como q_C. Se crea un nuevo vértice q_N en dirección a q_R partiendo de q_C y a una distancia ϵ de este, como se muestra en la Figura 6 (c). La distancia ϵ se define como un movimiento "seguro" entre configuraciones, es decir, se asume que al desplazarse entre q_C a q_N no existirá una colisión.

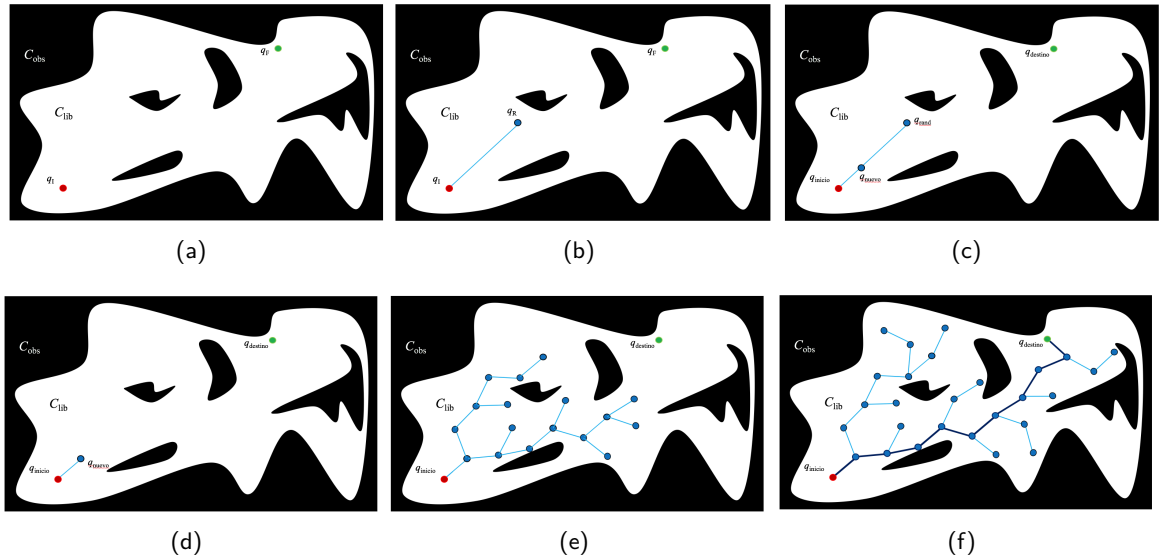


Figura 6: Construcción de un árbol de exploración con expansión rápida (RRT)

Una vez generado q_N se comprueba si este vértice se encuentra en $\mathcal{C}_{lib}$ de ser así q_N se agrega al árbol indicando que su predecesor es el vértice q_C. Si q_N se encuentra en colisión se elimina. El proceso anterior se repite iterativamente para agregar más vértices al árbol. En la Figura 6, se puede observar el proceso de construcción del RRT.

Para encontrar una solución por medio del RRT se necesita que cada cierto número i de iteraciones, q_F se elija como q_R, esto es, se sesga la expansión del árbol hacia la solución deseada. De momento no existe una heurística que indique el valor adecuado para i ya que depende de la cantidad de las iteraciones totales que se realicen, así como del espacio $\mathcal{C}$ en el que se está trabajando. Una vez que q_F se conecta satisfactoriamente al árbol, es posible detener el algoritmo debido a que se encontró una trayectoria.

Como se mencionó anteriormente, para la construcción del PRM y RRT, considerados como los principales algoritmos de planeación de movimiento basados en muestras, se requiere de un detector de colisiones para validar si un vértice se encuentran en $\mathcal{C}_{lib}$ y un método de búsqueda de vecinos para seleccionar los vértices más cercanos a una muestra específica. La búsqueda de vecinos es el componente del algoritmo que requiere más tiempo al realizar la construcción.

La comunidad de robótica, se ha enfocado en optimizar el funcionamiento interno de los algoritmos de planeación de movimiento, utilizando estrategias de muestreo distintas (Akgun y Stilman, 2011), minimizando la detección de colisiones (Bohlin y Kavraki, 2000) o mejorando las trayectorias que se encuentran (Elbanhawi y Simic, 2013). Sin embargo, las propuestas que buscan optimizar la velocidad

de búsqueda de vecinos más cercanos han sido pocas. En el siguiente capítulo, se hace un análisis del trabajo relacionado con la búsqueda de vecinos más cercanos en los algoritmos de planeación de movimiento.

Capítulo 3. Búsqueda de vecinos más cercanos

Como se mencionó en el capítulo anterior, los algoritmos de planeación de movimiento basados en muestreo requieren calcular los k-vecinos más cercanos para cada una de las muestras que se van agregando a la representación del espacio de configuraciones, ya sea un PRM o un RRT. A medida que se incluyen más muestras se mejora la calidad de la representación pero también aumenta el tiempo requerido para hacer el cálculo de los k-vecinos más cercanos.

Ante este problema, la comunidad en el área de robótica ha tomado dos enfoques. El primer enfoque y el más popular, consiste en diseñar algoritmos que requieran de una menor cantidad de muestras para obtener una solución. Por lo regular, este enfoque funciona solamente en problemas donde el espacio de configuraciones es considerado pequeño, con pocos o sin obstáculos en el mismo. Otro enfoque, menos explorado consiste en acelerar la búsqueda de vecinos más cercanos, ya que una característica deseada en los problemas de planeación de movimiento consiste en poder agregar la mayor cantidad de muestras en la menor cantidad de tiempo posible. A continuación, se describen los algoritmos de búsqueda de vecinos más cercanos, utilizados en los algoritmos de planeación de movimiento basados en muestreo.

3.1 Búsqueda secuencial

El algoritmo de búsqueda secuencial también conocido como método de fuerza bruta es la solución más simple al problema de los k-vecinos más cercanos. El procedimiento consiste en calcular la distancia desde el punto de consulta q hacia todos los elementos en un conjunto de datos S con n elementos, conservando los k más cercanos. En este caso, se requiere de un tiempo $O(n)$ para resolver el problema de los k-vecinos más cercanos. Esta estrategia se recomienda únicamente cuando se trabaja con una cantidad pequeña de muestras y el espacio de configuraciones tiene un dimensión baja, debido a que este método se vuelve intratable rápidamente y se cataloga como un cuello de botella en espacios de configuraciones de altas dimensiones.

3.2 Árbol k-dimensional

Un árbol k-dimensional (Atramentov y LaValle, 2002) se considera una generalización multidimensional de un árbol de búsqueda binario. El proceso de construcción del árbol k-dimensional, mostrado en la Figura 7 (b), es el siguiente. A partir de un conjunto de puntos P en $\mathbb{R}^2$ se ordenan los puntos con

respecto a la coordenada x. Se elige el punto medio $p \in P$ como la raíz del árbol, este punto pasa a dividir P en dos grupos por medio de un hiperplano, línea roja vertical en Figura 7 (a) correspondiente al nodo raíz de la Figura 7(b). Para cada una de las dos partes restantes, se ordenan los puntos por la coordenada y, se encuentran sus puntos medios y estos pasan a ser los hijos de la raíz del árbol, líneas azules horizontales en Figura 7(a). Este proceso se repite recursivamente en cada nivel alternando cada una de las coordenadas hasta obtener un resultado como el que se muestra en las Figuras 7(a) y 7 (c). El procedimiento anterior, se puede aplicar a conjuntos de puntos que pertenezcan a $\mathbb{R}^n$ alternando cada una de las n coordenadas para formar las subdivisiones.

Si se tiene un conjunto de datos fijos, es posible garantizar que se puede construir un árbol perfectamente balanceado. Este balanceo es importante ya que está asociado con un buen desempeño en los tiempos de búsqueda. En el caso del PRM, una vez que se define el conjunto de muestras que se encuentran en $\mathcal{C}_{lib}$ estas no se modifican, por lo que sería posible construir un árbol k-dimensional balanceado.

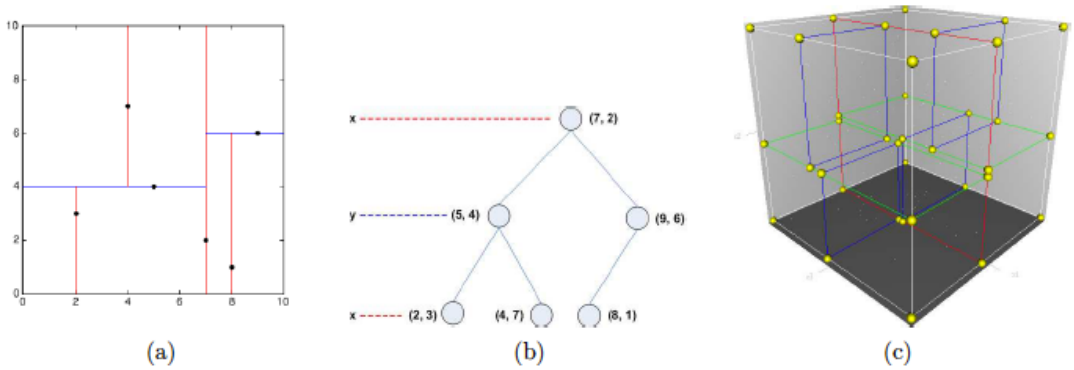


Figura 7: Ejemplos de un árbol k-dimensional: (a) de dimensión 2 (b) el árbol resultante con las subdivisones realizadas en (a) y en (c) se muestra un ejemplo de dimensión 3.

El algoritmo de búsqueda utilizando por el árbol k-dimensional (Algoritmo 2) se basa en la capacidad de ignorar una gran cantidad de vértices del árbol mediante la ejecución de una prueba que se explica a continuación. El algoritmo desciende a un vértice cuya región asociada contiene los puntos más cercanos a la consulta, las regiones en un árbol k-dimensional se delimitan por los hiperplanos generados en la construcción. Un vez identificada una región se calculan las distancias de los puntos de esta región hacia el punto de consulta y se conserva la más cercana. Posteriormente, de forma recursiva se visitan los vértices asociados a las regiones visitadas al descender y se valida si la distancia entre los vértices visitados y el punto de consulta son mayores al vértice más cercano encontrado hasta el momento. Esta operación se realiza en un tiempo $O(dn^{1-\frac{1}{d}})$.

Algoritmo 2 Búsqueda árbol k-dimensional

Entrada: Un árbol k-dimensional T, un nodo $consulta$, un nodo $raiz$, un entero $coord$

Valores iniciales $mejor\_distancia = infinito$, $mejor\_nodo = NULL$, $coord = 0$
$mejor\_nodo$, $mejor\_distancia$ y $dimension\_espacio$ son variables globales
function NNS(T, consulta, raiz, coord)
    si $T == NULL$|| distancia($consulta, raiz$) $> mejor\_distancia$ **entonces regresar**
    $dist\_consulta \leftarrow$ distancia($consulta, raiz$)
    si $dist\_consulta < mejor\_distancia$ **entonces**
        $mejor\_distancia \leftarrow dist\_consulta$
        $mejor\_nodo \leftarrow raiz$
    si $coord + 1 == dimension\_espacio$ **entonces**
        $coord = 0$
    si no
        $coord = coord + 1$
    si $consulta[coord] < raiz[coord]$ **entonces**
        NNS($consulta, raiz.hijo\_izquierdo, coord + 1$)
        NNS($consulta, raiz.hijo\_derecha, coord + 1$)
    si no
        NNS($consulta, raiz.hijo\_derecha, coord + 1$)
        NNS($consulta, raiz.hijo\_izquierda, coord + 1$)

Los árboles basados en particiones espaciales tales como el árbol k-dimensional son capaces de efectuar de manera eficiente la búsqueda exacta de vecinos más cercanos en espacios métricos euclidianos de baja dimensión (Ichnowski y Alterovitz, 2014). La popularidad de los árboles k-dimensionales en el área de planeación de movimiento se debe al buen desempeño en los tiempos de consulta que presentan, incluso ante un aumento en el número de elementos en el árbol. Sin embargo, el análisis realizado por (Plaku y Kavraki, 2006) demuestra que para ciertas métricas de distancia y distribuciones de datos, la eficiencia computacional del árbol k-dimensional disminuye cuando aumenta la dimensión del espacio. De hecho, se demuestra en (Hinneburg *et al.*, 2000) que después de una dimensión crítica, una búsqueda secuencial que examina todo el conjunto de datos es computacionalmente más eficiente que otros algoritmos de vecinos más cercanos.

3.3 Partición en cajas

En un intento por mejorar la búsqueda de vecinos más cercanos para el algoritmo RRT, (Svenstrup *et al.*, 2011) presentaron un algoritmo de búsqueda de vecinos más cercanos basado en una partición en cajas de tamaño fijo del espacio de configuraciones. La idea básica de esta propuesta consiste en minimizar el costo computacional de encontrar el vecino más cercano calculando la distancia a otros vértices solamente en aquellas cajas que se consideran relevantes, en lugar de tomar en cuenta a todos

los vértices en el espacio de configuraciones.

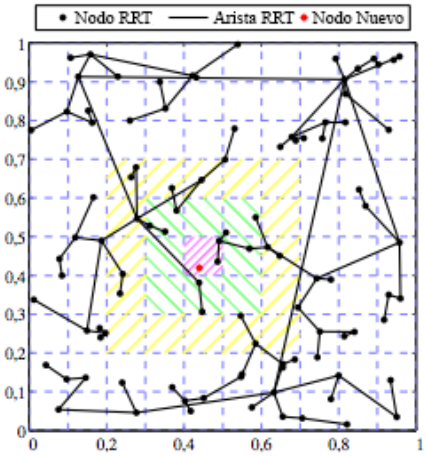


Figura 8: Ejemplo de una partición en cajas del espacio en dimensión 2. La caja central se muestra de color rosa, dos capas de cajas como expansión de la búsqueda en color verde y amarillo respectivamente.

El Algoritmo 3, describe el funcionamiento del método basado en cajas. Para agregar un nuevo vértice al RRT, se identifica a que caja pertenece por medio de la función EncontrarCaja y se realiza una búsqueda secuencial entre el nuevo elemento y todos los vértices que se encuentren en dicha caja, línea 3 del algoritmo. Una vez que se ha seleccionado un vecino potencial se comprueba si la distancia entre la consulta y el vecino es menor a la distancia entre la consulta y el borde más cercano, de ser así se considera que se ha encontrado al vecino más cercano. En caso contrario, se expande la búsqueda de manera secuencial a la siguiente capa de cajas adyacentes, línea 5 del algoritmo. Una vez que se identifican las cajas adyacentes se realiza una búsqueda secuencial en cada una de ellas, línea 6 del algoritmo. Una vez terminado este proceso existen dos posibles casos:

1. No fue posible encontrar un vecino más cercano al encontrado en la primera caja, por lo que el algoritmo regresa a ese elemento como el vecino más cercano de la consulta.

2. Se encontró un vecino más cercano al encontrado en la primera caja, por lo que el algoritmo valida si la distancia a este nuevo vecino es menor que la distancia al borde más cercano, dicho borde comprende las nuevas cajas adyacentes. Una vez validadas las distancias el algoritmo regresa al vecino más cercano o expande la cantidad de cajas a explorar.

Algoritmo 3 Búsqueda realizada por el método de cajas

Entrada: Un nodo $consulta$
Salida: El nodo más cercano a la consulta denotado como $nodo\_cercano$

```
1: cajaId ← EncontrarCaja(consulta)
2: areaDeBusqueda ← cajaId
3: nodo_cercano ← EncontrarVecino(areaDeBusqueda, consulta)
4: mientras distancia (consulta, borde_areaDeBusqueda) < distancia(consulta, nodo_cercano) hacer
5:     areaDeBusqueda ← expandirAreaDeBusqueda(cajaId)
6:     nodo_cercano ← EncontrarVecino(areaDeBusqueda, consulta)
7: InsertarEnCaja(consulta, cajaId)
8: regresar nodo_cercano
```

A pesar de que la idea general de este algoritmo consiste en descartar un gran número de vértices al seleccionar solamente un número limitado de cajas, es importante notar que la expansión hacia celdas adyacentes ocurre de manera recurrente cuando el número de vértices es muy pequeño ya que la mayoría de las cajas están vacias. Otro inconveniente, se presenta a medida que la dimensión del espacio de configuraciones aumenta, ya que la cantidad de cajas para construir una partición uniforme del espacio de configuraciones crece exponencialmente.

3.4 Celdas transformadas al azar

Los métodos descritos anteriormente se conocen como técnicas de búsqueda de vecinos exactas. En ocasiones, es posible sacrificar precisión por rapidez al utilizar métodos de aproximación, como en el caso de los algoritmos de planeación basados en muestras. El trabajo realizado por (Plaku y Kavraki, 2005) muestra que en ciertos problemas en espacios de altas dimensiones utilizar una búsqueda de vecinos aproximados mejora el desempeño de los algoritmos de planeación.

De acuerdo a lo anterior, (Kleinbort *et al.*, 2015) sugieren el uso de un método de búsqueda llamado celdas transformadas al azar, RTG por sus siglas en inglés. Este método de búsqueda aproximada consiste en dos pasos para resolver de manera aproximada el problema de los pares de puntos que se encuentran a una distancia r entre ellos, es decir, dado un conjunto P de n puntos y un radio de vecindad r encontrar todos los pares de puntos $p, q \in P$ tales que la distancia entre p y q sea como máximo r.

El radio utilizado en este trabajo corresponde al recomendado por los autores del PRM* Karaman y Frazzoli el cual es:

$$r_{PRM^*=(\log(n)/n)^{1/d}}$$

donde d corresponde a la dimensión del espacio y n al número total de muestras para la construcción del mapa de caminos. Note que el tamaño del radio disminuye con el número de muestras a medida que la dimensión del espacio incrementa. Si no se tiene cuidado en la utilización de ambos parámetros es posible que existan vértices que se encuentren completamente aislados de los demás, lo cual no es beneficioso para la búsqueda de trayectorias.

Algoritmo 4 RTG (P, r, c, m)

1: **para** $i = 0, ..., m$ **hacer**
2: Elegir una variación aleatoria de posición para la cuadrícula
3: $U \leftarrow \emptyset$
4: **para todo** $p \in P$ **hacer**
5: Calcular los vecinos de p en la celda u
6: $U \leftarrow U \cup u$
7: **para todo** $u \in U$ **hacer**
8: reportar todos los puntos en u que se encuentran a una distancia r

El Algoritmo 4 describe el RTG. Primeramente se coloca una cuadrícula de dimensión d en el que cada celda es de tamaño c, siendo c un "poco" más grande que r, esta cuadrícula se desplaza de acuerdo con un cambio uniforme elegido al azar y una vez definida cada elemento de P se asocia a una celda u. Para cada celda no vacía, se calcula la distancia entre todos puntos los asociados a una misma celda y se reportan los pares si la distancia calculada entre ellos es a lo más de tamaño r. El proceso anterior se repite m veces para garantizar, con alta probabilidad, que la mayoría de los vecinos de un punto a una distancia r serán encontrados. Para asegurar esto por cada iteración de m se utiliza una orientación aleatoria distinta de la cuadrícula como se muestra en la Figura 9.

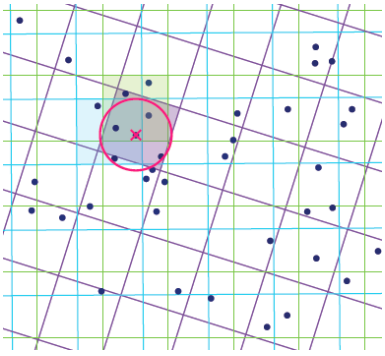


Figura 9: Ejemplo de un RTG en un espacio de dimensión 2.

Kleinbort et. al. recomiendan que el tamaño de las celdas c y el número m de cuadrículas desplazadas al azar se ajuste de acuerdo a las necesidades de cada problema, ya que ambos parámetros tienen un impacto importante en el rendimiento del algoritmo. Cuando se utiliza un valor de c o m muy pequeño,

el conjunto de vecinos reportados podría ser menor en comparación con el número real de vecinos más cercanos. Sin embargo, cuando se utiliza un valor grande de c, este algoritmo puede reportar todo el conjunto real de vecinos a distancia r a un costo mayor a realizar una búsqueda secuencial. Mientras más iteraciones se realicen con un valor de m mayor, mejores serán los resultados emitidos, sin embargo, existe un compromiso entre el tiempo de cálculo y la calidad de los vecinos encontrados y cualquier cambio sútil a c o m puede afectar el desempeño del algoritmo.

Además del problema mencionado anteriormente, los autores de este trabajo comentan que existe una restricción al construir una cuadrícula uniforme, ya que esta depende exponencialmente de la dimensión d. Por tal motivo este método, al igual que el de la partición en cajas, no son recomendados para problemas de planeación de movimiento en altas dimensiones.

Como resumen de este capítulo podemos resaltar varios puntos importantes. Una búsqueda secuencial es recomendable cuando se trabaja con una cantidad pequeña de muestras para la construcción del mapa de caminos. Cuando la cantidad de muestras aumenta se recomienda utilizar un árbol k-dimensional o uno de los métodos de partición del espacio, tomando en cuenta, que su desempeño depende de distintos parámetros así como de la dimensión del espacio de configuraciones con el que se este trabajando. Así mismo, cuando se tiene una estructura de datos adicional para realizar de la búsqueda de vecinos más cercanos, es importante considerar el costo computacional para su construcción ya que este puede superar al tiempo total de las búsquedas realizadas.

La búsqueda de vecinos más cercanos se considera un cuello de botella en los algoritmos de planeación de movimiento basado en muestras, en particular cuando la cantidad de vértices y la dimensión del espacio aumenta (Plaku y Kavraki, 2007). A veces es deseable sacrificar la precisión por velocidad mediante el uso de métodos de búsqueda aproximados (Indyk, 2004). Estos métodos pueden reducir drásticamente el tiempo de cálculo para las búsquedas de vecinos más cercanos. Sin embargo, no se ha estudiado si las pruebas de optimalidad para los métodos de planeación de movimiento asintóticamente óptimos siguen siendo válidas al utilizar búsquedas aproximadas.

Si bien existen muchas propuestas para resolver el problema de la búsqueda de los k-vecinos más cercanos, en la literatura identificamos una que cuenta con una serie de características que la vuelven atractiva al hacer frente a los problemas antes mencionados. El grafo de proximidad aproximada (Malkov *et al.*, 2012), conocido como APG por sus siglas en inglés, es una estructura de datos representada por un grafo que se construye de manera incremental al igual que el PRM. Además, el APG reporta un buen desempeño en los tiempos de búsqueda en espacios de altas dimensiones. Más detalles acerca de

la descripción e implementación del APG son presentados en el siguiente capítulo.

Capítulo 4. Grafo de proximidad aproximada

En este capítulo se describe el grafo de proximidad aproximada, conocido como APG por sus siglas en inglés. El APG es una estructura de datos presentada por Malkov *et al.* que ha atraído la atención de la comunidad especializada en búsquedas ya que es muy simple de construir y presenta tiempos de búsqueda menores a una búsqueda secuencial en espacios de altas dimensiones. La idea principal del APG consiste en construir un grafo, donde cada nodo está conectado con sus k-vecinos más cercanos.

De acuerdo a los autores de este trabajo, para obtener resultados precisos, un grafo de búsqueda debe contener el grafo Delaunay (De Berg *et al.*, 2008) como un subgrafo. Sin embargo, hay algunos problemas asociados a su construcción: 1) se requiere información acerca de la estructura interna del espacio métrico [1] y 2) se ve afectado por la maldición de la dimensionalidad [2] (Chávez *et al.*, 2001). Si la restricción de precisión en una búsqueda exacta es relajada, entonces el problema de encontrar de manera exacta a los vecinos más cercanos se puede transformar al problema de una búsqueda aproximada de vecinos más cercanos. Por lo que una representación exacta del grafo Delaunay ya no es necesaria. Este es el enfoque a seguir con el APG, el cual se diseño para resolver el problema de encontrar a los k-vecinos aproximados más cercanos en espacios métricos.

El proceso de construcción del APG se describe en el Algoritmo 5. Cabe resaltar que este algoritmo requiere la definición de una función Busqueda_Vecinos, la cual calcula los vecinos más cercanos en el grafo. Al presentar este trabajo (Malkov *et al.*, 2012), la búsqueda de vecinos más cercanos se realiza utilizando un enfoque voraz usando el mismo grafo, como se describe en el Algoritmo 6.

Algoritmo 5 Construcción APG

Entrada: Un conjunto $\mathcal{U}$ de n elementos
Salida: Un grafo $G = (V, E)$ conteniendo los k vecinos más cercanos de cada elemento en U

1: $G.V \leftarrow \emptyset$
2: $G.E \leftarrow \emptyset$
3: **para todo** $u \in U$ **hacer**
4: $\quad X_{cercano} \leftarrow$ Busqueda_Vecino(G, u, k)
5: $\quad G.V \leftarrow G.V \cup \{u\}$
6: $\quad$ **para todo** $v \in X_{cercano}$ **hacer** $G.E \leftarrow G.E \cup \{(u, v), (v, u)\}$
7: **regresar** G

[1] Un espacio métrico es un par (X, d) donde X es un conjunto no vacío de puntos y d es una función llamada distancia o métrica que satisface los axiomas de simetría, positividad y desigualdad del triángulo.

[2] La maldición de la dimensionalidad se refiere a varios fenómenos que se presentan al analizar y organizar los datos en espacios métricos de alta dimensión, por ejemplo, las distancias entre los elementos se vuelven similares entre sí.

Para conseguir una búsqueda logarítmica escalable por medio del algoritmo de búqueda voraz, es decir, que la complejidad del algoritmo incrementa logarítmicamente con el tamaño de la entrada, el grafo de búsqueda debe ser un grafo con la propiedad de mundo pequeño (Kleinberg, 2000). En el Algoritmo 5, esta propiedad es alcanzada sin conocimiento previo de la estructura interna del espacio de búsqueda. Por su construcción, el grafo tiene dos tipos de aristas con propósitos diferentes. Un subconjunto de aristas "cortas" que se utilizan como una aproximación del grafo Delaunay requerido por el algoritmo de búsqueda voraz, mientras que el subconjunto de aristas "largas" permiten escalar logarítmicamente en la búsqueda voraz.

Algoritmo 6 Búsqueda Voraz APG

Entrada: Un grafo $G = (V, E)$ un vertice inicial v_{inicio} y una consulta q
Salida: Un vertice $v_{min} \in G.V$ cuya distancia a q es un mínimo local

```
v_min ← v_inicio
d_min ← distancia(v_min, q)
v_siguiente ← NULL
X_cercanos ← {u ∈ G.V | (u, v_min) ∈ G.E}
para todo v ∈ X_cercanos hacer
    d_cercano ← distancia(v, q)
    si d_cercano < d_min entonces
        d_min ← d_cercano
        v_siguiente ← v
si v_siguiente != NULL entonces regresar Busqueda_Voraz(G, v_siguiente, q)
si no regresar v_min
```

El resultado de utilizar una búsqueda voraz puede ser ambiguo, en algunos casos se ha mostrado (Malkov *et al.*, 2012) que encuentra a los verdaderos vecinos más cercanos y en otros casos no. El resultado depende de la elección del vértice inicial para comenzar la búsqueda. Los vecinos encontrados para un nodo pueden presentar mínimos locales. Se dice que un elemento p es un mínimo local si sus vecinos se encuentran a una distancia mayor con respecto a una consulta, pero a su vez existe un elemento s el cual se encuentra más cerca de la consulta y no es vecino de p, como se muestra en la Figura 10. Así mismo, si no existe un elemento más cerca de la consulta que s se dice que este es un mínimo global.

Con la intención de reducir el error provocado por los mínimos locales, Malkov *et al.* propusieron utilizar m búsquedas iniciales, a partir de m vértices seleccionados aleatoriamente en el grafo y posteriormente elegir a los vecinos más cercanos del conjunto de elementos visitados. El procedimiento se describe en el Algoritmo 7. Este algoritmo requiere de la función Vertice_Aleatorio(G) la cual elige aleatoriamente un vértice del grafo G.

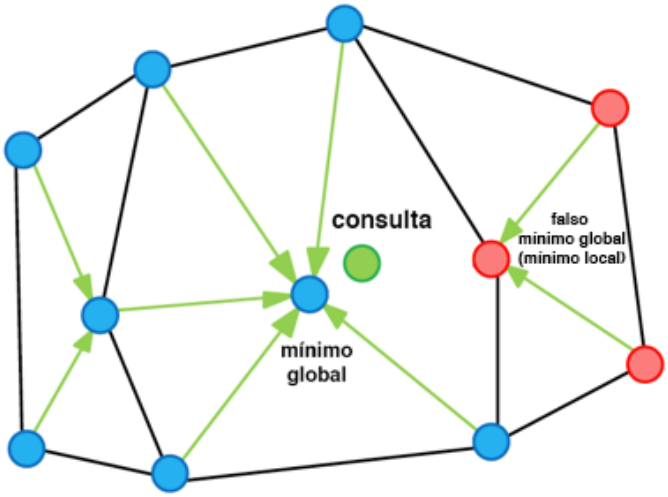


Figura 10: Problema de mínimo local en el APG.

Algoritmo 7 Búsqueda Múltiple

Entrada: Un grafo $G = (V, E)$, una consulta q y un número m de reinicios
Salida: Un conjunto $\mathcal{U}$ de vecinos más cercanos de q en G

1: $\mathcal{U} \leftarrow \emptyset$
2: **para** $i = 1, ..., m$ **hacer**
3: $\quad v_{init} \leftarrow Vertice\_Aleatorio(G)$
4: $\quad v_{min} \leftarrow Busqueda\_Voraz(G, v_{init}, q)$
5: $\quad$ **si** $v_{min} \notin \mathcal{U}$ **entonces**
6: $\quad\quad \mathcal{U} \leftarrow \mathcal{U} \cup \{v_{min}\}$
regresar $\mathcal{U}$

A partir de un vértice inicial que se elije aleatoriamente, existe una probabilidad p de encontrar los verdaderos vecinos más cercanos a un elemento particular q. Esta probabilidad no es nula debido a que siempre es posible encontrar de manera exacta a los vecinos más cercanos desde un vértice inicial.

Sin embargo, tampoco existe una garantía de que esto ocurrira cada vez que se realice una búsqueda. Para una consulta del elemento q la probabilidad de encontrar a los verdaderos vecinos más cercanos en un solo intento es p, entonces la probabilidad de encontrar a los verdaderos vecinos después de m intentos es $1 - (1 - p)^m$, así la precisión de la búsqueda incrementa exponencialmente con el número de intentos a probar. Si m es comparable a $|G.V|$, el algoritmo de búsqueda se convierte en una búsqueda exhaustiva, asumiendo que cada vértice se utiliza solamente una vez. Si el grafo del APG cuenta con la propiedad de mundo pequeño entonces es posible elegir un vértice aleatorio para relizar la búsqueda en una serie de pasos proporcial a $\log(n)$, manteniendo una complejidad logarítmica de la búsqueda.

Un parámetro importante del APG es el número de k-vecinos más cercanos a los que se conecta cada

nuevo vértice que se agrega. A medida que este k se incrementa la precisión y el tiempo de búsqueda de vecinos aumenta. Observe que este parámetro también esta relacionado con el tiempo de construcción del grafo. Malkov *et al.* sugieren la cantidad de vecinos $k = 3d$, donde d corresponde a la dimensión del espacio de búsqueda, como una buena elección para aplicaciones de búsqueda donde el costo de construir el APG se amortiza por el número de consultas que serán realizadas después de la construcción del grafo.

En (Malkov *et al.*, 2014) se propone un algoritmo más sofisticado para mejorar la búsqueda del APG. En este algoritmo, se utiliza una condición diferente. El algoritmo itera en elementos que no se han visitado previamente y que se encuentran cerca de la consulta. El algoritmo se detiene cuando después de una iteración los resultados de la consulta no se modifican. La lista de elementos visitados durante la búsqueda es compartida para evitar repetir la verficación de un vértice. El algoritmo de búsqueda se describe en el Algoritmo 8.

Algoritmo 8 Búsqueda Tabu

Entrada: Un grafo $G = (V, E)$, una consulta q y un número m de reinicios
Salida: Un conjunto $\mathcal{U}$ de vecinos más cercanos de q en G

Sea $\mathcal{U}$ una cola de prioridad mínima vacía de tamaño k
Sea $\mathcal{C}$ una cola de prioridad mínima vacía
Sea r la distancia al elemento más lejano a q en $\mathcal{U}$. Inicialmente $r = \infty$
$S \leftarrow \emptyset$
para $i = 1, ..., m$ **hacer**
    $c \leftarrow Vertice\_Aleatorio(G.V - S)$
    $S \leftarrow S \cup \{c\}$
    $Insertar(Distancia(c, q), c)$ en $\mathcal{U}$ y $\mathcal{C}$
    repetir
        Sea $(r_b, mejor)$ el elemento más cercano en $\mathcal{C}$
        Remover $mejor$ de $\mathcal{C}$
        si $r_b > r$ **entonces**
            interrumpir repetir
        $X_{cercanos} \leftarrow \{u \in G.V | (u, mejor) \in G.E\}$
        para todo $u \in X_{cercanos}$ **hacer**
            si $u \notin S$ **entonces**
                $S \leftarrow S \cup u$
                $Insertar(Distancia(q, u), u)$ en $\mathcal{U}$ y $\mathcal{C}$

En este trabajo de tesis como ya se mencionó, en lugar de construir una estructura de datos adicional para realizar la búsqueda de los k-vecinos más cercanos utilizados en la construcción del PRM, se hará uso de la información contenida en el mismo mapa de caminos. La motivación detrás de este enfoque radica en la similitud entre los algoritmos PRM y APG. En ambos casos, el objetivo consiste en construir un grafo en el cual cada nodo está conectado a sus k-vecinos más cercanos. Esto sugiere que el algoritmo

desarrollado para buscar en el APG se puede utilizar en el PRM. Es importante notar que en el caso del PRM, dos vértices vecinos se conectan si y sólo sí existe un camino libre de colisiones entre ellos, mientras que en el APG no se tiene noción de obstáculos en el espacio.

Naturalmente, se puede pensar que las arístas de mayor longitud cuentan con mayor probabilidad de colisionar con los obstáculos, así la mayoría de ellas no se consideran en la construcción del PRM. Por otro lado, estas aristas de mayor longitud son importantes para mantener la propiedad de mundo pequeño en el grafo del APG, la cual está relacionada con la complejidad logarítmica de búsqueda. Para mantener un balance, se hace uso del enfoque lazy PRM, descrito en el Capítulo 2. En este caso, el proceso de construcción del PRM es el mismo que el utilizado por el APG. Note que las aristas en el grafo generadas con el enfoque PRM perezoso únicamente se eliminan una vez que una trayectoria entre un par de vértices se calcula y valida.

Otro aspecto a tomar en cuenta es el número de vecinos más cercanos que se conecta a cada vértice. En el caso del PRM*, como se describió en el Capítulo 2, el valor de $k = 2e\log(n)$, donde n corresponde al número de muestras en el grafo, se sugiere que para alcanzar buenos resultados en la mayoría de las aplicaciones (Karaman y Frazzoli, 2011). Mientras que el APG requiere de una vecindad de $k = 3d$ para mantener su buena precisión de búsqueda. A medida que k se incrementa, la calidad de la búsqueda mejora, pero el tiempo de construcción del grafo se incrementa. En nuestro caso, estamos interesados en maximizar el número de vértices agregados al grafo en una cantidad de tiempo fija, pero también en mantener una buena precisión en el cálculo de los k-vecinos más cercanos para cada vértice que se agrega. Por tales motivos, en el siguiente capítulo, como parte de la experimentación se hará uso de ambas vecindades $k = 2e\log(n)$ y $k = 3d$ y se estudiará su desempeño.

Capítulo 5. Experimentación

Con el objetivo de evaluar el desempeño de los algoritmos de búsqueda de vecinos más cercanos utilizados en la planeación de movimiento basada en muestras, se presentan distintos experimentos que dan validez al método propuesto. Cada uno de los experimentos consta de un número n de muestras aleatorias, un número k vecinos más cercanos que estarán enlazados a cada muestra, así como un número d que corresponde a la dimensión del espacio de configuraciones en el que se construye cada PRM perezoso. Así mismo, se hará mención cuando se utilicen alguna de las variantes de los algoritmos mencionados en el Capítulo 2.

Para cada experimento se indica el incremento de la rapidez de construcción (3), la cual corresponde a la razón entre el tiempo de ejecución de la búsqueda de vecinos más cercanos utilizando el método secuencial y algún otro algoritmo según se indique, por ejemplo, el árbol k-dimensional o el APG, en un mismo problema, es decir:

$$rapidez = \frac{T_{secuencial}}{T_{APG}} \tag{3}$$

Cabe resaltar que si ambos tiempos son similares, el valor de la rapidez será cercano a uno. Idealmente, se espera que para el método propuesto este valor sea superior a uno tanto como sea posible.

La segunda métrica de comparación utilizada se denomina precisión (4), la cual corresponde a una medida de calidad de la solución obtenida con los algoritmos de búsqueda utilizados en este trabajo. Sea A_i el conjunto de vecinos más cercanos encontrados con algún método de búsqueda propuesto para el elemento i y sea B_i el conjunto de vecinos obtenidos al utilizar una búsqueda secuencial. Se denota como knn_i^c al número de elementos en $A_i \cap B_i$ y como knn_i^s al número de elementos en el conjunto B_i. Formalmente:

$$precision = \frac{1}{n} \sum_{i=1,\ldots,n} \frac{knn_i^c}{knn_i^s} \tag{4}$$

Para esta métrica se espera que el resultado obtenido sea lo más cercano a uno.

5.1 Experimento 1. Árbol k-dimensional en altas dimensiones

Para el primer experimento, se eligió el árbol k-dimensional para la búsqueda de vecinos más cercanos en la construcción de un PRM. Este experimento se realiza con la finalidad de tener un mejor entendimiento del desempeño que presentan los algoritmos de planeación de movimiento en altas dimensiones. La cantidad de muestras aleatorias que se utilizan va de 20,000 a 100,000, se consideraron

los 6 vecinos más cercanos y la dimensión del espacio de configuraciones varía entre 2 y 16. La elección de estos parámetros se deriva del trabajo realizado por (Plaku y Kavraki, 2007) en el que se indica que el desempeño de la búsqueda de vecinos más cercanos se degrada a una búsqueda secuencial a partir de una dimensión crítica la cual se encuentra entre 10 y 15, dependiendo del número de muestras que se utilicen.

En la Figura 11, se muestran los resultados obtenidos en la ejecución del experimento explicado anteriormente. Cada curva representa la cantidad de muestras utilizadas para construir un PRM perezoso. Es posible observar que a partir de una dimensión entre 12 y 16 la rapidez del árbol k-dimensional está por debajo de uno, esto quiere decir que el tiempo de ejecución utilizado por un árbol k-dimensional para la construcción de un PRM perezoso supera al tiempo de ejecución empleando una búsqueda secuencial para la construcción del mismo. Se puede observar que este patrón se mantiene al aumentar la dimensión del espacio o el número de muestras.

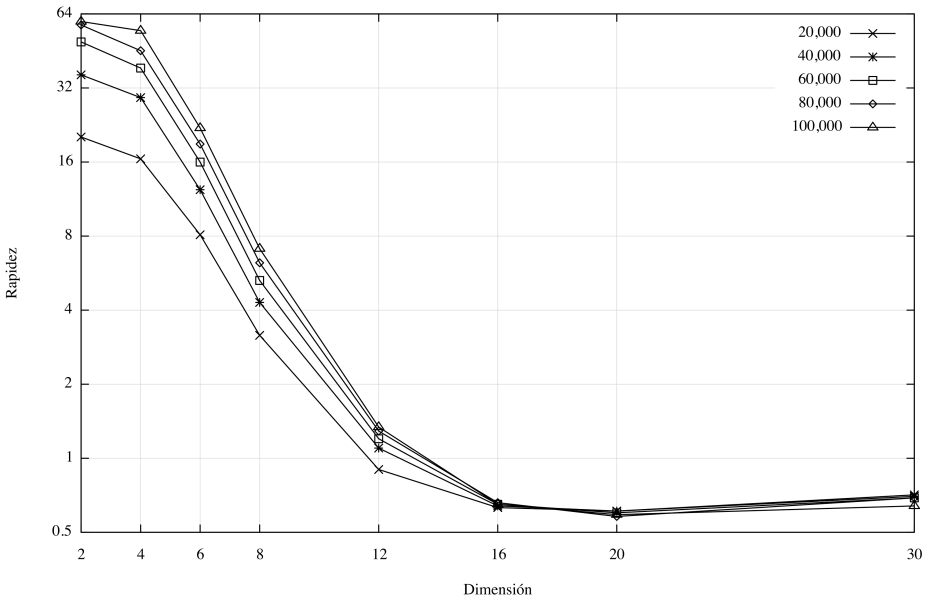


Figura 11: Se muestra la rapidez de utilizar una búsqueda de vecinos con un árbol k-dimensional en comparación con una búsqueda secuencial en distintas dimensiones. Cada curva muestra el número de muestras utilizadas para la construcción de un PRM perezoso.

5.2 Experimento 2. APG Voraz y APG Tabú

Con el objetivo de analizar el desempeño del APG como método de búsqueda en la construcción de un PRM, se utilizaron las estrategias de búsqueda voraz y tabú, explicadas en el Capítulo 4.

Los experimentos realizados en esta sección utilizaron el parámetro de vecindad recomendado por

Malkov et. al. de $k = 3d$ donde d corresponde a la dimensión del espacio.

En el primer experimento se emplearon 20,000 muestras para la construcción de un PRM perezoso y cada muestra está conectada con sus $3d$ vecinos más cercanos. El número de dimensiones utilizadas son 2, 4, 6, 8, 12, y 16. En el caso del APG voraz y tabú el número de reinicios para cada instancia fue de 10. Para este experimento, la selección de vecinos más cercanos hace uso de una búsqueda secuencial, KDTree, APG voraz y APG Tabú.

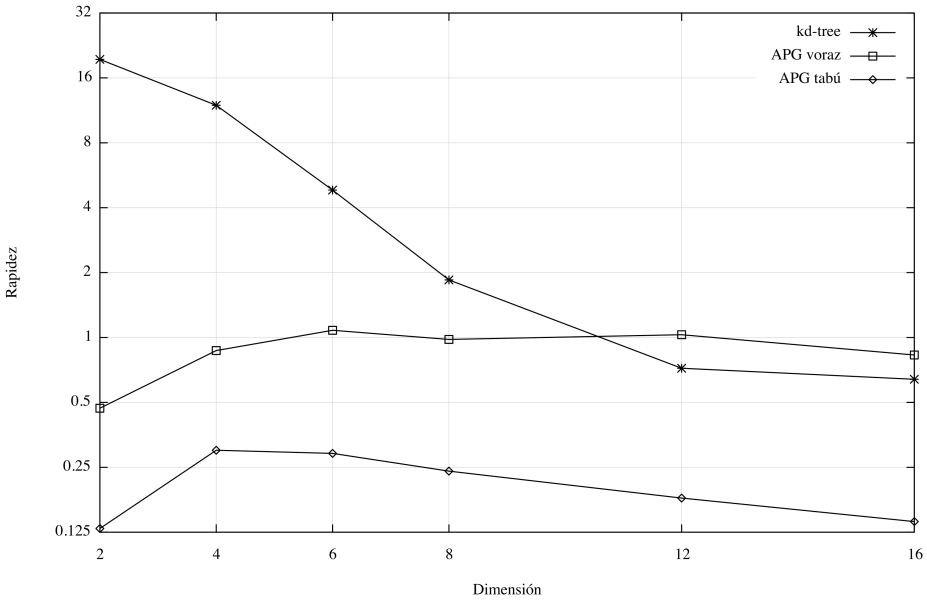


Figura 12: Se muestra la rapidez de la búsqueda en un árbol k-dimensional, APG voraz y tabú en comparación a una búsqueda secuencial en la construcción de un PRM perezoso considerando los 3d vecinos más cercanos a cada muestra.

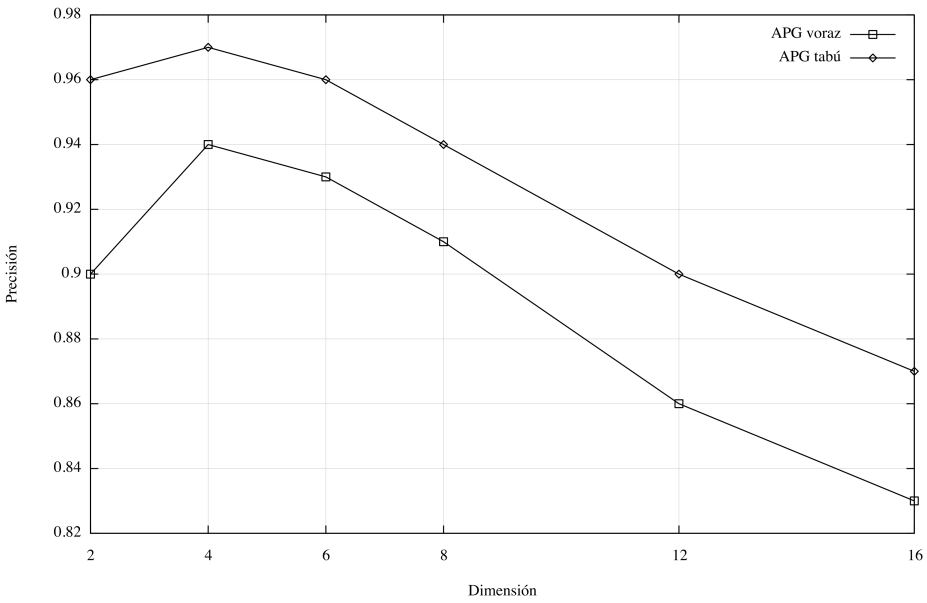


Figura 13: Se muestra la precisión de la búsqueda en un árbol k-dimensional, APG voraz y tabú en comparación a una búsqueda secuencial en la construcción de un PRM perezoso considerando los 3d vecinos más cercanos a cada muestra.

Los resultados que se muestran en las Figuras 12 y 13, nos indican que utilizar una búsqueda de vecinos más cercanos por medio del APG voraz o tabú para la construcción de un PRM perezoso con una vecinadad de $k = 3d$ es al menos tan costoso como realizar una búsqueda secuencial, para los casos explorados. Además, se puede observar que realizar la búsqueda con un APG tabú implica un mayor tiempo de construcción, sin embargo, este tiene una precisión mayor en comparación a utilizar el método voraz.

Aunque los resultados anteriores parecen pocos favorables, existen algunas consideraciones que se deben tener en cuenta:

- Los autores del APG Malkov *et al.*, recomiendan un número de vecinos $k = 3d$ para la construcción del índice y después realizar las consultas, donde generalmente, se busca una cantidad menor de vecinos para cada una de ellas. En nuestro caso, nos interesa que la construcción del PRM se realice lo más rápido posible manteniendo una precisión aceptable, para lo cual se explorará el desempeño de estos métodos al disminuir el número de vecinos.
- Es posible que en una dimensión mayor este método de búsqueda funcione como se espera con los parámetros utilizados. Sin embargo, lo ideal es que el método propuesto funcione a partir de que el árbol k-dimensional comienza a degradarse a una búsqueda secuencial.

- El número de reinicios en el APG es un factor que impacta en el desempeño del algoritmo. Si se aumentan los reinicios se incrementa la precisión de la búsqueda y a su vez el tiempo de construcción. Con el objetivo de encontrar un balance entre número de reinicios y precisión, en la siguiente subsección se realizaron experimentos donde se emplean distintos reinicios al realizar las búsquedas.

A partir de los resultados anteriores y las consideraciones mencionadas, se decidió realizar una serie de experimentos con un número menor de vecinos más cercanos. En este caso se decidió tomar 6 vecinos para cada muestra, manteniendo la cantidad de muestras en 20,000 y dimensiónes del espacio de 2, 4, 8, 12, 16 y 20. Nuevamente, con 10 reinicios para los métodos de búsqueda APG voraz y tabú.

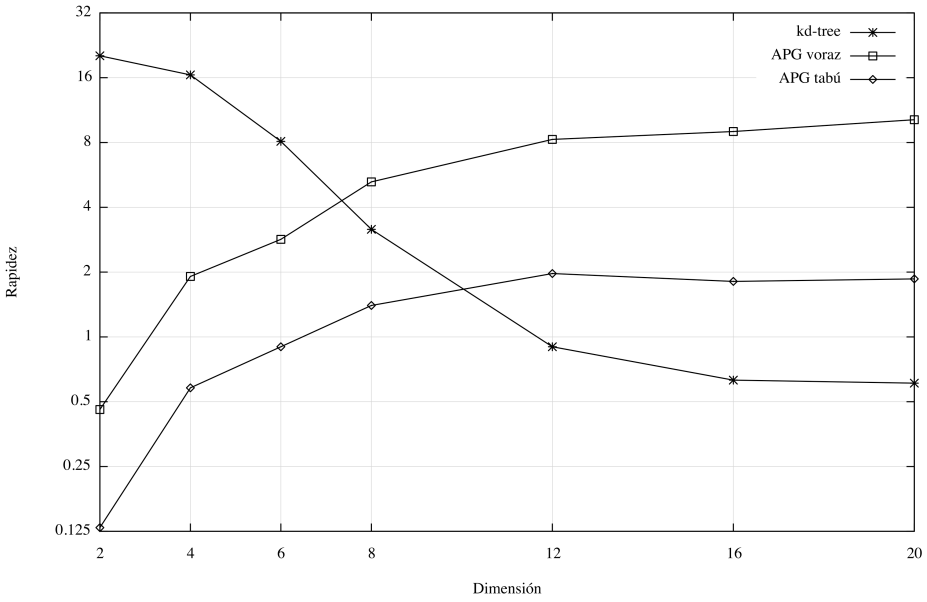


Figura 14: Se muestra la rapidez de la búsqueda en un árbol k-dimensional, APG voraz y tabú en comparación a una búsqueda secuencial en la construcción de un PRM perezoso considerando los 6 vecinos más cercanos a cada muestra.

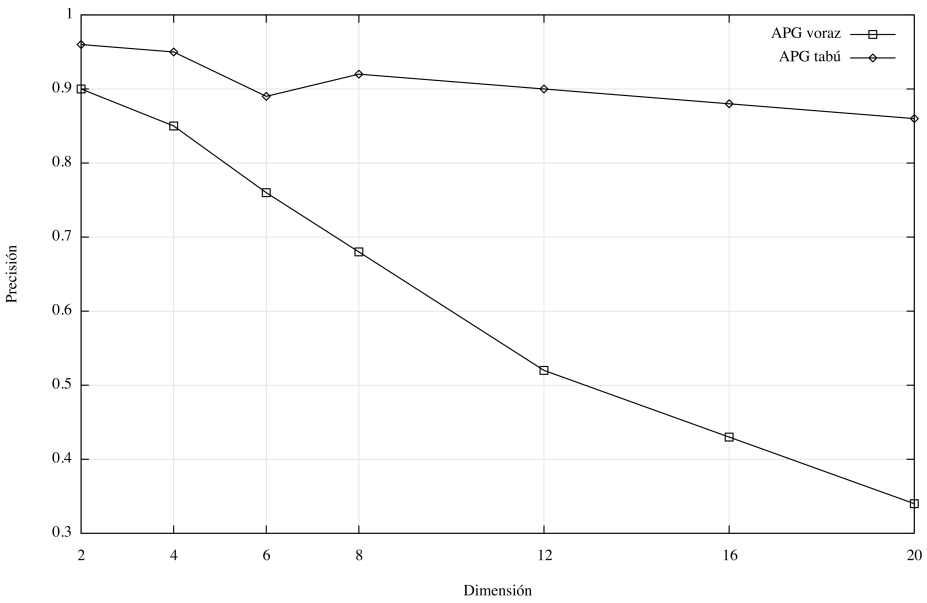


Figura 15: Se muestra la precisión de la búsqueda en un árbol k-dimensional, APG voraz y tabú en comparación a una búsqueda secuencial en la construcción de un PRM perezoso considerando los 6 vecinos más cercanos a cada muestra.

En los resultados que se muestran en la Figura 14 podemos observar que utilizar un APG ya sea voraz o tabú mejora el tiempo de construcción de un PRM en problemas en espacios de dimensión 8 y 12. Aunque, el APG voraz presenta un mejor desempeño en el tiempo de construcción, la precisión de este se degrada en mayor escala a medida que se aumenta la dimensión del espacio, ver Figura 15. Por tal motivo, se dejará de lado este método. Con la intención de mejorar el desempeño presentado hasta ahora por el APG tabú, en la siguiente sección nos enfocaremos en analizar la rapidez y precisión del mismo ante diferente número de reinicios, vecinos, nodos y dimensión.

5.3 Experimento 3. Heurísticas de mejora para el APG

En el experimento anterior, se analizó la rapidez y precisión de la búsqueda de vecinos más cercanos en la construcción de un PRM perezoso. Se observó que el número de vecinos que se busca por cada muestra afecta el desempeño del algoritmo. En esta ocasión, se utilizará la vecindad recomendada por Karaman y Frazzoli, $k = 2e\log(n)$, donde n corresponde al número de muestras que se utilizan para construir el grafo. Así mismo, se busca conocer el número mínimo de reinicios necesarios para que el APG Tabú mejore el tiempo de construcción, y a su vez, se obtenga la mayor precisión posible. En las siguientes figuras se muestran los resultados obtenidos al realizar la construcción de un PRM* perezoso de 10,000 a 50,000 muestras, utilizando de 1 a 5 reinicios para la búsqueda de APG tabú. Los experimentos se

realizaron en espacios de dimensión 12, 16 y 20, ya que son las dimensiones a partir de las cuales se presenta un mejor tiempo de construcción, hasta el momento, en comparación a un búsqueda secuencial.

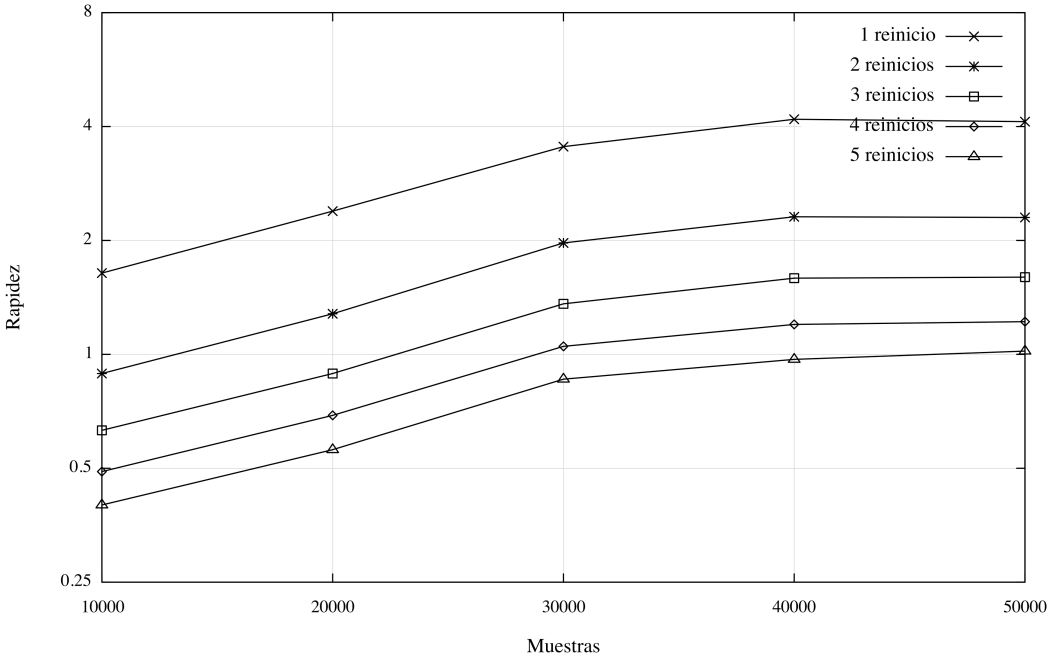


Figura 16: Se muestra la rapidez de la búsqueda APG tabú con distinto número de reinicios en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en dimensión 12.

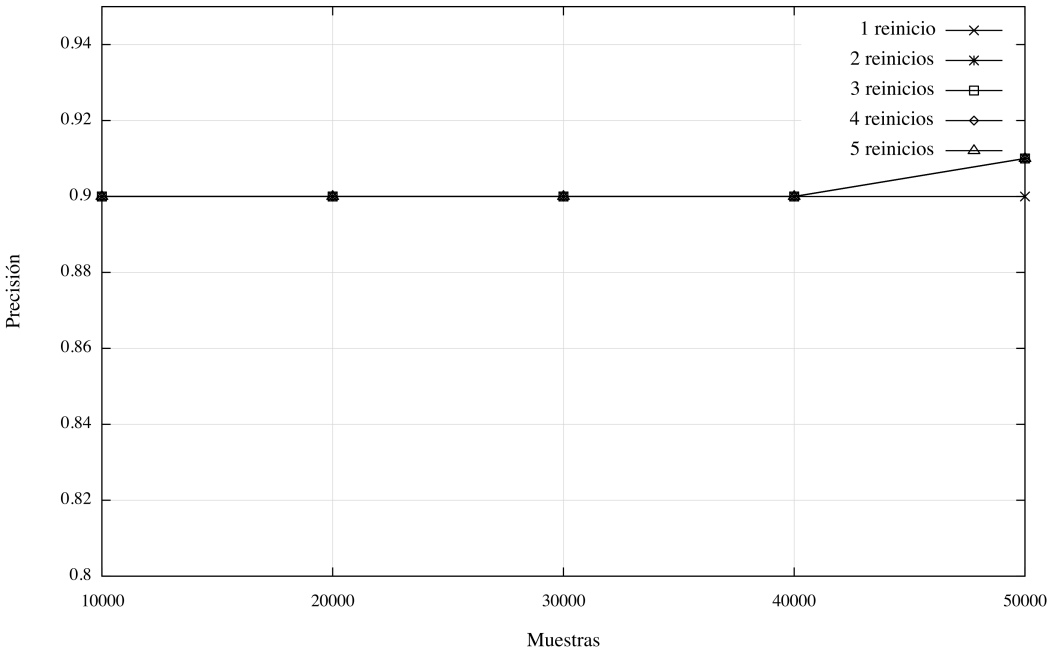


Figura 17: Se muestra la precisión de la búsqueda APG tabú con distinto número de reinicios en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en dimensión 12.

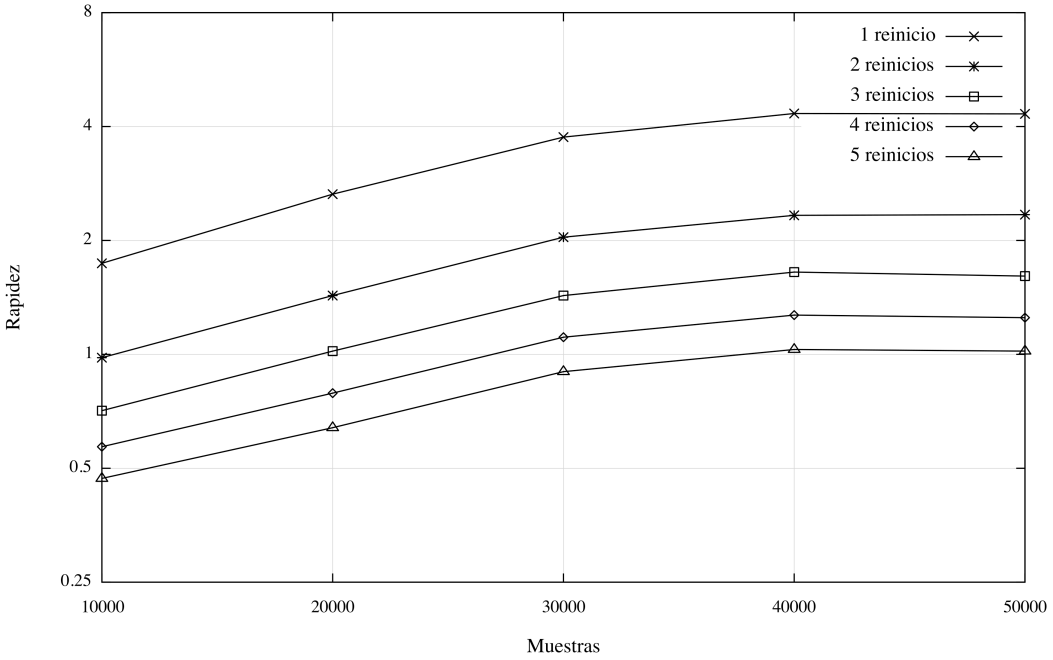


Figura 18: Se muestra la rapidez de la búsqueda APG tabú con distinto número de reinicios en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en dimensión 16.

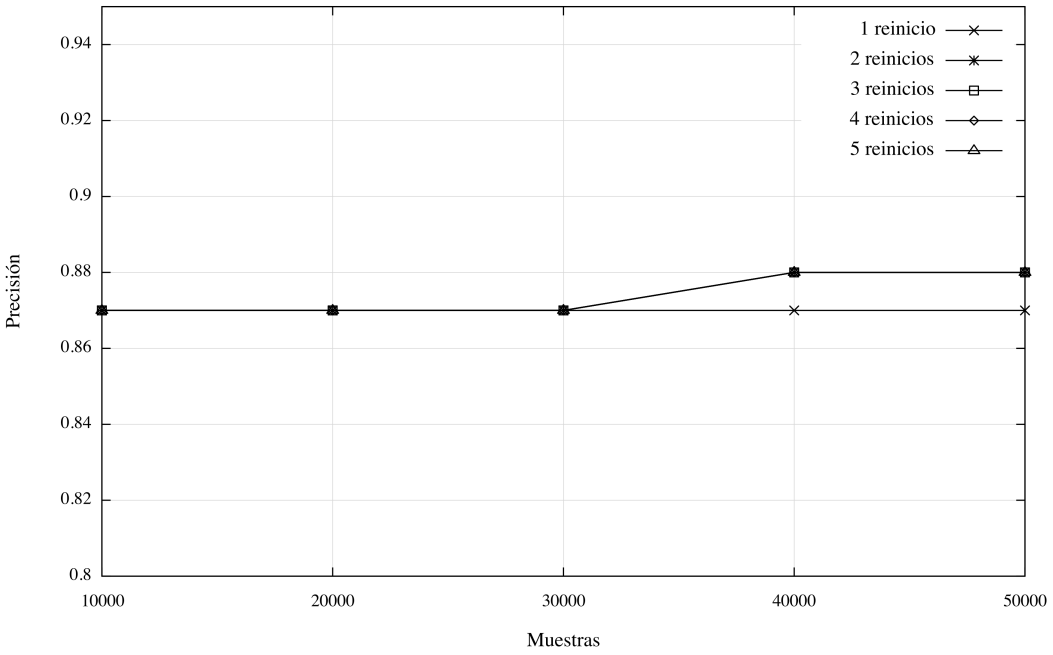


Figura 19: Se muestra la precisión de la búsqueda APG tabú con distinto número de reinicios en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en dimensión 16.

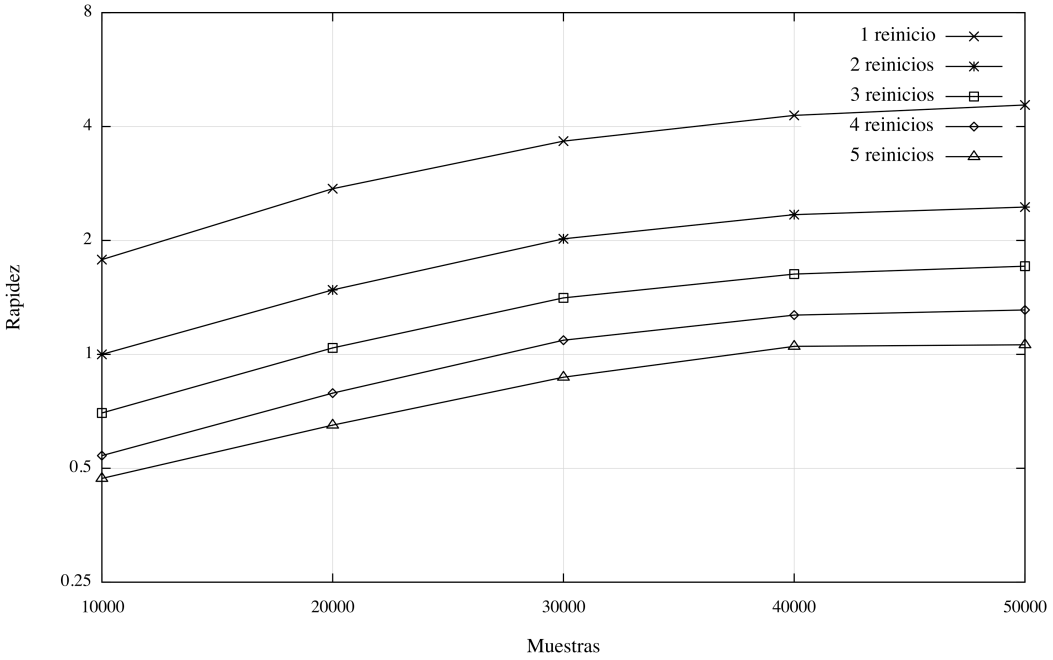


Figura 20: Se muestra la rapidez de la búsqueda APG tabú con distinto número de reinicios en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en dimensión 20.

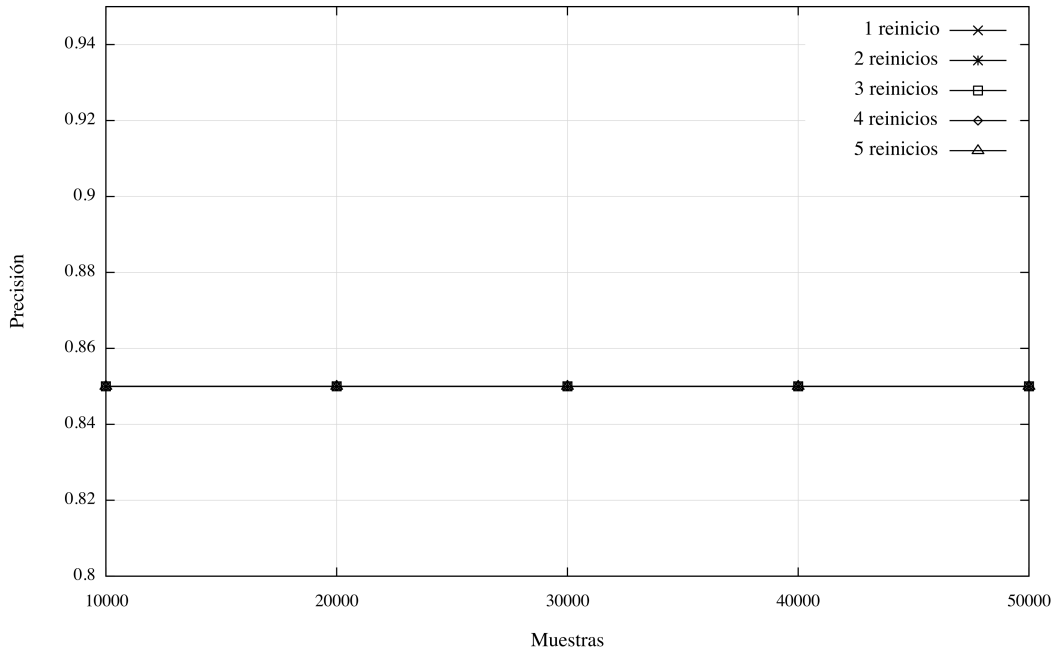


Figura 21: Se muestra la precisión de la búsqueda APG tabú con distinto número de reinicios en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en dimensión 20.

En las Figuras 16, 18 y 20 se pueden observar dos patrones interesantes, el primero muestra que la rapidez aumenta a medida que se incrementa la cantidad de muestras que se utilizan para construir el PRM* perezoso. Esto es algo positivo ya que como se comentó en los capítulos anteriores, es necesaria una gran cantidad de muestras para obtener una mejor representación del espacio y además nos acercamos a las trayectorias óptimas debido a la propiedad del algoritmo de ser asintóticamente óptimo. El otro patrón que se puede observar es que basta con un reinicio al momento de realizar la búsqueda de vecinos más cercanos para ser al menos dos veces más rápido que una búsqueda secuencial.

Por su parte, en las Figuras 17, 19 y 21 se observa que aunque se aumente el número de reinicios a utilizar en la búsqueda APG Tabú, la precisión que se obtiene es la misma que si se emplea un sólo reinicio en la mayoría de los casos. Esto se debe a que el algoritmo de búsqueda se ejecuta hasta que ya no es posible minimizar la lista de vecinos encontrados, por lo que la probabilidad de encontrar vecinos más cercanos que aún no han sido encontrados al elegir un nuevo reinicio aleatorio se reduce ya que una parte de muestras cercanas a la consulta ya han sido visitadas. Por tal motivo, en los siguientes experimentos cada vez que se realice la búsqueda con el APG Tabú esta se realizará con un sólo reinicio.

Después de probar distintas heurísticas de construcción del PRM* perezoso utilizando la búsqueda

del APG, se encontró una estrategia que mejora la precisión, la cual consiste en reiterar la construcción del grafo. Cuando se construye el PRM* perezoso por primera vez se toma cada una de las muestras generadas al azar y se conecta a sus vecinos más cercanos utilizando la búsqueda Tabú del APG. Si se itera nuevamente, dado que todas las muestras ya están conectadas entre si, es probable que si se repite la búsqueda para cada una de ellas se encuentren vecinos más cercanos distintos a los que se habían encontrado previamente.

Se realizó un experimento en el cual una vez creado un PRM* perezoso se reitera su construcción con las mismas muestras para comprobar si la suposición anterior es cierta. En este experimento, se calculó la rapidez y precisión al utilizar 10,000 muestras aleatorias para la construcción de un PRM* perezoso con $k = 2e\log(n)$ vecinos en espacios de dimensión dimensión 2, 4, 6, 8, 12, 16, 20, 25, 30, 40 y 50 reiterando la construcción de una a tres veces, a partir de las siguientes gráficas APG #x corresponde a la búsqueda tabú con # reiteraciones de la construcción. Los resultados obtenidos fueron los siguientes:

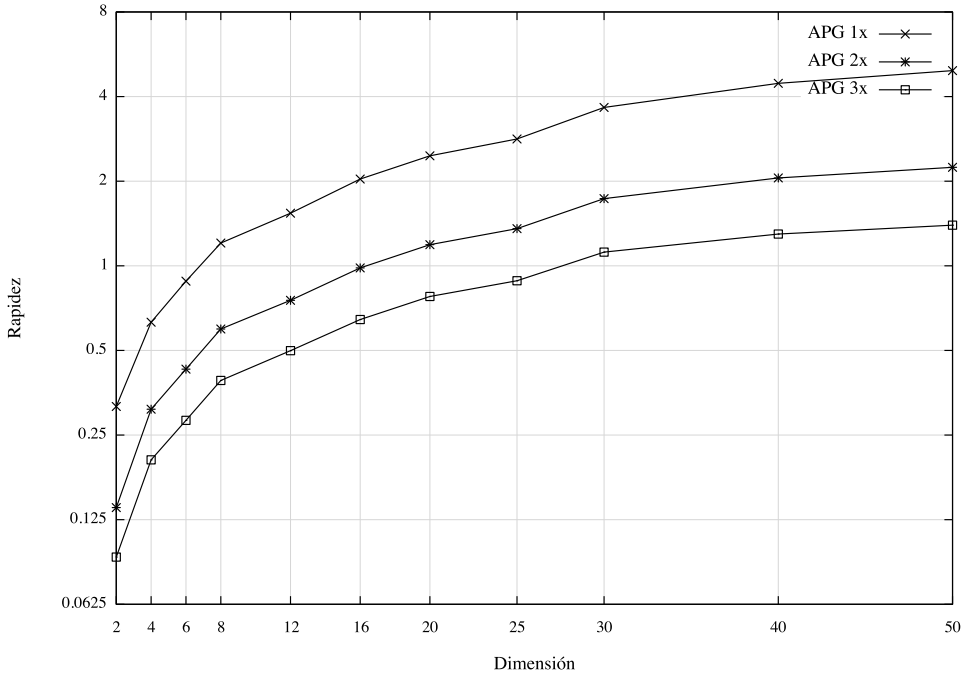


Figura 22: Se muestra la rapidez de la búsqueda APG tabú con una, dos y tres iteraciones de construcción, mostradas como APG 1x, APG 2x y APG 3x respectivamente, en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en distintas dimensiones.

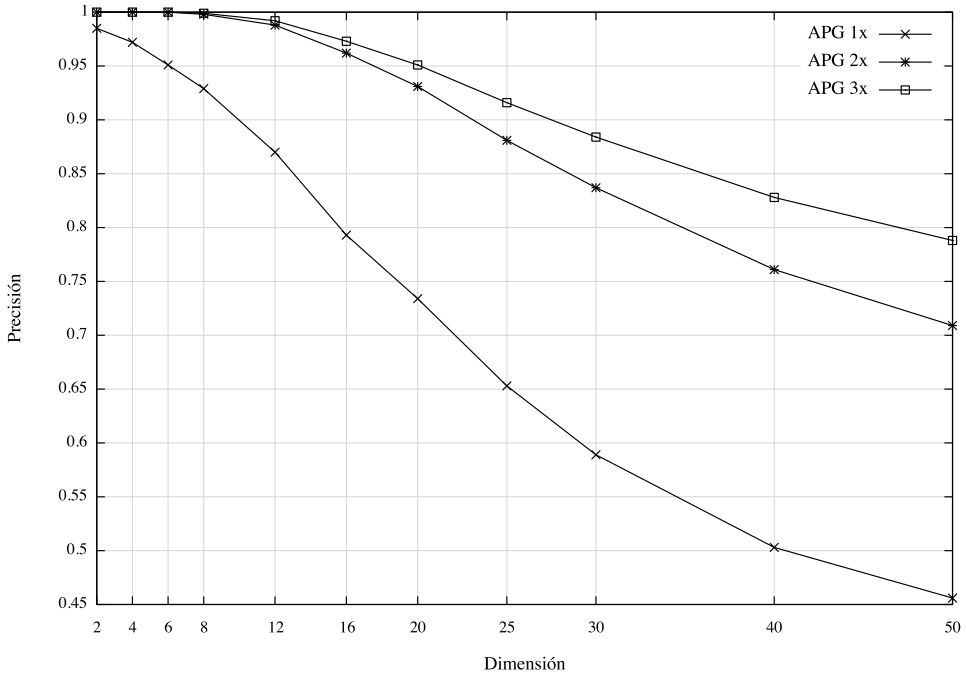


Figura 23: Se muestra la precisión de la búsqueda APG tabú con una, dos y tres iteraciones de construcción, mostradas como APG 1x, APG 2x y APG 3x respectivamente, en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso en distintas dimensiones.

De acuerdo a los resultados que se muestran en las Figuras 22 y 23, podemos observar que efectivamente existe una relación entre reiterar la construcción de PRM* perezoso con la búsqueda del APG y el aumento en la precisión de la misma. Aunque realizar la construcción una sola vez es sin duda más rápido, también es la que cuenta con una menor precisión en la búsqueda, por el contrario, al realizar la construcción tres veces, la precisión se mantiene por encima del 80 %, pero el tiempo que requiere es ligeramente menor a una búsqueda secuencial para dimensiones mayores a 30.

Si bien, realizar la búsqueda de vecinos más cercanos dos veces con el APG es mejor que una búsqueda secuencial a partir de dimensión 16, recordemos que a medida que el número de muestras aumenta mejorará el tiempo de búsqueda y construcción con respecto a la búsqueda secuencial. Es por eso que se decidió adoptar esta heurística de reiterar la construcción completamente una vez más para mejorar la precisión obtenida. En la siguiente sección, se estudia la rapidez y precisión de construir PRMs* contemplando distintos números de obstáculos y PRM* perezoso utilizando las heurísticas para el número de vecinos, reinicios e iteraciones de construcción obtenidos en esta sección.

5.4 Experimento 4. Construcción de PRM con obstáculos

El siguiente experimento se diseñó para probar el desempeño de la construcción de PRMs* utilizando la búsqueda de vecinos más cercanos con el APG Tabú, es decir, para estos casos se comprobará si cada muestra y arista se encuentran libre de colisiones, a su vez, con las mismas muestras se construirán PRM* perezosos con la finalidad de comprobar las ventajas que nos ofrece este método de construcción. Para este experimento se utilizaron 10,000, 50,000 y 100,000 muestras con una vecindad de $k = 2e\log(n)$ en dimensiones 4, 8, 12, 16, 20, 24, 28, 32, 40 y 50.

Los siguientes resultados corresponden a la construcción de PRM* perezoso con 10,000, 50,000 y 100,000 muestras:

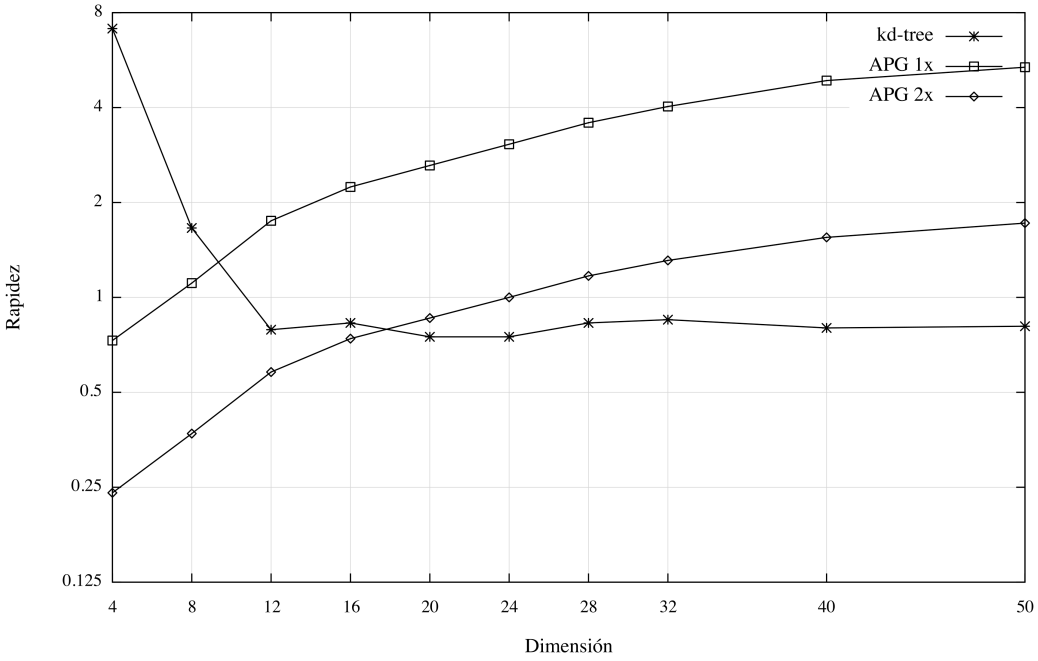


Figura 24: Se muestra la rapidez de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso de 10,000 muestras.

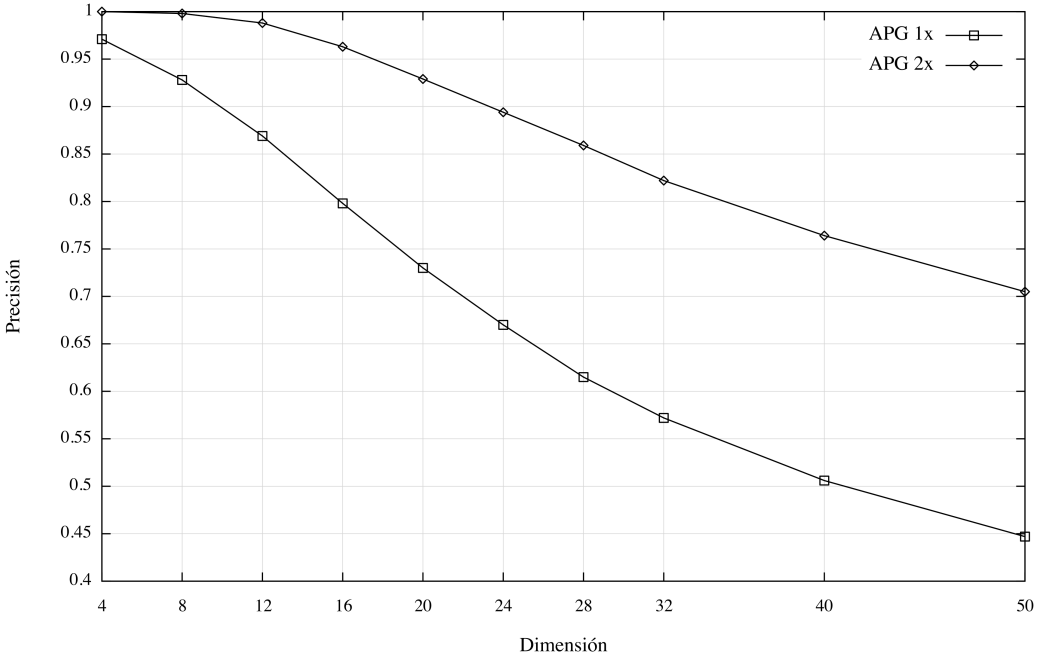


Figura 25: Se muestra la precisión de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso de 10,000 muestras.

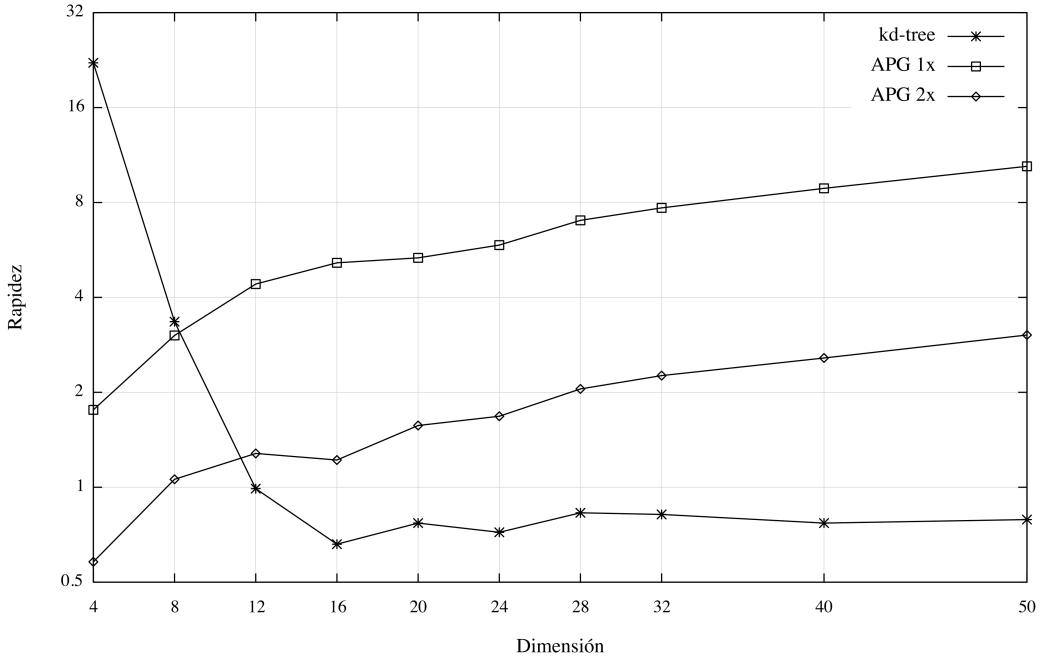


Figura 26: Se muestra la rapidez de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso de 50,000 muestras.

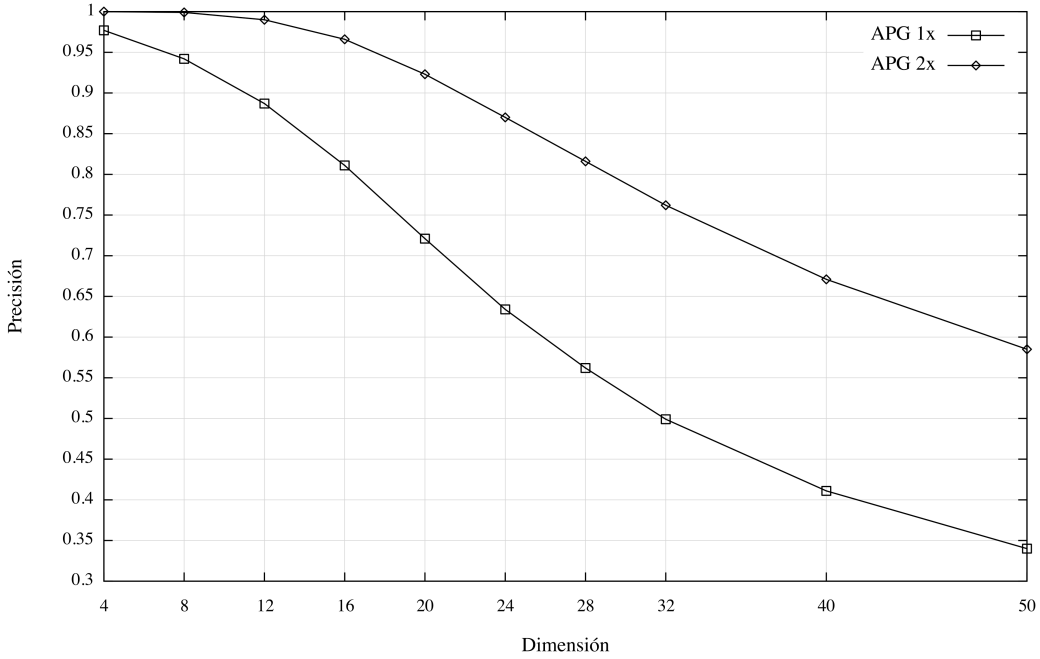


Figura 27: Se muestra la precisión de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso de 50,000 muestras.

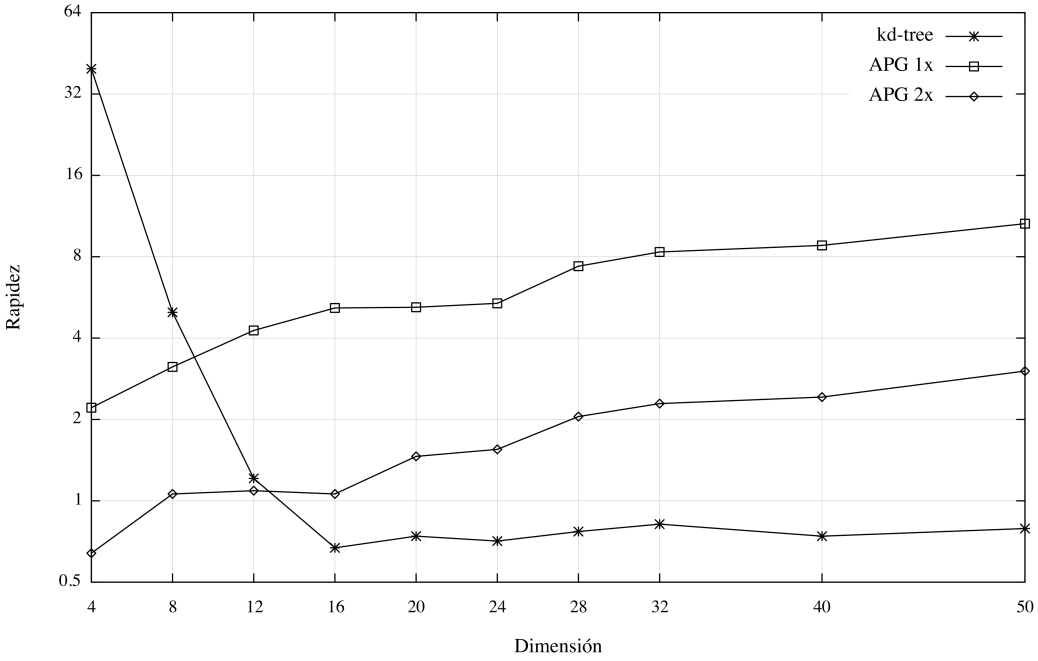


Figura 28: Se muestra la rapidez de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso de 100,000 muestras.

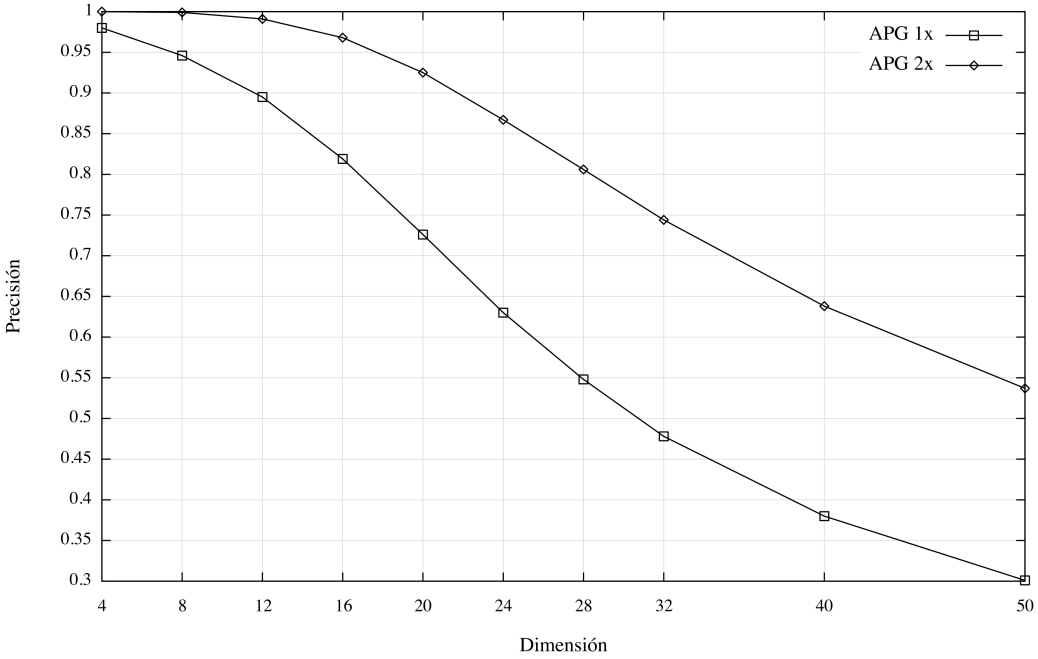


Figura 29: Se muestra la precisión de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* perezoso de 100,000 muestras.

Se puede ver en las Figuras 24, 26 y 28 que a partir de una dimensión 8, es más rápido utilizar el APG que hacer una búsqueda secuencial. Para dimensión 12 o mayor el APG supera al árbol k-dimensional en todas las instancias que se probaron. En las figuras anteriores es posible observar que a medida que se aumenta el número de muestras este método obtiene un mejor desempeño. Por ejemplo, en la Figura 24 se tiene que la rapidez del APG para 10,000 muestras en dimensión 12 es cercano a dos, es decir, casi dos veces más rápido que una búsqueda secuencial, pero cuando se utilizan 50,000 y 100,000 muestras la rapidez del APG es superior a 4 en ambos casos.

En las figuras se observa que reiterar la construcción del APG toma aproximadamente el doble de tiempo que la construcción estándar. Sin embargo, la importancia de esta reiteración puede verse en las Figuras 25, 27 y 29, donde la precisión que se obtiene mejora al menos un 10 % a partir de dimensión 12 y llega hasta un 25 % en dimensión 50, donde nuestra heurística es casi dos veces más rápida que realizar una búsqueda secuencial.

Se realizó una serie de experimentos en los cuales se construyó un PRM* en el cual se valida que cada una de las muestras y aristas se encuentren libre de colisión. En estos experimentos se utilizaron 20,000 muestras aleatorias con 5, 10 y 15 obstáculos aleatorios en cuanto a posición y diámetro en el espacio. De igual manera en estos experimentos se utilizó la estrategia de reiterar la construcción del grafo, a su vez en dicha reiteración se valida que muestras y aristas en el grafo se encuentren libre de colisiones. A continuación se muestran los resultados obtenidos:

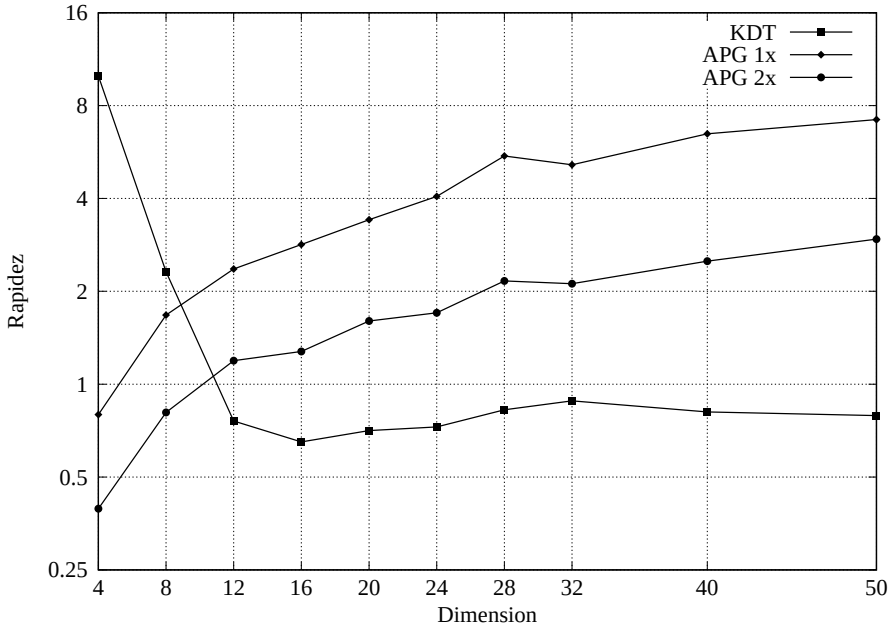


Figura 30: Se muestra la rapidez de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* con 20,000 muestras y 5 obstáculos aleatorios.

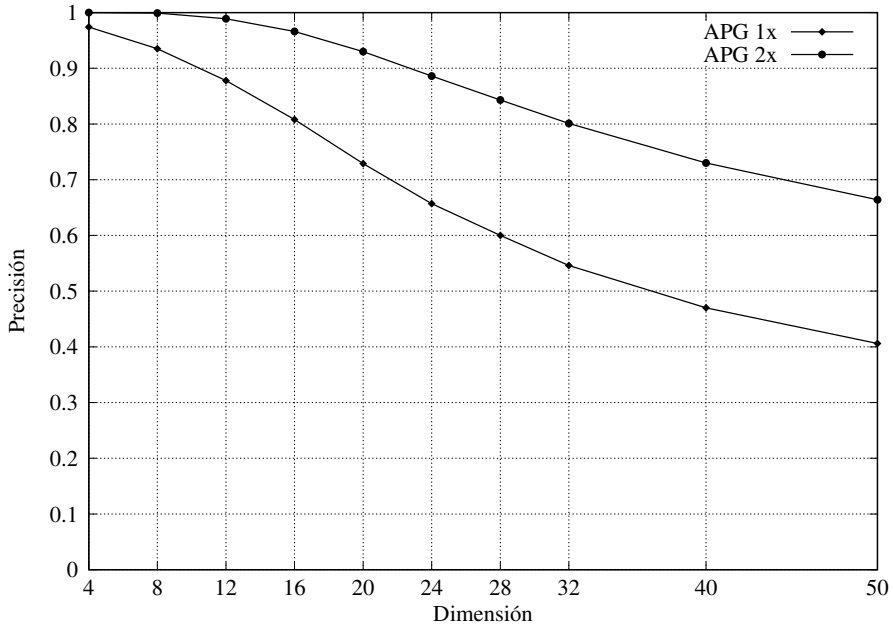


Figura 31: Se muestra la precisión de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* con 20,000 muestras y 5 obstáculos aleatorios.

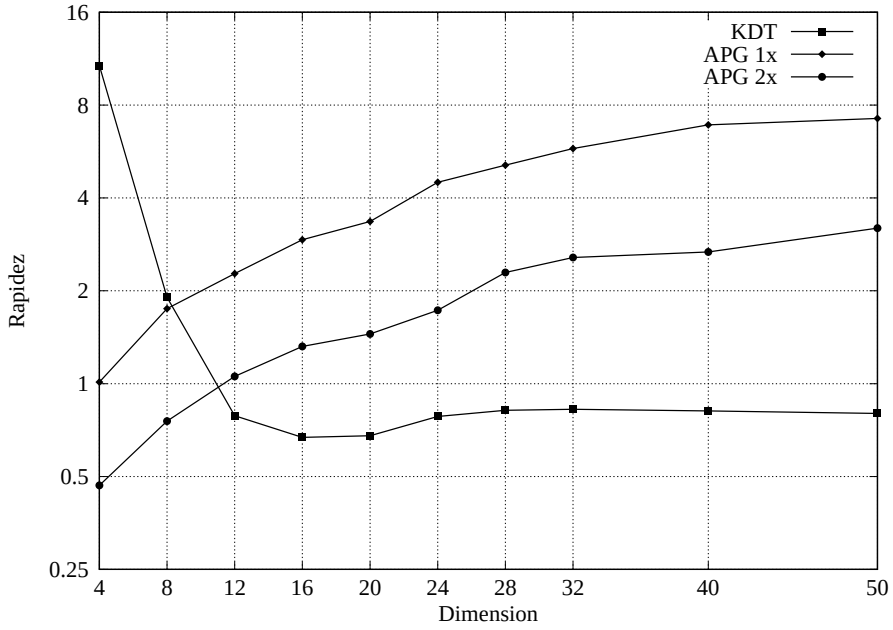


Figura 32: Se muestra la rapidez de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* con 20,000 muestras y 15 obstáculos aleatorios.

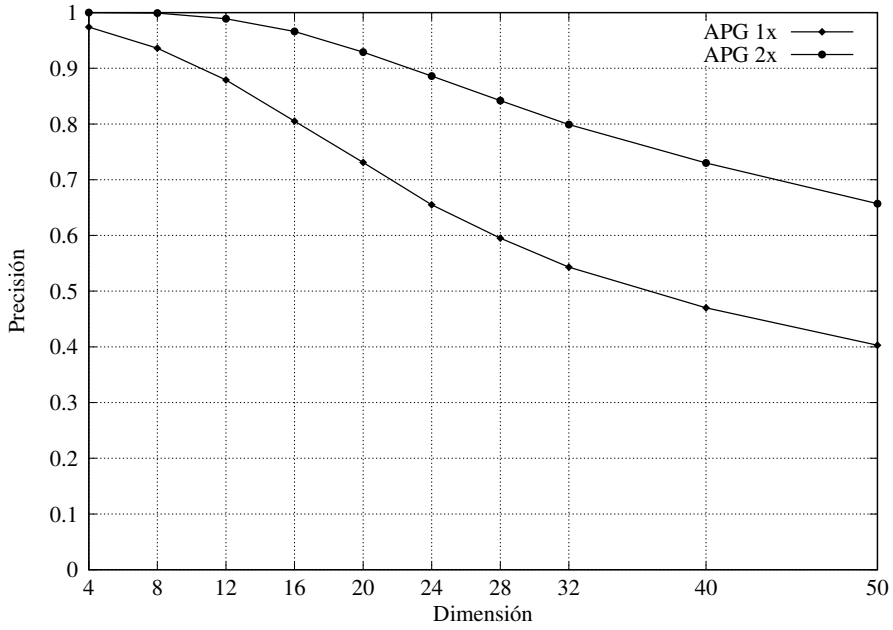


Figura 33: Se muestra la precisión de la búsqueda APG tabú en comparación a una búsqueda secuencial en la construcción de un PRM* con 20,000 muestras y 15 obstáculos aleatorios.

Si bien los resultados que se obtuvieron en las Figuras 30-33 son ligeramente distintos a los resultados mostrados en las figuras anteriores, esto se debe a que para cada experimento la distribución y tamaño de los obstáculos fue generada aleatoriamente. Así mismo, en estas figuras es posible apreciar que utilizar la búsqueda del APG permite construir en un menor tiempo un mapa de caminos PRM* para espacios en altas dimensiones, en comparación a una búsqueda secuencial o un árbol k-dimensional.

Después de observar los resultados que se obtuvieron en los experimentos realizados es posible identificar una dimensión crítica, a partir de la cual el desempeño del árbol k-dimensional con respecto a su tiempo de ejecución es mayor a realizar una búsqueda secuencial. En el caso del método propuesto el APG, se observó que al utilizar la vecindad recomendada por Malkov *et al.* de $k = 3d$, el tiempo de construcción que requiere es superior a una búsqueda secuencial y al utilizar la vecindad recomendada para construir un PRM* de $k = 2e\log(n)$, se obtienen un mejor tiempo de construcción que una búsqueda secuencial a partir de una dimensión 8 y que un árbol k-dimensional a partir de una dimensión entre 10 y 12. Disminuir el tiempo de construcción del APG con una vecindad menor a la recomendada, redujo la precisión de la búsqueda. La solución propuesta para este incoveniente consiste en reiterar la construcción del grafo una vez más, aunque esto aumenta el tiempo de construcción, se tiene un tiempo de construcción menor al de una búsqueda secuencial y un árbol k-dimensional a partir de una dimensión entre 12 y 16 dependiendo de la cantidad de nodos que se utilizan.

Capítulo 6. Conclusiones y trabajo futuro

6.1 Resumen

En esta tesis, se estudia el problema de calcular rápidamente los vecinos más cercanos en un mapa de caminos para espacios en altas dimensiones. Para lograrlo, se propone un enfoque que abandona la estrategia de utilizar una estructura de datos adicional, como los árboles k-dimensionales y se opta por utilizar el mismo mapa de caminos para realizar la búsqueda de los k-vecinos más cercanos. Para mejorar el tiempo de búsqueda, se hace uso de un grafo de proximidad aproximada (APG) (Malkov *et al.*, 2014), el cual ha sido diseñado para realizar búsqueda de vecinos más cercanos en espacios métricos de altas dimensiones.

Se diseñaron una serie de experimentos con el objetivo de evaluar el desempeño de los algoritmos de búsqueda de vecinos más cercanos utilizados en la construcción de PRMs. Se encontró que existe una dimensión crítica para el árbol k-dimensional, es decir, a partir de una dimensión de tamaño entre 10 y 12, el tiempo de cálculo es mayor al de una búsqueda secuencial. Además, se evaluó el desempeño de las búsquedas APG Voraz y APG Tabú, descritas en el Capítulo 4. Así mismo, se exploraron distintos parámetros de vecindad, reinicios y muestras con la intención de encontrar una heurística que nos permita alcanzar los objetivos planteados. Finalmente, se realizó la construcción de PRMs* y Lazy-PRMs* con obstáculos con la intención de validar que es posible realizar la construcción de un mapa de caminos en altas dimensiones en menor tiempo que al utilizar una búsqueda secuencial.

6.2 Conclusiones

Con base en el trabajo de tesis realizado, se concluye lo siguiente:

1. La búsqueda de vecinos más cercanos a través de un árbol k-dimensional requiere de un tiempo de ejecución similar o superior al de una búsqueda secuencial a partir de dimensión 12 en adelante. Este fenómeno de degradación, ocurre de manera más pronunciada a medida que se aumentan la cantidad de muestras y/o vecinos en los algoritmos para la construcción de un mapa de caminos en planeación de movimiento.

2. La búsqueda de vecinos basada en el método APG Voraz es mejor a una búsqueda secuencial a partir de una dimensión 4 y es superior a la búsqueda utilizando un árbol k-dimensional a

partir de dimensión 8, a diferencia del método APG Tabú, el cual es superior a los métodos antes mencionados a partir de dimensión 8 y 12, respectivamente. Sin embargo, la precisión que presenta la búsqueda APG Tabú es superior a la búsqueda APG Voraz y se encuentra dentro de rangos aceptables.

3. Se encontró que en la búsqueda APG Tabú sólamente es necesario utilizar un reinicio en la búsqueda para obtener un buen desempeño. Por lo que para este caso se desmiente experimentalmente que la cantidad de reinicios se refleja directamente en el desempeño del algoritmo al igual que la búsqueda APG Voraz, ya que en el caso del APG Tabú sólo aumenta el tiempo de ejecución pero no la precisión de la búsqueda.

4. El algoritmo APG presenta una mejora en el tiempo de ejecución a medida que se aumenta la dimensión del espacio y el número de muestras en el mismo, sin embargo, la precisión de la búsqueda disminuye. Para tratar este problema, se propuso una estrategia que mejora la precisión de la búsqueda local del APG, la cual consiste en reiterar a cada elemento en la construcción del mapa de caminos. A pesar de que esta estrategia disminuye la rapidez en la construcción del mapa de caminos, se sigue realizando en un tiempo menor a una búsqueda secuencial.

5. El algoritmo APG mostró ser superior experimentalmente a una búsqueda secuencial y a un árbol k-dimensional para realizar la búsqueda de vecinos más cercanos en la construcción de un mapa de caminos PRM y sus variantes en altas dimensiones, incluso en espacios de configuraciones que cuentan con obstáculos generados aleatoriamente.

6.3 Trabajo futuro

Los posibles temas a estudiar a partir de este trabajo son:

1. En los experimentos que se realizaron en esta tesis se utilizó una vecindad de $2e\log(n)$. Sin embargo, este parámetro corresponde al límite superior de la vecindad recomendada para construir un PRM*, la cual es $e(1+1/d)\log(n)$, y los resultados que se obtivieron sugieren que una vecindad de menor tamaño aumentaría la rapidez del tiempo de construcción del mapa de caminos. Al utilizar una vecindad de menor tamaño, se sabe que el tiempo de construcción se reduce, al igual que en el Experimento 4 del capítulo anterior. Sin embargo, habría que estudiar el impacto que esta vecindad produce en la precisión de la búsqueda y como se refleja en la trayectoría entre un par de puntos.

2. En el presente trabajo no se analiza el algoritmo árbol de exploración con expansión rápida (RRT). El cual en su variante asintóticamente óptima (RRT*) requiere de una búsqueda de $2e\log(n)$ vecinos más cercanos. Lo anterior sugiere que el APG podría servir como estructura de búsqueda y un subgrafo del mismo contendría al RRT*.

3. Recientemente, (Malkov y Yashunin, 2016) han publicado un trabajo al que llaman grafos mundo pequeño navegables (NSW) con jerarquía controlada,en el cual se construye incrementalmente una estructura en capas que consta de un conjunto jerárquico de grafos de proximidad para subconjuntos anidados de los elementos almacenados. El ejemplo de una búsqueda que se muestra en la Figura 33 se inicia desde un elemento de la capa superior (muestra en color rojo). Las flechas rojas indican la dirección del algoritmo voraz hacia el punto de consulta (muestra en color verde). Este método de búsqueda podría ser implementado para realizar la construcción del PRM en el que cada capa represente una prioridad para las muestras o asignar jerarquía a las dimensiónes del espacio, con el objetivo de mejorar los resultados obtenidos en este trabajo.

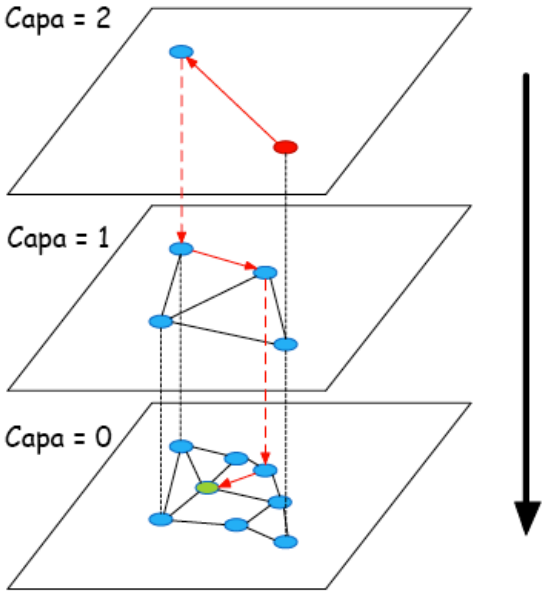


Figura 34: Se muestra la idea jerárquica en los grafos NSW.

Literatura citada

Akgun, B. y Stilman, M. (2011). Sampling heuristics for optimal motion planning in high dimensions. En: *IROS*.

Atramentov, A. y LaValle, S. M. (2002). Efficient nearest neighbor searching for motion planning. En: *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. IEEE, Vol. 1, pp. 632–637.

Bohlin, R. y Kavraki, L. E. (2000). Path planning using lazy prm. En: *ICRA*.

Canny, J. F., Donald, B. R., Reif, J. H., y Xavier, P. G. (1988). On the complexity of kinodynamic planning. En: *FOCS*.

Chávez, E., Navarro, G., Baeza-Yates, R. A., y Marroquín, J. L. (2001). Searching in metric spaces. *ACM Comput. Surv.*, **33**: 273–321.

Choset, H. M. (2005). *Principles of robot motion: theory, algorithms, and implementation*.

Craig, J. J. (1989). Introduction to robotics - mechanics and control (2. ed.). En: *DAGLIB*.

De Berg, M., Van Kreveld, M., Overmars, M., y Schwarzkopf, O. C. (2008). Computational geometry. En: *Computational Geometry, Algorithms and Applications*. Springer.

Elbanhawi, M. y Simic, M. (2013). On the performance of sampling-based optimal motion planners. En: *EMS*.

Hauser, K. (2015). Lazy collision checking in asymptotically-optimal motion planning. En: *ICRA*.

Hinneburg, A., Aggarwal, C. C., y Keim, D. A. (2000). What is the nearest neighbor in high dimensional spaces? En: *VLDB*.

Ichnowski, J. y Alterovitz, R. (2014). Fast nearest neighbor search in se(3) for sampling-based motion planning. En: *WAFR*.

Indyk, P. (2004). Nearest neighbors in high-dimensional spaces. En: *CG*.

Jaillet, L. y Porta, J. M. (2013). Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Trans. Robotics*, **29**: 105–117.

Karaman, S. y Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *I. J. Robotic Res.*, **30**: 846–894.

Kavraki, L. E. y Latombe, J.-C. (1998). Probabilistic roadmaps for robot path planning.

Kleinberg, J. M. (2000). The small-world phenomenon: an algorithmic perspective. En: *STOC*.

Kleinbort, M., Salzman, O., y Halperin, D. (2015). Efficient high-quality motion planning by fast all-pairs r-nearest-neighbors. *CoRR*, **abs/1409.8112**.

Lan, X. y Schwager, M. (2013). Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields. En: *ICRA*.

Latombe, J.-C., Lazanas, A., y Shekhar, S. (1991). Robot motion planning with uncertainty in control and sensing. *Artif. Intell.*, **52**: 1–47.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

LaValle, S. M. (2011). Motion planning. *IEEE Robotics & Automation Magazine*, **18**(1): 79–89.

LaValle, S. M. y Kuffner, J. J. (1999). Randomized kinodynamic planning. *I. J. Robotic Res.*, **20**: 378–400.

Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. *IEEE Trans. Computers*, **32**: 108–120.

Luo, J. y Hauser, K. K. (2014). An empirical study of optimal motion planning. En: *IROS*.

Malkov, Y., Ponomarenko, A., Logvinov, A., y Krylov, V. (2012). Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. En: *SISAP*.

Malkov, Y., Ponomarenko, A., Logvinov, A., y Krylov, V. (2014). Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.*, **45**: 61–68.

Malkov, Y. A. y Yashunin, D. A. (2016). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *ArXiv e-prints*.

Nechushtan, O., Raveh, B., y Halperin, D. (2010). Sampling-diagram automata: A tool for analyzing path quality in tree planners. En: *WAFR*.

Plaku, E. y Kavraki, L. E. (2005). Distributed sampling-based roadmap of trees for large-scale motion planning. En: *ICRA*.

Plaku, E. y Kavraki, L. E. (2006). Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. En: *WAFR*.

Plaku, E. y Kavraki, L. E. (2007). Distributed computation of the knn graph for large high-dimensional point sets. *J. Parallel Distrib. Comput.*, **67**: 346–359.

Reif, J. H. (1979). Complexity of the mover's problem and generalizations (extended abstract). En: *FOCS*.

Schwartz, Jacob T, S. M. (1983). On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics*, **4**(3): 298–351.

Svenstrup, M., Bak, T., y Andersen, H. J. (2011). Minimising computational complexity of the rrt algorithm a practical approach. En: *ICRA*.