Centro de Investigación en Matemáticas, A.C.

Un Planificador SST Eficiente para Trayectorias Mínimas en Tiempo para un DDR con Dinámica de Segundo Orden entre Obstáculos

T E S I S

Que para obtener el grado de
Maestro en Ciencias
con especialidad en
Computación y Matemáticas Industriales

P r e s e n t a:
Richard Fabian Arteaga Ospina

Directores de tesis:
Dr. Rafael Murrieta Cid

Dr. Israel Becerra Durán

Autorización de la versión final

Guanajuato, Gto., 30 de Noviembre de 2020

Agradecimientos

Agradezco a los doctores Rafael Murrieta Cid e Israel Becerra Durán por sus asesorías, enseñanzas, acompañamiento y dedicarme el tiempo necesario para el desarrollo de este trabajo de investigación.

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT), por el apoyo económico que me dieron para la realización de mis estudios de maestría. Al Centro de Investigación en Matemáticas (CIMAT), por haberse convertido en mi hogar dos años.

Resumen

En esta tesis se combinan herramientas fuertes de la teoria de control, como lo es el principio del máximo de Pontryagin (PMP), con métodos más practicos y modernos, como son los algoritmos de planificación de movimiento basados en muestreo, con el objetivo de investigar las trayectorias asintóticamente óptimas en tiempo considerando dinámica de segundo orden, con aceleración acotada, en el sistema noholonómico dado por un robot de manejo diferencial controlado por las aceleraciones en sus ruedas. En particular, se estudia el efecto de utilizar los controles localmente óptimos obtenidos con el PMP, como entradas para el método basado en muestreo SST y la versión asintóticamente óptima del mismo, llamada SST*, en términos de la velocidad de convergencia y del costo de las trayectorias encontradas.

Se demuestra que el SST* aplicado con los controles localmente óptimos se mantiene asintóticamente óptimo y probabilísticamente completo. Además, también se demuestra que la velocidad de convergencia, al costo óptimo es más rápida cuando se usan dichos controles en comparación con el conjunto entero de controles.

Finalmente, se presentan experimentos en simulación, que analizan el efecto del ajuste de algunos parámetros del método y confirman los resultados teóricos.

Índice

CAPÍTULO 1

Introducción

Este trabajo de investigación es parte del área de robótica, específicamente, en la intersección de la teoría de control óptimo y la planificación de movimiento. Se estudian soluciones de problemas que incluyen cinemática y dinámica (kinodinámicos) en sistemas noholonómicos con métodos de planificación basados en muestreo. Intuitivamente, un sistema noholonómico es aquel que tiene restricciones de movimiento, formalmente esta propiedad se puede determinar usando el *Teorema de Frobenius* [18]. El objetivo de un problema kinodinámico es determinar las entradas de control para conducir un sistema robótico desde una configuración inicial a una final, evitando colisiones en un ambiente con obstáculos y respetando restricciones dinámicas físicas, de este tipo de problemas existen numerosas investigaciones [21, 9, 2, 17, 18].

La teoría de control óptimo, como su nombre lo indica, tiene como objetivo principal controlar un sistema dinámico desde una configuración inicial a una final de manera óptima, respecto a la cuantificación de algún costo relacionado al sistema. El origen de esta teoría se remonta a finales del siglo XVII, al igual que el cálculo variacional, cuando el mátematico

Johann Bernoulli publicó el hoy conocido como el problema de la *curva braquistócrona* [12]. Desde los 60's aumentó el estudio y desarrollo de la teoría de control óptimo, impulsado por los *problemas de mínimo tiempo* [33] y su aplicación en otras áreas, como las ciencias económicas [35].

Otro pensamiento más convencional sostiene que la teoría del control óptimo nació con la publicación del *principio del máximo de Pontryagin* [28, 33], el cual proporciona condiciones que deben cumplir las trayectorias de control para la optimización, en caso de que existan trayectoriás óptimas. Por tanto ayuda a identificar o reducir los posibles candidatos a controles óptimos, lo que conlleva a aplicaciones directas en la planificación de movimientos.

Históricamente, la planificación de movimientos se refiere a encontrar un plan para conducir un sistema desde una posición inicial a una final sin chocar con obstáculos, semejante a la teoría de control óptimo pero sin la optimización de la trayectoria, como objetivo principal. Además, utiliza una gran variedad de herramientas y áreas de estudio como geometría computacional, programación dinámica, probabilidad, optimización, por supuesto teoría de control y muchas otras [16]. Se han propuesto planificadores de movimiento completos, es decir, aquellos que resuelven cualquier problema que tenga solución y dicen cuando es irresoluble, sin embargo, en general se requiere un tiempo exponencial en el número de grados de libertad del robot [7, 25]. Por tanto, tales planificadores completos no son útiles para la mayoría de problemas prácticos, que tienen un número elevado de grados de libertad, lo que conllevó a desarrollar algoritmos más aplicables y eficientes, como los planificadores basados en muestreo.

Los planificadores basados en muestreo realizan una exploración en el espacio de estados o de configuraciones del robot con un esquema de muestreo, el cual es más eficiente que hacer una construcción explícita de todo el espacio incluyendo los obstáculos, enfoque usual y hasta el momento necesario en los planificadores completos [20, 7]. Cabe mencionar que los planificadores basados en muestreo no se especializan en la detección de colisiones, para esta tarea utilizan otros algoritmos [11, 6, 10], lo que permite el desarrollo de algoritmos de planificación que son independientes de modelos geométricos particulares [15]. Aunque son

métodos que permiten abordar problemas complicados de planificación en altas dimesiones, en su desarrollo y en la busquedad de eficiencia práctica, pierden la deseable propiedad de completitud. Sin embargo, presentan propiedades que dan garantías para su aplicación, que son la *completitud probabilística* y la *optimalidad asintótica* [13, 22], el cumplimiento de estas garantías requería la solución de un problema de frontera de dos puntos (BVP), el cual no es fácil de hacer para muchos sistemas dinámicos interesantes, hasta que Yanbo Li, Zakary Littlefield y Kostas Bekris [22] describieron como lograr dichas propiedades sin solucionar un BVP con la invensión de los algoritmos *SST* y *SST**. Para el sistema dinámico en el que se enfoca este trabajo, un robot de manejo diferencial (DDR), no se conoce la solución a un BVP, lo que convierte a dichos algoritmos en buenos candidatos para resolver cuestiones de planificación de movimiento sobre este sistema.

El sistema dinámico noholonómico dado por un robot de manejo diferencial se caracteriza por tener dos ruedas con motores independientes, que le permite girar en su lugar. Existen numerosos problemas e investigaciones que involucran este sistema, por ejemplo, mantener la vigilancia a otro robot [27], estrategias óptimas en problemas de persecución y evasión [31, 26], experimentos en reconocimiento de objetos [4], trayectorias mínimas en tiempo con dinámica de primer orden [1] y por supuesto, el problema central del presente trabajo que son las trayectorias mínimas en tiempo con dinámica de segundo orden, considerando las aceleraciones de sus ruedas [29, 30].

En esta tesis se combinan herramientas fuertes de la teoria de control, como lo es el principio del máximo de Pontryagin, con métodos más practicos y modernos, como son los algoritmos de planificación de movimiento basados en muestreo, con el objetivo de investigar las trayectorias asintóticamente óptimas en tiempo considerando dinámica de segundo orden, con aceleración acotada, en el sistema noholonómico dado por un robot de manejo diferencial controlado por las aceleraciones en sus ruedas, utilizando las trayectorias localmente óptimas obtenidas con el principio del máximo de Pontryagin y los métodos de muestreo SST y SST*. La metodología que se desarrolla en el DDR para alcanzar el objetivo general de este trabajo se generaliza y mecaniza para su aplicación en otros sistemas. Se prueba que el SST* con

las modificaciones propuestas es probabilísticamente completo y asintóticamente óptimo, y se dan las condiciones para que converja más rápido que el SST* convencional, cuando se aplica en el DDR. Además se realiza un estudio experimental donde se observa que, el uso de los controles obtenidos con el principio del máximo de Pontryagin aumenta la velocidad de convergencia a un costo de trayectoria dado y reducen el costo de las trayectorias obtenidas por los algoritmos.

El contenido de esta tesis se organiza de la siguiente manera. En el capítulo 2 se describe la información y terminología necesaria para entender los métodos basados en muestreo, con enfoque en la solución de problemas kinodinámicos y noholonómicos, se define el problema a resolver y se explica el funcionamiento de los algoritmos PRM, RRT con sus versiones óptimas PRM* y RRT*, respectivamente. Finalmente, se describe en detalle el SST, el cual es el algoritmo principal en este trabajo. El capítulo 3 explica y define los conceptos necesarios para comprender el principio del máximo de Pontryagin, proporciona una interpretación geométrica, un ejemplo de como se utiliza este principio y encuentra los controles extremos del DDR que generan las trayectorias localmente óptimas. El capítulo 4 describe el marco de trabajo en el cual se basan los algoritmos SST y SST*, la adaptación de los algoritmos al sistema DDR con dinámica de segundo orden y con controles extremos a dicho marco de trabajo y el cumplimiento de las hipótesis, demuestra que dicha adaptación mantiene las propiedades relacionadas con completitud probabilística y optimalidad asintótica, generaliza la metodología que se desarrolló en el DDR para alcanzar el objetivo general de este trabajo, menciona algunos aspectos de la implementación y analiza la complejidad de cada una de las partes de los algoritmos, finalmente se analizan los resultados de las ejecuciones. El capítulo 5 contiene las conclusiones y los posibles futuros trabajos sobre esta tesis.

CAPÍTULO 2

Planificación de movimiento basada en muestreo

La planificación de movimiento basada en muestreo utiliza muestras aleatorias del espacio de configuraciones o el espacio de estados del robot para construir de manera rápida y efectiva una estructura de datos, espeficamente un grafo, donde los nodos son estados del robot y las aristas son caminos factibles entre los estados, el cual se utiliza para responder consultas de planificación. El muestreo surge por la necesidad de cubrir rápidamente un espacio de alta dimensión y por la complejidad o costo computacional de resolver el problema de conectar una configuración de inicio y una configuración objetivo a lo largo de una trayectoria valida. La mayoría de los métodos basados en muestreo garantizan completitud probabilística, esto significa que si existe solución, la probabilidad de encontrar una converge a uno a medida que el número de muestras o cardinalidad del grafo tiende a infinito. Por otro lado, los métodos basados en muestreo no son concluyentes a problemas sin solución, es decir, cuando no existe solución no tienen la capacidad de decir que el problema es irresoluble.

En este capítulo se describe la información y terminología necesaria para entender los métodos basados en muestreo, con enfoque en la solución de problemas kinodinámicos y

noholonómicos, se define el problema a resolver y se explica el funcionamiento de los algoritmos PRM, RRT con sus versiones óptimas PRM* y RRT*, respectivamente. Finalmente, se describe en detalle el SST, el cual es el algoritmo principal en este trabajo.

2.1. Formulación del problema

Se consideran sistemas dinámicos que cumplen restricciones diferenciable invariantes en el tiempo de la forma:

$$\dot{x} = f(x(t), u(t)) \qquad x(t) \in \mathbb{X}, u(t) \in \mathbb{U} \tag{2.1}$$

Donde $\mathbb{X}$ y $\mathbb{U}$ se definen acontinuación, así como otros conceptos importantes.

Definición 2.1.

- El **espacio de trabajo** es el espacio físico en el que se encuentra el robot. Generalmente es acotado y se supone que los límites del espacio representa un obstáculo para el robot. Se denota por $\mathcal{W}$.

- El **espacio de configuraciones** es el conjunto de configuraciones del robot, donde una configuración indica la posición y orientación en que el robot puede estar en el espacio de trabajo. Se supone que es una variedad, ya que permite propiedades útiles y necesarias, como la defición de la norma euclidiana y el concepto de bola, para la detección de colisiones. Se denota por $\mathcal{C}$. Además, el conjunto de las configuraciones para las cuales el robot está libre de colisión se denota por $\mathcal{C}_f$.

- El **espacio de estados** es el conjunto de los valores de las variables de estado, las cuales son un conjunto de variables linealmente independientes del sistema, que representan su estado dinámico completo en un determinado instante. Se denota por $\mathbb{X}$. Además, el conjunto de todos los estados para los cuales el robot está libre de colisión se denota por $\mathbb{X}_f$.

- El **espacio de controles** es el conjunto de los posibles controles que se pueden aplicar al sistema. Se denota por $\mathbb{U}$.

En el siguiente ejemplo se puede apreciar las anteriores definiciones y la representación dada por (2.1) de las restricciones diferenciales de un sistema.

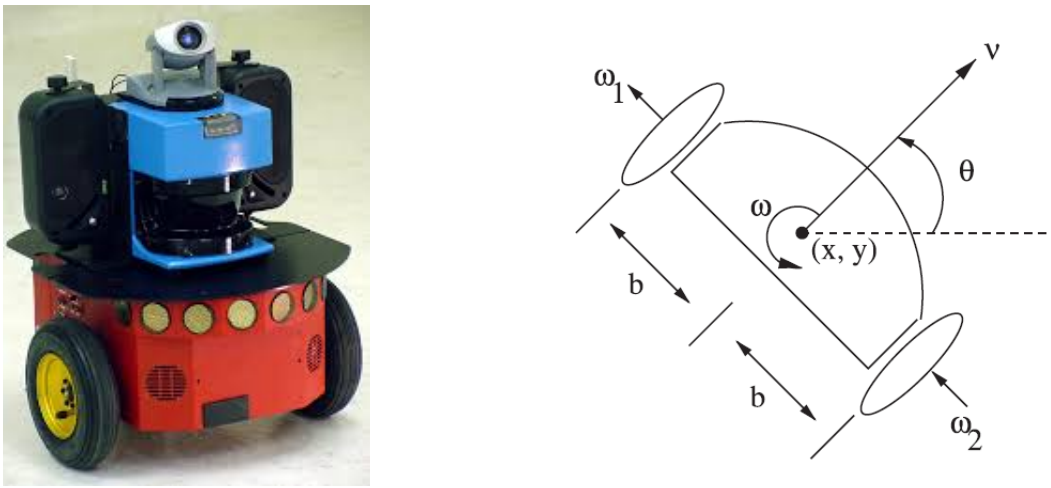


Figura 2.1: DDR (imagenes extraídas de [20, 31], respectivamente de izquierda a derecha)

Ejemplo 2.2. Considere un robot de manejo diferencial o DDR por sus siglas en inglés (figura 2.1), el cual se caracteriza por tener dos ruedas con motores independientes que permiten que el robot gire en su lugar. Se supone que las ruedas tienen radio r y distancia entre ellas igual a $2b$. Suponiendo que el robot tiene o no profundidad, se tendría que el espacio de trabajo es $\mathbb{R}^3$ o $\mathbb{R}^2$, respectivamente. Por otro lado, como la profundidad del robot es constante, entonces solo hay dos variables linealmente independientes y es conveniente definir $\mathcal{W} = \mathbb{R}^2$, ya que describe por completo la posición de robot y es de menor dimensión. Luego, $\mathcal{C} = \mathbb{R}^2 \times S^1$ donde (x, y) son las coordenadas del punto de referencia ubicado entre las ruedas y θ la orientación del robot con respecto al eje x. El estado del sistema puede ser expresado con $(x, y, \theta, v_r, v_l) \in \mathbb{R}^2 \times S^1 \times \mathbb{R}^2$, donde v_r y v_l denotan respectivamente las velocidades del punto de contacto de la rueda derecha e izquierda con el piso, por tanto $\mathbb{X} = \mathbb{R}^2 \times S^1 \times \mathbb{R}^2$. En la figura (2.1) se muestras las velocidades angulares de las ruedas ω_1 y ω_2, las cuales se relacionan con la velocidades tangenciales por las ecuaciones $v_1 = \omega_1 r$ y $v_2 = \omega_2 r$. Además,

es claro que $\omega = \dot{\theta}$.

Los controles del sistema son las aceleraciones a_l y a_r de la rueda izaquierda y derecha, respectivamente, si se supone que las ruedas tienen aceleración acotada por a, entonces $\mathbb{U} = [-a, a] \times [-a, a]$.

Las restricciones diferenciales o representación dinámica del sistema vienen dadas por la formula (sección 13.1.2.2 de [20]):

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_r \\ \dot{v}_l \end{pmatrix} = \begin{pmatrix} \frac{v_r+v_l}{2} r \cos\theta \\ \frac{v_r+v_l}{2} r \sin\theta \\ \frac{v_r-v_l}{2b} r \\ a_r \\ a_l \end{pmatrix} \tag{2.2}$$

La anterior ecuación también se conoce como *ecuación de transición de estado*, la cual coincide con la ecuación (2.1).

Definición 2.3.

- Un sistema **noholonómico** es aquel que tiene restricciones de movimiento, por ejemplo, no puede moverse instantáneamente en todas las direcciones. Más formalmente, la propiedad de no holonomía de un sistema se puede determinar usando el Teorema de Frobenius [18].

- El problema **kinodinámico** corresponde a determinar las entradas de control para conducir un sistema robótico desde una configuración inicial a una configuración final, evitando colisiones en un ambiente con obstáculos y respetando restricciones dinámicas físicas. Planificación kinodinámica fue propuesta en [8].

El sistema dado por un DDR, ejemplo (2.2), es un sistema noholonómico. Dado que no se puede mover en dirección perpendicular a la orientación de sus ruedas. Además, cuando se quiere encontrar controles que muevan el DDR de una configuración a otra, considerando que la velocidad y aceleración de las ruedas son acotadas, se trata de un problema kinodinámico.

Definición 2.4. Una **trayectoria** π es una función continua $\pi : [0, t_\pi] \to \mathbb{X}_f$, donde t_π es su duración, $\pi(0)$ es el estado donde comienza la trayectoria y $\pi(t_\pi)$ es el estado donde termina. La trayectoria π es generada desde $\pi(0)$ aplicando una **función de control** $\Psi : [0, t_\pi] \to \mathbb{U}$ por integración directa de la ecuación (2.1). Se dice que Ψ mueve el sistema de $\pi(0)$ a $\pi(t_\pi)$.

Note que en la definición de trayectoria se empleó como codominio a $\mathbb{X}_f$, por tanto trayectoria se definió equivalentemente, a lo que comúnmente se llama, trayectoria factible o valida. Ya que no considera a las funciones con imágenes en $\mathbb{X} - \mathbb{X}_f$, es decir, excluye las "trayectorias" con colisiones. Por otro lado, para sistemas controlables, como es el caso del ejemplo (2.2), una función de control dice qué controles se deben aplicar en cada instante de tiempo de la duración de la trayectoria, para generarla.

A partir de las definiciones anteriores, en principio el objetivo de un planificador de movimiento kinodinámico es encontrar una función de control, para generar una trayectoria en el espacio de estados del robot desde un estado inicial a un estado o región objetivo. Los planificadores basados en muestreo se implementan de modo que la función de control es constante por partes, concepto se define a continuación.

Definición 2.5. Una **función de control constante por partes** $\hat{\Psi}$ es una concatenación de funciones de control constantes de la forma $\Psi_i : [0, t_i] \to \{u_i\}$, donde $u_i \in \mathbb{U}$.

De la anterior definición se sigue que la duración de una función de control constante por partes $\hat{\Psi}$ es $\sum_i t_i$ y que t_i es el tiempo que se aplica el control u_i.

Después de la invención de algoritmos que asintóticamente garantizan encontrar una trayectoria solución a problemas de planificación de movimiento, se puede considerar tal enfoque en la solución del problema en que se centra la teoría de control óptimo. La cual tiene como objetivo principal la determinación de la trayectoria de costo óptima para las variables de control. Para el problema de este trabajo, el costo lo consideramos como el tiempo de la trayectoria, por tanto se hace la siguiente definición.

Definición 2.6. Una trayectoria $\pi^* : [0, t_{\pi^*}] \to \mathbb{X}_f$ se dice que es **óptima** u **óptima en tiempo** de x_s a x_f, si comienza en x_s, termina en x_f y para toda trayectoria $\pi : [0, t_\pi] \to \mathbb{X}_f$ tal que

$\pi(0) = x_s$ y $\pi(t_\pi) = x_f$, se cumple que $t_{\pi^*} \leq t_\pi$. Además, cualquier función de control $\Psi^* : [0, t_{\pi^*}] \to \mathbb{U}$ que genere a π^* se conoce como **función de control óptima** u **óptima en tiempo** de x_s a x_f.

Para el sistema noholonómico dado por un robot de manejo diferencial se conocen las trayectorias óptimas en tiempo, en un ambiente sin obtáculos y con dinámica de primer orden, es decir, velocidad acotada y puede ser discontinua [1]. Para dichas trayectorias se utilizaron los controles extremos generados por el principio del máximo de Pontryagin, los cuales generan trayectorias que contienen a las trayectorias óptimas, este método se explica con detalle en el capítulo 3.

Al considerar dinámica de segundo orden, es decir, velocidad y aceleración acotadas, no se conocen las trayectorias óptimas en tiempo para un DDR. Sin embargo, se conocen las trayectorias localmente óptimas como consecuencias del PMP, por tanto las trayectorias óptimas son concatenaciones de ellas. Es aquí donde se utilizan los metodos basados en muestreo, para hacer concatenaciones con las trayectorias localmente óptimas e investigar la óptimalidad global, es decir, si la solución tiende a ser óptima cuando aumenta el número de nodos en el grafo. A partir de lo anterior surge el objetivo general de este trabajo.

Objetivo general: *Investigar las trayectorias asintóticamente óptimas en tiempo considerando dinámica de segundo orden, con aceleración acotada, en el sistema noholonómico dado por un robot de manejo diferencial (DDR) controlado por las aceleraciones en sus ruedas, utilizando las trayectorias localmente óptimas obtenidas con el principio del máximo de Pontryagin y el método de muestreo SST.*

2.2. Planificadores basados en muestreo

En general los planificadores basados en muestreo tiene dos fases: la selección y la propagación del grafo. Donde la fase de selección es la que se encarga de escoger el o los posibles vértices del grafo para aplicar la propagación, se debe diseñar para aumentar la probabilidad de buscar en partes no exploradas de $\mathbb{X}_f$. La fase de propagación es la que apartir de un vérti-

ce del grafo y con algún método genera un nuevo estado que puede ser parte del grafo. En el algoritmo (2.1) se muestra un resumen en alto nivel de un planificador basado en muestreo, donde V y E son el conjunto de vértices y aristas respectivamente, y se contruye un árbol enraizado en el estado inicial (condición inicial del problema). Cabe mencionar que el hecho de que la estructura de datos sea un árbol y esté enraizado en el estado inicial, es un enfoque más moderno y difiere a los comienzos de los planificadores basados en muestreo, como se observa en la siguiente sección.

Algoritmo 2.1: Planificador ingenuo

```
1  V ← {x_0}; E ← ∅ ;
2  G = (V, E);
3  for i = 1, ..., n do
4      x_selected ← un elemento aleatorio de V;
5      x_new ← con información de x_selected y algún método se genera un nuevo vértice;
6      if la trayectoria de x_selected a x_new está libre de colisión then
7          V ← V ∪ {x_new};
8          E ← E ∪ {(x_selected, x_new)}
9      end
10 end
11 return G = (V, E)
```

2.2.1. PRM

El algoritmo PRM es el primer método de planificación basado en muestreo, está dirigido principalmente a aplicaciones de consultas [15]. Lo interesante de este método es que para robots con muchos grados de libertad, responde las consultas eficientemente, es decir, en un tiempo de ejecución razonable. Lo cual es destacable, ya que en el momento en que se publicó [15], pocos métodos efectivos de planificación de movimiento estaban disponibles para resolver problemas en altas dimensiones. Además, es probabilisticamente completo.

El método construye un grafo de trayectorias libres de colisiones, para robots que se mueven en un espacio de configuración con y sin obtáculos y procede en dos fases: una fase de aprendizaje y una fase de consulta.

Para describir el método se definen las siguientes funciones, la cuales también se utilizan en posteriores algoritmos:

- **Sample**(X): Genera una muestrea aleatoria de X.
- **Near**(Y, x, r): Genera el conjunto de puntos en Y que están a una distancia de x menor o igual a r.
- **Collision_Free**(x, y): Función booleana que devuelve 1 si la trayectoria que recorre el sistema dinámico para ir de x a y está libre de colisión, y 0 en otro caso.

Algoritmo 2.2: PRM

$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
$G = (V, E)$;
for $i = 1, \ldots, n$ **do**
    $x_{rand} \leftarrow$ Sample$(\mathbb{X}_f)$;
    $W \leftarrow$ Near(V, x_{rand}, r);
    $V \leftarrow V \cup \{x_{rand}\}$;
    for $x' \in W$, *en orden creciente de* $||x' - x_{rand}||$ **do**
        if x_{rand} *y* x' *no están en la misma componente conexa de* G **then**
            if *Collision_Free*(x', x_{rand}) **then**
                $E \leftarrow E \cup \{(x', x_{rand})\}$
            end
        end
    end
end
return $G = (V, E)$

La fase de aprendizaje, descrita en el Algoritmo (2.2), comienza con un grafo vacío. En cada iteración se muestrea un punto $x_{rand} \in \mathbb{X}_f$ y se agrega al conjunto de vértices V. Luego, se comprueban las conexiones entre x_{rand} y los vértices en V que están dentro de la bola de radio r centrada en x_{rand}, en orden creciente de distancia a x_{rand}, utilizando un planificador local, por ejemplo, conexión en línea recta. Cuando hay una conexión exitosa, es decir, sin colisión, se agrega una nueva arista al grafo, en especifico, al conjunto de aristas E. Como el objetivo del algoritmo es conectar una configuración con otra, para evitar cálculos innecesarios, no se conectan con x_{rand} los vértices que están en la misma componente conexa del grafo. Por lo tanto, el grafo creado por el PRM es un bosque, es decir, una colección de árboles.

Después de la fase de aprendizaje, se pueden responder múltiples consultas. Una consulta solicita una trayectoria entre dos configuraciones libres del robot. Para procesar una consulta, el método primero intenta encontrar una trayectoria desde las configuraciones de inicio y objetivo a dos nodos del grafo. Luego, se realiza una búsqueda en el grafo para encontrar un camino que conecte estos nodos del grafo. En caso de existir, ese camino encuentra una trayectoria para responder a la consulta.

2.2.2. RRT

Los árboles aleatorios de exploración rápida (RRTs) son el primer enfoque aleatorio de la planificación kinodinámica [19, 21], están principalmente dirigidos a aplicaciones de consulta única. El RRT es una estructura de datos y un algoritmo diseñado para buscar de manera eficiente en espacios no convexos de alta dimensión, mediante la construcción aleatoria de un árbol de trayectorias factibles, el cual tiene como raíz el estado inicial.

Se definen las siguientes funciones para describir el método, la cuales también se utilizan en posteriores algoritmos:

- **Nearest**(Y, x): Devuelve el elemento de Y que está más cerca a x, bajo alguna métrica.
- **Select_Input**(x, y): Selecciona algún control u, bajo algún método, muestreo o heurísti-

ca con el objetivo de minimizar la distancia de x a y.

- **New_State**(x, u): Devuelve el estado en el que se encuentra el sistema después de aplicar el control u desde x, por algún intervalo de tiempo.

En el Algoritmo (2.3) se proporciona un esquema del método estandar, el primero que se publicó [19]. El algoritmo se inicializa con un gráfo de un único vértice que representa el estado inicial y sin aristas. En cada iteración se muestrea un punto $x_{rand} \in \mathbb{X}_f$ y se encuentra el vértice más cercano a x_{rand}, en términos de alguna métrica, se denota por $x_{nearest}$. Luego, selecciona algún control u, que minimiza la distancia de $x_{nearest}$ a x_{rand}, es decir, se intenta expandir el árbol hacia x_{rand}. Se aplica el control, ya sea por un tiempo fijo o aleatorio en algún intervalo, para generar un nuevo estado x_{new} por integración directa de la ecuación de transición de estados. Finalmente, se comprueba que la trayectoria de $x_{nearest}$ a x_{new} está libre de colisiones, en caso afirmativo, se agrega x_{new} al conjunto de vértices y $(x_{nearest}, x_{new})$ al conjunto de aristas.

Algoritmo 2.3: RRT

$V \leftarrow \{x_0\}$; $E \leftarrow \emptyset$;
$G = (V, E)$;
for $i = 1, \ldots, n$ **do**
    $x_{rand} \leftarrow$ Sample$(\mathbb{X}_f)$;
    $x_{nearest} \leftarrow$ Nearest(G, x_{rand});
    $u \leftarrow$ Select_Input$(x_{rand}, x_{nearest})$;
    $x_{new} \leftarrow$ New_State$(x_{nearest}, u)$;
    if *Collision_Free*$(x_{nearest}, x_{new})$ **then**
        $V \leftarrow V \cup \{x_{new}\}$;
        $E \leftarrow E \cup \{(x_{near}, x_{new})\}$;
    end
end
return $G = (V, E)$;

Luego se utiliza el árbol para resolver el problema de planificación, basta con verificar si alguno de los vértices del árbol es el estado objetivo o está en la región objetivo.

El PRM y el RRT comparten la misma idea de conectar muestras aleatorias en el espacio de configuración o de estados, pero difieren en la forma en que construyen el grafo. Además, se puede resaltar que un RRT siempre es conexo, incluso el número de aristas es minimal. Al igual que el PRM, un RRT es probabilísticamente completo en condiciones muy generales [21]. Ambos métodos han demostrado que funcionan bien en la práctica, sin embargo, no proporcionan optimalidad en ningún criterio.

2.3. Planificadores óptimos basados en muestreo

Luego de la invención de algoritmos basados en muestreo como PRM y RRT, los cuales se demostró que son probabilísticamente completos, por un tiempo se dedicó poco esfuerzo al análisis formal de la calidad de la solución devuelta por dichos algoritmos. Hasta que en [13], se analizó rigurosamente el comportamiento asintótico del costo de la solución devuelta por los algoritmos basados en muestreo a medida que aumenta el número de muestras, además se presentaron los algoritmos PRM* y RRT*, que son asintóticamente óptimos, es decir, el costo de la solución devuelta converge casi seguramente al costo óptimo. Dichos algoritmos se presentaran en esta sección.

2.3.1. PRM*

Este algoritmo es similar a la versión estándar, con una diferencia en el tamaño de la bola centrada en x_{rand}. El algoritmo PRM comprueba conexiones entre vértices del grafo que están dentro de un radio fijo r entre sí, por tanto la constante r es un parámetro del algoritmo, mientras que en el PRM* el radio de conexión r se elige en función del número de muestras n por medio de la siguiente función:

$$r(n) = \gamma_{PRM}(\log(n)/n)^{1/d}$$

donde $\gamma_{PRM} > 2(1 + 1/d)^{1/d}(\mu(\mathcal{C}_f)/\zeta_d)^{1/d}$, d es la dimensión de $\mathcal{C}$, $\mu(\mathcal{C}_f)$ es la medida de Lebesgue de $\mathcal{C}_f$ y ζ_d es el volumen de la esfera unitaria de dimensión d. Claramente, el radio de conexión $r(n)$ disminuye a medida que aumenta el número de muestras n. En promedio, el número de conexiones de un vértice dado es del orden $\log(n)$ [13].

2.3.2. RRT*

El algoritmo RRT* es una extensión del RRT, en el sentido de que hacen lo mismo hasta agregar un nuevo nodo al árbol (linea 9 del algoritmo 2.3), aunque se formaliza la idea de expandir el árbol hacia x_{rand}, luego se hace un proceso de conexión para garantizar que se llegue a un vértice dado a través del camino más corto. Otra diferencia importante entre RRT y RRT*, es que RRT* considera el costo acumulativo desde la raíz del árbol hasta un nodo y hace un proceso de reconexión para obtener la optimización, ya que el RRT* es asintóticamente óptimo [13].

Antes de describir el algoritmo, es fundamental conocer la siguiente definición.

Definición 2.7. Dados dos puntos $x, y \in X$, **la función de dirección** Steer : $X \times X \to X$ devuelve un punto z tal que minimiza $||z - y||$ al mismo tiempo que mantiene $||z - x|| \leq \rho$, para un valor predeterminado $\rho > 0$, es decir:

$$\text{Steer}(x, y) = \arg \min_{z \in \mathcal{B}_{x,\rho}} ||z - y||$$

La definición anterior es la formalización de la idea de prolongar el árbol hacia x_{rand} en el RRT, fundamental para la construcción del árbol y la optimalidad asintótica. En el caso de un sistema dinámico, la función de dirección corresponde a la solución de un problema de frontera de dos puntos (BVP, por sus siglas en inglés). Abordar este problema concierne a resolver una ecuación diferencial, al tiempo que satisface ciertas condiciones de contorno. Sin embargo, no es fácil producir una solución a un BVP para muchos sistemas dinámicos interesantes.

Como se mencionó anteriormente el algoritmo RRT*, descrito en el algoritmo (2.4), agrega vértices a V de manera similar al RRT, aunque en este algoritmo se utiliza la función de

Algoritmo 2.4: RRT*

$V \leftarrow \{x_0\}$; $E \leftarrow \emptyset$;
$G = (V, E)$;
for $i = 1, \ldots, n$ **do**
  $x_{rand} \leftarrow$ Sample($\mathbb{X}_f$);
  $x_{nearest} \leftarrow$ Nearest(G, x_{rand});
  $x_{new} \leftarrow$ Steer($x_{nearest}, x_{rand}$);
  if *Collision_Free*($x_{nearest}, x_{new}$) **then**
    $X_{near} \leftarrow$ Near($G, x_{new}, r(i)$);
    $V \leftarrow V \cup \{x_{new}\}$;
    $x_{min} \leftarrow x_{nearest}$; $c_{min} \leftarrow$ Cost($x_{nearest}$) + Cost($(x_{nearest}, x_{new})$);
    for $x_{near} \in X_{near}$ **do**
      if *Collision_Free*(x_{near}, x_{new}) ***and***
       Cost(x_{near})+*Cost*((x_{near}, x_{new})) $< c_{min}$ **then**
        $x_{min} \leftarrow x_{near}$; $c_{min} \leftarrow$ Cost(x_{near}) + Cost((x_{near}, x_{new}));
      end
    end
    $E \leftarrow E \cup \{(x_{min}, x_{new})\}$;
    for $x_{near} \in X_{near}$ **do**
      if *Collision_Free*(x_{new}, x_{near}) ***and***
       Cost(x_{new})+*Cost*((x_{new}, x_{near})) <*Cost*(x_{near}) **then**
        $x_{parent} \leftarrow$ Parent(x_{near});
        $E \leftarrow (E \backslash \{(x_{parent}, x_{near})\}) \cup \{(x_{new}, x_{near})\}$
      end
    end
  end
end
return $G = (V, E)$;

dirección en vez de muestrear los controles, y para conectar el nuevo vértice x_{new} al árbol, solo considera los vértices dentro de una distancia dada $r(n)$ de x_{new}, que está determinada por el número actual de vértices n en el árbol. La ecuación que determina este radio es la siguiente:

$$r(n) = \text{mín}(\gamma_{RRT*}(\log(n)/n)^{1/d}, \eta)$$

donde, similar al RPM*, $\gamma_{RRT*} > 2(1 + /d)^{1/d}(\mu(\mathcal{C}_f)/\zeta_d)^{1/d}$, d es la dimensión de $\mathcal{C}$, $\mu(\mathcal{C}_f)$ es la medida de Lebesgue de $\mathcal{C}_f$, ζ_d es el volumen de la esfera unitaria de dimensión d y η es una costante que acota el tamaño del radio.

No todas las conexiones factibles dan como resultado nuevas aristas en el grafo. En particular, se crea una arista dentro del vecindario X_{near}, si esa arista está libre de colisiones y tiene la trayectoria de costo mínimo para x_{new} desde un vértice en X_{near}. Luego, en el proceso de reconexión, se crean aristas desde x_{new} a vértices en X_{near}, digamos x', si la trayectoria a través de x_{new} tiene un costo menor que la trayectoria actual a x'. En ese caso, la arista que une el vértice x' a su padre actual se elimina para mantener la estructura de árbol.

2.4. SST

Algunas de las propiedades deseables de los algoritmos de planificación de movimiento basados en muestreo, son la completitud probabilística y que sean asintotícamente óptimos. Para lograr esta última propiedad en sistemas dinámicos, se requiere resolver un BVP en el espacio de estados. Por ejemplo, en el RRT* en la utilización de la función Steer y en el PRM* se debe encontrar la trayectoria entre dos estados. Sin embargo, no es fácil producir una solución a un BVP para muchos sistemas dinámicos interesantes. Entonces, surgió el interrogante de qué si era posible lograr garantías de optimización al planificar sistemas sin resolver un BVP. Yanbo Li, Zakary Littlefield y Kostas Bekris [22] describieron como lograr dichas propiedades sin solucionar un BVP con la invensión de los algoritmos *SST* y *SST** que son asintóticamente casi óptimo y óptimo, respectivamente. Además, se muestra

que los algoritmos convergen rápidamente a trayectorias de alta calidad, mientras mantienen una estructura de datos de poco tamaño, comparado con otros métodos, lo que los hace computacionalmente eficientes [22].

Para lograr la optimalidad en los algoritmos SST y SST*, se introdujo un nuevo marco de trabajo y varias hipótesis que debe cumplir el sistema dado por la ecuación (2.1), descrito en la sección (4.1).

Adicional a las fases de selección y propagación, el SST agrega una fase de poda, la cual es su principal característica, y consiste en eliminar vertices para mantener el grafo de menor tamaño, aunque se debe hacer de tal forma que no afecte el objetivo del método. El algoritmo (2.5) describe el SST.

Algoritmo 2.5: SST$(N, \delta_{BN}, \delta_s)$

$V_{active} \leftarrow \{x_0\}$; $V_{inactive} \leftarrow \emptyset$; $E \leftarrow \emptyset$;
$G = (V = V_{active} \cup V_{inactive}, E)$;
$s_0 \leftarrow x_0$; $s_0.rep = x_0$; $S = \{s_0\}$;
for $i = 1, \ldots, N$ **do**
    $x_{selected} \leftarrow$ Best_First_Selection$(\mathbb{X}_f, V_{active}, \delta_{BN})$;
    $x_{new} \leftarrow$ Propagation$(x_{selected})$;
    if *Collision_Free*$(x_{selected}, x_{new})$ **then**
        if *Is_Locally_the_Best*(x_{new}, δ_s) **then**
            $V_{active} \leftarrow V_{active} \cup \{x_0\}$;
            $E \leftarrow E \cup \{(x_{selected}, x_{new})\}$;
            Prune_Dominated_Nodes(x_{new});
        end
    end
end
return $G = (V, E)$;

Antes de describir como funciona el algoritmo, o mejor dicho, como se realiza cada una

de las fases (selección, propagación y poda), se debe definir los siguiente:

- Los vértices del árbol V estan divididos en dos conjuntos disjuntos, a saber:
  - V_{active}: Contiene a los vértices que son localmente óptimos, es decir, en una vecindad o región local tienen el mejor costo de trayectoria desde la raíz.
  - $V_{inactive}$: Contiene a los vértices dominados, es decir, en una vecindad o región local no tienen el mejor costo de trayectoria desde la raíz, pero tienen hijos que son localmente óptimos. Estos vértices se conservan para mentener la conexidad con vértices que si son localmente óptimos, además de mantener y estructura de árbol.
- El SST tiene dos parámetros de entrada, δ_{BN} y δ_s, los cuales se utilizan de la siguente manera:
  - δ_{BN}: Influye en el número de vértices que se consideran para seleccionar el vértice a propagar, ya que se hace una bola de radio δ_{BN} centrada en la muestra aleatoria y se elige el de menor costo contenido en esa bola (figura 2.2). Entonces cuanto más grande sea este parámetro se consideran más vértices del gráfo y por tanto se generan trayectorias de mejor calidad. Aunque, es más probable que se ignore la exploración, ya que en regiones estrechas se pueden seleccionar con mayor probabilidad los mismos vértices para la propagación.
  - δ_s: Es el parámetro responsable de realizar la poda, cuanto mayor sea este parámetro, más vértices se eliminan de V_{active} y agregaran a $V_{inactive}$ o se eliminan del árbol. Por tanto, controla que tan disperso es el árbol.
- Se define un conjunto S y sus elementos se llaman "testigos", los cuales no se eliminan ni se modifican. Digamos que $s \in S$. A cada elemento de S se le asocia un vértice del árbol, se le conoce como representante del testigo, se denota con s_{rep} al representante de s. El vértice s_{rep} tiene el mejor costo de trayectoria desde la raíz dentro de una distancia δ_s al testigo s. Por la anterior propiedad que debe cumplir s_{rep}, se tiene que

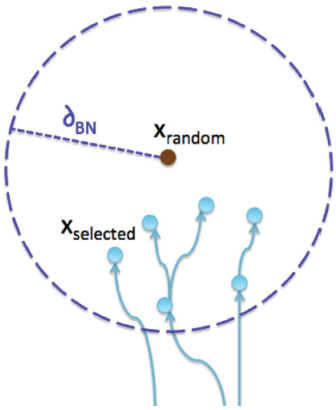


Figura 2.2: Imagen extraída de [22].

s_{rep} se modifica cada vez que un nuevo vértice tenga el mejor costo de trayectoria desde la raíz y esté a una distancia menor que δ_s a s.

Durante la descripción del algoritmo se van entendiendo los conceptos anteriores y él porqué de su definición.

Selección

El conjunto V_{active} se utiliza en el proceso de selección (linea 5 del algoritmo 2.5), aplicando el algoritmo (2.6). El proceso todavía promueve la selección de vértices en partes de X_f poco exploradas, como en los enfoques originales, pero localmente solo se seleccionan los vértices que corresponden a la mejor ruta desde la raíz. En caso de que no existan vértices del árbol en el vecindario local, devuelve el vértice de V_{active} más cercano a x_{rand}.

Algoritmo 2.6: Best_First_Selection($\mathbb{X}_f, V_{active}, \delta_{BN}$)

$x_{rand} \leftarrow$ Sample($\mathbb{X}_f$);
$X_{near} \leftarrow$ Near($V_{active}, x_{rand}, \delta_{BN}$);
if $X_{near} = \emptyset$ **then**
    return Nearest(V_{active}, x_{rand})
else
    return $\arg \min_{x \in X_{near}} Cost(x)$
end

Propagación

Se emplea un proceso de propagación totalmente aleatorio tanto en términos del control seleccionado como de la duración de la propagación (Algoritmo 2.7), es decir, dado un tiempo máximo de propagación T_{prop}, se muestrea la duración en que se va aplicar un control, también aleatorio, en el intervalo $[0, T_{prop}]$. Luego, con integración directa de la ecuación (2.1), se encuentra el estado del sistema después de aplicar el control seleccionado por el tiempo muestreado.

Algoritmo 2.7: Propagation($x_{selected}$)

$t_{rand} \leftarrow$ Sample($[0, T_{prop}]$);
$u \leftarrow$ Sample($\mathbb{U}$);
$x_{new} \leftarrow \int_0^{t_{rand}} f(x(t), u)\, dt + x_{selected}$;
return x_{new};

El estado x_{new} va ser parte del árbol si el camino hacia él es libre de colisiones y es el mejor localmente. El proceso de verificar que x_{new} es el mejor localmente, respecto al costo, está descrito en el algoritmo (2.8).

Algoritmo 2.8: Is_Locally_the_Best(x_{new}, δ_s)

$s_{new} \leftarrow$ Nearest(S, x_{new});
if $||x_{new} - s_{new}|| > \delta_s$ **then**
    $S \leftarrow S \cup \{x_{new}\}$;
    $s_{new} \leftarrow x_{new}$;
    $s_{new}.rep \leftarrow NULL$;
end
$x_{peer} \leftarrow s_{new}.rep$;
if $x_{peer} == NULL$ *and Cost*$(x_{new}) <$ *Cost*(x_{peer}) **then**
    return true;
end
return false;

Primero, se calcula el testigo más cercano a x_{new} del conjunto S, se denota por s_{new}. Si s_{new} tiene una distancia mayor a δ_s de x_{new}, entonces x_{new} se convierte en un nuevo elemento de S, por tanto él se convierte en el elemento de S más cercano a si mismo y por el momento no tiene representante. El representante de s_{new} se almacena en la variable x_{peer}. Luego, para agregar x_{new} al árbol se debe cumplir al menos una de las siguientes condiciones:

- El costo de x_{new} es menor que el costo del representante del testigo más cercano a él, el cual está almacenado en x_{peer}. En este caso, x_{new} remplazará al representante de s_{new}, ya que tiene mejor costo en la vecindad de s_{new} (figura 2.3).

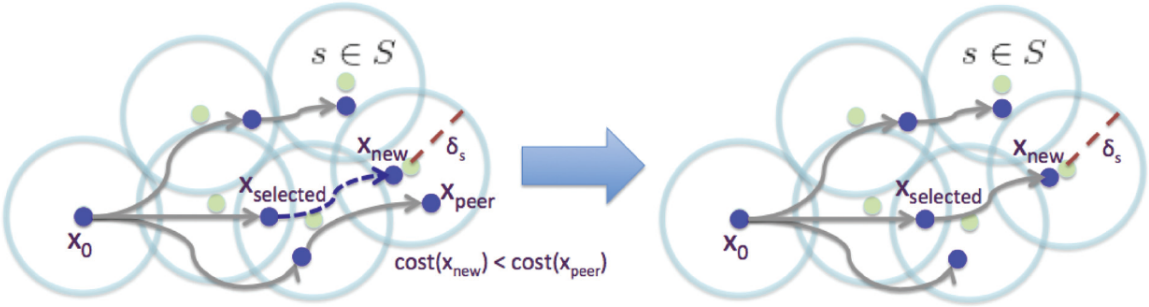


Figura 2.3: Imagen extraída de [22]. Los elementos de S son representados de color verde opaco y los vértices del árbol de azul. Se muestra el proceso de propagación cuando $Cost(x_{new}) < Cost(x_{peer})$. El vértice x_{peer} se elimina (poda) y el borde recién propagado se agrega al árbol.

- La variable x_{peer} es nula, es decir, la muestra x_{new} se acaba de agregar como testigo, esto significa que no hay un vecindario local alrededor de elementos de S que contenga a x_{new}, esto se puede observar en la figura (2.4). Por tanto x_{new} es locamente óptimo.

En caso de no cumplir ninguna de las anteriores condiciones, es decir, x_{new} está en una vecindad al rededor de un testigo y $Cost(x_{peer}) < Cost(x_{new})$, el vértice x_{peer} permanece en el árbol y la última propagación $x_{selected} \rightarrow x_{new}$ es ignorada y x_{new} no se agrega al árbol.

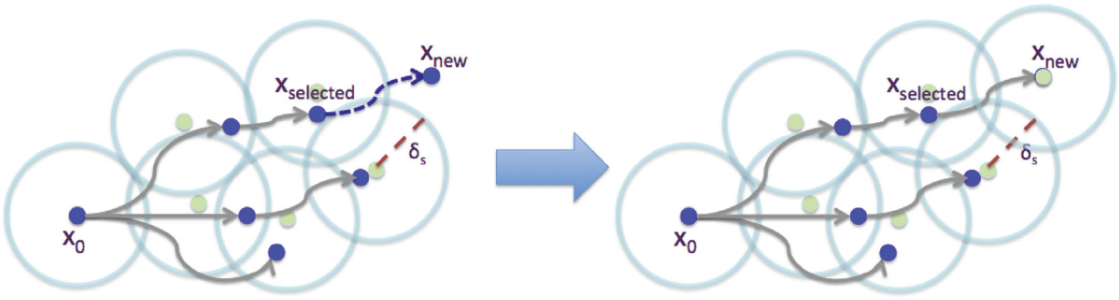


Figura 2.4: Imagen extraída de [22]. Se muestra el proceso de propagación cuando x_{new} no está cerca de un testigo. La arista $(x_{selected}, x_{new})$ se agrega al árbol y se agrega un testigo a S en la ubicación de x_{new}.

Poda

Los vecindarios centrados en los elementos de S, los cuales son bolas de radio δ_s, utilizados para decidir el agregar o no a x_{new} al árbol, también se utilizan para el proceso de poda (Algoritmo 2.9), que consiste en desactivar los vértices dominados y eliminar los que sean hojas.

Se definen las siguientes funciones, la cuales se utilizan en el algoritmo (2.9):

- **Is_Leaf**(x): Es una función booleana que devuelve 1 si x es un vértice hoja del árbol, y 0 en caso contrario.

- **Parent**(x): Devuelve el vértice padre de x, esto es, el primer vértice que está conectado con x en el camino que va desde x hasta la raíz.

Primero se encuentra s_{new} el testigo más cercano de x_{new}, el vértice que se acaba de agregar al árbol, y se almacena su representante en x_{peer}. Cuando existe x_{peer}, se elimina del conjunto activo de vértices V_{active} y se agrega al inactivo $V_{inactive}$, ya que está dominado por x_{new}, esto pasa cuando x_{new} está en alguna de las vecindades alrededor de los elementos de S. Luego, x_{new} reemplaza a x_{peer} como el representante de s_{new}. Si x_{peer} es un vértice hoja, entonces se elimina del árbol. Se mantiene en el árbol cuando no es un vértice hoja, lo cual mantiene la conexión con hijos que son localmente óptimos. La eliminación de x_{peer} puede

causar un efecto en cascada para sus padres, los cuales se eliminan del árbol si están en el conjunto inactivo $V_{inactive}$, ya que la única razón por la que se mantuvieron en el árbol fue porque estaban conduciendo a x_{peer}. Dicho proceso de cascada se detiene cuando se llega a un vértice que es localmente óptimo, es decir, está en V_{active}.

Algoritmo 2.9: Prune_Dominated_Nodes(x_{new})

```
1  s_new ← Nearest(S, x_new);
2  x_peer ← s_new.rep;
3  if x_peer != NULL then
4  |   V_active ← V_active \ {x_peer};
5  |   V_inactive ← V_inactive ∪ {x_peer};
6  end
7  s_new.rep ← x_new;
8  while Is_Leaf(x_peer) and x_peer ∈ V_inactive do
9  |   x_parent ← Parent(x_peer);
10 |   E ← E \ (x_parent, x_peer);
11 |   V_inactive ← V_inactive \ {x_peer};
12 |   x_peer ← x_parent;
13 end
```

El proceso de poda da como resultado una estructura de datos dispersa y de tamaño acotado, en lugar de aumentar el número de vértices indefinidamente a medida que el número de iteraciones crece. En la figura (2.5) se muestra la relación entre los conjuntos V_{active}, $V_{inactive}$ y S. También el proceso de cascada cuando se modifica el representante de un elemento de S.

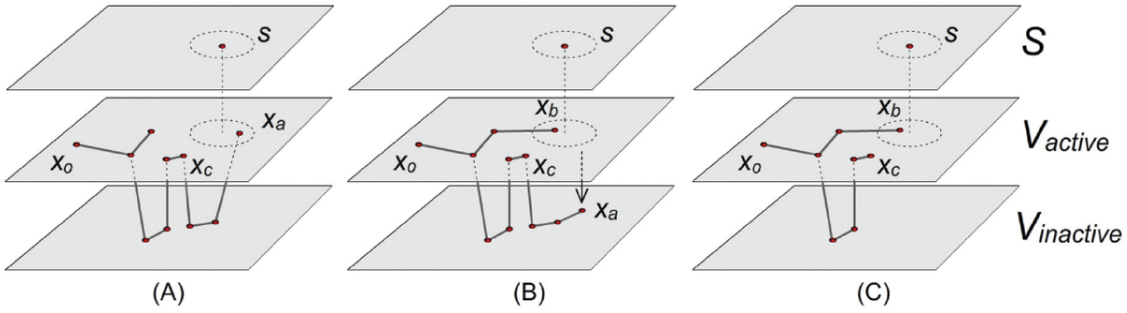


Figura 2.5: Imagen extraída de [22]. (A) Una trayectoria $x_o \rightarrow x_c \rightarrow x_a$ donde x_a es el representante de $s \in S$ y algunos de sus vértices están dominados localmente, por tanto pertenecen a $V_{inactive}$ y se mantienen en el árbol para la conexión con x_a. (B) Se genera un nuevo vértice x_b, el cual tiene un mejor costo que x_a. Luego, x_a se elimina de V_{active} y se agrega a $V_{inactive}$. (C) x_b es el nuevo representante de s. La trayectoria $x_c \rightarrow x_a$ que se encuentra en $V_{inactive}$ se elimina por el proceso de cascada generado por la poda.

2.4.1. SST*

El SST no es asintóticamente óptimo, principalmente debido a la operación de poda de tamaño fijo, la cual acota el tamaño del árbol. La solución a esto es reducir lentamente los parámetros δ_{BN} y δ_s, esto genera que eventualmente el SST converga en comportamiento al enfoque del algoritmo ingenuo de planificación basada en muestreo (algoritmo 2.1). En efecto, ya que cuando δ_{BN} tiende a cero, se consideran menos vértices en el proceso de selección y tiende a explorar más, hasta que funciona de manera uniforme, es decir, selecciona $x_{selected}$ de manera aleatoria y uniforme entre los nodos del árbol. Cuando δ_s tiende a cero aumentan el tamaño del árbol, lo que conlleva a mejores trayectorias en termino de costo, por tanto se hace más difícil realizar la poda, hasta el punto que se elimina el proceso de poda y se generarán todos los estados libres de colisión.

Siguiendo la idea anterior, se crea el meta-algoritmo SST* (algortimo 2.10) que ejecuta repetidamente al SST en una misma estructura de datos, es decir, en una ejecución del SST se continua el proceso con el árbol que devolvió la ejecución anteriór. Además se agrega un

proceso que va reduciendo los parámetros δ_{BN} y δ_s, mientras aumenta el número de iteraciones del SST (lineas 4-5). Donde $\xi \in (0, 1)$ y d y l son las dimensiones del espacio de estados y de controles, respectivamente.

Algoritmo 2.10: SST*$(n, N_0, \delta_{BN_0}, \delta_{s_0}, \xi)$

1 $N \leftarrow N_0$; $\delta_{BN} \leftarrow \delta_{BN_0}$; $\delta_s \leftarrow \delta_{s_0}$;
2 **for** $j = 1, \ldots, n$ **do**
3 SST$(N, \delta_{BN}, \delta_s)$;
4 $\delta_s \leftarrow \xi \cdot \delta_s$; $\delta_{BN} \leftarrow \xi \cdot \delta_{BN}$;
5 $N \leftarrow (1 + \log j) \cdot \xi^{-(d+l+1)j} \cdot N_0$;
6 **end**

Note que la relación entre el PRM con el PRM* y el SST con el SST* son muy similares, dado que el PRM* también es un meta-algoritmo que ejecuta repetidamente el PRM con un proceso de reducción de parámetro. Y igual que el PRM*, el SST* cumple con la propiedad deseable de la optimalidad asintótica.

CAPÍTULO 3

Principio del máximo de Pontryagin

La teoría de control óptimo tiene como objetivo principal la determinación de la trayectoria de costo óptima para una variable de control $u(t)$, en un sistema dinámico. Por supuesto, una vez se ha encontrado la trayectoria de control óptima $u^*(t)$, también se puede encontrar la trayectoria de estado óptima $x^*(t)$, que corresponde a la ejecución del control $u*(t)$. De hecho, las rutas óptimas $u^*(t)$ y $x^*(t)$ generalmente se encuentran en el mismo proceso, aplicando el principio del máximo de Pontryagin (PMP).

Este capítulo explica y define los conceptos necesarios para comprender el principio del máximo de Pontryagin, proporciona una interpretación geométrica y un ejemplo de como se utiliza este principio y encuentra los controles extremos del DDR.

3.1. Formulación de problema

El concepto de optimización en sistemas dinámicos está relacionado con maximizar o minimizar alguna cantidad estrechamente relacionada al problema en cuestión, tal que el valor de esa cantidad exprese la calidad de la solución encontrada, lo que motiva la definición

de funcional objetivo.

Definición 3.1. El **funcional objetivo** da una medida cuantitativa del comportamiento del sistema en el tiempo. Se consideran funcionales de la forma:

$$J(u) = \int_{t_s}^{t_f} L(x(t), u(t))\,\mathrm{d}t + G(x(t_f)) \tag{3.1}$$

donde:

- x es la variable de estado, tal que $x(t) \in \Omega$ conjunto abierto de $\mathbb{R}^n$ o $x(t) \in M$ una variedad n-dimensional, para todo t.
- u es la variable de control, tal que $u(t) \in U$ subconjunto de $\mathbb{R}^m$ de controles admisibles, para todo t.
- L es la función de costo del recorrido.
- $G(x(t_f))$ es la función de costo terminal.

siendo más riguroso, se debería escribir $x(t, u)$ en vez de $x(t)$, ya que la trayectoria que recorre el sistema en el espacio de estados está determinada por la trayectoria de control. Por tanto, se debe entender que $x(t)$ es la trayectoria de se generó con $u(t)$.

En términos del funcional objetivo, el problema a resolver en la teoria de control óptimo se expresa de la siguiente manera: Dado un sistema dinámico y dos estados x_s y x_f se quiere encontrar una trayectoria de control que sea admisible, diriga el sistema del estado x_s a x_f y haga que el funcional objetivo alcance el valor mínimo. Expresado en términos matemáticos:

$$\begin{aligned} &\min_{u \in U} \quad J(u) \\ &\text{sujeto a : } \dot{x} = f(x, u) \\ &\text{con : } x(t_s) = x_s \text{ y } x(t_f) = x_f. \end{aligned} \tag{3.2}$$

donde $f(x, u)$ es la función de transición de estado y $x(t_s) = x_s$ y $x(t_f) = x_f$ son las condiciones de frontera del sistema. Se tiene una expresión análoga cuando se quiere maximizar el funcional objetivo.

Antes de enunciar el principio del máximo de Pontryagin es fundamental la siguiente definición.

Definición 3.2. El **Hamiltoniano** asociado al problema (3.2) se define como:

$$H(x, u, \psi) = \langle \psi(t), f(x, u) \rangle + L(x, u)$$

donde $\langle \cdot, \cdot \rangle$ es el producto interno usual y a la variable $\psi(t)$ se le conoce como **adjunta** o de **coestado**.

Teorema 3.3 (Principio del Máximo de Pontryagin)**.** *Sean $x^*(t)$ y $u^*(t)$ las trayectorias óptimas de estado y control, respectivamente, del problema (3.2). Entonces existe $\psi^*(t)$ tal que:*

- $\dot{\psi}^* = -\frac{\partial H(x^*, u^*, \psi^*)}{\partial x}$ *con* $\psi^*(t_f) = \frac{\partial G(x^*(t_f))}{\partial x}$. *La ecuación anterior se conoce como ecuación adjunta.*
- $\dot{x}^*(t) = f(x^*(t), u^*(t))$.
- $u^*(t)$ *optimiza el Hamiltoniano, es decir, cuando se quiere maximizar el funcional se cumple que* $H(x^*, u^*, \psi^*) \geq H(x^*, u, \psi^*)$ *y cuando se quiere minimizar se tiene que* $H(x^*, u^*, \psi^*) \leq H(x^*, u, \psi^*)$. *Además* $H(x^*(t), u^*(t), \psi^*(t))$ *es constante en el dominio de t.*

La demostración rigurosa de este teorema se encontra en [28]. Utilizando métodos de cálculo variacional, en [34], está la demostración del teorema para el caso particular cuando las variables de estado y control son unidimensionales.

Es importante mencionar que el principio del máximo de Pontryagin proporciona condiciones que deben cumplir las trayectorias para la optimización, en caso de que existan trayectoriás óptimas, no establece ninguna información acerca de la existencia o no de dichas trayectorias. El problema de la existencia no se aborda en este trabajo, se encontran resultados sobre esto en [16, 23]. El PMP solo ayuda a identificar los posibles controles óptimos y cada uno de estos candidatos se debe analizar más a fondo para determinar si es realmente óptimo, estos controles se nombran **controles extremos**.

3.2. Interpretación geométrica

No se pretende dar una demostración del PMP, para esto ya se mencionó anteriormente dos referencias, la idea es mostrar la relación entre la optimización del funcional objetivo, el hamiltoniano y las consecuencias del PMP. Tal interpretación se obtuvo de [16] al igual que la figura (3.1), donde se supone que la variable de estados tiene dos dimensiones.

Para obtener un significado geométrico del PMP, es conveniente considerar el valor del funcional J como una nueva variable de estado. Entonces se define $\hat{x}(t) = (J(x(t), u(t)), x(t))$ y su vector velocidad es $\dot{\hat{x}}(t) = \hat{f}(x, u) = (L(x(t), u(t)), f(x(t), u(t)))$. Con esta representación el problema a optimizar (maximizar) se puede describir de la siguiente manera:

> *Sea D la línea que pasa a través de $(0, x_f)$ paralela al eje $\hat{x}_0$. Entre todas las trayectorias que comienzan en $\hat{x}_s = (0, x_s)$ y alcanzan D, se busca la que maximiza la primera coordenada x_0 del punto de intersección $x = (x_0, x_f)$ con D.*

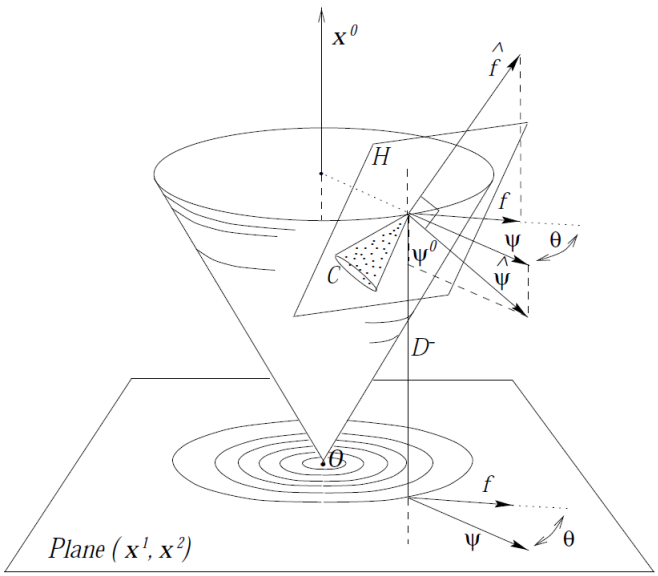


Figura 3.1: Imagen extraida de [16]. Interpretación geométrica del PMP.

Por el principio de optimalidad, si $u^*(t)$ es la trayectoria de control óptima, entonces en todo momento t debe ocurrir que $\hat{x}(t)$, la trayectoria de estado asociada a $u^*(t)$, esta en $\partial\mathcal{R}(\hat{x}_s, t)$, donde $\mathcal{R}(\hat{x}_s, t)$ es el área alcanzable desde $\hat{x}_s$ en el tiempo t.

Considere otras trayectorias de control $\hat{u}(t)$, tales que difieren de $u^*(t)$ en pequeños intervalos de tiempo y están infinitesimalmente cerca de $u^*(t)$. Luego el conjunto de puntos $x(t, \hat{u}(t))$ alcanzados por esas trayectorias "perturbadas", constituye un cono C con vértice en $x(t, u^*(t))$, que esta contenido en $\mathcal{R}(x_s, t)$. Este cono se aproxima localmente al conjunto $\mathcal{R}(x_s, t)$ y no intersecta la semirrecta D^- que comienza en $x(t, u^*(t))$ en la dirección de disminución de x_0, de no ser así, contradice la optimalidad de $u^*(t)$ (ver figura 3.1).

Luego es posible encontrar un hiperplano H tangente a C en $x(t, u^*(t))$, separando C y la semirrecta D^-, y que contiene el vector $\hat{f}(x(t), u^*(t))$ tangente a la trayectoria $\hat{x}(t)$ para todo t. Luego, como el vector adjunto $\hat{\psi} = (\psi_0, \psi^*)$ está definido como el que optimiza y hace constante el Hamiltoniano, para la trayectoria óptima en cada instante, entonces $\hat{\psi}$ es ortogonal al hiperplano H en cada momento. Siguiendo el principio de evolución óptima, la proyección del vector $\hat{f}(x(t), u^*(t))$ en la línea paralela a $\hat{\psi}(t)$, pasando a través de $x(t, u^*(t))$, debe ser máxima para el control u.

3.3. Aplicación del PMP

Veamos con un ejemplo como se puede utilizar el PMP para resolver un problema de control óptimo como (3.2). Se considera el siguiente ejercicio:

$$\begin{aligned} \max_u \quad & \int_1^5 (u(t)x(t) - u(t)^2 - x(t)^2)\,\mathrm{d}t \\ \text{sujeto a : } & \dot{x} = x + u \\ \text{con : } & x(1) = 2 \end{aligned}$$

Identificando lo anterior con lo definido en la sección (3.1), se tiene que:

$$L(x(t), u(t)) = u(t)x(t) - u(t)^2 - x(t)^2, \qquad G(x(5)) = 0, \qquad f(x, u) = x + u$$

entonces $H(x, u, \psi) = \psi(x + u) + ux - u^2 - x^2$. Luego, por consecuencia del PMP:

$$\dot{\psi}^* = -\frac{\partial H(x^*, u^*, \psi^*)}{\partial x} = 2x^* - u^* - \psi^* \ \text{ con } \ \psi^*(5) = \frac{\partial G(x^*(5))}{\partial x} = 0 \tag{3.3}$$

$$\dot{x}^* = x^* + u^* \quad \text{con} \quad x^*(1) = 2 \tag{3.4}$$

y u^* maximiza el hamiltoniano, por tanto

$$\begin{aligned} 0 =& \frac{\partial H(x^*, u^*, \psi^*)}{\partial u} = \psi^* + x^* - 2u^* \\ u^* =& \frac{\psi^* + x^*}{2} \end{aligned} \tag{3.5}$$

con el criterio de la segunda derivada se verifica que es un máximo. Remplazando (3.5) en (3.3) y (3.4) se tiene que:

$$\dot{\psi}^* = \frac{3}{2}x^* - \frac{3}{2}\psi^* \quad \text{con} \quad \psi^*(5) = 0 \tag{3.6}$$

$$\dot{x}^* = \frac{3}{2}x^* + \frac{1}{2}\psi^* \quad \text{con} \quad x^*(1) = 2 \tag{3.7}$$

De (3.7) se sigue que

$$\begin{aligned} \psi^* &= 2\dot{x}^* - 3x^* \\ \dot{\psi}^* &= 2\ddot{x}^* - 3\dot{x}^* \end{aligned} \tag{3.8}$$

sustituyendo las anteriores ecuaciones en (3.6) se tiene que

$$\ddot{x}^* - 3x^* = 0$$

entonces $x^* = Ae^{\sqrt{3}t} + Be^{-\sqrt{3}t}$. Por tanto, con la ecuación (3.8), se puede encontrar ψ^*; con las condiciones de (3.6) y (3.7) se pueden encontrar las constantes A y B; por último con la ecuación (3.5) se encuentra el control óptimo u^*.

A partir de lo anterior se podría "mecanizar", en el sentido de seguir unas instrucciones, la aplicación del PMP.

1. Identificar los elementos del problema para definir el hamiltoniano.

2. Optimizar el hamiltoniano para encontrar una expresión del control óptimo.

3. Resolver el sistema de ecuaciones del PMP, es decir, la ecuación adjunta y la ecuación de transición de estado, para encontrar las trayectoria óptimas.

3.4. Controles extremos del DDR

En esta sección se utiliza el PMP en el sitema dado por un DDR controlado por las aceleraciones en sus ruedas (ejemplo 2.2), para encontrar los controles extremos. Se quiere minimizar el tiempo del movimiento del robot entre dos estados, con $z = (x, y, \theta, v_r, v_l)$, donde el estado inicial es $z_s = (0, 0, 0, 0, 0)$ y el estado final z_f. Se define el funcional objetivo como:

$$J(u) = \int_{t_s}^{t_f} \mathrm{d}t = t_f - t_s$$

y según la notacion de la sección (3.1) se tiene que $L(z, u) = 1$ y $G(z(t_f)) = 0$. Entonces, con la ecuación de transición de estado (2.2) que define a $f(z, u)$, se tiene que el Hamiltoniano del sistema es:

$$H(z, u, \psi) = \psi_1 \left(\frac{v_r + v_l}{2}\right) r\cos\theta + \psi_2 \left(\frac{v_r + v_l}{2}\right) r\sin\theta + \psi_3 \left(\frac{v_r - v_l}{2b} r\right) + \psi_4 a_r + \psi_5 a_l + 1$$

donde $u = (a_r, a_l)$ y $\psi = (\psi_1, \psi_2, \psi_3, \psi_4, \psi_5)$ son las variables control y adjunta, respectivamente. Suponga que $z^*(t)$ y $u^*(t)$ son trayectorias que minimizan el funcional objetivo, entonces aplicando el PMP se tiene la existencia de $\psi^*(t)$, tal que:

$$\frac{\partial \psi^*}{\partial t}(t) = -\frac{\partial H(z^*(t), u^*(t), \psi^*(t))}{\partial z} \quad \text{con} \quad \psi^*(t_f) = 0 \tag{3.9}$$

$$\frac{\partial z^*}{\partial t}(t) = f(z^*(t), u^*(t)) \tag{3.10}$$

$$H(x^*, u^*, \psi^*) \leq H(x^*, u, \psi^*) \qquad \forall u \neq u^* \tag{3.11}$$

Por la definición del hamiltoniano y que $u^* = (a_r^*, a_l^*)$ lo minimiza, está claro que cuando las variable ψ_4^* y ψ_5^* son ambas distintas de cero, el control extremo es **saturado** (la magnitud es máxima) y los signos de las coordenadas depende de ψ_4^* y ψ_5^* de la siguiente manera:

$$a_r^* = -sing(\psi_4^*)a \qquad a_l^* = -sing(\psi_5^*)a \tag{3.12}$$

A continuación se prueba que cuando ψ_4^* o ψ_5^* se anulan en un intervalo no degenerado de tiempo, necesariamente los controles extremos son saturados.

De la ecuación adjunta (3.10) se tiene que:

$$\dot{\psi}_1^* = -\frac{\partial H}{\partial x} = 0,\ \dot{\psi}_2^* = -\frac{\partial H}{\partial y} = 0,\ \dot{\psi}_3^* = -\frac{\partial H}{\partial \theta},\ \dot{\psi}_4^* = -\frac{\partial H}{\partial v_r},\ \dot{\psi}_5^* = -\frac{\partial H}{\partial v_l} \tag{3.13}$$

Sin perdida de generalidad se supone que la función $\psi_4^*(t)$ se anula en un intervalo de tiempo distinto de cero, entonces:

$$\dot{\psi}_4^* = -\frac{\partial H}{\partial v_r} = -\frac{\psi_1^*}{2} r\cos\theta - \frac{\psi_2^*}{2} r\sin\theta - \frac{\psi_3^*}{2b} r = 0$$

$$\therefore \qquad \psi_3^* = -b(\psi_1^*\cos\theta + \psi_2^*\sin\theta)$$

al derivar ψ_3^* se tiene que:

$$\dot{\psi}_3^* = b(\psi_1^*\sin\theta - \psi_2^*\cos\theta)\dot{\theta}$$

Por la ecuación de transición de estados:

$$\dot{\psi}_3^* = \frac{v_r - v_l}{2} r(\psi_1^*\sin\theta - \psi_2^*\cos\theta) \tag{3.14}$$

Además, de la ecuación adjunta se sigue que:

$$\dot{\psi}_3^* = -\frac{\partial H}{\partial \theta} = \frac{v_r + v_l}{2} r(\psi_1^*\sin\theta - \psi_2^*\cos\theta) \tag{3.15}$$

Igualando las ecuaciones (3.14) y (3.15) se obtiene:

$$v_l(\psi_1^*\sin\theta - \psi_2^*\cos\theta) = 0$$

Se tiene entonces dos casos:

1. $v_l = 0$.

2. $\psi_1^*\sin\theta - \psi_2^*\cos\theta = 0$.

El primer caso correponde a una rotación en sitio, que depende de v_r, ya que el punto de contacto de la rueda izquierda con el piso no está en movimiento. El segundo caso, dado que ψ_1^* y ψ_2^* son constantes, consecuencia de la ecuación (3.13), entonces θ es constante y por tanto corresponde a moverse en linea recta. En los dos casos, una condición necesaria para que el movimiento sea óptimo en tiempo es que la aceleración de las ruedas tengan magnitud máxima. Utilizando razonamiento análogo en el caso que $\psi_5^*(t)$ se anule en un intervalo no degenerado de tiempo, se demuestra que los controles extremos son saturado.

En conclusión, la trayectoria de control óptima $u^*(t)$ toma valores solo en el conjunto $\mathbb{U}_1 = \{-a, a\} \times \{-a, a\}$ y al cambiar entre estos valores genera discontinuidades, que en este caso se conocen como **interruptores**, ya que estas características definen a $u^*(t)$ como **bang-bang** [23].

El conjunto $\mathbb{U}_1$ es el de controles extremos para el sistema DDR controlado por las aceleraciones en sus ruedas.

Dado que las trayectorias óptimas se obtienen con $|a_r| = |a_l| = a$, se tiene dos tipos de curvas:

1. Cuando $a_r = -a_l = \pm a$ se tiene que la aceleración lineal del robot es nula, entonces $v(t) = v_0$, cuando $v_0 = 0$ se produce la rotación pura. Se sigue que la longitud del recorrido es $s(t) = v_0 t$ y las ecuaciones de las velocidades son:

$$v_r(t) = \pm at + v_{r0} \qquad v_l(t) = \mp at + v_{l0} \qquad \omega(t) = \pm\frac{a}{b}t + \omega_0.$$

   por tanto la curvatura de la trayectoria es:

$$\kappa(t) = \frac{\omega(t)}{v(t)} = \pm\frac{a}{bv_0^2}s(t) + \frac{\omega_0}{v_0}$$

   entonces la curvatura en cualquier punto es proporcional a la distancia a lo largo de la curva medida desde el origen, esto define una clotoide. Bajo condiciones para el estado inicial [16], se puede encontrar la parametrización de la curva con integrales de Fresnel, así se obtiene la figura (3.2).

2. Cuando $a_r = a_l = \pm a$ se tiene que la aceleración angular del robot es nula, entonces $\omega(t) = \omega_0$ y $\theta(t) = \omega_0 t + \theta_0$, cuando $\omega_0 = 0$ se mueve en linea recta. Además, la aceleración lineal es $\dot{v}(t) = sign(a_r)a$ y $v(t) = sign(a_r)at + v_0$. El radio de curvatura de la trayectoria es:

$$\rho(t) = \frac{v(t)}{\omega(t)} = sign(a_r)\frac{a}{\omega_0^2}(\theta(t) - \theta_0) + \frac{v_0}{\omega_0}.$$

   tal radio de curvatura define una involuta de un círculo. Análogamente al caso anterior, bajo condiciones para el estado inicial, se puede encontrar la parametrización de la curva [16], así se obtiene la figura (3.3) involuta de círculo:

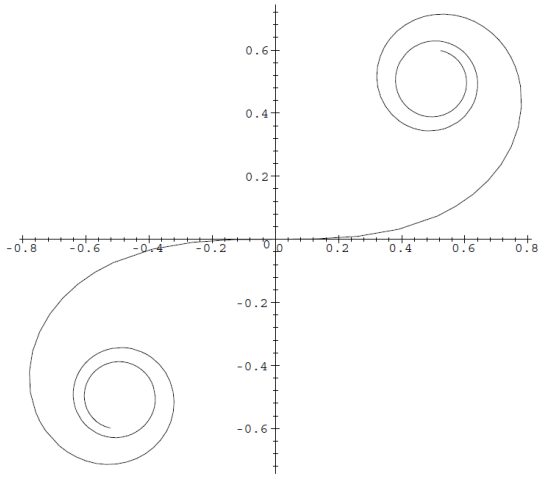


Figura 3.2: Imagen extraida de [16]. Clotoide obtenida con $a_r = -a_l = a$

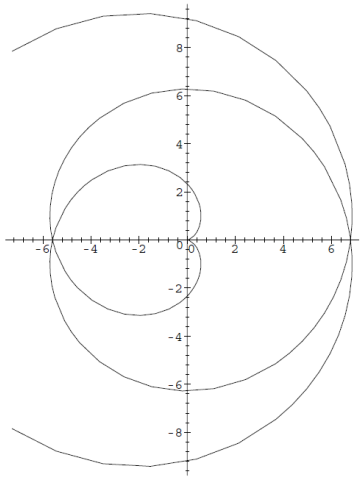


Figura 3.3: Imagen extraida de [16]. Involuta de círculo obtenida con $a_r = a_l = a$

CAPÍTULO 4

Aplicación del SST y SST* al sistema DDR con controles extremos

Los algoritmos SST y SST* no requieren solucionar un BVP para lograr la optimalidad asintótica, convirtiéndolos en buenos candidatos para resolver cuestiones de planificación de movimiento para el sistema dado por un DDR controlado por las aceleraciones en sus ruedas, ya que para dicho sistema *no se conoce la solución a un BVP*. Además, como se vio en el capítulo 3, para dicho DDR se sabe que los controles óptimos en tiempo están contenidos en $\{-a, a\} \times \{-a, a\}$, lo cual reduce a cuatro la cantidad de controles a considerar en la implementación del SST y SST*. En ese sentido se puede ver la ventaja o utilidad de la integración de resultados encontrados con fuertes herramientas teóricas, como lo es el PMP, con métodos más practicos y modernos, como lo son los planificadores basados en muestreo.

La optimalidad en los algoritmos SST y SST* se logró con la definición de un nuevo marco de trabajo y varias hipótesis que debe cumplir el sistema dinámico [22]. Por tanto, para justificar la aplicación del SST y SST* al sistema DDR con controles extremos, se muestra que el problema de planificación a tratar cumple o se adapta a dicho marco de trabajo.

4.1. Introducción

En esta sección se describe el marco de trabajo en el cual se basan los algoritmos SST y SST*. Además, la adaptación de los algoritmos al sistema DDR con dinámica de segundo orden y con controles extremos a dicho marco de trabajo y el cumplimiento de las hipótesis. La siguiente definición es el concepto principal en el que se basa la teoria que justifica la optimalidad de los algorimos SST y SST*.

Definición 4.1 (Trayectorias δ-similares)**.** Las trayectorias π y π' son δ-similares si para una función de escala, continua y no decreciente $\sigma : [0, t_\pi] \to [0, t_{\pi'}]$ se cumple que $\pi'(\sigma(t)) \in \mathcal{B}_\delta(\pi(t))$, donde $\mathcal{B}_\delta(\pi(t))$ es la bola abierta centrada en $\pi(t)$ de radio δ.

Intuitivamente dos trayectorias son δ-similares cuando al hacer un recorrido simultaneo por ambas trayectorias (uso de la función σ), los puntos no se alejan una distancia mayor a δ. La función $\sigma : [0, t_\pi] \to [0, t_{\pi'}]$ es la típica biyección entre dos intervalos, la cual en este caso tiene la siguiente forma:

$$\sigma(x) = \frac{t_{\pi'}}{t_\pi} x.$$

Hipótesis 4.2. El sistema descrito por (2.1) satisface las siguientes propiedades:

i) Condición de Chow sobre accesibilidad local en pequeño tiempo, es decir, se cumple que el conjuto de estados alcanzables $R(x, T) \subset V$, desde cualquier estado x en un tiempo menor o igual a T, sin salirse de una vecindad $V \subset \mathbb{X}$ de x, tiene la misma dimensión del espacio de estados del sistema $\mathbb{X}$.

ii) Tiene segunda derivada acotada, es decir, existe $M \in \mathbb{R}$ tal que $|\ddot{x}(t)| \leq M$.

Veamos que el DDR con controles extremos cumple la hipótesis i). Para $z = (x, y, \theta, v_r, v_l)$ se definen los campos vectoriales

$$h_0(z) = \begin{pmatrix} \frac{v_r+v_l}{2} r\cos\theta \\ \frac{v_r+v_l}{2} r\sin\theta \\ \frac{v_r-v_l}{2b} r \\ 0 \\ 0 \end{pmatrix}, \qquad h_1(z) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \qquad h_2(z) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

entonces el sistema (2.2) es igual a

$$(\dot{x}, \dot{y}, \dot{\theta}, \dot{v_r}, \dot{v_l}) = h_0(z) + a_l h_1(z) + a_r h_2(z).$$

Veamos que el algebra de Lie $\mathcal{L}(\Delta)$, de la distribución $\Delta = \{h_0, h_1, h_2\}$, tiene dimensión 5 [20]. Se completan los campos vectoriales para una base a partir del corchete de Lie $[\cdot, \cdot]$. Se definen los siguientes campos vectoriales:

$$h_3(z) = [h_1, h_0], \qquad h_4(z) = [h_2, h_0], \qquad h_5(z) = [h_2, [h_0, h_4]]$$

$$h_3(z) = \begin{pmatrix} \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta \\ -\frac{r}{2b} \\ 0 \\ 0 \end{pmatrix}, \qquad h_4(z) = \begin{pmatrix} \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta \\ \frac{r}{2b} \\ 0 \\ 0 \end{pmatrix}, \qquad h_5(z) = \begin{pmatrix} \frac{r^2}{2b}\sin\theta \\ -\frac{r^2}{2b}\cos\theta \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

El determinante de la matriz $[h_1\ h_2\ h_3\ h_4\ h_5]$, que tiene las columnas a los h_i, es igual a $-\frac{r^2}{4b^2}$. Como r y b son distintos a cero, entonces h_1, h_2, h_3, h_4 y h_5 son linealmente independientes y por tanto el algebra de Lie $\mathcal{L}(\Delta)$ tiene dimensión 5, por tanto cumple la condición de Chow descrita en [14] y esto implica que el sistema localmente accesible en pequeño tiempo [32], es decir, cumple la hipótesis i).

Dado que el algoritmo sólo genera funciones de control constantes por partes, el supuesto de la hipótesis (4.7) y la discretización del espacio de control, entonces carece de sentido las derivadas de las variables de control. Por tanto, para la validez de las hipótesis ii) sólo se consideran las variables de estado (x, y, θ), entonces sus primeras derivadas se expresan en

el siguiente sistema:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{v_r+v_l}{2} r\cos\theta \\ \frac{v_r+v_l}{2} r\sin\theta \\ \frac{v_r-v_l}{2b} r \end{pmatrix} \tag{4.1}$$

De la ecuación (4.1) se sigue que:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{a_r+a_l}{2} r\cos(\theta) - \frac{v_r+v_l}{2} r\sin(\theta)\dot{\theta} \\ \frac{a_r+a_l}{2} r\sin(\theta) + \frac{v_r+v_l}{2} r\cos(\theta)\dot{\theta} \\ \frac{a_r-a_l}{2b} r \end{pmatrix} \tag{4.2}$$

Utilizando la norma infinito se tiene que:

$$\begin{aligned} ||(\ddot{x},\ddot{y},\ddot{\theta})||_\infty &= \text{máx}\left\{ \begin{array}{c} |\frac{a_r+a_l}{2} r\cos(\theta) - \frac{v_r+v_l}{2} r\sin(\theta)\dot{\theta}|, \\ |\frac{a_r+a_l}{2} r\sin(\theta) + \frac{v_r+v_l}{2} r\cos(\theta)\dot{\theta}|, \\ |\frac{a_r-a_l}{2b} r| \end{array} \right\} \\ &\leq \text{máx}\left\{ \begin{array}{c} |\frac{a_r+a_l}{2} r\cos(\theta)| + |\frac{v_r+v_l}{2} r\sin(\theta)\dot{\theta}|, \\ |\frac{a_r+a_l}{2} r\sin(\theta)| + |\frac{v_r+v_l}{2} r\cos(\theta)\dot{\theta}|, \\ |\frac{a_r-a_l}{2b} r| \end{array} \right\} \end{aligned}$$

dado que la magnitud de las aceleraciones y las velocidades están acotadas por a y v, respectivamente, se cumple lo siguiente:

$$\begin{aligned} ||(\ddot{x},\ddot{y},\ddot{\theta})||_\infty &\leq \text{máx}\{a\cdot r + v\cdot r\cdot|\dot{\theta}|, a\cdot r + v\cdot r\cdot|\dot{\theta}|, \frac{a}{b}r\} \\ &\leq \text{máx}\{a\cdot r + v\cdot r\cdot\frac{v}{b}r, \frac{a}{b}r\} \\ &= a\cdot r\cdot\text{máx}\{1+\frac{v^2}{a\cdot b}r, \frac{1}{b}\} := M. \end{aligned}$$

Lo que demuestra que el sistema (4.1) cumple con la hipótesis ii). Se utilizó la norma infinito por facilidad de cálculo, pero el resultado es valido para cualquier norma, ya que en espacios de dimesión finita todas las normas son equivalentes [24].

Hipótesis 4.3. La función f del sistema (2.1) es lipschitz para los dos argumentos, es decir, existen $K_x, K_u \geq 0$ tales que:

$$||f(x_0, u_0) - f(x_1, u_0)|| \leq K_x||x_0 - x_1||$$

$$||f(x_0, u_0) - f(x_0, u_1)|| \leq K_u||u_0 - u_1||.$$

Sea $(f_1, f_2, f_3, f_4, f_5)$ la representación en funciones coordenadas de la función f del sistema dado por el DDR (2.2) que corresponde a la representación en (2.1), por tanto:

$$f_1(x, u) = \frac{v_r + v_l}{2} r\cos\theta, \quad f_2(x, u) = \frac{v_r + v_l}{2} r\sin\theta, \quad f_3(x, u) = \frac{v_r - v_l}{2b} r$$

$$f_4(x, u) = a_r, \quad f_5(x, u) = a_l.$$

Sean $\overline{x}_0, \overline{x}_1 \in \mathbb{X}$ y $u_0, u_1 \in \mathbb{U}$ con $\overline{x}_0 = (x_0, y_0, \theta_0, v_{0r}, v_{0l}), \overline{x}_1 = (x_1, y_1, \theta_1, v_{1r}, v_{1l}), u_0 = (a_{0r}, a_{0l})$ y $u_1 = (a_{1r}, a_{1l})$. Veamos que f es lipschitz en la variables de estado:

$$||f(\overline{x}_0, u_0) - f(\overline{x}_1, u_0)|| = \left\| \begin{pmatrix} f_1(\overline{x}_0, u_0) - f_1(\overline{x}_1, u_0) \\ f_2(\overline{x}_0, u_0) - f_2(\overline{x}_1, u_0) \\ f_3(\overline{x}_0, u_0) - f_3(\overline{x}_1, u_0) \\ 0 \\ 0 \end{pmatrix} \right\|$$

$$= \left\| \begin{pmatrix} f_1(\overline{x}_0, u_0) \\ f_2(\overline{x}_0, u_0) \\ f_3(\overline{x}_0, u_0) \end{pmatrix} - \begin{pmatrix} f_1(\overline{x}_1, u_0) \\ f_2(\overline{x}_1, u_0) \\ f_3(\overline{x}_1, u_0) \end{pmatrix} \right\|$$

La última igualdad expresa que f en la variable de estado es lipschitz si y sólo si la función

$$g(\overline{x}) = \begin{pmatrix} f_1(\overline{x}, u_0) \\ f_2(\overline{x}, u_0) \\ f_3(\overline{x}, u_0) \end{pmatrix} = \begin{pmatrix} \frac{v_r+v_l}{2} r\cos\theta \\ \frac{v_r+v_l}{2} r\sin\theta \\ \frac{v_r-v_l}{2b} r \end{pmatrix}$$

es lipschitz. La función g coincide con la parte derecha de la ecuación (4.1). El desarrollo que le sigue a la ecuación (4.1) demuestra que la función g tiene derivada acotada, lo cual

implica que g, y por tanto f en la variable de estado, es lipschitz (corolario 1 de la sección 5 del capítulo 5 de [24]).

Para el cumplimiento de la hipótesis (4.3) falta probar que es lipschitz en las varialbles de control, lo cual se demuestra a continuación:

$$\begin{aligned} ||f(\overline{x}_0, u_0) - f(\overline{x}_0, u_1)|| &= ||(0, 0, 0, a_{0r} - a_{1r}, a_{0l} - a_{1l})|| \\ &= ||(a_{0r}, a_{0l}) - (a_{1r}, a_{1l})|| \\ &= ||u_0 - u_1|| \end{aligned}$$

más aún se sigue que $K_u = 1$.

El concepto de *despeje dinámico* (Teorema 4.4) es fundamental en los sistemas en que se aplican los métodos basados en muestreo, ya que el hecho de querer encontrar una trayectoria en específico es desacertado, dado que tiene probabilidad cero de ocurrir. Por tanto los planificadores se centran en encontrar trayectorias de alguna manera "similares" a la trayectoria objetivo, en este caso, trayectorias δ-similares.

Teorema 4.4. *Sea π una trayectoria para un sistema como (2.1) que satisface la condicion de Chow sobre accesibilidad local en pequeño tiempo. Entonces existe $\delta_0 > 0$, llamado el* ***despeje dinámico****, tal que $\forall \delta \in (0, \delta_0], \forall {x'}_0 \in \mathcal{B}_\delta(\pi(0))$ y $\forall {x'}_1 \in \mathcal{B}_\delta(\pi(t_\pi))$, existe una trayectoria π' tal que $\pi'(0) = {x'}_0$, $\pi'(t'_\pi) = {x'}_1$ y π y π' son trayectorias δ similares.*

Demostración. Apéndice A en [22]. □

Definición 4.5.

- El **despeje de obtáculos** ϵ de una trayectoria π es la mínima distancia desde los obstáculos a todos los estados en π, es decir, $\epsilon = \text{ínf}_{t \in [0, t_\pi], x_o \in \mathbb{X}_o} ||\pi(t) - x_o||$, donde $\mathbb{X}_o = \mathbb{X} \backslash \mathbb{X}_f$.
- Una trayectoria para un sistema dinámico que sigue la ecuación (2.1), es llamada δ **robusta** si su despeje de obstáculos y despeje dinámico son mayores que δ.

A partir de las definiciones anteriores se puede definir el problema que trata de resolver este marco de trabajo.

Definición 4.6. Dado un sistema dinámico que sigue la ecuación (2.1), el subconjunto libre de colisiones $\mathbb{X}_f \subseteq \mathbb{X}$, un estado inicial $x_0 \in \mathbb{X}_f$, una región objetivo $\mathbb{X}_G \subseteq \mathbb{X}$ y que existe una trayectoria δ robusta que conecta x_0 con un estado en $\mathbb{X}_G$, *la planificación de movimiento factible δ robusta* encuentra una trayectoria solución π para la cual $\pi(0) = x_0$ y $\pi(t_\pi) \in \mathbb{X}_G$.

Note que la definición anterior no pide que la trayectoria solución sea δ-similar a alguna trayectoria, pero es necesario que exista una trayectoria δ robusta que sea solución al problema de planificación. Además, dado que los planificadores basados en muestreo se implementan de modo que la función de control es constante por partes, es necesario hacer la siguiente hipótesis.

Hipótesis 4.7. Para un problema de planificación de movimiento factible δ robusto, existe una trayectoria δ robusta π generada por una función de control constante por partes Ψ.

Como se puede notar, la base del marco de trabajo son las trayectorias δ robustas, entonces propiedades deseables en planificadores de movimineto se deberían redefinir en este enfoque. Por tanto, la siguiente definición relaja y adapta el concepto de completitud probabilística para los algoritmos con propiedades que dependen del concepto de trayectorias δ robusta.

Definición 4.8 (Completitud probabilistica δ-robusta)**.** Sea $\prod_n^{ALG}$ el conjunto de las trayectorias obtenidas por un algoritmo basado en muestreo en la iteración n. Un algoritmo basado en muestreo es probabilísticamente δ-robusto completo, si para cualquier problema de planificación de movimiento factible δ robusto $(\mathbb{X}_f, x_0, \mathbb{X}_G, \delta)$ se tiene que:

$$\liminf_{n\to\infty} \mathbb{P}(\exists \pi \in \prod_{n}^{ALG} \ : \ \pi \text{ es solución de } (\mathbb{X}_f, x_0, \mathbb{X}_G, \delta)) = 1$$

Un algoritmo es probabilísticamente δ robusto completo cuando eventualmente encuentra trayectorias de solución, en el caso de que exista una trayectoria solución δ robusta.

De igual manera, se debe adaptar el concepto de optimalidad asintótica para los algoritmos con propiedades que dependen del concepto de trayectorias δ robusta.

Definición 4.9 (Asintoticidad δ-robusta casi óptima)**.** Sea c^* el costo mínimo de todas las trayectorias solución para un problema de planificación de movimiento factible δ robusto

$(\mathbb{X}_f, x_0, \mathbb{X}_G, \delta)$. Sea Y_n^{ALG} la variable aleatoria que representa costo mínimo de las trayectorias en $\prod_n^{ALG}$. El algoritmo es asintóticamente δ robusto casi óptimo si

$$\mathbb{P}(\limsup_{n\to\infty} Y_n^{ALG} \leq h(c^*, \delta)) = 1$$

donde $h\colon \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ es función del costo óptimo y el despeje dinámico δ tal que $h(c^*, \delta) \geq c^*$.

La propiedad en la definición de la función h garantiza que el costo de la solución encontrada tiene una cota superior relacionada con el costo óptimo. En los algoritmos propuestos en [22] se exhibe la propiedad anterior donde h tiene la forma: $h(c^*, \delta) = (1 + \alpha \cdot \delta) \cdot c^*$ para alguna constante $\alpha \geq 0$.

Si se argumenta que un algoritmo satisface las definiciones (4.8) y (4.9) para todos los valores decrecientes de δ, entonces este algoritmo satisface las propiedades tradicionales de completitud probabilística y optimalidad asintótica [22].

4.2. Completitud y optimalidad asintótica

Está demostrado que en un sistema que cumple el marco de trabajo definido en la sección anterior, el algoritmo SST es probabilísticamente δ robusto completo y asintóticamente δ robusto casi óptimo; y el algoritmo SST* es completo y asintóticamente óptimo [22]. Propiedades deseables para los algoritmos que se implementan en este trabajo, los cuales difieren con el enfoque clásico, como ya se ha mencionado, que en la propagación no se considera todo el conjunto de controles, solamente los obtenido con el PMP, es decir:

$$\mathbb{U} = [-a, a] \times [-a, a] \quad \Longrightarrow \quad \mathbb{U} = \{-a, a\} \times \{-a, a\}. \tag{4.3}$$

Por tanto, al demostrar que el método de propagación, con esa modificación, sigue teniendo los atributos requeridos para obtener las propiedades deseables de los algoritmos, se obtiene el cumplimiento de las mismas propiedades en los algoritmos implementados en este trabajo.

Se requiere que el método de propagación cumpla el Teorema 17 de [22], el cual garantiza que la probabilidad de que el método de propagación genere una trayectoria δ similar

a otra es distinta de cero. Esta propiedad muestra que la propagación es válida para su uso en un planificador de movimiento asintóticamente óptimo. Antes de enunciar y demostrar el resultado análogo a dicho teorema, el cual es el 4.11, se necesita el siguiente lema.

Lema 4.10. *El cumplimiento de la hipótesis (4.3) implica que para dos trayectorias π, π' y $T \geq 0$ tal que $\pi(0) = \pi'(0) = x_0$ y $\Delta u = \sup_t\{||u(t) - u'(t)||\}$, se cumple que:*

$$||\pi(T) - \pi'(T)|| < K_u \cdot T \cdot e^{K_x \cdot T} \cdot \Delta u$$

Demostración. Apéndice B en [22]. □

Teorema 4.11. *Suponga que el conjunto de control $\mathbb{U}$ de un sistema, que satisface las hipótesis (4.2) y (4.3), es finito y tiene N elementos. Sea π una trayectoria generada por una función de control constante por partes Ψ. La probabilidad de éxito para que la propagación genere una trayectoria π' δ-similar a π, cuando se ejecuta desde el estado $\pi'(0) = \pi(0)$, está acotada inferiormente por un valor positivo $\rho_\delta > 0$.*

Demostración. Suponga que π es concatenación de n trayectorias $\pi_1, \pi_2, \ldots, \pi_n$ las cuales son generadas por las funciones de control constantes que concatenadas generan a Ψ, los controles que se aplican son $u_1, u_2, \ldots, u_n$ y con duraciones $T_1, T_2, \ldots, T_n$, respectivamente, y son menores a T_{prop}, el tiempo máximo en la propagación. Además, que π' es generada por los controles $u'_1, u'_2, \ldots, u'_n$ con duraciones $T'_1, T'_2, \ldots, T'_n$, respectivamente. Sea $\lambda \in (0, 1)$, se define los siguientes conjuntos para un estado x que pertenece a la trayectoria π, es decir, $x \in Imagen(\pi)$:

$$\Omega_\lambda(x) = \{z \in Imagen(\pi) \mid \mathcal{B}_{\lambda\delta}(z) \subset \mathcal{B}_\delta(x)\}$$

$$\Delta_\lambda(x) = \bigcup_{z \in \Omega_\lambda(x)} \mathcal{B}_{\lambda\delta}(z)$$

Note que $\Omega_\lambda(x) \subset \Delta_\lambda(x) \cap Imagen(\pi)$ y $\Delta_\lambda(x) \subset \mathcal{B}_\delta(x)$. Considere el conjunto $\Omega_\lambda(\pi_1(T_1))$ el cual coincide con una trayectoria contenida en π. Sea $[t_1, \hat{t}_1]$ el intervalo de tiempo en el que π recorre dicha trayectoria. El Teorema (4.4) garantiza la existencia de una trayectoria δ similar a π_1 que tiene estado terminal en $\Delta_\lambda(\pi_1(T_1))$ y aplicando el lema (4.10) se tiene que

$$||\pi'(T'_1) - \pi_1(T'_1)|| < \lambda\delta \qquad \text{cuando} \qquad ||u'_1 - u_1|| < \frac{\lambda\delta}{K_u \cdot T'_1 \cdot e^{K_x \cdot T'_1}} \tag{4.4}$$

entonces para que π' sea δ similar a π_1 y más aún su estado terminal esté en $\Delta_\lambda(\pi_1(T_1))$ se debe cumplir que u_1' satisface la desigualdad (4.4) y $T_1' \in [t_1, \hat{t}_1]$. Por tanto la probabilidad para que el método de propagación desde $\pi'(0)$ genere una trayectoria δ similar a π_1 es

$$\frac{\hat{t}_1 - t_1}{T_{prop}} \cdot \frac{\mu_1}{|\mathbb{U}|} = \frac{\hat{t}_1 - t_1}{T_{prop}} \cdot \frac{\mu_1}{N}$$

donde μ_1 es el número de controles u_1' que cumplen la desigualdad (4.4). Como consecuencia del Teorema (4.4) existe una trayectoria δ similar a π_2 que empieza en $\pi'(T_1)$, por tanto siguiendo un proceso análogo desde $\pi'(T_1)$ se genera un trayectoria δ similar a $\pi_1|\pi_2$ con probabilidad

$$\frac{\hat{t}_1 - t_1}{T_{prop}} \cdot \frac{\mu_1}{N} \cdot \frac{\hat{t}_2 - t_2}{T_{prop}} \cdot \frac{\mu_2}{N} = \frac{(\hat{t}_1 - t_1)(\hat{t}_2 - t_2)}{N^2 T_{prop}^2} \cdot \mu_1\mu_2$$

continuando este proceso, se sigue que la probabilidad de que la propagación genere una trayectoria δ similar a π es

$$\frac{\prod_{i=1}^{n}(\hat{t}_i - t_i)}{N^n T_{prop}^n} \cdot \prod_{i=1}^{n} \mu_i \tag{4.5}$$

dado que $\mu_i \geq 1$ para todo i, una cota inferior para la probabilidad es

$$\rho_\delta = \frac{\prod_{i=1}^{n}(\hat{t}_i - t_i)}{N^n T_{prop}^n}. \tag{4.6}$$

□

Aplicando el anterior teorema al DDR con controles extremos, donde $N = 4$, se tiene que la probabilidad de que el método de propagación genere una trayecteria δ similar a alguna π, generada por una función de control constante por partes, está acotada inferiormente por

$$\frac{\prod_{i=1}^{n}(\hat{t}_i - t_i)}{4^n T_{prop}^n}. \tag{4.7}$$

En conclusión, dado que el sistema DDR con controles extremos cumple con el marco de trabajo y el teorema anterior garantiza que el cambio en el método de propagación (4.3) no afecta las propiedades requeridas por los algoritmos, entonces el SST aplicado a dicho sistema es probabilísticamente δ robusto completo y asintóticamente δ robusto casi óptimo; y el algoritmo SST* es completo y asintóticamente óptimo.

4.2.1. Mayor velocidad de convergencia

El Teorema 4.11 tiene la hipótesis de que el conjunto de control es finito, ya que se quería ver que el DDR con el subconjunto de controles extremos cumple el marco de trabajo para aplicar el SST, pero también es valido si se considera todo el conjunto de controles (Teorema 17 de [22]), en el caso del DDR es $\mathbb{U} = [-a, a] \times [-a, a]$. A partir de la demostración del Teorema 4.11 se puede argumentar, bajo algunas suposiciones, que es más probable que para un problema de planificación con el sistema DDR, se encuentre una solución más rápido si se restringe a los controles extremos.

Considerando todo el conjunto de controles en el DDR y siguiendo los mismo pasos, al igual que la terminología, de la demostración del Teorema 4.11, la ecuación (4.5) se transforma en:

$$\frac{\prod_{i=1}^{n}(\hat{t}_i - t_i)}{T_{prop}^{n}} \cdot \prod_{i=1}^{n} \frac{\tilde{\mu}_i}{|\mathbb{U}|} = \frac{\prod_{i=1}^{n}(\hat{t}_i - t_i)}{T_{prop}^{n}} \cdot \frac{\prod_{i=1}^{n} \tilde{\mu}_i}{(4a^2)^n} \tag{4.8}$$

donde el $4a^2$ que aparece en el denominador es la medida del conjunto de controles $[-a, a] \times [-a, a]$ y $\tilde{\mu}_i$ es la medida (área) del conjunto de controles u_i' que satisfacen la desigualdad:

$$||u_i' - u_i|| < \frac{\lambda\delta}{K_u \cdot T_i \cdot e^{K_x \cdot T_i}}. \tag{4.9}$$

Comparando la cota inferior de la probabilidad de que el método de propagación, en el DDR con controles extremos, genere una trayectoria δ similar a otra (fórmula 4.7), con la parte derecha de la ecuación (4.8), se observa que difieren en los términos

$$\frac{1}{4^n} \qquad \text{y} \qquad \frac{\prod_{i=1}^{n} \tilde{\mu}_i}{(4a^2)^n}$$

respectivamente. Para homogeneizar un poco los anteriores términos, se supone que $a = 1$. En este caso, la diferencia depende totalmente de $\prod_{i=1}^{n} \tilde{\mu}_i$, se da la igualdad cuando ese producto es igual a 1. Entonces, si $\prod_{i=1}^{n} \tilde{\mu}_i < 1$ conlleva a que el término (4.7) sea mayor y por tanto la cota inferior de la probabilidad de que la propagación genere una trayectoria δ similar a otra, en el sistema DDR con todo el conjunto de controles, es menor al sistema DDR con el conjunto de controles extremos. En términos prácticos, puede significar mayor

velocidad de convergencia en la ejecución del SST con los controles extremos, esto se puede apreciar en los resultados de la sección 4.5.3.

Para que $\prod_{i=1}^{n} \tilde{\mu}_i < 1$, es suficiente que cada $\tilde{\mu}_i$ sea menor que 1. El valor máximo de $\tilde{\mu}_i$ es la medida de la bola de radio $\frac{\lambda\delta}{K_u \cdot T_i \cdot e^{K_x \cdot T_i}}$ centrada en u_i de $\mathbb{R}^2$, en caso de que esté contenida en $[-a, a] \times [-a, a]$ (4.9). Entonces, $\tilde{\mu}_i < 1$ se cumple cuando:

$$\frac{\lambda\delta}{K_u \cdot T_i \cdot e^{K_x \cdot T_i}} < \frac{1}{\sqrt{\pi}} \tag{4.10}$$

ya que $\frac{1}{\sqrt{\pi}}$ es el radio de la bola unitaria de $\mathbb{R}^2$. La validez de la desigualdad (4.10) se puede justificar por cualquiera de los siguientes comentarios:

- En la sección 4.1 se probó que $K_u = 1$ y no se encontró un valor para K_x, pero se puede utilizar un valor suficientemente grande, condicionado a los valores de T_i de la trayectoria, tal que para todo i se satisfaga (4.10).

- El valor de δ es intrínseco del sistema y del espacio de trabajo, pero por la definición de una trayectoria δ robusta, es valido sustituir a δ por cualquier elemento de $(0, \delta]$. Por tanto, se puede considerar δ suficientemente pequeño para que se cumpla la desigualdad (4.10), para todo i. Análogamente, se puede considerar cualquier valor para λ en $(0, 1)$, ya que así se definió en la demostración del Teorema 4.11.

4.3. Metodología General

Parte del contenido y desarrollo de secciones anteriores se enfocan en el sistema DDR, ya que en este sistema está orientado el objetivo general de este trabajo, que se definió en la sección 2.1. Por otro lado, hay contenido no relacionado a un sistema en específico, por ejemplo, casi todo el capítulo 2 y el Teorema 4.11, esto permite generalizar y mecanizar la metodología que se desarrolló en el DDR para otros sistemas.

Es importante recordar que el principal aporte en esta metodología es la combinación de herramientas fuertes en la teoria de control, como lo es el PMP, en métodos más practicos y modernos, como son los algoritmos SST y SST*. La causalidad de esta combinación es la

finitud del conjunto de controles extremos del DDR, teniendo esto en cuenta a continuación se describe, en resumen, lo que debe cumplir o los pasos que se tienen que hacer para aplicar lo hecho en este trabajo a un sistema dinámico, además de sus consecuencias.

En primera instancia, la metodología se aplica a sistemas dinámicos considerando el conjunto de controles extremos, el cual debe ser de cardinalidad finita. Luego, suponiendo que (2.1) representa su ecuación de transición de estado, para dicho sistema se deben comprobar las siguientes tres propiedades:

- La condición de Chow sobre accesibilidad local en pequeño tiempo.
- Tiene segunda derivada acotada, es decir, existe $M \in \mathbb{R}$ tal que $|\ddot{x}(t)| \leq M$.
- La función de transición de estado del sistema es lipschitz para los dos argumentos, a saber, la variable de estados y la de controles.

Como consecuencia del cumplimiento de estas propiedades y el Teorema 4.11, al aplicar los algoritmos SST y SST* a dicho sistema muestreando sólo el conjunto de controles extremos, se cumplen sus propiedades sobre completitud y optimalidad asintótica. Finalmente, como resultado destacable, se tiene que la aplicación del SST* a un problema de planificación de movimiento, obtiene la trayectoria óptima en costo.

4.4. Implementación y Complejidad

Se menciona algunos aspectos de la implementación de los algoritmos SST y SST*, con las que se obtienen los datos de la siguiente sección, con el fin de obtener la complejidad de dichas implementaciones. Se analiza la complejidad de cada una de las partes del algoritmo (selección, propagación y poda), pero antes es conveniente hacer la siguiente definición.

Definición 4.12. Si la ejecución del sistema se emplea en un subconjunto acotado del espacio de estados, esto es lo usual, se detona por M_{δ_s} al máximo número de vertices que puede tener el árbol.

Note que M_{δ_s} está indizado por δ_s, ya que este parámetro es el que controla la dispersion del árbol y por tanto M_{δ_s} está completamente determinado por el valor de δ_s.

La complejidad se va a definir en términos del número de muestras N.

1. **Selección**: Cada vez que se aplica el proceso de selección se recorre el conjunto V_{active}, el cual puede crecer linealmente a medida que aumenta el número de muestras, hasta que la cantidad de vértices se aproxima a M_{δ_s}. Por tanto, el proceso de selección empieza con complejidad cuadrática y luego lineal. Suponiendo que N_m es la iteración donde la cardinalidad de V_{active} es igual a M_{δ_s}, entonces:

$$O(N) = \begin{cases} O(N^2) & \text{si } N \leq N_m \\ O(M_{\delta_s} N) & \text{si } N > N_m \end{cases} \tag{4.11}$$

   Por comodidad se utiliza la cota lineal para la complejidad del proceso de selección, es decir, $O(M_{\delta_s} N)$. Es claro que esta cota también es válida para el comienzo del algoritmo, por la definición de M_{δ_s}.

2. **Propagación**: El proceso de propagación utiliza los algoritmos (2.7) y (2.8). Para el primero se utiliza un método de integración númerica compuesta, más específicamente *la regla de Simpson de 3 subintervalos*, la cual hace una aproximación de la forma:

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3} \sum_{i=0, i=i+2}^{n-2} [f(x_i) + 4f(x_{i+1}) + f(x_{i+2})]$$

   donde h es la longitud de una partición regular del intervalo $[a, b]$ en n subintervalos [5]. La complejidad de este método depende de la partición del intervalo, y como en la implementación de utilizó una partición de cadinalidad constante, entonces esta parte de la propagación tiene orden constante, es decir, $O(1)$. Lo que conlleva a una complejidad lineal en el SST, es decir, $O(N)$.

   La segunda parte de la propagación tiene el mismo comportamineto que la selección, ya que en cada interación debe recorrer todos los elementos de S, el cual tiene la misma cardinalidad de V_{active}.

En conclusión, al igual que la selección, una cota para la complejidad de la propagación es $O(M_{\delta_s}N)$.

3. **Poda**: Las funciones Is_Leaf y $Parent$ que se utilizan en el algoritmo (2.9) tiene complejidad 1, ya que esa información se guarda en el vértice del árbol. El buscar el elemento más cercano de S (linea 1 de 2.9), tiene la misma complejidad de la segunda parte de la propagación. Luego, el saber si $x_{new} \in V_{inactive}$ es orden logarítmico en el tamaño de $V_{inactive}$, entonces está acotado por $\log(M_{\delta_s})$. Suponiendo que el camino más largo desde la raíz a una hoja es de tamaño L, entonces el bucle *while* se repite a lo más L veces.

   Todo lo anterior ocurre en cada ejecución de la fase de poda, entonces en la ejecución del SST esta fase tiene la complejidad acotada por:

$$O(M_{\delta_s}N + L\log(M_{\delta_s})N) = O(M_{\delta_s}N)$$

4. **Detector de colisiónes**: Esta función se utiliza en la muestra aleatoria y en la trayectoria que se genera cuando se propaga un vértice, por tanto es bastante utilizada por el SST. Se implementó una *detección de colisiones mediante recubrimientos simpliciales* [11], más especificamente, la detección de colisión 2D de una circunferencia, que contiene el robot, con polígonos. La complejidad de este algoritmo depente de la cantidad de vértices de los poligonos, por tanto tiene complejidad constante, ya que el espacio de trabajo es fijo. Entonces, la detección de colisiones tiene complejidad $O(N)$ en la ejecución del SST.

En conclusión, el SST tiene complejidad lineal en el número de muestras por el máximo número posible de vértices del árbol, es decir, $O(M_{\delta_s}N)$.

4.5. Resultados

El algoritmo SST tiene varios parámetros para su ejecución, los cuales influyen de forma directa en el resultado. Por tal motivo, es de interés ver el comportamiento del algoritmo con

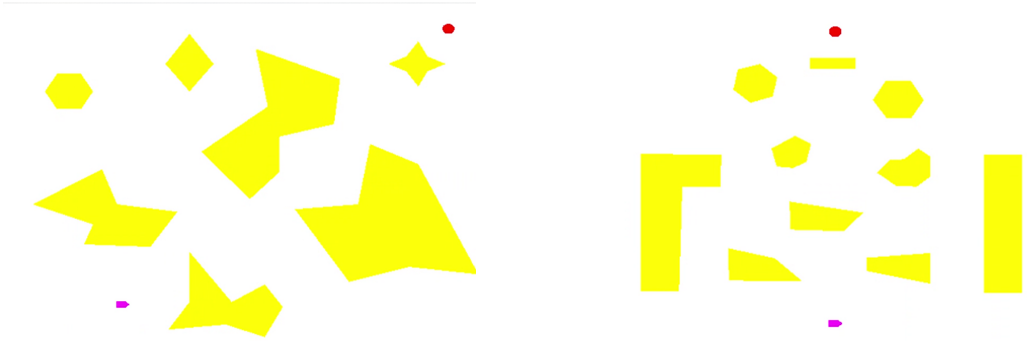
la variación de los parámetros. En esto consiste la presente sección.
Las variables correspondientes a las características del robot y algunas componentes del algoritmo se fijan para todas las ejecuciones con las que se obtiene el contenido de esta sección, a menos que se especifique lo contrario. Las variables que se fijan son las siguientes:

- El robot inicialmente se encuentra en el estado $x = y = \theta = v_l = v_r = 0$.
- Radio de las ruedas: 1
- Distancia entre el centro del robot y las ruedas: 0.71
- Magnitud máxima de velocidad : 5
- Magnitud máxima de aceleración: 1
- Intervalo del tiempo de propagación: $[0, 2]$
- Número de iteraciones: 150000

Para el análisis del funcionamiento del SST y SST* se hacen 25 ejecuciones en cualquier configuración de los parámetros a analizar. Además, como el principal interés es ver el comportamiento del algoritmo cuando se cambia el conjunto de controles por los controles extremos, entonces se fijan las muestras del espacio de estados y tiempo de propagación con las que se van a ejecutar el algoritmo. Con el objetivo de que cuando se comparen resultados de ejecuciones con los mismos parámetros no influya la parte aleatoria en el resultado, sólo el cambio del conjunto de controles. Además, se utilizan dos espacios de trabajo (figura 4.1) y cada una tiene una región objetivo diferente:

- Región objetivo 1, espacio de la figura (4.1a): círculo de radio 1 con centro en $(55, 55)$, no se considera la componente θ.
- Región objetivo 2, espacio de la figura (4.1b): círculo de radio 1 con centro en $(0, 55)$, de igual manera sólo se considera la posición y no la orientación.

Los resultados en las tablas de esta sección significan lo siguiente:



(a) **(b)**

Figura 4.1: Espacios de trabajo a utilizar en las ejecuciones. El robot es la figura color morado, la región objetivo es el círculo rojo y los polígonos de color amarillo son los obtáculos.

- **C.E.**: Especifica si el algoritmo se ejecutó con el conjunto de controles extremos o no.
- **Exitos**: Dice el número de veces, que en las 25 ejecuciones, se encontró una solución al problema de planificación.
- **Mínimo**: Es el menor costo que se obtuvo para resolver el problema de planificación.
- **Media**: Es la media aritmética de los costos obtenidos en las ejecuciones exitosas.
- **Máximo**: Es el mayor costo que se obtuvo para resolver el problema de planificación.
- **Des. estándar**: Es la desviación estándar de los costos obtenidos.
- **Activos**: Es la media aritmética de la cantidad de vértices activos en los árboles que encontraron una solución al problema de planificación.
- **Inactivos**: Es la media aritmética de la cantidad de vértices inactivos en los árboles que encontraron una solución al problema de planificación.

4.5.1. Variación de δ_s en el SST

El parámetro δ_s es el que controla que tan disperso es el árbol, más específicamente, la región donde está uno y sólo uno de los nodos del árbol. Por tanto, se espera que a menor valor de δ_s mayor la cantidad de nodos en el árbol y mejor costo, como se ve observa en los resultados. Además, el parámetro δ_{BN} se fijó en 3.

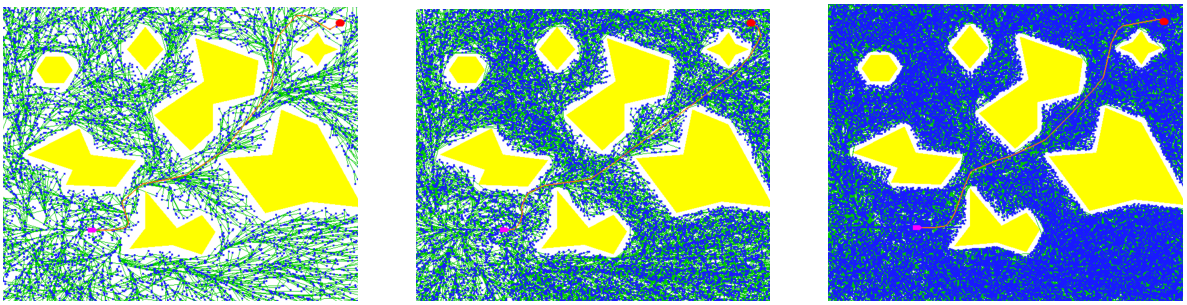
Se comienza analizando los datos obtenidos en el espacio de trabajo de la figura (4.1a).

C.E.	δ_s	Exitos	Mínimo	Media	Máximo	Des. estándar	Activos	Inactivos
No	1.6	23	22.1506	**25.8749**	33.0772	2.9162	2934	1907
No	1.1	25	21.8181	**23.5459**	26.8525	1.31601	8787	4887
No	0.6	25	20.9869	**23.2918**	27.8029	1.4627	29337	9386
Sí	1.6	23	22.3595	**24.916**	32.3435	2.63386	4208	2337
Sí	1.1	25	21.1239	**23.0156**	25.293	1.04612	10330	5200
Sí	0.6	25	21.0409	**22.4064**	24.5296	0.84457	33737	9091

Tabla 4.1: Variación de δ_s en la región objetivo 1.

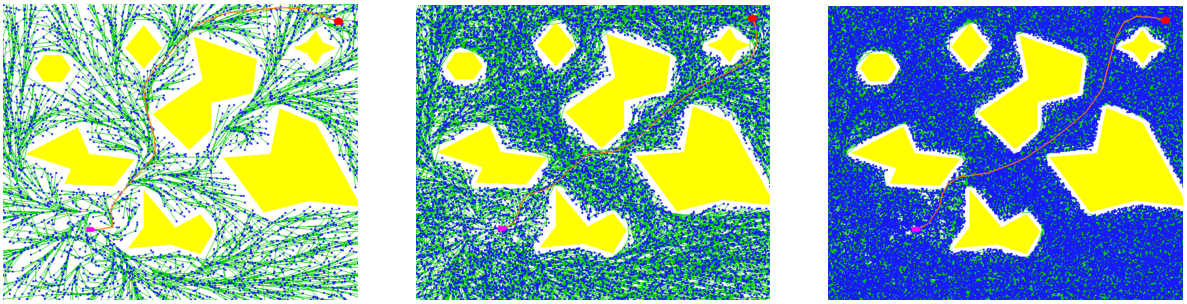
Observando la media aritmética y la desviación estándar se puede decir que el algoritmo tiene mejor rendimiento y es más estable a medida que δ_s disminuye, independiente de si los controles son extremos o no. Esa mejora no es gratis, ya que se ve un aumento considerable del número de vértices del árbol y esto conlleva a un mayor tiempo de ejecución. Dichos aumentos, del tamaño del árbol y tiempo de ejecución, se pueden ver en las figuras (4.2) y (4.3), donde los puntos azules son los vertices del árbol, los segmentos verdes son las aristas y se distingue el camino de menor costo, el resultado del algoritmo, con el color naranja. La figura (4.2) corresponden a la aplicación de los controles extremos y la figura (4.3) a todo el conjunto de controles.

Cabe aclarar que el camino que se distingue en las figuras (4.2) y (4.3) no es la trayectoria real del robot en el espacio de configuraciones, sino las aristas que unen los nodos del árbol, que son lineas rectas. La trayectoria real del robot no difiere mucho en forma, pero se puede apreciar que el movimiento es suave. Con la figura (4.4) se ve la semejanza global y algunas



(a) **(b)** **(c)**

Figura 4.2: Se utilizó el conjunto de controles extremos y $\delta_{BN} = 3$ (a) $\delta_s = 1.6$, costo= 24.85 y tiempo de ejecución 32.46 seg. (b) $\delta_s = 1.1$, costo= 23.62 y tiempo de ejecución 68.88 seg. (c) $\delta_s = 0.6$, costo= 22.18 y tiempo de ejecución 194.95 seg.



(a) **(b)** **(c)**

Figura 4.3: Se utilizó todo el conjunto de controles y $\delta_{BN} = 3$ (a) $\delta_s = 1.6$, costo= 26.69 y tiempo de ejecución 21.04 seg. (b) $\delta_s = 1.1$, costo= 24.23 y tiempo de ejecución 74.92 seg. (c) $\delta_s = 0.6$, costo= 23 y tiempo de ejecución 210.49 seg.

partes donde difieren.

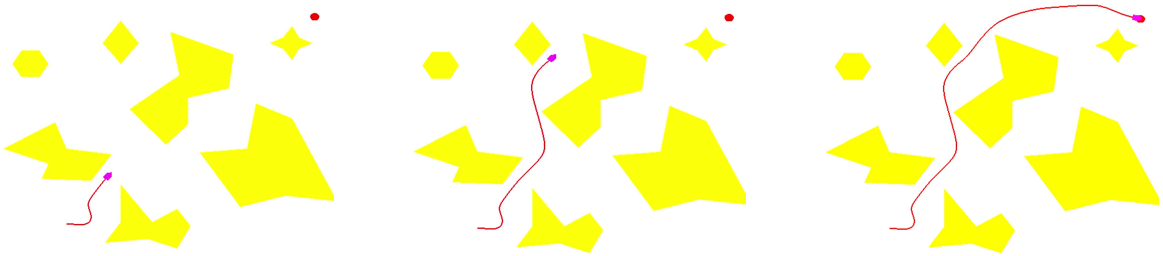


Figura 4.4: La trayectoria encontrada con el árbol de la figura (4.3a) que sigue el robot.

Comparando los resultado en base a los controles no cabe duda que el algoritmo tiene mejor rendimiento con los controles extremos en todos los datos que se analizaron, ya que tiene mejor media aritmética en el costo de la trayectoria encontrada y es más estable (menor desviación estándar). Además, no se aprecia ningún efecto contraproducente, por ejemplo, no hay una relación significante entre el número de vértices del árbol y el conjunto de controles a considerar, por tanto no se afecta el tiempo de ejecución del algoritmo. Lo cual es de esperar, dado que cuando se cambia el conjunto de controles, de tal manera que todo el espacio siga siendo accesible, no debería de producir un cambio significante en el tiempo de ejecución por como está diseñado el algoritmo.
Se continua con los datos obtenidos en el espacio de trabajo de la figura (4.1b). Se fijó el valor de δ_{BN} en 3.5.

C.E.	δ_s	Exitos	Mínimo	Media	Máximo	Des. estándar	Activos	Inactivos
No	1.6	23	18.5735	**20.5786**	23.4071	1.27582	3213	2040
No	1.1	25	17.3841	**19.2869**	21.0414	1.02427	9519	5117
No	0.6	25	17.6257	**18.8793**	20.5908	0.70956	31932	10425
Sí	1.6	25	17.4214	**19.8232**	21.505	0.99553	4047	2154
Sí	1.1	24	17.3393	**18.778**	20.1891	0.835335	10206	4928
Sí	0.6	25	16.7247	**18.1696**	19.4867	0.659846	33126	9356

Tabla 4.2: Variación de δ_s en la región objetivo 2.

Los anteriores resultados coinciden, en comportamiento, con los de la tabla (4.1). Por tanto, los comentarios hechos al respecto de los datos de la tabla (4.1) también son validos para la variación del δ_s en la región objetivo 2.

Las siguientes figuras muestran trayectorias solución para la región objetivo 2, donde la figura (4.5) corresponde a los controles extremos y la figura (4.6) a todo el conjunto de controles.

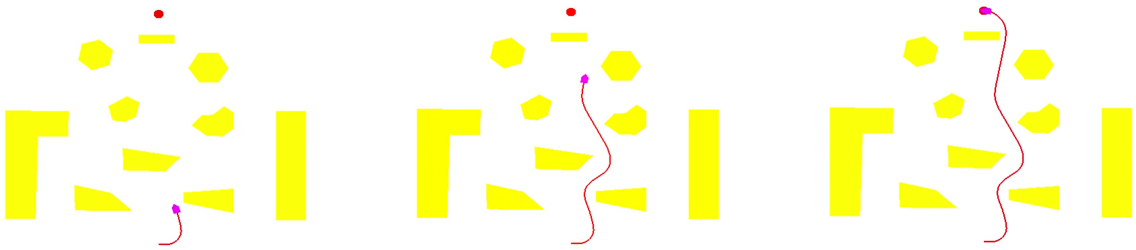


Figura 4.5: Controles extremos, $\delta_{BN} = 3.5$, $\delta_s = 0.6$, costo$= 17.43$ y tiempo de ejecución 214.45 seg.

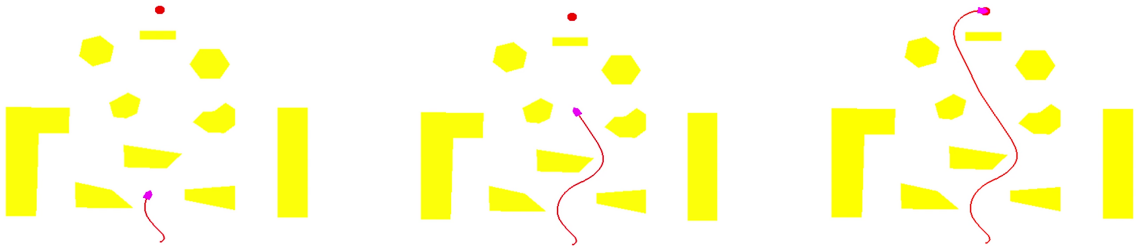


Figura 4.6: Conjunto completo de controles, $\delta_{BN} = 3.5$, $\delta_s = 0.6$, costo$= 19.9$ y tiempo de ejecución 205.5 seg.

4.5.2. Variación de δ_{BN} en el SST

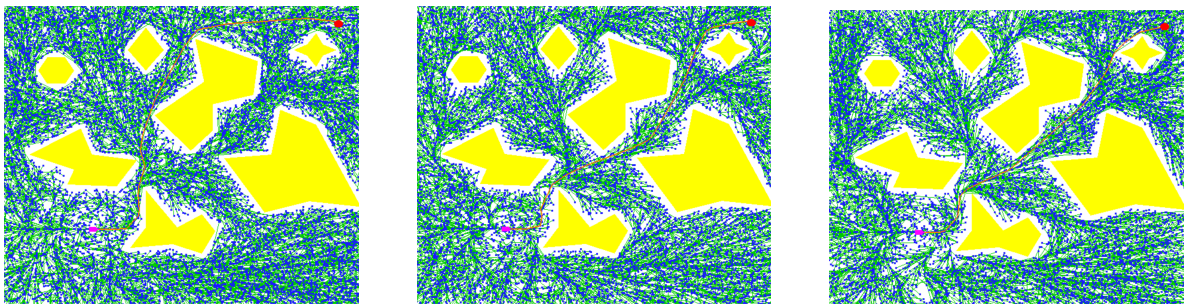
El parámetro δ_{BN} define el tamaño de la vecindad alrededor de la muestra aleatoria, para la selección del nodo con el que se hace la propagación. Por tanto, se espera que a mayor valor de δ_{BN} se seleccionen nodos de mejor calidad, lo que conlleva a rutas de menor costo, lo cual se puede apreciar en los datos de esta sección.

La siguiente tabla muestra los datos obtenidos en el espacio de trabajo de la figura (4.1a), donde el parámetro δ_s se fijó en 1.

C.E.	δ_{BN}	Exitos	Mínimo	Media	Máximo	Des. estándar	Activos	Inactivos
No	2	25	23.2764	**28.8982**	35.5592	3.40427	13238	6836
No	3	25	22.0702	**24.3506**	29.0564	1.66873	11137	5770
No	4	25	21.6289	**23.1105**	24.9665	0.978608	10384	5309
Sí	2	25	22.5136	**27.251**	33.6239	2.88247	16873	7766
Sí	3	25	21.6273	**22.8603**	24.8791	0.915915	12858	6011
Sí	4	25	21.2718	**22.3563**	24.2754	0.693503	11854	5527

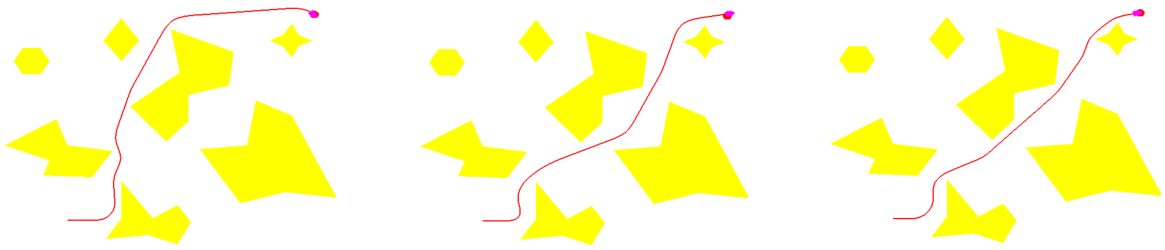
Tabla 4.3: Variación δ_{BN} en la región 1.

Los resultados concuerdan con lo que se espera. En efecto, a mayor valor de δ_{BN} mejor costo en promedio y menor desviación estándar, independiente de si los controles son extremos o no. Análogo a la variación δ_s, tal mejora no es gratis, ya que aumenta el tiempo de computo al considerar más vértices para seleccionar el que se propaga. Aunque, este aumento de tiempo es menor comparado con el que proporciona la variación de δ_s, ya que el tamaño del árbol no depende tanto de δ_{BN} como de δ_s, esto se puede apreciar en la figura (4.7) y comparando con la variacion de δ_s en la figura (4.2). En la figura (4.8) se puede ver, con más detalle, las trayectorias solución de la figura (4.7).



(a) **(b)** **(c)**

Figura 4.7: Se utilizaron los controles extremos y $\delta_s = 1$. (a) $\delta_{BN} = 2$, costo$=$ 24.33 y tiempo de ejecución 43.67 seg. (b) $\delta_{BN} = 3$, costo$=$ 22.84 y tiempo de ejecución 45.73 seg. (c) $\delta_{BN} = 4$, costo$=$ 20.97 y tiempo de ejecución 48.54 seg.



(a) **(b)** **(c)**

Figura 4.8: (a) trayectoria de la figura (4.7a). (b) trayectoria de la figura (4.7b). (c) trayectoria de la figura (4.7c)

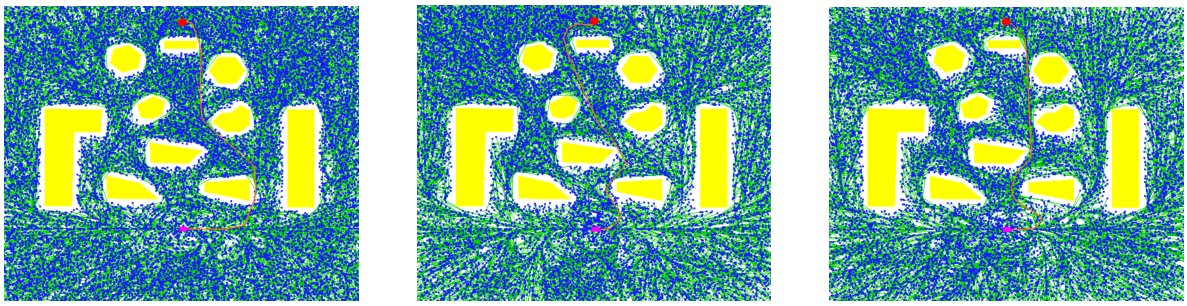
Se continua con los datos obtenidos en el espacio de trabajo de la figura (4.1b). Se fijó el valor de δ_s en 0.9.

C.E.	δ_{BN}	Exitos	Mínimo	Media	Máximo	Des. estándar	Activos	Inactivos
No	2.5	25	17.8023	**20.233**	22.7573	1.31228	17878	8442
No	3.5	25	17.6186	**19.2117**	20.5853	0.80627	15176	7135
No	4.5	25	17.0747	**18.3405**	20.6235	0.824177	13351	6364
Sí	2.5	25	17.5062	**19.0992**	20.2248	0.646655	19565	8216
Sí	3.5	25	17.5454	**18.6663**	19.9323	0.634101	15539	6576
Sí	4.5	25	17.1304	**18.2113**	19.3288	0.520687	13836	5889

Tabla 4.4: Variación δ_{BN} en la región 2.

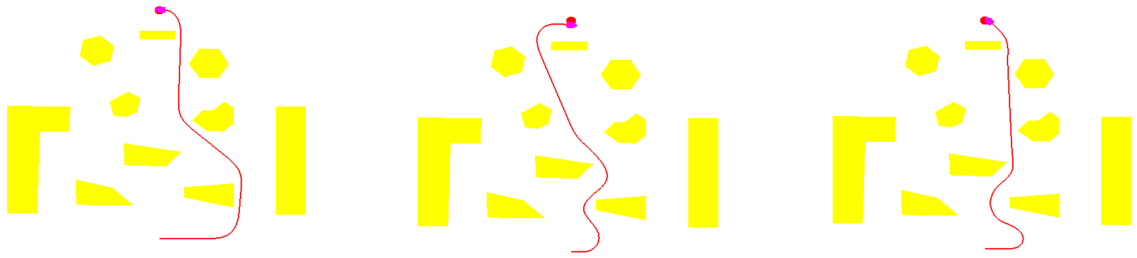
Al igual que en la variación del δ_s, los resultado de la región objetivo 2 tiene el mismo comportamiento que los obtenido en la región objetivo 1 al variar a δ_{BN}, lo que da más de soporte a lo esperado teóricamente. Además, se puede notar una diferencia un poco más significativa en el número de vértices del árbol al variar a δ_{BN}, que en los resultados de la región 1. Disminuye el número de vértices a medida que crece δ_{BN}. Ya que entre más grande es δ_{BN} se tiende a perder la exploración y como la cantidad máxima de vértices esta controlada por δ_s, esa perdida de exploración se ve reflejada en el número de vértices.

Este comportamiento es más notable en la tabla (4.4) que en la tabla (4.3) ya que se utilizó un valor de δ_s más pequeño, esto conlleva a que halla más vértices en la misma región del espacio de trabajo y por tanto una mayor diferencia en el tamaño del árbol cuando se reduce la exploración. En las figuras (4.9a) y (4.9b) se puede apreciar ligeramente esta diferencia del tamaño de los árboles.



(a) **(b)** **(c)**

Figura 4.9: Se utilizaron los controles extremos y $\delta_s = 1$. (a) $\delta_{BN} = 2.5$, costo= 19.41, tamaño del árbol 20904 y tiempo de ejecución 105.36 seg. (b) $\delta_{BN} = 3.5$, costo= 18.79, tamaño del árbol 17380 y tiempo de ejecución 89.63 seg. (c) $\delta_{BN} = 4.5$, costo= 18.53, tamaño del árbol 16085 y tiempo de ejecución 89.5 seg.



(a) **(b)** **(c)**

Figura 4.10: (a) trayectoria de la figura (4.9a). (b) trayectoria de la figura (4.9b). (c) trayectoria de la figura (4.9c)

Comparando los resultado en base a los controles, en un algoritmo con los mismo parámetros y ambiente, se tiene mejor rendimiento con los controles extremos, ya que tiene mejor

media aritmética y es más estable. Además, tampoco se aprecia ningún efecto contraproducente como en la sección anterior.

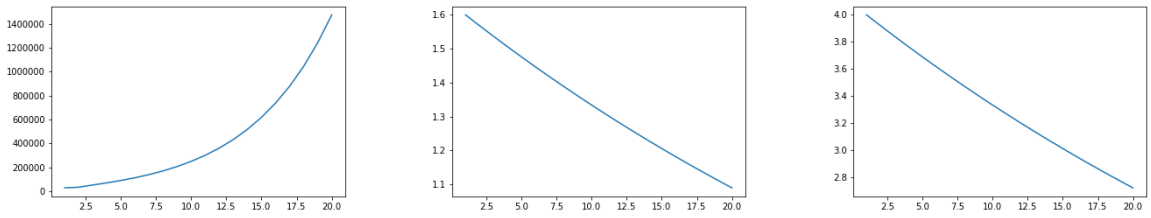
4.5.3. SST*

El meta-algoritmo SST* (2.10) tiene un proceso que va reduciendo a δ_{BN} y δ_s mientras aumenta N, parámetros del SST, el cual depende de un $\xi \in (0, 1)$. El proceso de incremento de N es exponencial, por tanto puede aumenta considerablemente el tiempo de ejecución del SST en cada iteración, dependiendo del valor de ξ. Cuando ξ es "cercano" a 1 los parámetros varian más lento que cuando se "aleja" de 1. Además, una variación "lenta" de δ_s y δ_{BN} hace que el número de vértices del árbol también aumente "lento", por lo que se necesita un valor de n (ver algoritmo 2.10) más grande para lograr llenar todo el espacio y así alcanzar la optimalidad (figura 4.11). Por otro lado, una variación "rápida" de δ_s, δ_{BN} y N hace que el número de vértices del árbol aumente "rápido", lo que conlleva al tiempo de ejecución del SST aumentar enormemente y sólo se pueda utilizar valores pequeños de n (figura 4.12).

Las figuras 4.11 y 4.12 muestran la variación de los parámetros del DDR para dos valores de ξ, donde el eje horizontal son las iteraciones del SST en el SST*. En la figura 4.11 se observa una variación lenta de los parámetros, se pueden hacer hasta 17 ejecuciones del SST antes que N llegue al orden de 10^6, pero en las 20 ejecuciones del SST ya se habría acumulado una cantidad aproximada de 9×10^6 muestras aleatorias, donde los parámetros δ_{BN} y δ_s sólo variaron de 4 a 2.72 y de 1.6 a 1.09, respectivamente. En la figura 4.12 se observa una variación más rápida de los parámetros, con una variación más significativa de δ_s y δ_{BN}, pero sólo se pueden hacer 5 ejecuciones del SST antes que N llegue al orden de 10^6 y en sólo con $n = 10$ se acumula más de 23×10^6 muestras aleatorias, lo que conlleva a gran tiempo de ejecución.

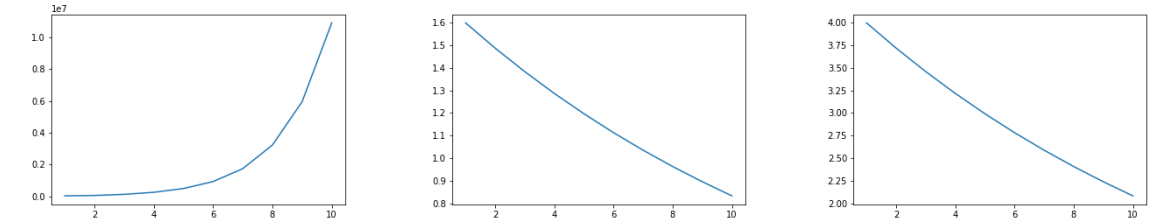
En conclusión es difícil, computacionalmente hablando, llegar a una reducción significativa de δ_s y δ_{BN} para apreciar el máximo potencial del algoritmo. Aún así, se hace la ejecución del SST* para las dos regiones objetivo que se estan considerando, con $\xi = 0.95$ que permite realizar 7 ejecuciones del SST antes que N alcanze un orden cercano a 10^6 y una reducción

considerable de δ_s y δ_{BN}.



(a) **(b)** **(c)**

Figura 4.11: Se definen $N_0 = 30000$, $\delta_{s_0} = 1.6$, $\delta_{BN_0} = 4$ y $\xi = 0.98$ (a) Variación de N. (b) Variación de δ_s. (c) Variación de δ_{BN}.



(a) **(b)** **(c)**

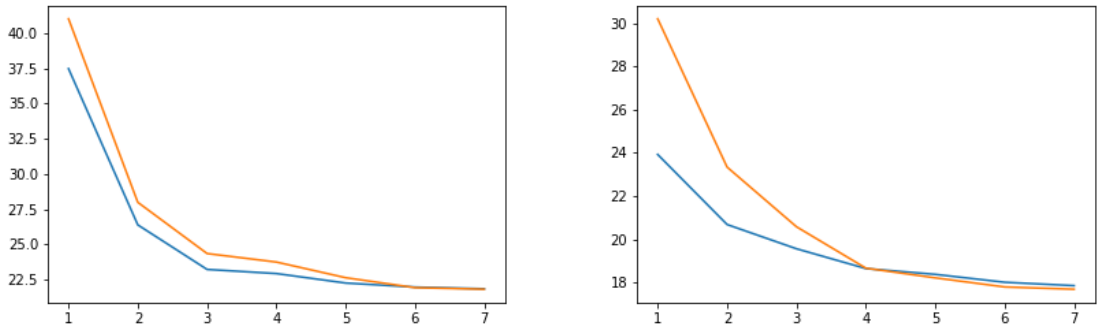
Figura 4.12: Se definen $N_0 = 30000$, $\delta_{s_0} = 1.6$, $\delta_{BN_0} = 4$ y $\xi = 0.93$ (a) Variación de N. (b) Variación de δ_s. (c) Variación de δ_{BN}.

Análogo a la sección anterior, se hacen 25 ejecuciones del SST* con muestras aleatorias fijas y los resultados se muestran en la tabla 4.5. Para las dos regiones objetivo se utilizaron los mismos parámetros, los cuales son $N_0 = 30000$, $\delta_{s_0} = 1.6$, $\delta_{BN_0} = 4$, $n = 7$ y $\xi = 0.95$. Cada ejecución utilizó aproximadamente 1.6×10^6 muestras aleatorias.

Región	C. Extremos	Mínimo	Media	Máximo	Des. estándar	Activos	Inactivos
1	No	20.9366	**21.8188**	22.4758	0.389646	8397	5019
1	Sí	20.8318	**21.8505**	22.9701	0.507127	9136	5009
2	No	16.604	**17.6786**	18.5719	0.5324	9535	5333
2	Sí	17.1365	**17.8446**	18.5764	0.387081	9785	4885

Tabla 4.5: SST* con $\xi = 0.95$.

No se puede extraer información significativa de la tabla 4.5 para comparar el cambio del conjunto de controles, por la semejanza de los resultados, entonces se analiza el comportamiento del SST* en el transcurso de su ejecución de la siguiente manera: en cada iteración del SST, en el SST*, se guarda el menor costo de una trayectoria a la región objetivo, en caso de que exista, para cada una de las 25 ejecuciones del SST*. Con esos valores se hace un promedio y así se genera la figura 4.13. El eje horizontal marca la iteración del SST en la ejecución del SST* y el vertical el costo promedio.



(a) **(b)**

Figura 4.13: La gráfica azul corresponde al SST* con controles extremos y la de color naranja al SST* usual. (a) Región objetivo 1. (b) Región objetivo 2.

Además se hacen experimentos con una modificación de los parámetros diferente a la establecida en el algoritmo 2.10, con el fin tener más datos para observar el comportamiento del SST*, ya que es limitada la información que se encuentra con sólo las 7 ejecuciones del SST que se realizaron en los anteriores experimentos. Se opta por una modificación lineal de los parámetros, con el objetivo de hacer más iteraciones y poder hacer una reducción más significativa de δ_s y δ_{BN}, sin que se aumente enormemente el tiempo de ejecución del SST. Esta actualización también permite considerar parámetros iniciales más grandes, lo cual se aplicó a δ_{s_0} y δ_{BN_0}. Sin embargo, con una actualización lineal de los datos es más dificil alcanzar optimalidad asistótica del SST*, ya que el aumento de N no es suficiente para ma-

nejar el aumento del tamaño del árbol que resulta en la reducción de δ_s y δ_{BN}, pero estos experimentos se hacen bajo la premisa de que el comportamiento obtenido con el cambio de controles no tiene porque diferir al que se obtiene con la actualización usual, como muentran los resultados.

Análogo a como se realiza la toma de datos en los experimentos anteriores, se hacen 25 ejecuciones del SST* con muestras aleatorias fijas y los resultados se muestran en la tabla 4.7. En este caso, la actualizacíon de los parámetros está en la tabla 4.6 y en ambas regiones objetivo el algoritmo SST se ejecuta 13 veces, en una ejecución del SST*, por tanto al final se tienen los parámetros $\delta_s = 0.6$, $\delta_{BN} = 2$ y $N = 150000$ para el SST.

Parámetros	Región objetivo 1	Región objetivo 2
Iniciales	$N_0 = 30000$ $\delta_{s_0} = 1.8$ $\delta_{BN_0} = 4.4$	$N_0 = 30000$ $\delta_{s_0} = 1.8$ $\delta_{BN_0} = 5$
Actualización	$N = N_0 + 10000 \cdot i$ $\delta_s = \delta_{s_0} - 0.1 \cdot i$ $\delta_{BN} = \delta_{BN_0} - 0.2 \cdot i$	$N = N_0 + 10000 \cdot i$ $\delta_s = \delta_{s_0} - 0.1 \cdot i$ $\delta_{BN} = \delta_{BN_0} - 0.25 \cdot i$

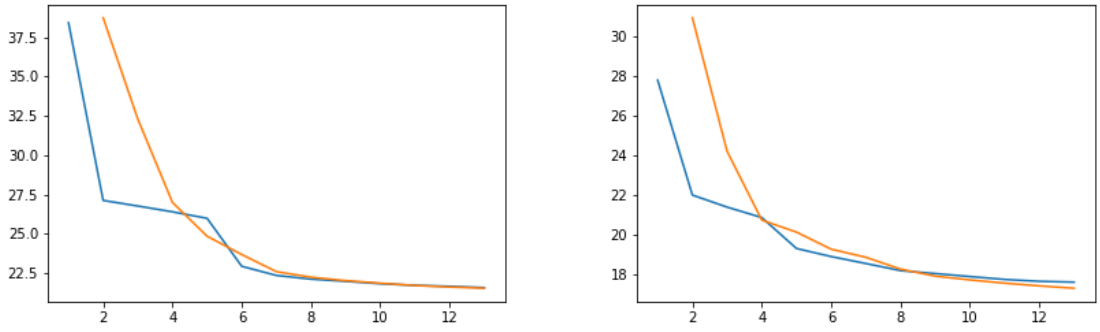
Tabla 4.6: Párametros iniciales y actualizacion de ellos en los experimentos con variación lineal.

Región	C. Extremos	Mínimo	Media	Máximo	Des. estándar	Activos	Inactivos
1	No	20.5528	**21.5342**	22.2097	0.442993	42715	12350
1	Sí	20.8637	**21.5716**	22.3695	0.406407	50399	14027
2	No	16.3869	**17.3313**	18.2553	0.533243	47944	13908
2	Sí	16.8044	**17.6388**	18.3213	0.421347	54783	13408

Tabla 4.7: SST* con actualización lineal de los parámetros.

Como se ve en la tabla 4.7, el único valor que difiere significativamente al cambiar el conjunto de controles es el número de vértices en el árbol. Por tanto, el tiempo de ejecución

con los controles extremos es mayor y sólo en ese sentido, a partir de los datos que muestra la tabla, se puede afirmar que el SST* tiene peor rendimiento considerando sólo los controles extremos, ya que se tiene resultados similares en el costo, con mayor tiempo de ejecución. Sin embargo, esto no es suficiente para analizar los resultados, por tanto se observa el comportamiento del SST* en el transcurso de su ejecución con la figura 4.14, de la misma forma que se generó la figura 4.13.



(a) **(b)**

Figura 4.14: La gráfica azul corresponde al SST* con controles extremos y la de color naranja al SST* usual. (a) Región objetivo 1. (b) Región objetivo 2.

Como primera observación, las figuras 4.13 y 4.14 concuerdan con lo mencionado en la sección 4.2.1, de que al considerar el conjunto de controles extremos se tiene mayor velocidad de convergencia. En efecto, ya que en la primera ejecución del SST, en el SST* con actualización lineal, se encontraron trayectorias que solucionan los problemas de planificación de movimiento, caso contrario al considerar todo el conjunto de controles. Además, en iteraciones del SST también se observa mayor cantidad de éxitos con los controles extremos, como se ve en las tablas 4.8 y 4.9. Por tanto, es más probable encontrar una solución, con menor número de muestras aleatorias, con el conjunto de controles extremos.

El anterior análisis fue en términos de solucionar el problema de planificación, ahora se analiza la calidad de las soluciones. En principio, con los controles extremos se encuentran soluciones de mejor calidad, pero a medida que aumenta las iteraciones del SST se van igua-

lando los resultados, hasta el punto de no poder distinguir las gráficas (figuras 4.13a y 4.14a). Además, no se puede apreciar un dominio en toda la ejecución del algoritmo, ya que se ven intersecciones de las gráficas (figura 4.13b y 4.14b). Este comportamiento puede explicarse por la optimalidad asintótica del SST*, ya que cualquier conjunto de controles que mantenga las condiciones expuestas en la sección 4.3, hace que el algoritmo converga al óptimo, por tanto tiende a dar resultado similares al aumentar el número de ejecuciones del SST.

Éxitos en ejecuciones del SST		Iteración			
Región	C. Extremos	1	2	3	4
1	No	15	22	23	23
1	Sí	19	24	24	25
2	No	19	23	23	25
2	Sí	23	25	25	25

Tabla 4.8: Número de éxitos en las primeras 4 ejecuciones del SST, en las 25 ejecuciones del SST* con $\xi = 0.95$.

Éxitos en ejecuciones del SST		Iteración			
Región	C. Extremos	1	2	3	4
1	No	0	13	23	23
1	Sí	18	22	23	23
2	No	0	15	23	24
2	Sí	19	23	24	24

Tabla 4.9: Número de éxitos en las primeras 4 ejecuciones del SST, en las 25 ejecuciones del SST* con actualización lineal.

4.6. Modificaciones en el SST

En esta sección se proponen dos modificaciones en el SST, aplicado a un sistema con el conjunto de controles extremos finito, con el fin de mejorar el costo de las soluciones

encontradas, a saber, una heurística y un método exhaustivo. Las modificaciones se hacen en el método de propagación, de la siguiente manera:

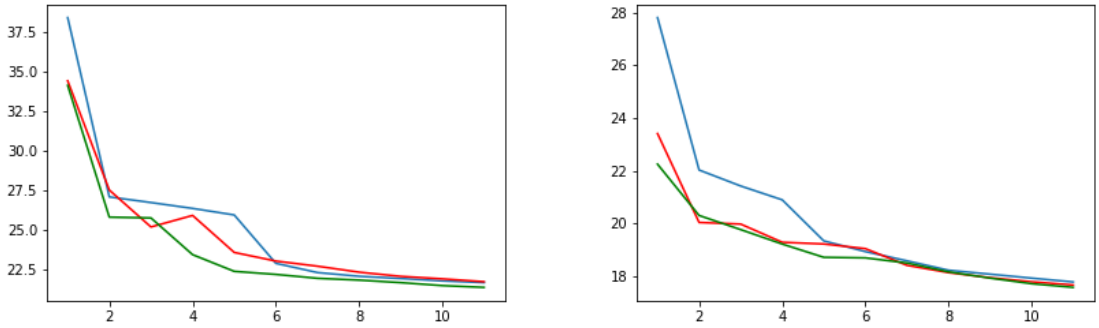
- **Método exhaustivo**: Desde el vértice seleccionado a propagar se van generando nuevos estados, con integración directa, aplicando cada uno de los controles extremos, con la misma muestra para el tiempo de integración, y cada vez que se genere un estado que cumple la condición para ser un nuevo vértice, esto es que sea localmente el mejor (Algoritmo 2.8), se agrega al árbol con su respectiva arista, es decir, se hace una propagación.

- **Heurística**: Similar al método exhaustivo se van generando nuevos estados desde el vértice a propagar con cada uno de los controles extremos, pero en este caso, se va seleccionando el control a aplicar de manera aleatoria (uniforme) y sólo se agrega al árbol el primer estado que cumpla lo requerido para ser el nuevo vértice. Análogamente, el tiempo de integración es el mismo para los estados generados.

El método exhaustivo se aparta de la metodología clásica de la planificación de movimiento basada en muestreo, ya que en una iteración se puede agregar más de un vértice al árbol, los cuales estan conectados con el mismo vértice. A diferencia de la heurística que a lo más agrega un vértice, aunque similarmente, puede que utilice todos los controles extremos.

Cabe mencionar que no se analizará con rigor estas modificaciones, ya que son propuestas para futuros trabajos, sólo se mostraran algunos resultados para ver su comportamiento. Además, solo se aplican a la actualización linal de parámetros.

Para analizar estas modificaciones, se ejecutan en el SST* con el DDR y se generan gráficas como 4.13 y 4.14 que muestran el comportamiento del SST, con sus iteraciones en el SST*. La actualización y el comienzo de los parámetros del SST es la misma, en cada región objetivo, que en la sección anterior y además se hacen las 25 ejecuciones con las mismas muestras, para evitar mayor influencia de la parte aleatoria, aunque en este caso sólo se hacen 11 iteración del SST en el SST*. Los resultados de esta propuesta de análisis de muestran en la figura (4.15), además, para hacer una comparación, también se muestran de los resutados del



(a) **(b)**

Figura 4.15: Las gráficas de color azul corresponden al SST* usual con el conjunto de controles extremos, son las mismas de la sección anterior, las rojas al método exhaustivo y las verdes a la heurística. (a) Región objetivo 1. (b) Región objetivo 2.

SST* usual con el conjunto de controles extremos, graficas azules de la sección anterior.

Se obtienen los resultados esperados, ya que en las primeras iteraciones del SST las dos modificaciones encuentran soluciones de mejor calidad. Esta mejora no es gratis, pues se aumenta las operaciones que debe hacer el algoritmo y conlleva a un mayor tiempo de ejecución. Además, se puede apreciar un mejor comportamiento con la heurística, ya que en gran parte de las ejecuciones es la que presenta los menores costos y el tiempo de ejecución es menor que el del método exhaustivo. Por otro lado, a medida que aumentan las iteraciones del SST, se encuentran resultados similares, esto debido a la propiedad de optimalidad asintótica del SST*, como se mencionó en la sección anterior.

CAPÍTULO 5

Conclusiones y futuros trabajos

En este trabajo se estudió el efecto de utilizar los controles localmente óptimos obtenidos con el principio del máximo de Pontryagin, como entradas para los métodos basados en muestreo SST y SST*, en términos de la velocidad de convergencia y del costo de las trayectorias encontradas, aplicados a problemas kinodinámicos en el sistema noholonómico dado por un robot de manejo diferencial controlado por las aceleraciones en sus ruedas, con lo que se logró concluir que el uso de esos controles aumenta la velocidad de convergencia a un costo de trayectoria dado y reducen el costo de las trayectorias obtenidas por los algoritmos, para un número fijo de iteraciones. Además, debido a que el SST* es asintóticamente óptimo, se obtienen las trayectorias mínimas en tiempo.

En cuanto a las aportaciones de esta tesis, la metodología que se desarrolló en el DDR para alcanzar el objetivo general de este trabajo se generalizó y mecanizó para su aplicación en otros sistemas, se demostró que el SST* con las modificaciones propuestas es probabilísticamente completo y asintóticamente óptimo. Además, se dieron justificaciones para que los algoritmos con controles extremos converjan más rápido que los convencionales, aplicado al

sistema DDR, esto acompañado de un estudio experimental.

Como futuros trabajos se puede buscar más sistemas dinámicos para aplicar la metodología que se generalizó en esta tesis, hacer un análisis profundo sobre la velocidad de convergencia y la calidad de las soluciones encontradas por los algoritmos con las modificaciones propuestas en 4.6, incorporar trayectorias obtenidas con la metodología, en algún sistema, en cascos de realidad virtual para exponer a usuarios humanos a dichas las trayectorias y determinar si les son agradables [3].

Referencias

[1] Balkcom, D. J. & Mason, M. T. (2002). Time optimal trajectories for bounded velocity differential drive vehicles. *The International Journal of Robotics Research*, *21*(3), 199–217.

[2] Barraquand, J. & Latombe, J.-C. (1993). Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, *10*(2-4), 121.

[3] Becerra, I., Suomalainen, M., Lozano, E., Mimnaugh, K. J., Murrieta-Cid, R., & LaValle, S. M. (2020). Human perception-optimized planning for comfortable vr-based telepresence. *arXiv preprint arXiv:2002.10696*.

[4] Becerra, I., Valentín-Coronado, L. M., Murrieta-Cid, R., & Latombe, J.-C. (2016). Reliable confirmation of an object identity by a mobile robot: A mixed appearance/localization-driven motion approach. *The International Journal of Robotics Research*, *35*(10), 1207–1233.

[5] Burden, R. L. & Faires, J. D. *Numerical Analysis 9th edition, 2011*. Brooks/Cole.

[6] Cameron, S. (1990). *Collision detection by four-dimensional intersection testing*. Oxford University. Computing Laboratory. Programming Research Group.

[7] Canny, J. (1988). *The complexity of robot motion planning*. MIT press.

[8] Donald, B., Xavier, P., Canny, J., & Reif, J. (1993). Kinodynamic motion planning. *Journal of the ACM (JACM)*, *40*(5), 1048–1066.

[9] Hsu, D., Kindel, R., Latombe, J.-C., & Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, *21*(3), 233–255.

[10] Hubbard, P. M. (1993). Interactive collision detection. In *Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium*, (pp. 24–31). IEEE.

[11] Jimenez, J. (2006). *Detección de Colisiones Mediante Recubrimientos Simpliciales*. PhD thesis, Dpto. de Informática-Universidad de Jaén.

[12] Johann, B. (1696). Problema novum ad cujus solutionem mathematici invitantur. *Acta Eruditorum*, *15*, 264–269.

[13] Karaman, S. & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, *30*(7), 846–894.

[14] Karaman, S. & Frazzoli, E. (2013). Sampling-based optimal motion planning for nonholonomic dynamical systems. In *2013 IEEE International Conference on Robotics and Automation*, (pp. 5041–5047). IEEE.

[15] Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, *12*(4), 566–580.

[16] Laumond, J.-P. et al. (1998). *Robot motion planning and control*, volume 229. Springer.

[17] Laumond, J.-P., Jacobs, P. E., Taix, M., & Murray, R. M. (1994). A motion planner for nonholonomic mobile robots. *IEEE Transactions on robotics and Automation*, *10*(5), 577–593.

[18] Laumond, J.-P., Sekhavat, S., & Lamiraux, F. (1998). Guidelines in nonholonomic motion planning for mobile robots. In *Robot motion planning and control* (pp. 1–53). Springer.

[19] LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

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

[21] LaValle, S. M. & Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. *The international journal of robotics research*, *20*(5), 378–400.

[22] Li, Y., Littlefield, Z., & Bekris, K. E. (2016). Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, *35*(5), 528–564.

[23] Liberzon, D. (2011). *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press.

[24] Lima, E. L. (2008). Curso de análise, vol. 2, projeto euclides. *Rio de Janeiro: IMPA*.

[25] Lin, M. C. (1993). *Efficient collision detection for animation and robotics*. PhD thesis, PhD thesis, University of California, Berkeley.

[26] Macias, V., Becerra, I., Murrieta-Cid, R., Becerra, H. M., & Hutchinson, S. (2018). Image feedback based optimal control and the value of information in a differential game. *Automatica*, *90*, 271–285.

[27] Murrieta-Cid, R., Ruiz, U., Marroquin, J. L., Laumond, J.-P., & Hutchinson, S. (2011). Tracking an omnidirectional evader with a differential drive robot. *Autonomous Robots*, *31*(4), 345.

[28] Pontryagin, L., Boltyanskij, V., Gamkrelidze, R., & Mishchenko, E. (1962). The mathematical theory of optimal processes. *New York, John Wiley & Sons*.

[29] Reister, D. B. & Pin, F. G. (1994). Time-optimal trajectories for mobile robots with two independently driven wheels. *The International Journal of Robotics Research*, *13*(1), 38–54.

[30] Renaud, M. & Fourquet, J.-Y. (1997). Minimum time motion of a mobile robot with two independent, acceleration-driven wheels. In *Proceedings of International Conference on Robotics and Automation*, volume 3, (pp. 2608–2613). IEEE.

[31] Ruiz, U., Murrieta-Cid, R., & Marroquin, J. L. (2013). Time-optimal motion strategies for capturing an omnidirectional evader using a differential drive robot. *IEEE Transactions on Robotics*, *29*(5), 1180–1196.

[32] Sussmann, H. J. (1987). A general theorem on local controllability. *SIAM Journal on Control and Optimization*, *25*(1), 158–194.

[33] Sussmann, H. J. & Willems, J. C. (1997). 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, *17*(3), 32–44.

[34] Tena, E. C. (2003). *Optimizacion dinamica*. Pearson Educación.

[35] Thompson, G. L. (2006). *Optimal Control Theory: Applications to Management Science and Economics*. Springer.