



UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

*“Estudio de técnicas de optimización modernas para
la planificación de trayectorias en robots móviles”*

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRÍA EN INGENIERÍA ELÉCTRICA

PRESENTA:

Ing. Daniel Fernando Zambrano Gutiérrez

DIRECTORES:

Dr. Horacio Rostro González

Dr. Juan Gabriel Aviña Cervantes

Dedicatoria

A mis queridos padres: Sr. Jose Fernando Zambrano García y Sra. Betty Enoemi Gutiérrez Sánchez

A mi amada esposa: Madame, Grecia Carolina Duque Giménez

A mis abuelos, Franco Gutiérrez y Carmen Sánchez que tanto han creído en mí.

A mi Abuela Maria Celina García †, siempre presente en mi corazón.

A mi Tía Míriam, siempre presente en cada paso que doy.

A mis queridos hermanos, Maria zambrano, Yohana zambrano, Jhosep Zambrano los cuales son la principal fuente de mi motivación.

A mis Suegros, Oswaldo Duque y Carolina Giménez.

Daniel Zambrano

Agradecimientos

Al Dr. Gabriel Aviña por su confianza, apoyo, orientación, dirección y formación durante la maestría, siendo pilar importante en mi desarrollo académico y profesional.

Al Dr. Horacio Rostro por su apoyo, orientación y formación durante la maestría.

Al Dr. Daniel Jauregui Vazquez por todo su apoyo, ayuda y confianza.

Al Dr. Jesús Yañez por todo su apoyo y consejos.

A mi amiga, Reyna Pernía Perez, por su ayuda y colaboración, deseándole el mejor de los éxitos en su vida profesional.

A todas aquellas personas que de una u otra manera hicieron posible la realización de este trabajo de Tesis.

Daniel Zambrano

Agradecimientos Institucionales

Expreso mi más sincera gratitud hacia la Universidad de Guanajuato, especialmente a la División de Ingenierías del Campus Irapuato-Salamanca (DICIS) por la formación académica y por el apoyo financiero que he recibido durante mis estudios en esta institución.



A mi Alma Mater, la Universidad Nacional Experimental del Táchira (Venezuela), base de mi sólida formación académica.



Este trabajo de tesis fue realizado gracias al apoyo invaluable recibido a través del Consejo Nacional de Ciencia y Tecnología CONACyT de México, bajo el número de becario 766748 y CVU 1046000.



Índice General

Dedicatoria	i
Agradecimientos	ii
Agradecimientos institucionales	iii
Índice de Figuras	ix
Índice de Tablas	xii
Resumen	xiv
1 Introducción	1
1.1 Introducción	1
1.2 Antecedentes	3
1.3 Justificación	4


1.4	Objetivos	5
1.4.1	Objetivo general	5
1.4.2	Objetivos específicos	6
1.5	Planteamiento general del problema	6
1.6	Organización de la tesis	7
	Abstract	1
2	Marco Teórico	8
2.1	Conceptos de movilidad y navegación robótica	9
2.2	Robots móviles autónomos	10
2.3	Detección robótica	11
2.4	Planificación de trayectorias	11
2.4.1	Enfoques de planificación de trayectorias	15
2.4.2	Roadmap	16
2.4.3	Descomposición de Celdas	16
2.4.4	Campos Potenciales Artificiales	18
2.4.5	Programación Matemática	19
2.4.6	Algoritmo Genético (AG)	19
2.4.7	Optimización por Enjambre de Partículas (PSO)	19
2.4.8	Redes Neuronales Artificiales	20
2.5	Clasificación de los enfoques de planificación de trayectorias	21
3	Campos Potenciales	23

3.1	Introducción	23
3.2	Campos Eléctricos	24
3.3	Generación de Trayectorias mediante potencial eléctrico.	26
3.4	Campo de atracción electrostático	27
3.5	Campo repulsivo electrostático.	29
3.6	Campo potencial Total	31
3.7	Problema de mínimos locales	34
4	Métodos de optimización	35
4.1	Introducción	36
4.2	Método de Newton	37
4.3	Método del Descenso del Gradiente	39
4.4	Método de Levenberg-Marquardt	40
4.5	Método de Dog-Leg	41
4.6	Funciones de Prueba	43
4.6.1	Validación del método de Newton	45
4.6.2	Validación del método del Descenso del Gradiente	47
4.6.3	Validación del método de Levenberg-Marquardt	49
4.6.4	Validación del método de Dog-Leg	51
4.7	Campos Potenciales con Levenberg-Marquardt y Dog-Leg	53
5	Métodos Metaheurísticos	56
5.1	Introducción	57

5.2	Optimización por enjambres de partículas (PSO)	58
5.3	Evolución Diferencial (ED)	61
5.4	Cuckoo search (CS)	63
5.5	Algoritmo de búsqueda gravitacional (GSA)	65
5.6	Funciones de Prueba	70
5.6.1	Validación del Algoritmo Metaheurístico PSO	72
5.6.2	Validación del Algoritmo Metaheurístico ED	73
5.6.3	Validación del Algoritmo Metaheurístico CS	76
5.6.4	Validación del Algoritmo Metaheurístico GAS	78
5.6.5	Comparación de los Algoritmos PSO, ED, CS, GAS	79
5.7	Integración APF con el algoritmo CS	82
5.7.1	Validación de método APF+CS	87
6	Conclusiones	90
	Referencias	92

Índice de Figuras

1.1	Principio de funcionamiento del algoritmo de Descomposición por Celdas [Patle et al., 2019] ©.	3
1.2	Principio de funcionamiento del algoritmo de Campos Potenciales Artificiales .	4
1.3	Mínimo local presente en los Campos Potenciales Artificiales	5
2.1	Clasificación del problema de planificación de trayectorias de los robots móviles.	12
2.2	Planificación de trayectoria basada en la hoja de ruta y Gráfico de visibilidad [Patle et al., 2019] ©.	17
2.3	Descomposición de celdas a) Espacio de configuración. b) Trayectoria generada al conectar el gráfico de conectividad en el espacio libre [Patle et al., 2019] ©.	17
2.4	Campos potenciales Artificiales [Patle et al., 2019] ©.	18
3.1	Representación de las líneas de fuerza generadas a partir de a) carga puntal negativa y a una b) carga puntal positiva ©.	25

3.2	La carga negativa atrae al robot y la carga positiva lo repele, lo que da como resultado una ruta óptima libre de obstáculos.	27
3.3	Campos potenciales de atracción.	28
3.4	Representación gráfica de la fuerza de atracción hacia la meta (circulo verde).	29
3.5	Campo de Potencial Repulsivo $U_r(\vec{X})$	30
3.6	Campo de Potencial Electrostático Total.	31
3.7	Espacio de trabajo. El cual puede ser sustituido por un mapa topológico.	32
3.8	Evolución del Robot mediante la interacción de campos potenciales artificiales.	33
3.9	Interacción de fuerzas y ruta calculada, donde el punto rojo es el punto de partida del robot y el verde es la meta.	34
4.1	Funciones de prueba para validación de algoritmos de optimización.	44
4.2	Funciones de prueba minimizadas por el método de Newton.	46
4.3	Funciones de prueba minimizadas por el método del descenso del gradiente.	48
4.4	Funciones de prueba minimizadas por el método de Levenberg-Marquardt.	50
4.5	Funciones de prueba minimizadas por el método de Dog-Leg.	52
4.6	Mapa generado con cuatro obstáculos fijos. Con un punto rojo se muestra la meta la cual el robot debe alcanzar.	53
4.7	Ruta generada mediante Potenciales Eléctrico y Levenberg-Marquardt.	54
4.8	Ruta generada mediante Potenciales Eléctrico y Dog-Leg. Las regiones de confianza son resaltadas en color rojo.	55
4.9	Problema de mínimos locales con el método de Dog-Leg y Levenberg-Marquardt.	55
5.1	Comportamiento de enjambres de peces y aves 	58

5.2	Representación esquemática del movimiento de una partícula en PSO, moviéndose hacia el mejor global g^* y el mejor actual x_i^* para cada partícula i .	59
5.3	Comportamiento de las Aves <i>cuckoo</i>  .	63
5.4	Fuerza de atracción entre masas  .	67
5.5	Funciones de prueba para validación de algoritmos de optimización Metaheurística.	71
5.6	Convergencia VS Iteración con el algoritmo PSO.	73
5.7	Comportamiento del enjambre de partículas ante la función de prueba <i>Hölder table function</i> .	74
5.8	Convergencia VS Iteración con el algoritmo ED.	75
5.9	Convergencia Vs Iteración con el algoritmo CS.	77
5.10	Comportamiento de las aves <i>cuckoo</i> en la función de prueba <i>Hölder table function</i> .	78
5.11	Comportamiento del algoritmo GAS ante la función de prueba <i>Ackley function</i> .	79
5.12	Convergencia Vs Iteración con el algoritmo GAS.	80
5.13	Comparación de convergencia de los algoritmos PSO, ED, CS, GAS.	82
5.14	Comparación del tiempo de ejecución de los algoritmos PSO, ED, CS, GAS.	83
5.15	Problema de mínimo local APF	84
5.16	Distorsión de las líneas de fuerza del método APF	85
5.17	Rutas generadas a partir de la modificación de las fuerzas del método APF	85
5.18	Diagrama de flujo del método APF+CS	87
5.19	Optimización del método APF mediante el algoritmo CS	89

Índice de Tablas

2.1	Diferentes escenarios para el problema de planificación de trayectorias.	14
2.2	Comparación de ventajas y desventajas de algunos métodos de planificación de trayectorias.	21
4.1	Funciones de referencia para evaluación de algoritmos de optimización	43
4.2	Resultados obtenidos por el método de Newton para las funciones de referencia 4.1.	45
4.3	Resultados obtenidos por el método del descenso del gradiente para las funciones de referencia 4.1.	47
4.4	Resultados obtenidos por el método de Levenberg-Marquardt para las funciones de referencia 4.1.	49
4.5	Resultados obtenidos por el método de Dog-Leg para las funciones de referencia 4.1.	51
4.6	Parámetros usados para el ajuste de los algoritmos de optimización	53
4.7	Resultados numéricos para llegar al vector objetivo dado por $\vec{X}_g = [940, 800]$	54

5.1	Funciones de referencia para evaluación de algoritmos Metaheurísticos	70
5.2	Resultados obtenidos por el algoritmo PSO para las funciones de referencia 5.1.	72
5.3	Resultados obtenidos por el algoritmo ED para las funciones de referencia 5.1.	74
5.4	Resultados obtenidos por el algoritmo CS para las funciones de referencia 5.1.	76
5.5	Resultados obtenidos por el algoritmo GAS para las funciones de referencia 5.1.	79
5.6	Factores de ajuste para los algoritmos de optimización	81
5.7	Resultados obtenidos por el método APF+CS con los escenarios de la Figura 5.19.	88

List of Algoritmos

1	Método de los Campos de Potenciales Artificiales.	32
2	Método de Newton para n variables	38
3	Método del Descenso de gradiente	40
4	Método de Levenberg Marquardt.	41
5	Método de Optimización Dog-Leg.	42
6	Algoritmo PSO.	60
7	Algoritmo Evolución Diferencial.	62
8	Algoritmo de búsqueda de Cuckoo Search	65
9	Algoritmo de búsqueda Gravitacional	69

Resumen

Este trabajo presenta una revisión de la aplicación de los fundamentos teóricos de las leyes de interacción de cargas electrostáticas aplicada a la robótica. Concretamente la aplicación de estas leyes se emplean para la planificación de trayectorias a robots móviles en entornos dinámicos y tareas multiobjetivo. Así, en su forma más básica, un robot puede representarse como una carga puntual a la que se le permite moverse libremente en un área de trabajo o mapa disponible; y tiene como objetivo alcanzar una meta preestablecida. En consecuencia, para planificar una trayectoria evitando obstáculos, la interacción electrostática entre los elementos del entorno (obstáculos) y el robot permite determinar dinámicamente una trayectoria suave y continua, además de utilizar la mínima cantidad de energía en la interacción. Además, incluye un estudio del desempeño de diferentes técnicas de optimización robustas para minimizar el problema de la interacción de cargas electrostáticas, conduciendo a la estimación y ejecución de una trayectoria rápida y estable. En este tenor, se simula el movimiento del robot, incluyendo un control robusto de posición o velocidad, para mejorar y obtener un movimiento natural y más suave. Bajo este esquema, el robot puede evolucionar en entornos complejos y dinámicos siempre que se disponga de información sensorial adecuada.

CAPÍTULO 1

Introducción

1.1. Introducción

En la actualidad, los robots móviles se encuentran inmersos en diferentes actividades del mundo real, tales como operaciones militares, procesos agrícolas, transporte y manipulación de materiales peligrosos, entre otros [Ha et al., 2019; Seyyedhasani et al., 2020; Evans et al., 1989]. Un robot móvil puede llegar a ser esencial en múltiples tareas, donde la intervención humana pueda estar comprometida, limitada o se requiera una alta precisión. En un robot móvil, los procesos autónomos son altamente priorizados. En este sentido, la navegación autónoma permite al robot desplazarse de una posición inicial a una posición final sin intervención humana. Durante este proceso, el robot debe planificar sus movimientos y evitar colisionar con obstáculos de forma dinámica. Para mejorar los procesos de navegación la navegación, se debe disponer de algoritmos robustos de planificación de trayectorias basados en métodos de optimización rápidos y estables.

La planificación de trayectorias de un robot es un tema al cual se le ha prestado especial atención en los últimos años y es el tema que se aborda en este trabajo de tesis. Esta se basa en dos enfoques principales, clásicos y metaheurísticos. [Wahab et al., 2020]. La planificación

de trayectorias con métodos clásicos tiene ciertas restricciones que reducen su aplicabilidad en situaciones generales. Algunos de los métodos clásicos son: el gráfico de visibilidad [Mitchell, 1988], mapeo de cuadrículas [Weigl et al., 1993], diagramas de Voronoi [Candeloro et al., 2013], entre otros.

Estos métodos son útiles debido a que su implementación es simple. Sin embargo, presentan algunos problemas para la planificación de trayectorias en entornos complejos, ya que las rutas encontradas no garantizan ser una trayectoria globalmente óptima. Por otro lado, en los últimos años han aparecido un gran número de algoritmos metaheurísticos de optimización que emulan o se inspiran en comportamientos biológicos, físicos o químicos. Estos algoritmos han demostrado tener mejores resultados que los métodos tradicionales basados en gradientes. En nuestro caso, el objetivo consiste en obtener una trayectoria globalmente óptima y para ellos se pueden utilizar algoritmos tales como el algoritmo de optimización de colonias de hormigas [Castillo et al., 2015], algoritmos genéticos [Sarkar et al., 2020], algoritmos de optimización de enjambres de partículas [Das, 2020], entre otros.

Este trabajo tiene como objetivo principal, dar solución al problema de la planeación de trayectorias, dicho problema es de vital importancia para definir a un robot móvil como autónomo, en el sentido de la navegación. En este sentido, se puede definir la navegación autónoma como la tarea que realiza un robot móvil para trasladarse de un punto de origen a un punto destino mediante una trayectoria definida. En la antigüedad, este término se aplicaba principalmente en la navegación marítima [Rogoff, 1988] utilizando cartas o sistemas de referencias muy limitados.

Durante la navegación de un robot, este deberá interactuar con su entorno y evadir los obstáculos que se presenten para poder ejecutar el seguimiento a la trayectoria planeada y llegar al destino sin ningún tipo de ayuda externa. Para lograr este objetivo se requiere una metodología integral que permita guiar al robot móvil. La metodología integral implica la fusión de algoritmos de optimización que se inspiran o simulan comportamientos de la naturaleza, los cuales son conocidos como métodos metaheurísticos. El algoritmo de optimización principal y base de este trabajo se denomina Campos Potenciales Artificiales (APF por sus siglas en inglés Artificial Potential Field), el cual solo requiere la inclusión de una función objetivo (también llamada de costo) para lograr la tarea planteada.

1.2. Antecedentes

La planificación de trayectorias es un área de estudio actual, en el que están surgiendo constantemente nuevas soluciones y métodos para abordar los problemas relacionados con la robótica móvil. Actualmente, ninguno no se tiene una metodología definida que capaz de obtener resultados con características deseables, estas características pueden ser: baja carga computacional, eficiencia, rapidez, trayectorias suaves, soluciones óptimas, entre otras. Debido a esto se requieren algoritmos robustos que puedan cumplir con estos requerimientos.

En las últimas décadas, las investigaciones referentes a la planificación de trayectorias se pueden clasificar en dos enfoques principales: clásico y heurístico [Masehian and Sedighzadeh, 2007]. En el enfoque clásico se encuentran la descomposición por celdas (Cell Decomposition, CD) [Lingelbach, 2004] y campos potenciales artificiales (APF) [Khatib, 1986] teniendo este último gran participación en años recientes [Li, 2020; Lin et al., 2020].

La idea del método de CD es descomponer el espacio de trabajo (C-espacio) en un conjunto de celdas y así determinar la proximidad entre ellas como se puede observar en la Figura 1.1. Esta metodología representa los obstáculos mediante una serie de restricciones en los parámetros de configuración y la idea principal es minimizar los parámetros del modelo, con el fin de encontrar una trayectoria óptima entre el punto de inicio y el punto destino [Siegwart et al., 2004].

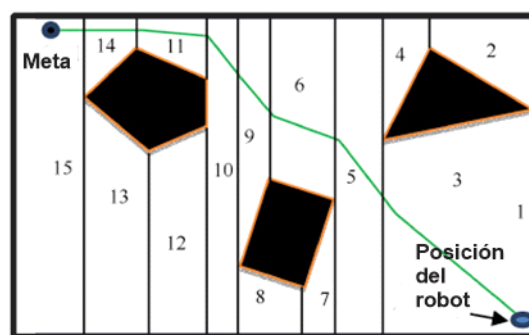


Figura 1.1. Principio de funcionamiento del algoritmo de Descomposición por Celdas [Patle et al., 2019] ©.

Por otra parte, el concepto de APF fue introducido por [Khatib, 1985], el cual propone una interacción de fuerzas electrostáticas. Esto con la finalidad de generar una fuerza de repulsión y de atracción, esto se consigue asignando a la meta una carga eléctrica de signo opuesto del robot y a los obstáculos una carga del mismo signo 1.2.

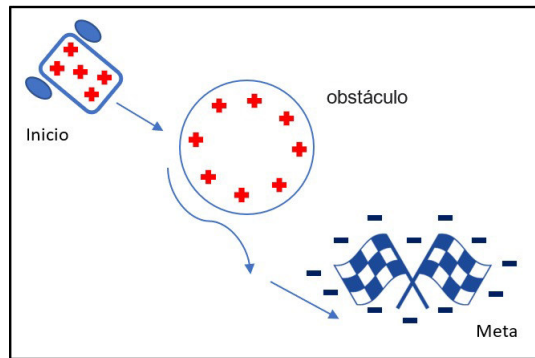


Figura 1.2. Principio de funcionamiento del algoritmo de Campos Potenciales Artificiales

La tarea asignada al robot consiste en llegar a su objetivo a partir de la planificación de trayectoria generada dinámicamente, apoyándose en el área de trabajo, representado en parte por un campo eléctrico conservativo sobre el cual se desplaza el robot.

Si bien la tarea de planificación de trayectorias puede resolverse mediante el modelado de sistemas y como un problema de optimización, se requiere implementar varios algoritmos para comparar su desempeño individual y escoger el método adecuado. En este sentido, entre los enfoques heurísticos y metaheurísticos están las Redes Neuronales, los Algoritmos Genéticos ([Genetic algorithms -GA](#)) [[Dashkevich et al., 2020](#)], Optimización por Colonia de Hormigas (Ant Colony Optimization) [[Qingzhen et al., 2007](#)], Optimización por enjambre de partículas (Particle Swarm Optimization) [[Gao et al., 2015](#)], Búsqueda gravitacional (Gravitational Search) [[Rashedi et al., 2009a](#)], Búsqueda Tabu (Tabu Search) [[Chelouah and Siarry, 2000](#)] y Lógica Difusa (Fuzzy Logic) [[Castillo et al., 2015](#)]. En muchas ocasiones, algunos de estos métodos son combinados o hibridizados con el objeto de mejorar el desempeño individual y reducir los comportamientos desventajosos de cada método individual [[Paterega, 2011](#)].

1.3. Justificación

En robótica existen diferentes métodos para la planificación de trayectorias [[Rosa, 2004](#); [García et al., 2010](#)], siendo el método APF uno de los más atractivos por estar fundamentado en la teoría de campos eléctricos [[Lee, 2004](#)]. Sin embargo, como todo problema de optimización, este método es susceptible a caer en soluciones de mínimos locales [[Khatib, 1986](#)]. Esto se debe a la interacción de fuerzas producidas por los campos de atracción y repulsión. En la [Figura 1.3](#) podemos observar como el robot puede quedar atascado debido a la interacción de las fuerzas,

visto de otra forma, la fuerza de repulsión y la de atracción tienen la misma orientación pero en sentidos opuestos.

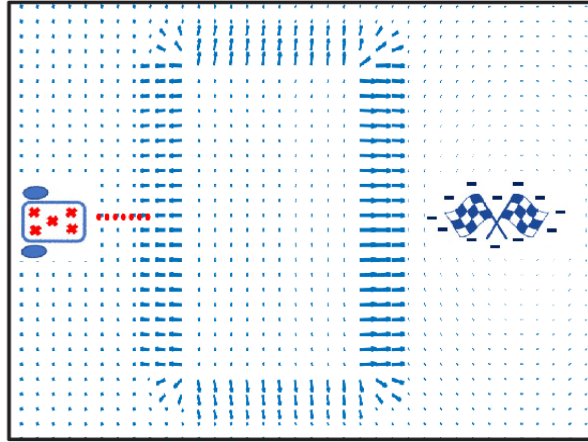


Figura 1.3. Mínimo local presente en los Campos Potenciales Artificiales

En la literatura [existen](#) diversas propuestas para atender esta problemática y dar robustez a las soluciones planteadas [[Barraquand et al., 1991](#); [Caselli et al., 2001](#); [Chengqing et al., 2000](#)]. Como en cualquier aplicación real, es de esperarse que de momento no exista una solución definitiva y simple a este problema, requiriéndose la implementación de complejos algoritmos y una metodología sofisticada o y metodologías sofisticadas para resolver el problema de planificación de trayectorias. Debido a lo anterior, se aprecia la posibilidad de explorar la capacidad que tienen los más recientes y robustos algoritmos metaheurísticos para evadir mínimos locales utilizando el esquema de leyes electrostáticas y de la interacción de cargas en campos potenciales.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar y evaluar algoritmos de planificación de trayectorias para robots móviles aplicando técnicas de optimización moderna.

1.4.2. Objetivos específicos

1. Realizar una revisión del estado del arte de las metodologías actuales para la optimización moderna de trayectorias de robots móviles.
2. Codificar y evaluar el algoritmo de planificación de trayectoria basado en campos potenciales artificiales (APF).
3. Analizar y evaluar el desempeño de los principales algoritmos determinísticos de optimización basados en gradientes: Newton, Descenso del gradiente, Levenberg-Marquardt y Dog-Leg.
4. Codificar y evaluar los metaheurísticos más representativos: Evolución diferencial, Optimización por Enjambre de Partículas (PSO), Cuckoo Search (CS), Búsqueda Gravitacional (GAS).
5. Desarrollar algoritmos híbridos que combinen APF con el método metaheurístico que mejor se ajuste.
6. Validar el algoritmo propuesto para la solución del problema de planificación de trayectorias en robots móviles.

1.5. Planteamiento general del problema

En la actualidad, las nuevas tendencias de la investigación en robótica y el desarrollo constante de nuevas y mejores tecnologías han tomado gran importancia debido a sus diferentes áreas de aplicación. Dentro de los temas con mayor importancia en el área de la robótica, encontramos la planificación de trayectorias. La planificación de trayectorias debe procesar su entorno y así poder indicarle al robot cuál será su siguiente movimiento, cada movimiento modificará la solución que se genera, ya que la planificación de trayectoria depende de la posición del robot. Dadas las exigencias actuales, donde los entornos son dinámicos, es necesario desarrollar metodologías inteligentes que tengan la capacidad de resolver tareas complejas sin la necesidad de ayuda externa (operador). Por esto los algoritmos óptimos que puedan dar solución a la navegación, han sido objeto de muchos estudios. En este trabajo de tesis se presenta la propuesta de solución al problema de planificación de trayectorias de robots móviles, combinando el método de APF con algoritmos de optimización metaheurísticos y determinísticos basados en gradientes.

Los APF es un método matemático basado en la interacción de fuerzas electrostáticas (fuerza de atracción y de repulsión) los cuales son ampliamente utilizados para la planificación de trayectoria en robots móviles, dado que genera soluciones estables [Zhu et al., 2013; ASİL and USLU, 2020]. El objetivo deseado es conocido como mínimo global y es representado por la fuerza de atracción. Este mínimo global representa la meta o punto final donde se quiere que llegue el robot móvil.

Los APF presentan un inconveniente, cuando la interacción de fuerzas ocasionan mínimos locales [Lee and Park, 2003], ocasionando que el robot quede atascado. Para solventar el problema, en este trabajo de tesis se plantea la fusión del método APF con algoritmos metaheurísticos, a fin de realizar ciertas modificaciones en los parámetros originales del APF y así salir de los mínimos locales. Estos algoritmos presentan un mejor esquema de exploración del espacio de trabajo, es decir una relación de exploración/explotación mejorada [Dokeroglu et al., 2019].

1.6. Organización de la tesis

Este trabajo de tesis está organizado en 5 capítulos, los cuales se describen brevemente a continuación:

Capítulo 1: Presenta la introducción y los objetivos de la presente obra.

Capítulo 2: Describe los conceptos necesarios para la planificación de trayectorias en robots móviles.

Capítulo 3: Formula la metodología de los campos potenciales artificiales.

Capítulo 4: Presenta algoritmos de optimización dependientes del gradiente y la integración con los campos potenciales artificiales.

Capítulo 5: Presenta algoritmos de optimización Metaheurísticos y la integración con los campos potenciales artificiales.

CAPÍTULO 2

Marco Teórico

2.1	Conceptos de movilidad y navegación robótica	9
2.2	Robots móviles autónomos	10
2.3	Detección robótica	11
2.4	Planificación de trayectorias	11
2.4.1	Enfoques de planificación de trayectorias	15
2.4.2	Roadmap	16
2.4.3	Descomposición de Celdas	16
2.4.4	Campos Potenciales Artificiales	18
2.4.5	Programación Matemática	19
2.4.6	Algoritmo Genético (AG)	19
2.4.7	Optimización por Enjambre de Partículas (PSO)	19
2.4.8	Redes Neuronales Artificiales	20
2.5	Clasificación de los enfoques de planificación de trayectorias	21

La planificación de trayectorias de robots móviles consiste en encontrar una ruta libre de colisiones a través de un entorno, desde una ubicación de inicial hasta una ubicación de meta deseada. En este capítulo se realiza una clasificación de los distintos enfoques de planificación de trayectorias y una revisión general sobre los métodos tradicionales de planificación de trayectorias en distintos entornos, como por ejemplo, el método del gráfico de visibilidad, el método de descomposición de celdas y el método del campo potencial artificial. Además, se discuten los inconvenientes relacionados con el método del campo potencial. Por otro lado se definen tres términos útiles que se utilizan habitualmente en las técnicas de planificación de trayectorias basadas en gráficos, los cuales son: espacio de configuración ($C_{espacio}$), espacio de obstáculos ($C_{obstaculo}$), espacio libre (C_{libre}) y trayectoria libre.

2.1. Conceptos de movilidad y navegación robótica

Definición 2.1.1 (Movilidad). La movilidad del robot se define el número de grados de libertad con el que los robots pueden moverse libremente en el entorno (o volumen de trabajo en caso de robots fijos o manipuladores industriales).

El primer tipo de robot práctico fue el denominado “Unimate”, que es un robot manipulador [Devol, 1961]. Estos robots se han utilizado en fábricas y están programados para realizar una secuencia de acciones repetidas una y otra vez de forma más rápida, barata y precisa que la de los humanos.

Definición 2.1.2 (Autonomía). Es una cualidad de un robot de naturaleza bastante abstracta y difícil de medir de forma paramétrica. Hay una clara distinción entre lo automático y lo autónomo, donde automático significa que el robot hará lo que programemos y autónomo significa que el robot tiene la capacidad de tomar una decisión tomando en cuenta todos los agentes internos y la dinámica actual de un entorno cambiante u hostil.

Definición 2.1.3 (Espacio de configuración ($C_{espacio}$)). El espacio de configuración nos da información sobre la posición y orientación que el robot móvil puede tomar en el espacio de trabajo, este puede ser representado mediante un mapa de navegación en 2D o 3D.

Cuando el robot no presenta ninguno altercado con un obstáculo, se dice que se encuentra en el espacio libre y, por lo tanto, la configuración es libre. El espacio de trabajo de

un robot puede ser definir, como un espacio de configuraciones, el cual consiste en el conjunto de todas las posiciones y orientaciones que el robot puede tomar.

Por lo tanto, este se utiliza básicamente en la planificación de trayectorias de robots en un entorno que incluye obstáculos estacionarios conocidos. Finalmente, en resumen, el espacio de configuración es una transformación del espacio físico en el cual el robot y el obstáculo tienen el tamaño real a otro espacio, donde el robot es tratado en una representación simplificada como una partícula o con un volumen regular, evitando complejas representaciones dinámicas o geométricas. Esto se consigue reduciendo el tamaño del robot a un punto y ampliando los obstáculos en el tamaño del robot [[Lozano-Perez, 1983](#)]

Definición 2.1.4 (Espacio libre (C_{libre})). Se define como el conjunto de áreas que no están ocupadas por los obstáculos del espacio de configuración.

Definición 2.1.5 (Espacio de obstáculos ($C_{obstaculo}$)). Es el conjunto de Áreas que se encuentran ocupadas en el espacio de configuración, es decir, se definen como un conjunto de configuraciones no factibles que representan los obstáculos existentes en él $C_{espacio}$. En un ambiente dinámico, se pueden establecer esquemas de cartografía que actualizan constantemente estos espacios libres.

Definición 2.1.6 (Trayectoria libre). Es la trayectoria entre el punto de partida y el punto de llegada que se encuentra completamente en el espacio libre y no entra en contacto con ningún obstáculo del entorno.

2.2. Robots móviles autónomos

Una área importante de la robótica es permitir que el robot se adapte a su entorno de trabajo, que puede ser terrestre, subacuático, aéreo, subterráneo o espacial. Para este fin, el robot necesita varias capacidades para ser autónomo y poder operar de forma inteligente. Consecuentemente, para que un robot sea totalmente autónomo, debe tener la capacidad de obtener información sobre el entorno, trabajar durante un periodo prolongado sin ayuda humana, moverse por sí mismo en su entorno de trabajo y evitar los obstáculos. Estas capacidades se dividen en tres categorías: detección del robot, planificación de la trayectoria y control [[Sharir, 1989](#)].

2.3. Detección robótica

Es una rama de la robótica que pretende dotar a los robots de capacidades sensoriales, de modo que el robot pueda realizar estas de manera como lo hacen los seres vivos, en particular animales y humanos. El proceso de detección proporciona principalmente las facilidades para que el robot vea, toque y oiga, y a partir de esta información el robot se mueva. Para que esto suceda los algoritmos incorporados al robot utilizarán información del entorno. Por lo tanto, los robots autónomos deben tener incorporados una serie de sensores para poder realizar sus tareas y los datos sensoriales registrados deben ser analizados y transformados en una forma numérica para que puedan ser procesados por el mismo robot.

2.4. Planificación de trayectorias

La planificación de trayectorias o planificación de movimientos es un componente esencial en la navegación de un robot móvil [Qin et al., 2004]. El problema se enfoca a encontrar trayectorias entre un entorno sin colisiones a lo largo de un camino donde un robot se mueve desde su posición inicial a un destino final. Así mismo, la trayectoria estimada debe incluir las restricciones de movilidad del robot y los límites del mapa para la seguridad del robot. Entre las aplicaciones de la planificación de trayectorias podemos mencionar, la capacidad de que un robot logre conseguir su objetivo de manera autónoma o la planificación de movimientos de manipuladores en la industria para servicios o fabricación.

En Robótica, la planificación de trayectorias se considera un problema importante, ya que su complejidad aumenta exponencialmente con la dimensión del espacio de configuración o del entorno de trabajo.

La representación del entorno, por lo tanto, puede ser conocida o puede ser descubierta por métodos de localización y mapeo. Así, dependiendo de la información que se disponga del entorno, los problemas de planificación de trayectorias pueden clasificarse en dos categorías: estáticos y dinámicos [Hwang and Ahuja, 1992]. En un problema del tipo estático toda la información del entorno es conocida, es decir se conoce exactamente la posición de los obstáculos, y a partir de esta información el movimiento del robot se planifica. Caso contrario ocurre con los problemas del tipo dinámico, en estos solo se tiene información parcial del entorno y el movimiento del robot se planificará conforme este se mueve y logre obtener más

información para continuar con la ruta planificada.

Cada una de estas categorías anteriores podría dividirse a su vez en dos subcategorías basadas en el grado de conocimiento del robot sobre toda la información de los entornos circundantes.

- Planificación de la trayectoria del robot en un entorno claramente conocido en el que el robot ya conoce el mapa del entorno antes de empezar a moverse.
- Planificación de la trayectoria del robot en un entorno parcialmente conocido o incierto en el que el robot no tiene el conocimiento del entorno.

La Figura 2.1 muestra la clasificación de este problema de planificación de trayectorias desde el punto de vista del conocimiento del entorno.

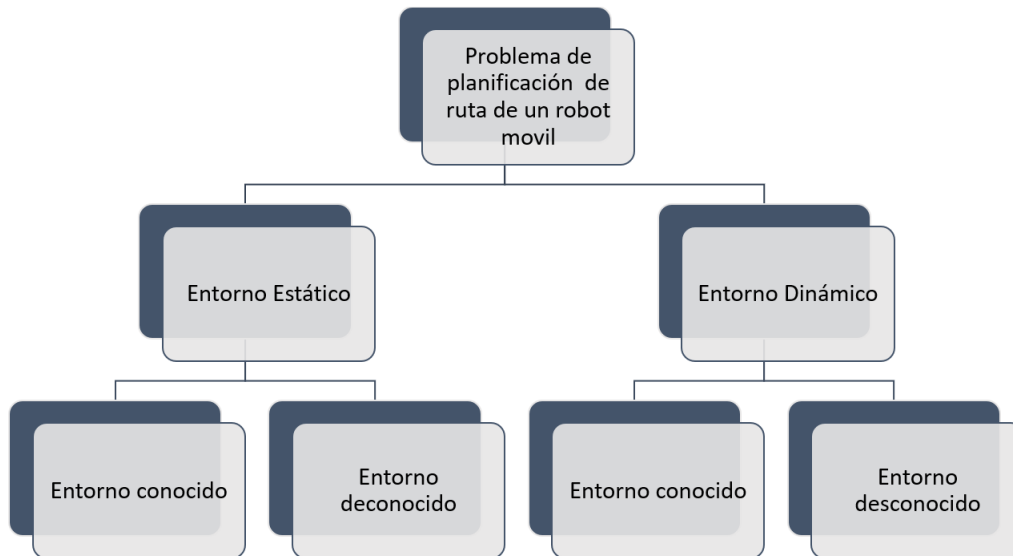


Figura 2.1. Clasificación del problema de planificación de trayectorias de los robots móviles.

- **Planificación de trayectorias en un entorno estático conocido:** En un entorno estático conocido, el robot debe conocer toda la información del entorno antes de empezar a moverse. Por lo tanto, el robot puede calcular la trayectoria libre de colisiones antes de la misión.
- **Planificación de trayectorias en un entorno dinámico conocido:** En este tipo de entorno el robot tiene conocimiento de los objetos en movimiento, es decir que el robot es capaz de obtener información sobre su ubicación, velocidad aceleración, etc.

Esta información puede recibirse de cada objeto en movimiento o de un sistema de comunicación.

- **Planificación de trayectorias en un entorno estático y desconocido:** En un entorno estático desconocido, la navegación del robot es más difícil que en entorno estático conocido. Esto se debe a la incertidumbre de la información recibida de los sensores del entorno. Por lo tanto, no se puede obtener el resultado óptimo y el robot tiene que utilizar la información local para calcular la trayectoria durante el movimiento.
- **Planificación de trayectorias en un entorno dinámico y desconocido:** Los problemas de planificación de trayectorias en entornos dinámicos y desconocidos son los más complejos. En estos entornos dinámicos desconocidos, el robot no puede utilizar ninguna técnica de planificación de trayectorias globales y la planificación de trayectorias locales es la mejor opción. El robot tiene que estar dotado con una gran cantidad de sensores para adquirir la información del entorno y hacer la planificación de la trayectoria en línea en tiempo real. El tiempo de planificación del robot debe ser corto, ya que el robot necesita un intervalo de tiempo suficiente para ajustar su movimiento y evitar los obstáculos que se presenten.

En los problemas de planificación de trayectorias en entornos conocidos, la trayectoria resultante debe ser corta, minimizando la distancia entre los puntos e incluye la optimización de tiempo, energía y suavidad. La distancia de recorrido es uno de los criterios de optimización más típicos. Un camino más corto entre la posición de partida y la de llegada permite un desplazamiento más rápido y minimiza el consumo de energía. Sin embargo, la mayoría de los enfoques de planificación de trayectorias convencionales no tienen en cuenta la seguridad y la suavidad de la trayectoria. La restricción de seguridad es, por lo tanto, un factor importante tanto para el robot como para los objetos que lo rodean.

El grado de dificultad de la planificación de trayectorias robóticas varía mucho en función de un par de factores que se mencionan a continuación:

- Si toda la información relativa al obstáculo (tamaño, ubicación, movimiento, etc.) se conoce antes de empezar a mover el robot.
- Si se trata de obstáculos dinámicos que se mueven o de obstáculos estáticos que permanecen en un lugar mientras el robot se mueve.

Los diferentes escenarios de los problemas de planificación de trayectorias se muestran en la Tabla 2.1.

Tabla 2.1. Diferentes escenarios para el problema de planificación de trayectorias.

Tipos de entorno	Obstáculos	
	Estáticos	Dinámicos
Totalmente conocido	Caso I	Caso II
Parcialmente conocido	Caso III	Caso IV
Completamente desconocido	Caso V	Caso VI

El caso I todos los obstáculos están fijos en su posición y todos los detalles sobre los obstáculos se conocen a priori. Los problemas de planificación de trayectorias, especialmente en esta categoría, suelen resolverse de esta forma:

- Definir un gráfico para representar la geometría del entorno (representación geométrica).
- Realizar una búsqueda de trayectorias para encontrar la conectividad entre los nodos que contienen el punto de partida y el punto de destino.

La representación geométrica del entorno difiere según el enfoque que se utilice en el problema de planificación de trayectorias en un entorno estático conocido. Hay tres enfoques comúnmente utilizados en estas condiciones.

- Roadmap method: es un método que proporciona un gráfico de red de posibles trayectorias en un determinado entorno basado en espacios libres y ocupados. En este método, los nodos y las conexiones se crean basándose en los parámetros establecidos en el algoritmo Roadmap method. El número de nodos es específico del problema y también afecta el rendimiento del algoritmo cuando es sometido a entornos complejos. La red de nodos conectados es utilizada por el algoritmo para encontrar una trayectoria libre de colisiones para el robot móvil. [[Santiago et al., 2017](#)]
- Método de Descomposición por celdas: el método de descomposición de celdas son soluciones aplicables a la planificación de trayectorias. La idea de estos métodos es encontrar regiones libres de obstáculos, y construir un gráfico de adyacencia para ellas. [[LaValle, 2006](#)]

- Método del campo potencial: el método de campos potenciales tiene sus fundamentos en la interacción de cargas eléctricas, donde un obstáculo presenta una carga contraria al robot y la meta u objetivo tiene la misma carga que el robot, con esto el robot se aleja de los obstáculos mientras que se sentirá atraído a la meta. [Khatib, 1986]

Estos tres enfoques resuelven los problemas básicos de planificación de trayectorias del “Caso I” y en otras situaciones se han introducido otras técnicas de planificación de trayectorias.

2.4.1. Enfoques de planificación de trayectorias

Los enfoques de planificación de trayectorias pueden clasificarse en dos categorías basadas en la complejidad y alcance [Patle et al., 2019; Qin et al., 2004; Hacene and Mendil, 2013], por un lado los métodos clásicos y por otro los heurísticos. Los métodos clásicos tratan de encontrar un camino óptimo si existe o prueban que no hay solución. Los métodos heurísticos intentan encontrar una solución mejor en poco tiempo, pero no garantizan encontrar una solución siempre [Hwang and Ahuja, 1992]. Algunos ejemplos de enfoques para la clasificación anterior pueden ser los siguientes:

1. Enfoque clásico

- Método de mapa de camino (Roadmap).
- Descomposición de celdas.
- Campos potenciales artificiales.
- Programación matemática.

2. Enfoque heurístico.

- Redes neuronales.
- Algoritmo genético.
- Optimización por enjambre de partículas.
- Búsqueda de cuco.
- Recocido simulado.

2.4.2. Roadmap

El método de Roadmap es uno de los primeros métodos de planificación de rutas que se han empleado ampliamente para resolver el problema del camino más corto. Este enfoque depende del concepto de espacio de configuración (C_{espacio}) y de camino continuo. En este enfoque, se utiliza el (C_{espacio}) y la característica clave de este enfoque es la construcción de una hoja de ruta. El método de Roadmap se basa en la conectividad del espacio libre del robot en forma de una red de curvas en una dimensión (líneas rectas). Este conjunto de líneas rectas que conectan dos nodos de diferentes obstáculos poligonales se encuentran en el espacio libre (C_{libre}) representan la hoja de ruta. Se dibujan todos los segmentos que conectan un vértice de un obstáculo con un vértice de otro sin entrar en el interior de ninguno de los obstáculos poligonales.

Si se puede encontrar un camino continuo en el espacio libre de la hoja de ruta, los puntos inicial y de meta se conectan a este camino para llegar a la solución final o un camino libre. Si se encuentra más de un camino continuo y el número de nodos del grafo es relativamente pequeño, se suele utilizar el algoritmo del camino más corto de Dijkstra para encontrar el mejor camino.

Uno de los métodos más antiguos de planificación de trayectorias es el método del gráfico de visibilidad, introducido por N.J. Nilsson a principios de 1969. El gráfico de visibilidad es el conjunto de líneas en el espacio libre que conecta una característica de un objeto con la de otro. En su forma principal, estas características son vértices de obstáculos poligonales. En la Figura 2.2 se muestra un ejemplo básico de construcción de un gráfico de visibilidad. Una de las desventajas del grafo de visibilidad es que los caminos más cortos resultantes tocan los obstáculos en los vértices o incluso en las aristas, por lo que no es seguro.

2.4.3. Descomposición de Celdas

El método de descomposición de celdas es muy utilizado en la literatura en problemas de planificación de trayectorias. La idea básica de este enfoque es encontrar una trayectoria entre el punto inicial y el punto de destino que se puede determinar subdividiendo el espacio libre de la configuración del robot en regiones más pequeñas llamadas celdas. En esta representación del entorno se reduce el espacio de búsqueda o, en otras palabras, el C_{libre} se descompone en celdas. Después de este proceso de descomposición, se utiliza un operador de búsqueda para

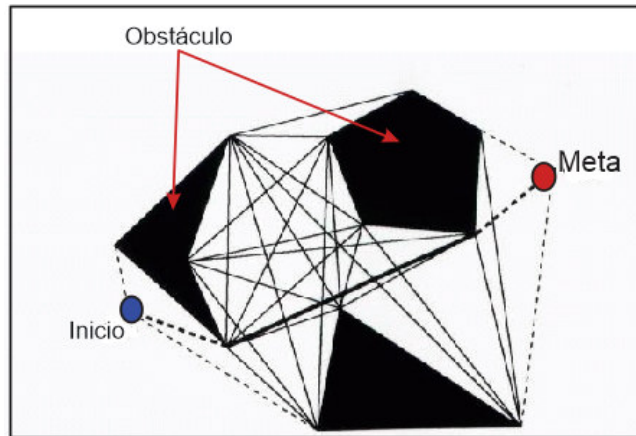


Figura 2.2. Planificación de trayectoria basada en la hoja de ruta y Gráfico de visibilidad [Patle et al., 2019] ©.

encontrar una secuencia de celdas libres de colisiones desde el punto de partida hasta el punto objetivo. Este gráfico de conectividad se genera según las relaciones de adyacencia entre las celdas, donde los nodos representan las celdas en el espacio libre, y los enlaces entre los nodos muestran que las celdas correspondientes son adyacentes entre sí. A partir de este gráfico de conectividad, se puede determinar un camino o canal continuo simplemente conectando las celdas libres adyacentes desde el punto inicial hasta el punto de destino.

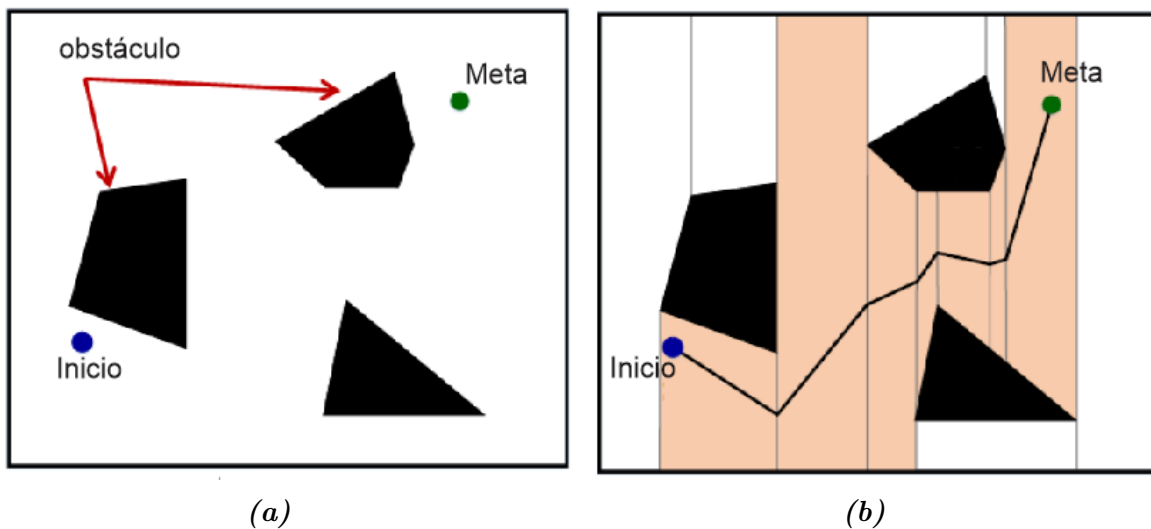


Figura 2.3. Descomposición de celdas a) Espacio de configuración. b) Trayectoria generada al conectar el gráfico de conectividad en el espacio libre [Patle et al., 2019] ©.

2.4.4. Campos Potenciales Artificiales

Este método consiste en modelar el robot como una partícula que se mueve bajo la influencia del campo de potencial artificial que generan los obstáculos y el objetivo del espacio de configuración. El método del campo de potencial se basa en la idea de las fuerzas de atracción/repulsión; la fuerza de atracción tiende a tirar del robot hacia la configuración de la meta, mientras que la fuerza de repulsión empuja al robot lejos de los obstáculos, como se puede observar en la Figura 2.4. En cada paso, la fuerza potencial total generada por la función potencial en la posición actual del robot, cambia la dirección y mueve al robot de forma incremental hacia la siguiente posición. De este modo, la información calculada se utiliza directamente en la planificación de la trayectoria del robot y no se desperdicia potencia de cálculo.

El concepto de campo de potencial artificial fue introducido por primera vez por Khatib como un enfoque para evitar colisiones locales [Khatib, 1986], la cual se aplica cuando el robot no tiene conocimiento a priori sobre el entorno, pero si puede percibir el entorno circundante durante la ejecución del movimiento. El único inconveniente de este método es el problema del mínimo local, ya que este enfoque es local y no global (es decir, se considera el mejor curso de acción inmediato). El robot puede quedarse atascado en un mínimo local del campo potencial en lugar de su mínimo global, que es el destino. Esto se denomina generalmente “punto muerto” en la planificación de trayectorias de robots. Escapar del mínimo local es posible construyendo una función de campo potencial que no contenga ningún mínimo local o acoplando este método con algunas otras técnicas heurísticas que puedan escapar del mínimo local.

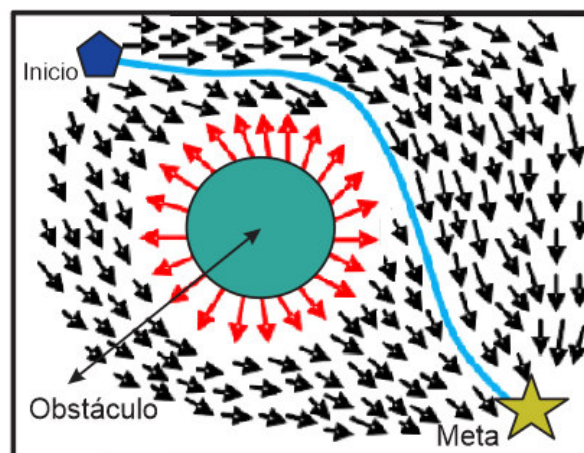


Figura 2.4. Campos potenciales Artificiales [Patle et al., 2019] ©.

2.4.5. Programación Matemática

La programación matemática es otro enfoque convencional de planificación de trayectorias. Este enfoque representa las restricciones como un conjunto de desigualdades para evitar obstáculos en el espacio de configuración. El problema de planificación de trayectorias se formula como un problema de optimización matemática, la cual encuentra una solución que define una curva entre el punto de partida y el punto de llegada, minimizando ciertos parámetros. Dado que este tipo de problemas de optimización son no lineales y con muchas restricciones de desigualdad, se utilizan métodos numéricos para encontrar una solución [Shibata and Ohnishi, 1992].

2.4.6. Algoritmo Genético (AG)

El algoritmo genético fue introducido por John Holland en la década de 1960 y anexado al proceso de evolución biológica para resolver los problemas. Esta técnica se aplica con éxito en los problemas de optimización, como el clásico problema del viajante comprador. Se han realizado varios estudios sobre el AG en problemas de planificación de trayectorias [Zeqing et al., 2008; Cao et al., 2016]. El AG es uno de los algoritmos más utilizados en planificación de trayectorias debido a su capacidad de optimización global [Jianguo et al., 2012]. La planificación de trayectorias mediante AG puede evitar obstáculos y planificar trayectorias en entornos desconocidos, pero al aumentar la longitud de los individuos aumenta la codificación binaria y esto provoca una baja eficiencia de la memoria ocupada o recursos computacionales [Sugihara and Smith, 1997].

2.4.7. Optimización por Enjambre de Partículas (PSO)

La optimización por enjambre de partículas (PSO), es un algoritmo basado en poblaciones, inspirado en el comportamiento de ciertos animales, se utiliza para encontrar el mínimo global mediante el uso del comportamiento partículas y así obtener la influencia de los comportamientos sociales y cognitivo del enjambre. En el PSO, las partículas se definen en función de su posición y su velocidad en el espacio de búsqueda. Las partículas son atraídas hacia posiciones en el espacio de búsqueda, los cuales representan su mejor hallazgo personal y el mejor hallazgo del enjambre (posiciones mejor local y posiciones globales) [Stacey et al., 2003].

Sin embargo, el PSO tiene sus propias debilidades en términos del control de los parámetros, convergencia prematura, y falta de ajuste dinámico, esto da como resultado la incapacidad de escalar la solución [Ratnaweera et al., 2004]. Para superar estos inconvenientes asociados a la PSO, se han introducido algunas versiones modificadas para la planificación de trayectorias y la navegación de robots móviles. Pero en muchos estudios, se ha demostrado que PSO tiene mejor rendimiento que el AG [Kennedy and Spears, 1998; Pugh and Martinoli, 2006].

2.4.8. Redes Neuronales Artificiales

Las redes neuronales artificiales consisten en el estudio de la comprensión del funcionamiento del cerebro humano. Las redes neuronales han sido utilizadas ampliamente en problemas de reconocimiento de patrones y optimización, debido a su capacidad de proporcionar soluciones óptimas y sencillas. Las redes neuronales se definen como el estudio de nodos adaptables que se ajustan para resolver repetidamente problemas basados en el conocimiento experimental, obtenido a partir del proceso de aprendizaje. Las características generales de funcionamiento de la red se definen en función de los potenciales y de la naturaleza de la interconexión de las neuronas. Los métodos basados en redes neuronales pueden clasificarse en función de dos factores:

- La configuración de sus capas: monocapa, multicapa, capa competitiva.
- Su metodología de entrenamiento: entrenamiento supervisado, entrenamiento no supervisado, pesos fijos (sin entrenamiento), y entrenamiento autosupervisado.

Las redes neuronales han sido satisfactoriamente aplicadas a la navegación de robots y consisten en tres etapas [Zou et al., 2006]:

- Interpretación de los datos sensoriales.
- Evasión de obstáculos.
- Planificación de trayectorias

Los enfoques híbridos de redes neuronales combinados con otros métodos basados en la inteligencia artificial como lógica difusa o enfoques evolutivos son más apropiados para abordar el problema de la navegación robótica en aplicaciones del mundo real[Zou et al., 2006].

2.5. Clasificación de los enfoques de planificación de trayectorias

Los enfoques de planificación de trayectorias pueden clasificarse de nuevo en dos categorías: enfoques locales y globales. Los problemas de planificación de trayectorias globales tienen información completa sobre el entorno, es decir el mapa del entorno es conocido por completo además se sabe las posiciones de la localización de los obstáculos. La planificación de trayectorias global por lo general son convergentes en entornos estáticos. Sin embargo, pueden perder eficacia si aparecen obstáculos dinámicos, como resultado de no incluir la información del obstáculo en el mapa conocido. Debido a esto no se puede garantizar un movimiento libre de colisiones.

Por otro lado, los métodos de planificación de trayectorias locales utilizan directamente datos sensoriales del robot y no tiene ningún conocimiento a priori sobre el entorno. Dado que no se dispone de un mapa del entorno definido, todas las acciones de control se basan en la percepción del entorno del robot. Estos algoritmos demandan un bajo esfuerzo computacional.

Una comparación general de algunos de los métodos de planificación de trayectorias se puede observar en la Tabla 2.2

Tabla 2.2. Comparación de ventajas y desventajas de algunos métodos de planificación de trayectorias.

Algoritmo	Ventaja	Desventaja
Campos potenciales	Implementación en tiempo real artificiales	Mínimos locales 2D o 3D
Descomposición de celdas	Implementación en 2D o 3D	Costo computacional
Heurística	Menos tiempo de búsqueda	Sintonización de sus parámetros
Gráfico de visibilidad	Trayectoria completa 2D	No óptimo, se necesita un sensor de largo alcance
Diagrama de Voroni	Trayectoria de longitud óptima, 2D o 3D	Trayectoria muy cercana a los obstáculos

En el siguiente capítulo se explorará a detalle el método de campos potenciales,

realizando una revisión detallada de la interacción de las fuerzas que permiten calcular la trayectoria para un robot móvil.

CAPÍTULO 3

Campos Potenciales

3.1	Introducción	23
3.2	Campos Eléctricos	24
3.3	Generación de Trayectorias mediante potencial eléctrico.	26
3.4	Campo de atracción electrostático	27
3.5	Campo repulsivo electrostático.	29
3.6	Campo potencial Total	31
3.7	Problema de mínimos locales	34

3.1. Introducción

Uno de los métodos empleados en la planificación de trayectorias [Khatib, 1986] utilizado en este trabajo de tesis es el uso de campos potenciales artificiales (APF), como se ha mencionado anteriormente. La idea de los APF para la planificación de trayectorias de robots se introdujo a principios de los años ochenta y fue explotada por varios científicos, destacándose

por ser uno de los métodos para aplicaciones en tiempo real. El APF es una función vectorial cuyo gradiente se utiliza para calcular y estimar coordenadas de posición, las cuales se traducen en un par de torsión en los neumáticos, al cual será sometido el robot, obligándolo a alcanzar su meta mientras evita los obstáculos.

Los APF dependen de la meta y de la geometría o mapa del entorno donde se desplazará el robot, teniendo sus valores máximos de repulsión en los obstáculos y su mínimo global en la meta. El uso del APF para realizar un determinado movimiento implica que la trayectoria del robot no se conoce ni se calcula de antemano. Esto significa que el robot “elige” de forma autónoma o en el peor de los casos en forma “reactiva” su camino para alcanzar su posición objetivo.

Algunas características atractivas del uso de los APF son:

1. Implementación en tiempo real.
2. Aplicable a robots redundantes y no redundantes.
3. Incorporación del modelo real del robot.

En este capítulo, se estudian entonces los fundamentos y conceptos generales para la planificación de trayectorias empleando APF, en el escenario de evasión dinámica de obstáculos.

3.2. Campos Eléctricos

La interacción electrostática entre las partículas puede interpretarse mediante el concepto de campo eléctrico. Bajo esta premisa, la interacción entre dos agentes con una carga específica se puede describir en términos del efecto que produce una carga en el espacio libre, es decir, que cualquier otro agente con cualquier tipo de carga, generara una fuerza de origen eléctrico. Dicho de otra forma, puede decir que se creó un campo eléctrico definido por la primera carga, que afecta puede afectar a cualquier otra carga en el espacio.

De igual manera, el segundo agente cargado crea un campo eléctrico que afecta al primero. Para probar la existencia de un campo eléctrico se toma un agente pequeño cargado el cual se ubica en cualquier punto. Si este agente experimenta una fuerza de origen eléctrico, entonces existe un campo eléctrico. A esta pequeña carga se la llama carga de prueba y comúnmente se le denota como q_0 .

Para la definición del campo eléctrico se puede utilizar una carga de prueba positiva o negativa, el signo únicamente contribuirá con la orientación de las líneas de fuerza y la magnitud de la carga con su intensidad. La Figura 3.1b muestra un ejemplo ilustrativo de un campo eléctrico producido por una carga puntual.

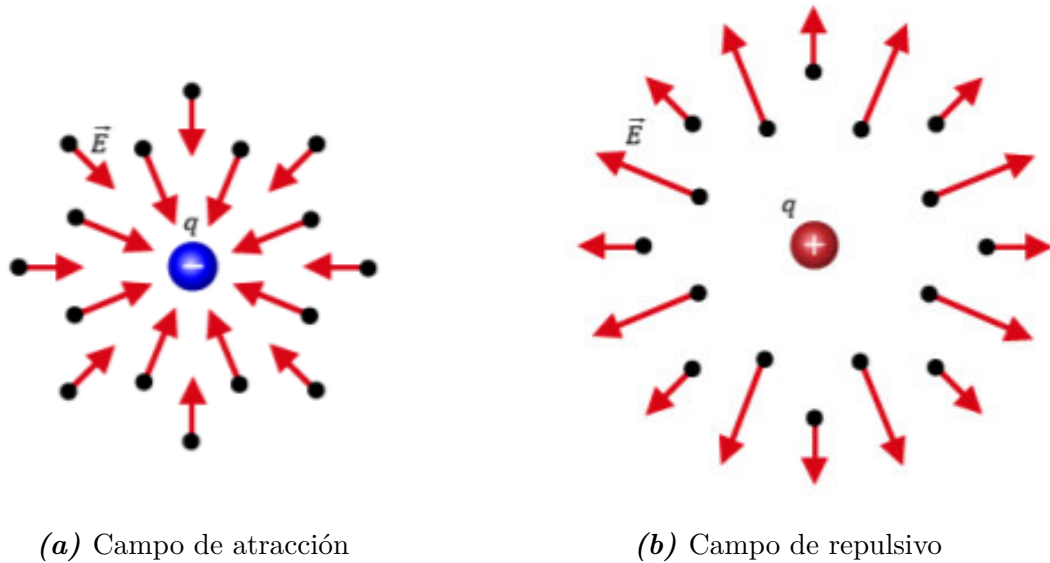


Figura 3.1. Representación de las líneas de fuerza generadas a partir de a) carga puntual negativa y a una b) carga puntual positiva CC .

Se define el campo eléctrico como la fuerza que actúa sobre la carga de prueba por unidad de carga,

$$\vec{E} = \frac{\vec{F}}{q_0} \quad (3.1)$$

Se puede observar que $\vec{F} = q\vec{E}$, la fuerza sobre una carga positiva tiene la dirección del campo eléctrico, mientras que la fuerza sobre cargas negativas tiene dirección opuesta al campo, las unidades de la intensidad del campo eléctrico son $[\text{N C}^{-1}]$.

Para calcular la magnitud del campo eléctrico debido a una carga puntual se puede utilizar la relación

$$\vec{E} = k \frac{q}{r^2} \hat{a}_r \quad (3.2)$$

Donde \hat{a}_r es el vector radial emanado de la carga prueba.

3.3. Generación de Trayectorias mediante potencial eléctrico.

El método de campos potenciales es una de las técnicas utilizadas para la generación de trayectorias para robots móviles, la cual, por ser eficiente, robusta y de implementación relativamente simple, permite realizar controles de locomoción por posición o desplazamiento con base en la formulación matemática completamente determinista.

Khatib [1985] propone que los obstáculos y el robot tengan una carga eléctrica del mismo signo. Esto con la finalidad de generar una fuerza de repulsión, mientras que la meta tiene asociada una carga eléctrica de signo opuesto, para atraer el robot al punto destino.

La tarea asignada al robot consistirá en llegar a su objetivo a partir de la planificación de trayectoria generada dinámicamente, apoyándose en el campo eléctrico conservativo sobre el cual se desplaza el robot. El proceso se inicia definiendo la posición del robot por el vector,

$$\vec{X} = [x, y]^T \in \mathbb{R}^2, \quad (3.3)$$

y una función potencial de valor real, continua y, por tanto, diferenciable,

$$U(\vec{X}) : \mathbb{R}^2 \rightarrow \mathbb{R}. \quad (3.4)$$

El valor de una función potencial puede verse como energía almacenada, y por lo tanto, el gradiente del potencial nos genera el vector campo eléctrico, como se establece en las leyes de Maxwell.

En seguida, se observa que en presencia de una carga externa q_o se puede generar una fuerza electrostática por efecto del campo eléctrico del entorno. Lo cual permite establecer que el campo eléctrico se obtiene como $\vec{E} = -\nabla U$, y la fuerza a su vez se calcula por

$$\vec{F} = q_o \vec{E} = -q_o \vec{\nabla} U \quad (3.5)$$

. Note que el operador gradiente está definido por

$$\vec{\nabla} U(\vec{X}) = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right]^T U(\vec{X}). \quad (3.6)$$

Con el objeto de que el robot sea atraído hacia el objetivo y evite exitosamente los obstáculos

del entorno circundante, se debe estimar un campo eléctrico resultante total \vec{E}_T que incluya la acción de los campos de repulsión con los obstáculos y de atracción hacia el objetivo.

La fuerza total que experimenta la carga externa (representada por el robot), inducirá por medio de la segunda Ley de Newton una aceleración, lo cual propiciará el desplazamiento por medio de las ecuaciones de cinemática tradicional, $\vec{x}(t) = \frac{1}{2}\vec{a}t^2 + \vec{v}_o t + \vec{x}_o$. El campo eléctrico resultante viene entonces dado por

$$\vec{E}_T = \vec{E}_a + \vec{E}_r, \quad (3.7)$$

donde, \vec{E}_a y \vec{E}_r son los campos de atracción y repulsión, respectivamente.

Dado que el robot está representado por una carga q_o^+ del mismo signo que los obstáculos y signo opuesto a la posición destino, se obtiene la fuerza de atracción hacia el objetivo y repulsión hacia los obstáculos, como se muestra en la Figura 3.2.

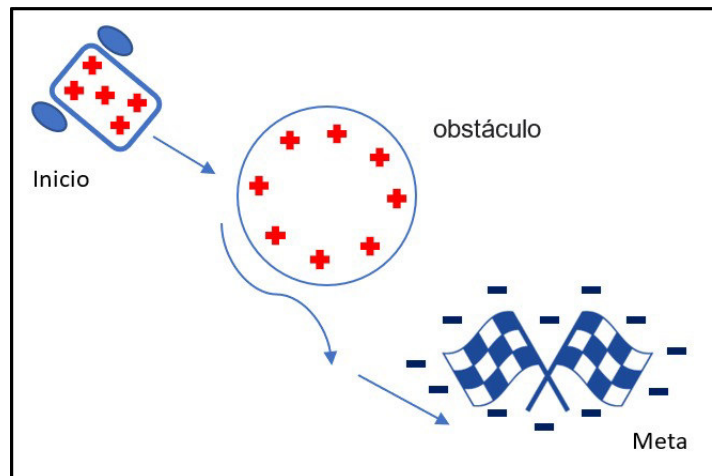


Figura 3.2. La carga negativa atrae al robot y la carga positiva lo repele, lo que da como resultado una ruta óptima libre de obstáculos.

3.4. Campo de atracción electrostático

El campo de atracción electrostática es el encargado de hacer que el robot sea atraído a la meta, dependerá de la configuración del robot y del destino. Dicho campo debe cumplir ciertos criterios a la hora de seleccionar la función del potencial $U_a(\vec{X})$.

El potencial de atracción debe ser una función convexa, diferenciable y real además

debe tener su mínimo global en la configuración destino, típicamente se emplean las funciones cuadráticas o cónicas, como se puede observar en la Figura 3.3. Las funciones cuadráticas

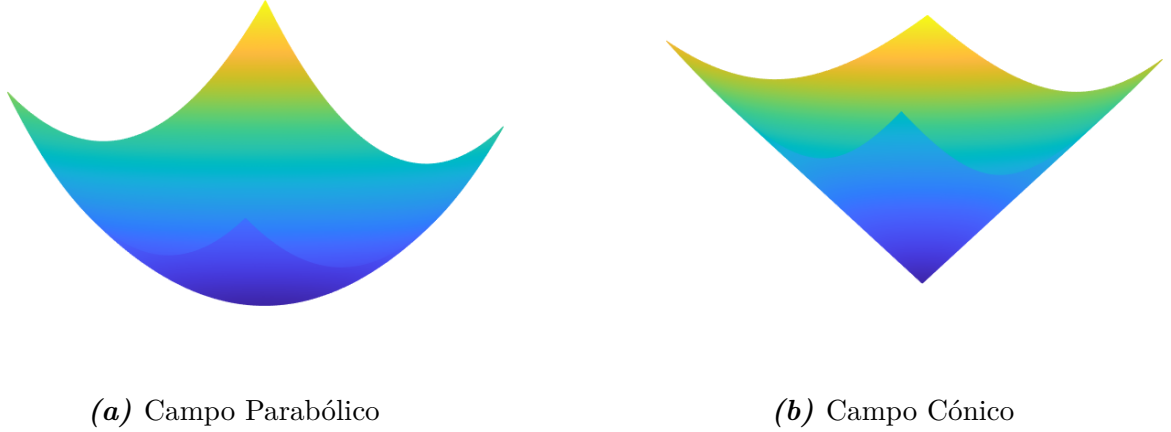


Figura 3.3. Campos potenciales de atracción.

pueden tomar valores muy elevados lejos de la configuración destino, pero son funciones suaves en las proximidades del destino, Figura 3.3a, por otro lado la configuración cónica es proporcional a la distancia entre la configuración del robot y del destino, sin embargo en las proximidades del destino cambia de forma abrupta, Figura 3.3b.

Una opción es utilizar una función cuadrática del tipo,

$$U_a(\vec{X}) = \frac{1}{2}(\vec{X} - \vec{X}_g)^\top \mathbf{Q}(\vec{X} - \vec{X}_g), \quad (3.8)$$

la cual es una función parabólica, donde, \mathbf{Q} es una matriz función de cargas electrostáticas, la cual matemáticamente debe ser simétrica y definida positiva y se encargara de que la función parabólica no tenga valores muy elevados lejos de la configuración destino, $\vec{X}_g = [x_g, y_g]^\top$ es la coordenada de la posición meta.

Por lo tanto, el campo de atracción se define como

$$\vec{E}_a = -\vec{\nabla}U_a = -\vec{\nabla}\left\{\frac{1}{2}(\vec{X} - \vec{X}_g)^\top \mathbf{Q}(\vec{X} - \vec{X}_g)\right\}, \quad (3.9)$$

$$\vec{E}_a = -\mathbf{Q}(\vec{X} - \vec{X}_g). \quad (3.10)$$

La fuerza de atracción entre el robot móvil y el objetivo viene finalmente dada por

$$\vec{F}_a(\vec{X}) = -q_o^+ \vec{E}_a. \quad (3.11)$$

En la Figura 3.4 se puede observar el comportamiento de la fuerza de atracción del campo potencial generado en la Ec. (3.8), donde todas las líneas del campo están dirigidas hacia la configuración destino.

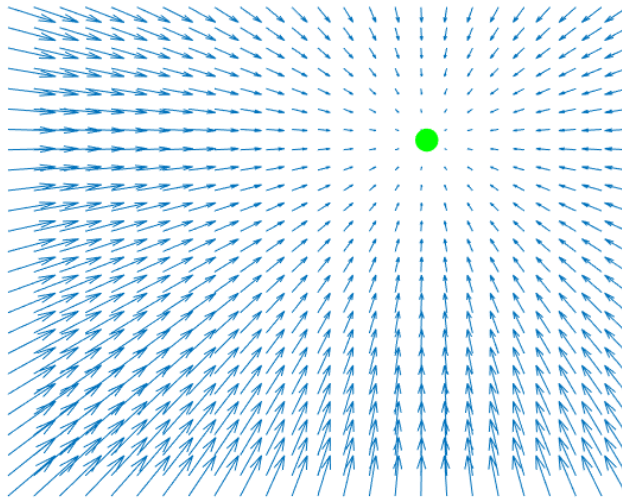


Figura 3.4. Representación gráfica de la fuerza de atracción hacia la meta (círculo verde).

3.5. Campo repulsivo electrostático.

La definición de potencial repulsivo consiste en crear una barrera de potencial alrededor de cada obstáculo que no pueda ser atravesada por el robot. Esto implica que el potencial repulsivo debe ser una función no negativa, continua y diferenciable. Además, es deseable que el potencial repulsivo no afecte al movimiento del robot cuando esté lo suficientemente lejos de los obstáculos.

Dado que físicamente la proximidad de dos cargas llevaría a generar una fuerza electrostática muy grande, se considera en el análisis de interacción la aproximación hasta un cierto punto de seguridad entre las cargas, representadas por el robot y los obstáculos, sin provocar inconsistencias físicas por singularidades en las ecuaciones. Se pueden emplear

distintas formas para el diseño de este potencial repulsivo. Usaremos una función potencial continua y suave para definir un conjunto de obstáculos.

La función potencial eléctrico de referencia para n obstáculos se calcula de la siguiente manera,

$$U_r(\vec{X}) = \eta \sum_{i=1}^n \frac{Q_r}{R(\vec{X})}, \quad (3.12)$$

donde, $R(\vec{X}) = \|\vec{X} - \vec{X}_r\|_2$ es la distancia Euclidiana o radial entre dos cargas y Q_r son las cargas puntuales asociadas a cada obstáculo. El gradiente de este potencial de repulsión, $E_r(\vec{X}) = -\vec{\nabla}U_r(\vec{X})$, es quien en consecuencia genera la fuerza resultante de repulsión para n obstáculos que afectan a la carga prueba (robot).

Esta resultante de repulsión para n obstáculos dinámicos es definida por

$$F_r(\vec{X}) = K \sum_{i=1}^n \frac{q_o^- Q_r}{R^2} \left(\frac{x - x_{r_i}}{R} \hat{a}_x + \frac{y - y_{r_i}}{R} \hat{a}_y \right), \quad (3.13)$$

donde \hat{a}_x y \hat{a}_y son vectores unitarios en los ejes x y y , respectivamente. K es la constante de Coulomb y se define como $K = \frac{1}{4\pi\epsilon_o}$, siendo $\epsilon_o = 8.85418 \times 10^{-12}$ [F/m], la permitividad en el vacío. Un campo de repulsión puede representarse tal como se muestra en la Figura 3.5, donde cada obstáculo genera sus propias fuerzas de repulsión.

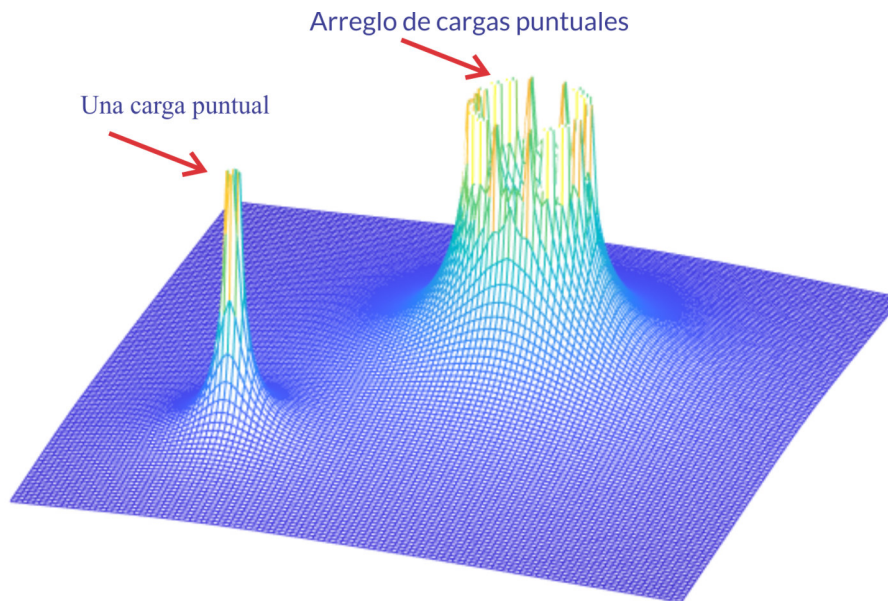


Figura 3.5. Campo de Potencial Repulsivo $U_r(\vec{X})$.

3.6. Campo potencial Total

El campo potencial total es la combinación de los campos potenciales de atracción y de repulsión Ec. (3.14),

$$U_T(\vec{X}) = U_a(\vec{X}) + U_r(\vec{X}), \quad (3.14)$$

el cual genera una función con características (continúa, diferenciable, monótona decreciente) que permitirán al robot llegar a la meta. El robot, al evolucionar en su entorno, estará afectado por la fuerza repulsiva de los obstáculos $\vec{F}_r(\vec{x})$ y la fuerza de atracción $\vec{F}_a(\vec{X})$ hacia el objetivo. De esta manera, la fuerza resultante se calcula por medio de

$$\vec{F}_T(\vec{X}) = -q_o^+ \vec{\nabla} U_a(\vec{X}) + \eta \sum_{i=1}^n \left(\frac{x - x_{r_i}}{R^{3/2}} \hat{a}_x + \frac{y - y_{r_i}}{R^{3/2}} \hat{a}_y \right), \quad (3.15)$$

donde $\eta = \frac{q_o^- Q_r}{4\pi\epsilon_o}$ es una constante de regularización que permite controlar la proximidad del robot hacia los obstáculos.

La función resultante es la función de costo que se desea maximizar (o minimizar) según los signos asignados a la Ec. 3.15. Para valores muy cercanos de cargas, la fuerza crecería enormemente, por lo cual, se puede agregar una función Heaviside $u_h(\vec{X})$ en el término de $F_r(\vec{X})$ para saturar la respuesta a distancias cortas. El campo potencial global para este estudio se muestra en la Figura 3.6.

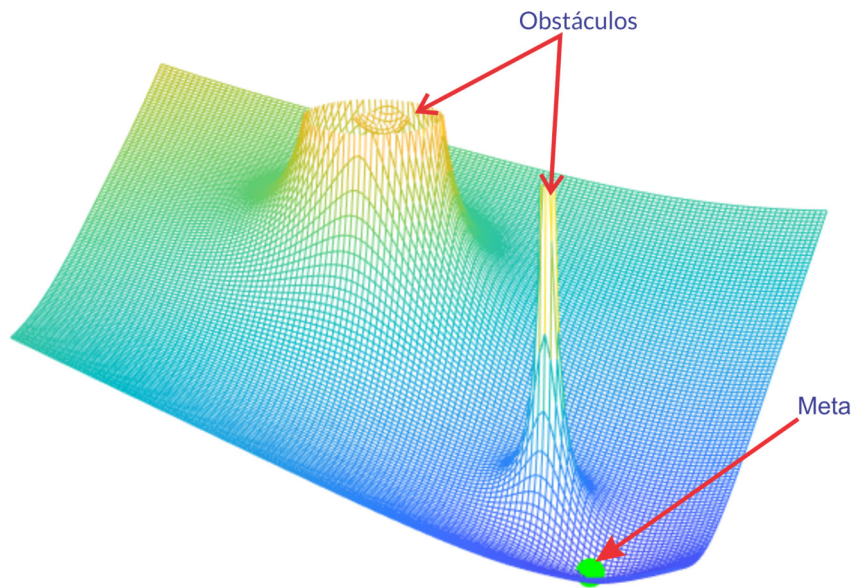


Figura 3.6. Campo de Potencial Electrostático Total.

Finalmente, el método APF en su forma general está descrito por el Algoritmo 1. Para observar el comportamiento del algoritmo, se planteó un escenario como el que se puede

Algoritmo 1 Método de los Campos de Potenciales Artificiales.

Input : $U_T(\vec{X})$, \vec{X}_g , $iter$, $Tolerancia$

Output: Coordenadas de la meta $[x^*, y^*]$

$\vec{X}_k \leftarrow [x, y]$

▷ Posición inicial del robot

$\alpha \leftarrow 0.5$

▷ Constante de amortiguamiento

while $Error > Tolerancia$ **do**

$\vec{X}_{k+1} = \vec{X}_k - \alpha \nabla U(\vec{X}_k) / \|\nabla U(\vec{X}_k)\|_2^2$

$Error = \|\vec{X}_{k+1} - \vec{X}_g\|_2^2$

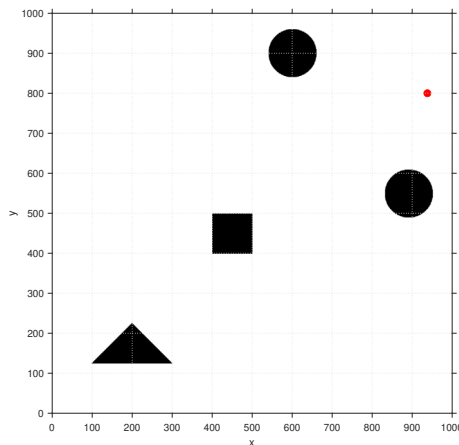
$iter \leftarrow iter + 1$

$\vec{X}_k \leftarrow \vec{X}_{k+1}$

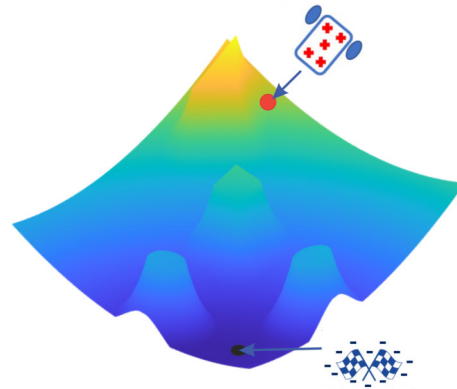
end

observar en la Figura 3.7a, donde se realizó un arreglo de cargas puntuales para obtener formas geométricas para los obstáculos.

El Potencial total del espacio de trabajo planteado, lo se observa en la Figura 3.7b, donde el máximo del potencial se encuentra en donde inicia el robot, y el mínimo global se encuentra en la meta.



(a) Entorno



(b) Función potencial

Figura 3.7. Espacio de trabajo. El cual puede ser sustituido por un mapa topológico.

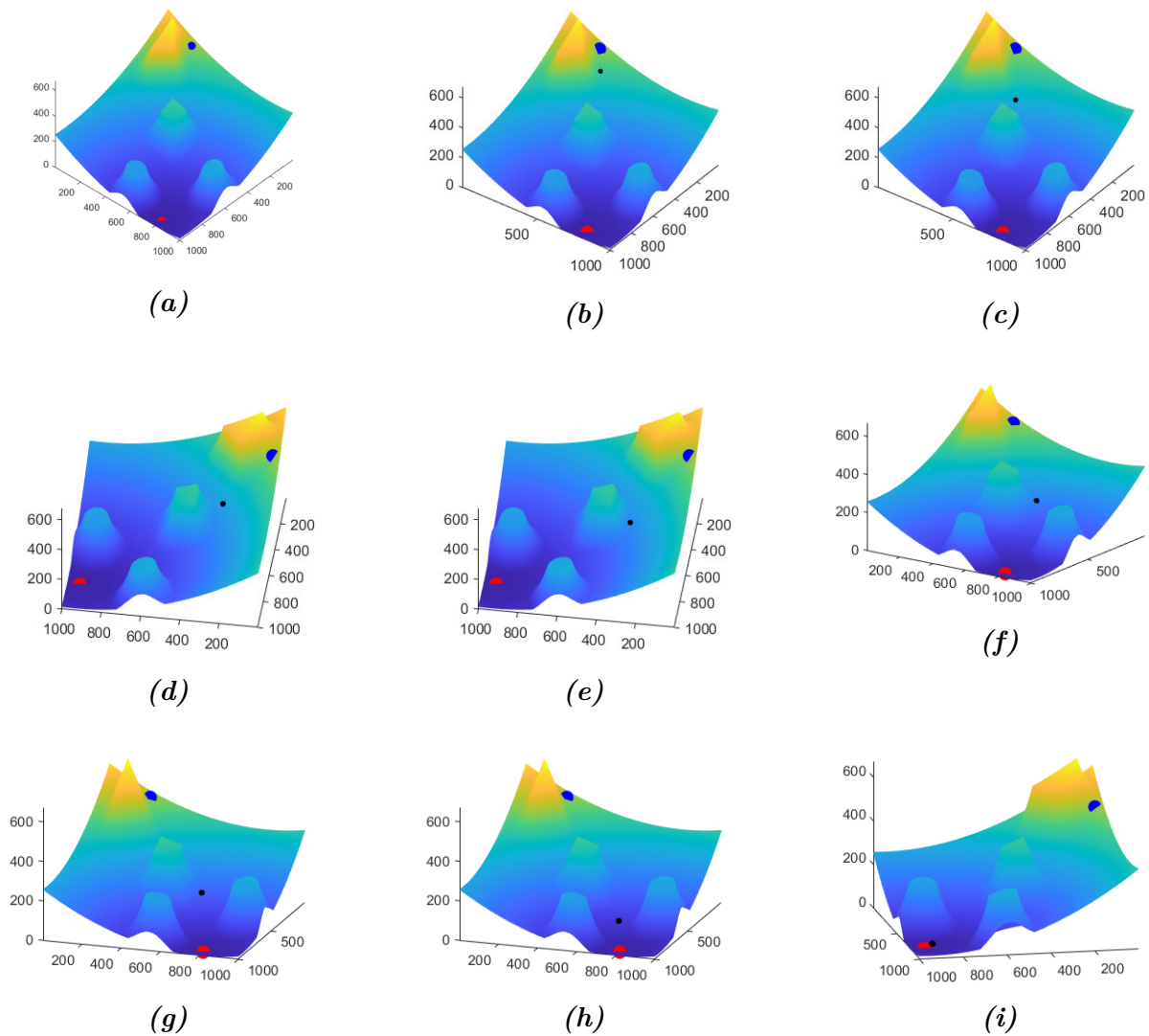


Figura 3.8. Evolución del Robot mediante la interacción de campos potenciales artificiales.

En la Figura 3.8 podemos observar la evolución del algoritmo, desde una posición inicial hasta la meta establecida, donde el robot está representado por la partícula de color negro, es evidente ver que logra evitar los obstáculos y conseguir su objetivo, pero esto lo consigue en un número de iteraciones de 320. En la próxima sección de esta tesis, abordaremos algoritmos con mayor exploración y explotación, logrando hacer que este número de iteraciones disminuya.

De igual forma, en la Figura 3.9 se observa la interacción de fuerzas y la ruta calculada por el algoritmo, el robot logra llegar del punto “a” al punto “b” evadiendo los obstáculos presentes en su camino.

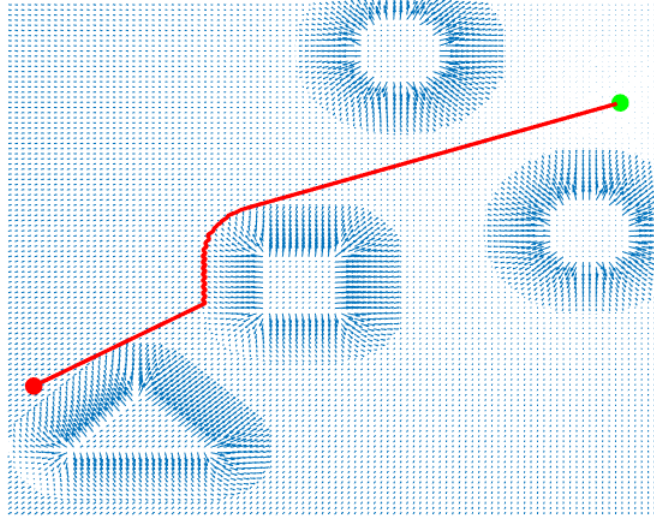


Figura 3.9. Interacción de fuerzas y ruta calculada, donde el punto rojo es el punto de partida del robot y el verde es la meta.

3.7. Problema de mínimos locales

El problema de mínimos locales se presenta, cuando el robot detiene su movimiento, debido a que el gradiente de la función en dicho punto es cero, por lo general esto sucede cuando la interacción de fuerzas de repulsión y de atracción se anulan [Bing et al., 2011]. Por lo general ocurre cuando la meta se encuentra entre el arreglo de obstáculos, también puede producirse cuando el obstáculo y la meta generan líneas de fuerza en la misma dirección, pero en sentido opuesto, es decir que el potencial de atracción puede ser igual al de repulsión, por lo tanto, la suma total de fuerzas es cero.

Para solucionar estos problemas producidos por mínimos locales, se emplean algoritmos donde la exploración y explotación sea mayor que la propuesta hecha por el algoritmo original (Gradiente descendente).

En el siguiente capítulo se discutirán las técnicas de optimización dependientes del gradiente y las metaheurística, con el objeto de utilizar las técnicas más robustas y estables bajo el criterio de la mejor exploración/explotación.

CAPÍTULO 4

Métodos de optimización

4.1	Introducción	36
4.2	Método de Newton	37
4.3	Método del Descenso del Gradiente	39
4.4	Método de Levenberg-Marquardt	40
4.5	Método de Dog-Leg	41
4.6	Funciones de Prueba	43
4.6.1	Validación del método de Newton	45
4.6.2	Validación del método del Descenso del Gradiente	47
4.6.3	Validación del método de Levenberg-Marquardt	49
4.6.4	Validación del método de Dog-Leg	51
4.7	Campos Potenciales con Levenberg-Marquardt y Dog-Leg	53

4.1. Introducción

Los adelantos en la computación han permiten actualmente a los científicos estudiar sistemas físicos, biológicos y sociales cada vez más complejos. La modelación matemática es una herramienta sistemática y poderosa para manejar la numerosa información que se requiere para entender dichos sistemas.

En las últimas décadas, se han multiplicado las ramas del conocimiento que usan la modelación matemática como parte de su metodología. Las aplicaciones de esta ciencia son numerosísimas: desde el estudio de las proteínas hasta la planificación de trayectorias en robots móviles.

La optimización matemática o también llamada programación matemática se define como la selección del mejor elemento, siguiendo algún criterio a partir de las alternativas disponibles. En este sentido, un problema de optimización consiste en maximizar o minimizar una función real y de esta forma encontrar el óptimo global, en otras palabras, los métodos de optimización buscan y seleccionan los mejores valores disponibles de función objetivo, esto se puede representar matemáticamente de la siguiente manera [Baquela and Redchuk, 2013] ;

Definición 4.1.1. Dada una función $f : \Omega \rightarrow \mathbb{R}$ de un conjunto Ω , se busca un elemento $\mathbf{x}_0 \in \Omega$, tal que $f(\mathbf{x}_0) \leq f(\mathbf{x})$ para todo $\mathbf{x} \in \Omega$ (“proceso de minimización”) o tal que $f(\mathbf{x}_0) \geq f(\mathbf{x})$ para todo $\mathbf{x} \in \Omega$ (“proceso maximización”).

En la actualidad algunas técnicas clásicas pueden resolver problemas de optimización con ciertas características específicas. Estos métodos clásicos son deterministas o dependientes del gradiente de la función de costo.

En este capítulo se abordarán métodos de optimización determinísticos aplicados a los campos potenciales para la planificación de trayectorias en robots móviles.

4.2. Método de Newton

El método de Newton es un método de optimización iterativo que se basa en aproximar la función a optimizar por medio de la serie de Taylor hasta orden 2.

Supongamos que se desea minimizar una función f de de una sola variable real x y además cada punto x^k podemos calcular $f(x^k), f'(x^k), f''(x^k)$. Podemos ajustar una función cuadrática a través de x que hace coincidir su primera y segunda derivada con la de la función f . Esta función cuadrática tiene esta definida como,

$$q(x) \approx f(x^k) + f'(x^k)(x - x^k) + \frac{1}{2}f''(x^k)(x - x^k)^2, \quad (4.1)$$

si definimos $q(x^k) = f(x^k), q'(x^k) = f'(x^k)$ y $q''(x^k) = f''(x^k)$, entonces en lugar de minimizar f , minimizamos su aproximación q . Donde la condición necesaria de primer orden para lograr dicha minimización de q es,

$$0 = q'(x) = f'(x^k) + f''(x^k)(x - x^k), \quad (4.2)$$

si $x = x^{k+1}$ se obtiene,

$$x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)} \quad (4.3)$$

Donde la Ecuación 4.3 es la forma general del método de Newton para conseguir de forma iterativa el mínimo de una función dada. Dicho método a la hora de llevarlo a un espacio de dimensión mayor, tiene sus inconsistencias matemáticas, por ello se debe realizar las siguientes modificaciones:

Suponga que se desea minimizar la función $f(\vec{X})$ con n variables y que esta se aproxima utilizando el desarrollo de Taylor. Así que,

$$f(\vec{X}) \approx q(\vec{X}) = f(\vec{X}_k) + (\vec{X} - \vec{X}_k)' \nabla f(\vec{X}_k) + \frac{1}{2}(\vec{X} - \vec{X}_k)' \mathbf{H}_f(\vec{X}_k) (\vec{X} - \vec{X}_k) \quad (4.4)$$

Si la aproximación de $f(\vec{X})$ por $q(\vec{X})$ es buena, un mínimo relativo $f(\vec{X})$ se podría aproximar por un mínimo relativo de por $q(\vec{X})$. Supongamos que \vec{X}_{k+1} es un mínimo relativo de $q(\vec{X})$, entonces \vec{X}_{k+1} es un punto estacionario para $q(\vec{X})$, así $\nabla q(\vec{X}_{k+1}) = 0$. Desarrollando

el gradiente de $q(\vec{X})$, sustituyendo \vec{X}_{k+1} por X e igualando a 0 tenemos,

$$\nabla f(\vec{X}_k) + \mathbf{H}_f(\vec{X}_k)(\vec{X}_{k+1} - \vec{X}_k) = 0 \quad (4.5)$$

Si la matriz hessiana $\mathbf{H}_f(\vec{X}_k)$ es invertible, tenemos que

$$\vec{X}_{k+1} = \vec{X}_k - \mathbf{H}_f^{-1}(\vec{X}_k) \nabla f(\vec{X}_k) \quad (4.6)$$

La Ecuación 4.6 se utiliza como una ecuación de recurrencia dado un punto inicial generar una sucesión de puntos que deben converger al mínimo local de $f(x)$. Como calcular la inversa de una matriz tiene una mayor complejidad que resolver un sistema de ecuaciones, la expresión $\mathbf{H}_f^{-1}(\vec{X}_k) \nabla f(\vec{X}_k)$ se obtiene resolviendo el sistema.

$$\left[\mathbf{H}_f(\vec{X}_k) \mid \nabla f(\vec{X}_k) \right] \quad (4.7)$$

Finalmente, el método Newton en su forma general para n variables está descrito por el Algoritmo 2.

Algoritmo 2 Método de Newton para n variables

Input : $f(\vec{X})$, \vec{X}_0 , $iter$, $Tolerancia$

Output: Mínimo encontrado $[x^*, y^*]$

$\vec{X}_k \leftarrow [x, y]$

▷ Posición inicial

while $Error > Tolerancia$ **do**

$$\vec{X}_{k+1} = \vec{X}_k - \mathbf{H}_f^{-1}(\vec{X}_k) \nabla f(\vec{X}_k)$$

$$Error = \|\vec{X}_{k+1} - \vec{X}_k\|_2^2$$

$$iter \leftarrow iter + 1$$

$$\vec{X}_k \leftarrow \vec{X}_{k+1}$$

end

4.3. Método del Descenso del Gradiente

El descenso de gradiente es un algoritmo iterativo que permite estimar numéricamente donde una función f presenta sus valores mínimos. Es un método muy similar al método de Newton, pero con una ventaja en cuanto su estabilidad, dadas por su formulación matemática, el cual se define como,

$$\vec{X}_{k+1} = \vec{X}_k - \lambda \nabla f(\vec{X}_k) \quad (4.8)$$

el valor de α permite hacer que el tamaño de paso sea mayor o menor.

El uso del método de descenso de gradiente negativo, como dirección para la minimización de una función f , fue realizado por primera vez por Cauchy [CAUCHY, 1847]. Este método inicia en un punto de prueba \vec{X}_k y se mueve iterativamente a lo largo de las direcciones de descenso más pronunciado hasta encontrar el punto óptimo.

El método de descenso del gradiente se puede resumir en los siguientes pasos:

- Comenzar con un punto inicial arbitrario \vec{X}_k y establecer el número de iteración como $k = 1$.
- Encontrar la dirección de búsqueda \vec{S}_k como

$$\vec{S}_k = -\nabla f_i = -\nabla f(\vec{X}_k)$$

- Determinar la longitud de paso óptima λ en la dirección \vec{S}_k y establecer

$$\vec{X}_{k+1} = \vec{X}_k + \lambda \vec{S}_k = \vec{X}_k - \lambda_i \nabla f_k$$

Finalmente, el método del descenso del Gradiente en su forma general está descrito por el Algoritmo 3.

Algoritmo 3 Método del Descenso de gradiente**Input** : $f(\vec{X})$, \vec{X}_0 , $iter$, $Tolerancia$ **Output:** Minimo encontrado $[x^*, y^*]$ $\vec{X}_k \leftarrow [x, y]$ ▷ Posición inicial $\lambda \leftarrow$ valor para generar estabilidad ▷ Constante de amortiguamiento**while** $Error > Tolerancia$ **do**

$$\vec{S}_k = \nabla f(\vec{X}_k)$$

$$\vec{X}_{k+1} = \vec{X}_k - \lambda \vec{S}_k$$

$$Error = \|\vec{X}_{k+1} - \vec{X}_k\|_2^2$$

$$iter \leftarrow iter + 1$$

$$\vec{X}_k \leftarrow \vec{X}_{k+1}$$

end

4.4. Método de Levenberg-Marquardt

El algoritmo de Levenberg-Marquardt genera una solución al problema de minimización de una función generalmente de naturaleza no lineal [Marquardt, 1963]. El citado algoritmo es una técnica iterativa que encuentra el mínimo de una función, con ciertos parámetros de inicialización. En este algoritmo no se necesita calcular una matriz Hessiana exacta, sino una aproximación la cual se define como,

$$\nabla^2 U(\vec{X}) \approx J(\vec{X})^\top J(\vec{X}), \quad (4.9)$$

donde, $J(\vec{X})$ es el Jacobiano de la función a minimizar. Considerando la siguiente aproximación,

$$H(\vec{X}) = \nabla^2 f(\vec{X}) \approx J(\vec{X})^\top J(\vec{X}) + \lambda \mathbf{I}, \quad (4.10)$$

donde λ es un parámetro que se escoge dinámicamente para que $H(\vec{X})$ sea definida positiva y \mathbf{I} es una matriz identidad, se obtiene finalmente el algoritmo de Levenberg-Marquardt (LM) (4.11),

$$\vec{X}_{k+1} = \vec{X}_k - (J(\vec{X}_k)^\top J(\vec{X}_k) + \lambda \mathbf{I})^{-1} J(\vec{X}_k)^\top. \quad (4.11)$$

Finalmente, el método de Levenberg-Marquardt en su forma general está descrito por el Algoritmo 4.

Algoritmo 4 Método de Levenberg Marquardt.**Input** : $f(\vec{X})$, $iter$ **Output:** Coordenadas de la meta $[x^*, y^*]$ $\vec{X}_k \leftarrow [x, y]$

▷ Posición inicial del robot

 $\lambda \leftarrow 0.001$

▷ Constante de amortiguamiento

while $Error > tolerancia$ **do**

$$J(\vec{X}_k) = \nabla f(\vec{X}_k)$$

$$H = J(\vec{X}_k)^T J(\vec{X}_k) + \lambda I$$

$$\alpha = H^{-1} J(\vec{X}_k)^T$$

$$\vec{X}_{k+1} = \vec{X}_k - \alpha$$

$$Error = \|\vec{X}_{k+1} - \vec{X}_k\|_2^2$$

$$iter \leftarrow iter + 1$$

$$\vec{X}_k \leftarrow \vec{X}_{k+1}$$

end

4.5. Método de Dog-Leg

El método de Dog-Leg es un algoritmo iterativo basado en regiones de confianza que busca optimizar una función e incluyendo mejoras a los métodos tradicionales como el gradiente descendente y Gauss-Newton. En cada iteración se genera una nueva región de confianza. Dependiendo de los resultados obtenidos en esta región se toma las decisiones de aceptar o no el nuevo punto obtenido o de aumentar, reducir o mantener el radio de la región de confianza [Chauhan et al., 2018]. En su forma general el paso de actualización del algoritmo está descrito por,

$$\vec{X}_{k+1} = \vec{X}_k + \mathbf{P}_k, \quad (4.12)$$

donde \mathbf{P}_K es el incremento o tamaño de paso de la k -ésima iteración. Para conseguir \mathbf{P}_K que aproxime a la solución óptima, se trabaja con un subproblema más simple que consiste en encontrar un \mathbf{P}_K^*

$$\mathbf{P}_K^* = \arg \min [m_k(\mathbf{P}), \mathbf{P} \in Q_k], \quad (4.13)$$

donde $m_k(\mathbf{P})$ es la función de costo y está definida como

$$\min_{\mathbf{P} \in \mathbb{R}^n} m_k(\mathbf{P}) = f_k + \nabla f_k^T \mathbf{P} + \frac{1}{2} \mathbf{P}^T b_k \mathbf{P} \quad (4.14)$$

f_k es el valor de la función de costo en la iteración k y $b_k \approx \nabla^2 f_k$, Q_k es la región de confianza y está definida como, $Q_k = \{\mathbf{P} \mid \|\mathbf{P}\| < \Delta_k\}$, donde Δ_k es el radio de la región de confianza en k -ésima iteración. Para ajustar el parámetro Δ_{k+1} en cada iteración, es necesario el cálculo de los vectores P_U y P_B tal que

$$\Delta_{k+1} = \|P_B + P_U\|_2, \quad (4.15)$$

$$P_U = -\frac{J(\vec{X}_k)^T J(\vec{X}_k)}{J(\vec{X}_k)^T H(\vec{X}_k) J(\vec{X}_k)}, \quad (4.16)$$

$$P_B = -H(\vec{X}_k)^{-1} J(\vec{X}_k), \quad (4.17)$$

donde $H(\vec{X})$ es la matriz hessiana y $J(\vec{X})$ es el gradiente del potencial. Formalmente, el método de Dog-Leg se describe tal como se muestra en el Algoritmo 5.

Algoritmo 5 Método de Optimización Dog-Leg.

Input : $f(\vec{X})$, $\vec{X}_g, \Delta > 0, \Delta_0 \in (0, \Delta)$, η , *iter*

Output: Coordenadas de la meta $[x^*, y^*]$

$\vec{X}_k \leftarrow [x, y]$

▷ Posición inicial del robot

Δ_0

▷ Radio inicial de la región de confianza

while *Error* > *tolerancia* **do**

$\mathbf{P}_K = \arg \min [m_k(\mathbf{P}), \mathbf{P} \in Q_k]$

$\Delta_{k+1} = \|P_B(X_k) + P_U(X_k)\|$

if $\mathbf{P}_K < 1/4$ **then**

$\Delta_{k+1} = \Delta_k/4$

else

if $\mathbf{P}_K > 3/4$ *and* $|\mathbf{P}_K| = \Delta_k$ **then**

$\Delta_{k+1} = \min(2\Delta_k, \Delta)$

else

$\Delta_k + 1 = \Delta_k$

end

end

if $\mathbf{P}_K > \eta$ **then**

$\vec{X}_{k+1} = \vec{X}_k + \mathbf{P}_k$

else

$\vec{X}_{k+1} = \vec{X}_k$

end

$\text{Error} = \|\vec{X}_{k+1} - \vec{X}_g\|_2^2$

iter \leftarrow *iter* + 1

end

4.6. Funciones de Prueba

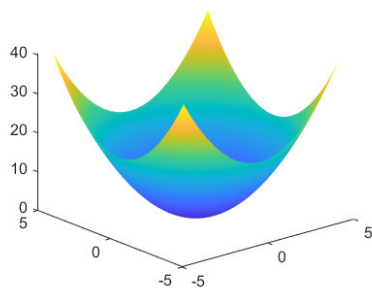
Con el propósito de validar los métodos de optimización propuestos, se sometieron a pruebas de desempeño que consisten en encontrar los mínimos en las funciones de prueba (*funciones benchmark*) [Wikipedia, 2020]. Este tipo de funciones presentan un gran reto para los algoritmos y nos permiten identificar y comparar el desempeño de los métodos propuestos.

Se sometió cada algoritmo a diferentes funciones de prueba como se pueden observar en la Tabla 4.1, las cuales están descritas brevemente, donde se mencionan sus rangos de búsqueda, su formulación matemática y sus mínimos globales teóricos. Como se puede observar en la Figura 4.1, se observa cada función de prueba, donde los algoritmos deberán evolucionar para conseguir cada mínimo global.

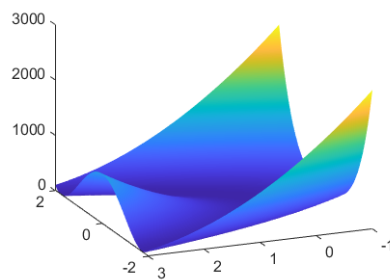
Se evaluarán pruebas de desempeño dependiendo del número de iteraciones, error cuadrático medio (RMSE), tiempo de ejecución del algoritmo y prueba de estabilidad al repetir el experimento 100 veces.

Tabla 4.1. Funciones de referencia para evaluación de algoritmos de optimización

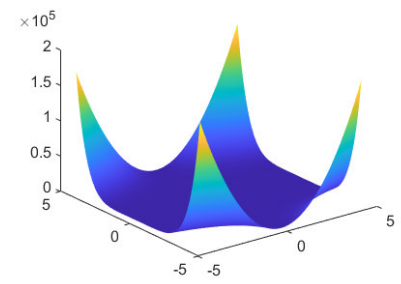
No.	Nombre	Formula	Mínimo global	Dominio de búsqueda
F_1	Sphere function	$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$	$f(0, 0) = 0$	$-5 \leq x_i \leq 5$
F_2	Rosenbrock function	$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$f(1, 1) = 0$	$-\infty \leq x_i \leq \infty$
F_3	Beale function	$f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$	$f(3, 0.5) = 0$	$-4.5 \leq x, y \leq 4.5$
F_4	Goldstein-Price function	$f(x, y) = [1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)] [30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)]$	$f(0, -1) = 3$	$-2 \leq x, y \leq 2$
F_5	Booth function	$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$	$f(1, 3) = 0$	$-10 \leq x, y \leq 10$
F_6	Bukin function N.6	$f(x, y) = 100\sqrt{ y - 0.01x^2 } + 0.01 x + 10 $	$f(-10, 1) = 0$	$-15 \leq x \leq -5, -3 \leq y \leq 3$
F_7	Matyas function	$f(x, y) = 0.26(x^2 + y^2) - 0.48xy$	$f(0, 0) = 0$	$-10 \leq x, y \leq 10$
F_8	Three-hump camel function	$f(x, y) = 2x^2 - 1.05x^4 + \frac{1}{6}x^6 + xy + y^2$	$f(0, 0) = 0$	$-5 \leq x, y \leq 5$
F_9	Himmelblau's function	$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$	$f(3.0, 2.0) = 0$ $f(-2.805118, 3.131312) = 0$ $f(-3.779310, -3.283186) = 0$ $f(3.584428, -1.848126) = 0$	$-5 \leq x, y \leq 5$



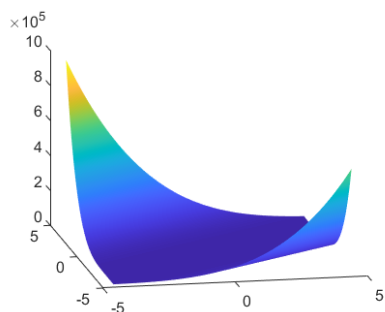
(a) Sphere function



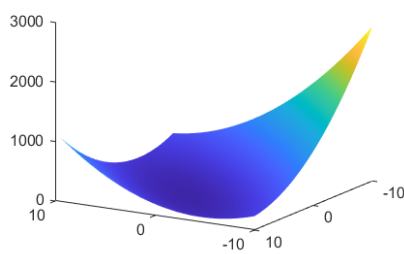
(b) Rosenbrock function



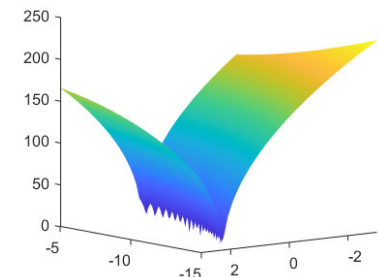
(c) Beale function



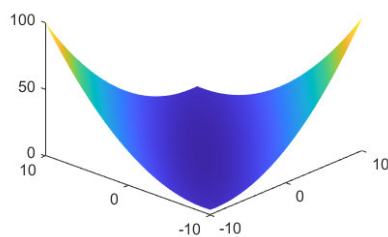
(d) Goldstein-Price function



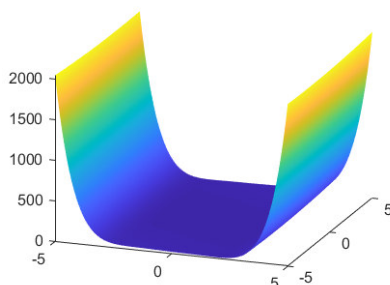
(e) Booth function



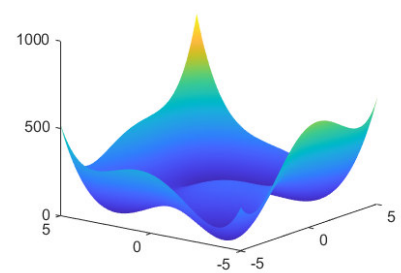
(f) Bukin function N.6



(g) Matyas function



(h) Three-hump camel function



(i) Himmelblau's function

Figura 4.1. Funciones de prueba para validación de algoritmos de optimización.

4.6.1. Validación del método de Newton

Se sometió el método de Newton a 9 funciones de prueba, dichas funciones presentan dificultades al momento de calcular su Hessiana, y, por lo tanto, puede crear inestabilidad. Por ello en algún caso se debió ubicar el punto inicial para la evolución del algoritmo más cerca de la solución.

La Tabla 4.2 muestra los resultados numéricos del método de Newton sometido a las funciones de referencia descritas en la Tabla 4.1. El objetivo es encontrar los mínimos de las funciones de prueba con la mejor exactitud posible.

En la Figura 4.2 se observa como el algoritmo evoluciona ante cada función de referencia, consiguiendo el mínimo en cada una de estas funciones, excepto la función *Bukin function* (F_6) debido a que la matriz hessiana es inestable. Además, se observa que el método presenta un número de iteraciones grandes ante funciones que son monótonas decrecientes, esto se debe a que el tamaño de paso está sujeto al cálculo de la de la hessiana, ya que de ella depende el nuevo paso a conseguir.

Tabla 4.2. Resultados obtenidos por el método de Newton para las funciones de referencia 4.1.

Función No.	Óptimo $F(x,y)$	Mínimo encontrado	Metodo de Newton		
			RMSE	Iteraciones	Tiempo [seg]
F_1	0	2.2568E-3	5.0931E-6	45	0.00812
F_2	0	3.6187E-3	1.309498E-5	90	0.02331
F_3	0	1.486E-4	2.20819E-8	115	0.04572
F_4	3	3.003245	1.0530E-5	175	0.07266
F_5	0	2.8564E-2	8.15902E-4	86	0.03389
F_6	0	-	-	-	-
F_7	0	2.3589E-2	5.564409E-4	79	0.05045
F_8	0	3.2547E-4	1.059307E-7	66	0.078179
F_9	0	6.453E-2	0.004164E-4	53	0.02357

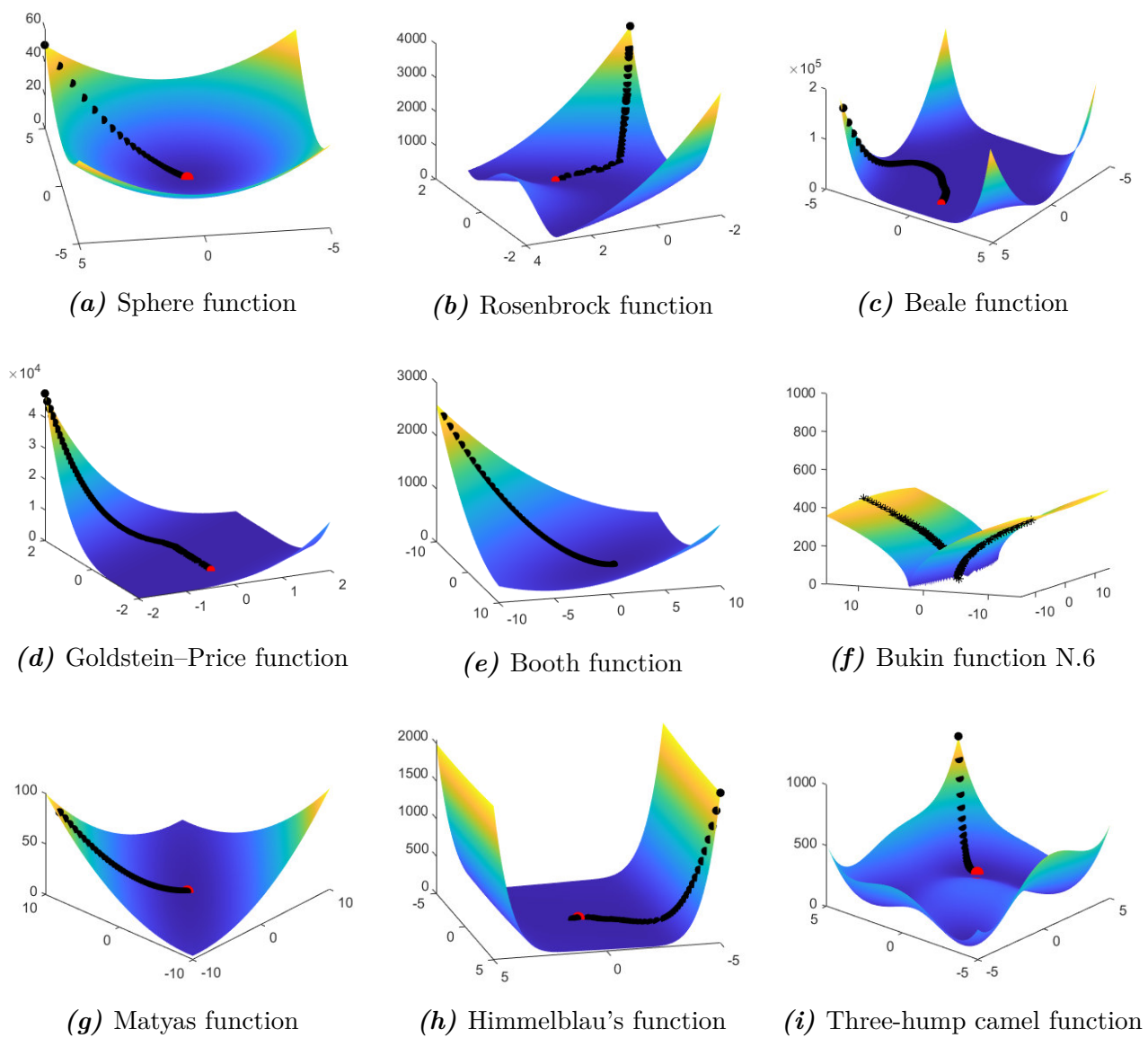


Figura 4.2. Funciones de prueba minimizadas por el método de Newton.

4.6.2. Validación del método del Descenso del Gradiente

La Tabla 4.3 muestra los resultados numéricos del método del descenso de gradiente sometido a las funciones de referencia descritas en la Tabla 4.1. El objetivo en este experimento es encontrar los mínimos de las funciones de prueba con la mejor exactitud posible.

En la Figura 4.3 se observa como el algoritmo evoluciona ante cada función de referencia, consiguiendo el mínimo en casi todas las funciones de referencia, fue necesario realizar ajustes en la constante de amortiguamiento del método de descenso de gradiente, específicamente en las funciones f_9 . En este caso se ajustó $\lambda = 0.000008$, este valor es muy pequeño y genera un número considerable de iteraciones para conseguir la solución.

Se observó que el método logra conseguir una solución con un número de iteraciones aceptable ajustando de manera correcta λ , pero de igual forma que el método de Newton no consigue dar solución a la función F_6 .

Tabla 4.3. Resultados obtenidos por el método del descenso del gradiente para las funciones de referencia 4.1.

Función No.	Óptimo $F(x,y)$	Minimo encontrado	Metodo del Descenso del Gradiente		
			RMSE	Iteraciones	Tiempo [seg]
F_1	0	1.02356E-4	1.04767507E-8	32	0.0065
F_2	0	2.65447E-4	7.04621098E-8	65	0.0085
F_3	0	1.2547E-4	1.5743E-08	85	0.0095
F_4	3	2.999861	1.9321E-8	75	0.007321
F_5	0	3.2544E-3	1.05911E-5	54	0.0052475
F_6	0	-	-	-	-
F_7	0	3.3589E-3	1.12822E-5	45	0.00221
F_8	0	2.5458E-5	6.4810976E-10	35	0.001278
F_9	0	4.68753E-3	2.19729E-5	82	0.012542

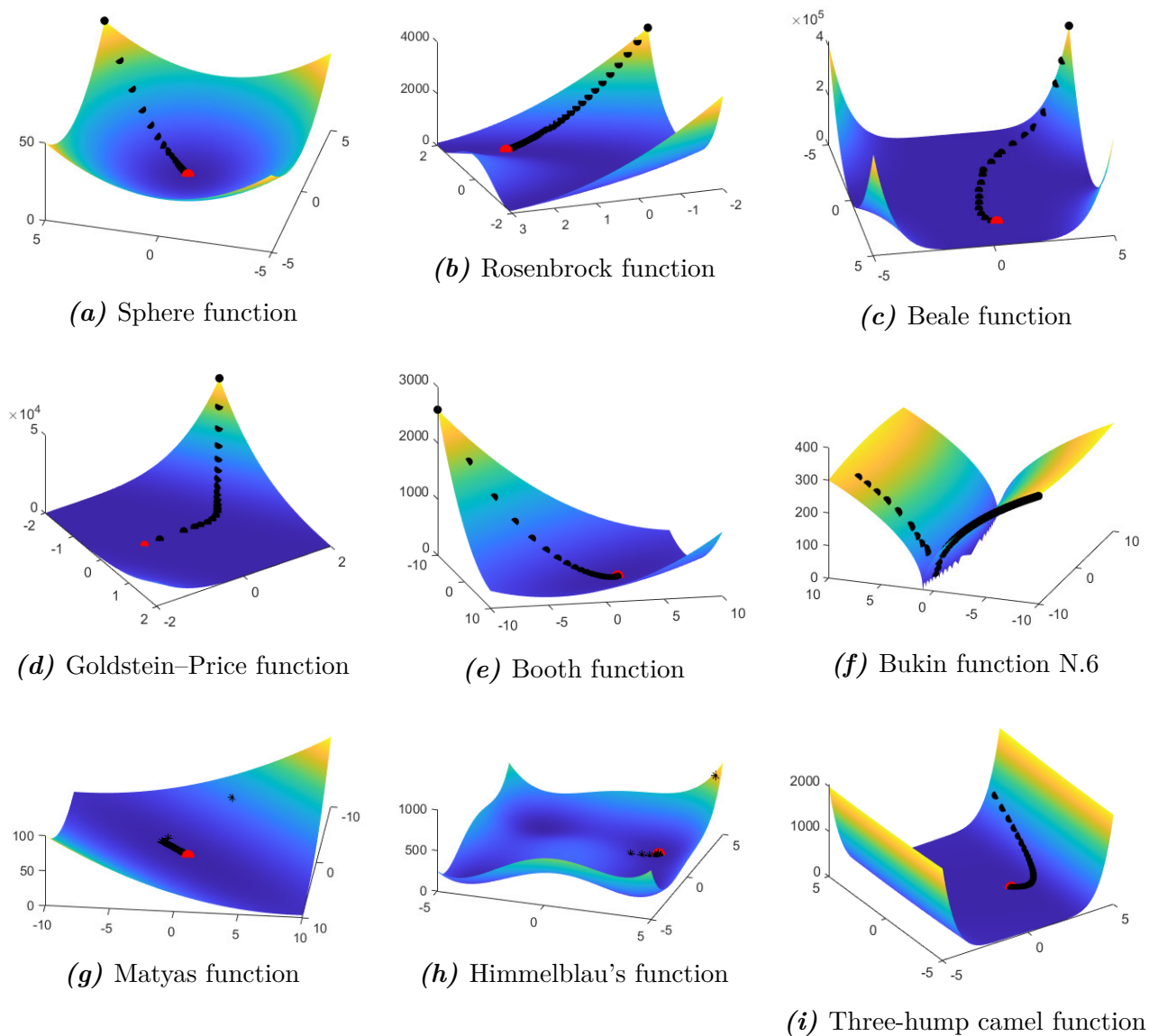


Figura 4.3. Funciones de prueba minimizadas por el método del descenso del gradiente.

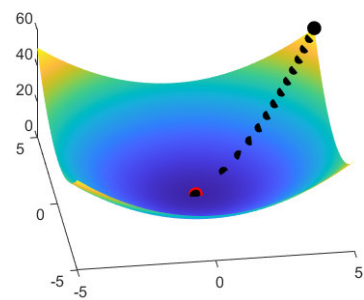
4.6.3. Validación del método de Levenberg-Marquardt

La Tabla 4.4 muestra los resultados numéricos del método de Levenberg-Marquardt, el cual fue sometido a las funciones de referencia descritas en la Tabla 4.1. En la Figura 4.4 se observa como el algoritmo evoluciona ante cada función de referencia, consiguiendo el mínimo en cada una de estas funciones. La aproximación de la hessiana actúa como un elemento que favorece al algoritmo ante funciones que presentan dificultad para el cálculo de sus derivadas, esto se puede observar en la función de referencia. F_6 Bukin function N.6, donde su derivada se torna compleja. El algoritmo logró conseguir el mínimo de esta función, la cual no se había logrado con los métodos anteriores descritos.

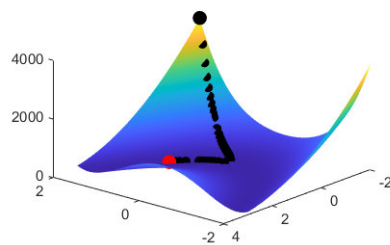
Para garantizar que la aproximación de la hessiana tenga una matriz inversa, se ajusta el parámetro de amortiguamiento λ , de tal forma que la aproximación de la hessiana sea definida positiva. Es importante resaltar que para cada función fue necesario ajustar este parámetro para conseguir el mejor desempeño del algoritmo.

Tabla 4.4. Resultados obtenidos por el método de Levenberg-Marquardt para las funciones de referencia 4.1.

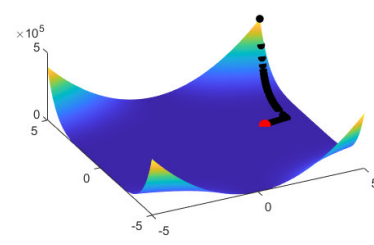
Función No.	Óptimo F(x,y)	Mínimo encontrado	Metodo de Levenberg-Marquardt		
			RMSE	Iteraciones	Tiempo [seg]
F_1	0	4.332E-6	1.8766E-11	16	0.000095
F_2	0	6.3325E-5	4.0101E-9	54	0.003325
F_3	0	7.2632E-5	5.2754E-9	37	0.0019
F_4	3	3.000015	2.2500E-10	42	0.002147
F_5	0	2.295E-6	5.2670E-12	30	0.00154
F_6	0	7.58419E-4	5.7520E-07	46	0.0023514
F_7	0	1.288367E-4	1.6599E-8	25	0.000148437
F_8	0	8.2658E-6	6.8323E-11	29	0.000172186
F_9	0	9.2532E-6	8.5622e-11	41	0.002095



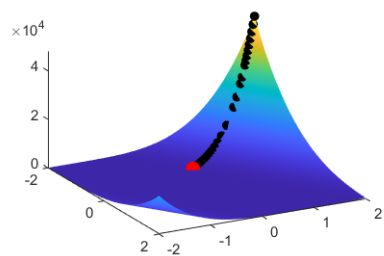
(a) Sphere function



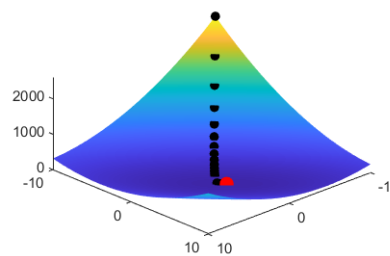
(b) Rosenbrock function



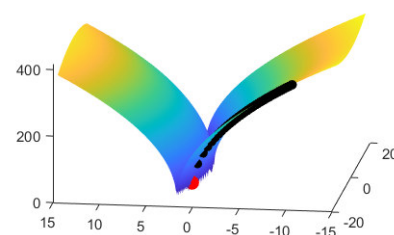
(c) Beale function



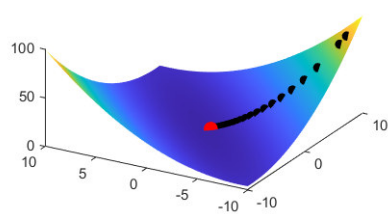
(d) Goldstein-Price function



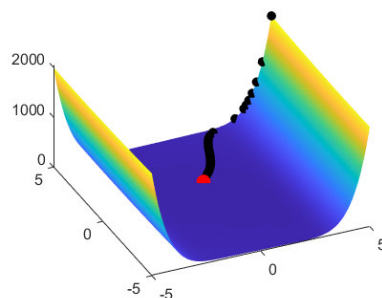
(e) Booth function



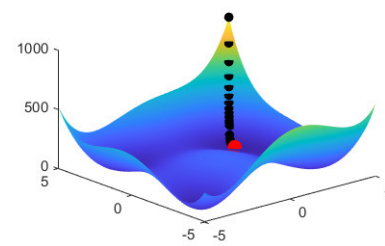
(f) Bukin function N.6



(g) Matyas function



(h) Himmelblau's function



(i) Three-hump camel function

Figura 4.4. Funciones de prueba minimizadas por el método de Levenberg-Marquardt.

4.6.4. Validación del método de Dog-Leg

La Tabla 4.5 muestra los resultados numéricos del método de Dog-Leg, el cual fue sometido a las funciones de referencia descritas en la Tabla 4.1. En la Figura 4.5. Se observa como el algoritmo evoluciona ante cada función generando regiones de confianza descritas por los círculos rojos, consiguiendo el mínimo en cada una de estas funciones.

Las regiones de confianza reducen el problema a un subproblema, con esto no es necesario abordar toda función de referencia y solo se reduce a encontrar el mínimo de la región de confianza.

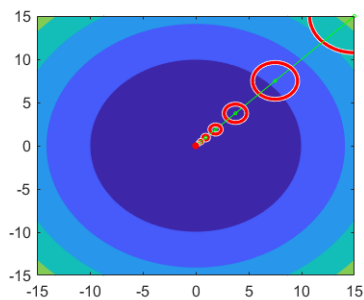
La estabilidad del método dependerá del tamaño de la región de confianza, esta aumentará o disminuirá dependiendo de que tan bueno sea el mínimo encontrado y si es mejor al anterior.

Se observa que el número de iteraciones disminuye notablemente con respecto a los métodos anteriores.

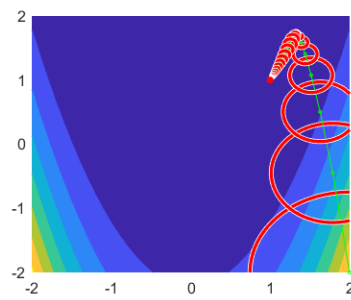
Tabla 4.5. Resultados obtenidos por el método de Dog-Leg para las funciones de referencia 4.1.

Función No.	Óptimo $F(x,y)$	Mínimo encontrado	Metodo de de Dog-Leg		
			RMSE	Iteraciones	Tiempo [seg]
F_1	0	1.029556E-6	1.06E-12	9	0.00009156
F_2	0	2.95817E-5	8.7508E-10	29	0.0005026
F_3	0	8.45332E-5	7.1459E-9	40	0.00006933
F_4	3	3.000453	2.05208E-7	10	0.0000984
F_5	0	3.2544E-3	1.05911E-5	26	0.00082475
F_6	0	3.2544E-3	1.05911E-5	12	0.0001258
F_7	0	3.3589E-3	1.12822E-5	19	0.00012521
F_8	0	2.5458E-5	6.4810976E-10	26	0.00078425
F_9	0	4.68753E-3	2.19729E-5	18	0.00011335

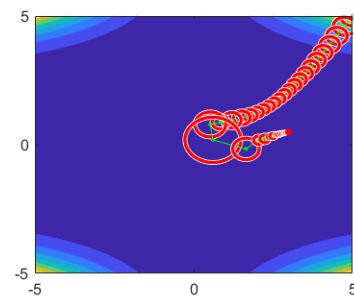
Los métodos de Levenberg-Marquardt y de Dog-Leg, fueron los que presentaron mejor desempeño ante el problema de minimización de las funciones de referencia 4.1, Debido a esto se seleccionaron estos dos métodos para la integración con el método APF, el cual veremos a continuación.



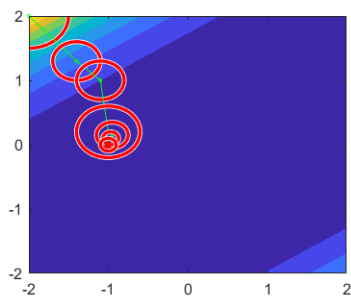
(a) Sphere function



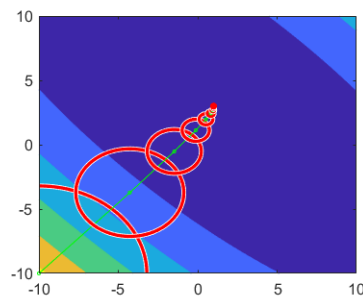
(b) Rosenbrock function



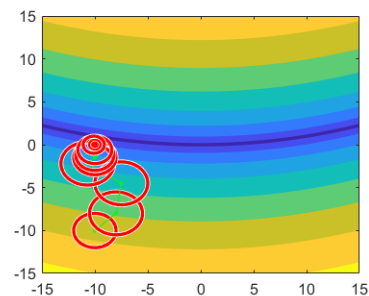
(c) Beale function



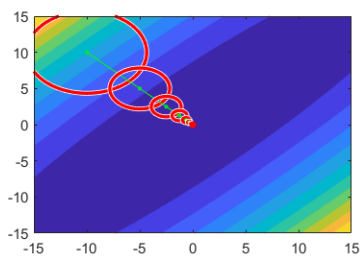
(d) Goldstein-Price function



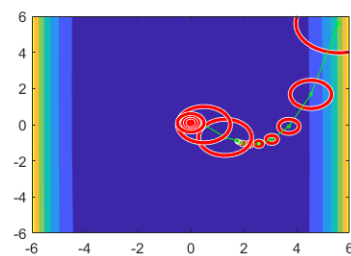
(e) Booth function



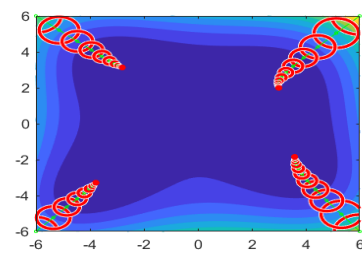
(f) Bukin function N.6



(g) Matyas function



(h) Himmelblau's function



(i) Three-hump camel function

Figura 4.5. Funciones de prueba minimizadas por el método de Dog-Leg.

4.7. Campos Potenciales con Levenberg-Marquardt y Dog-Leg

Con el fin de observar el comportamiento de las trayectorias empleando el método del Levenberg-Marquardt y el método de Dog-Leg, se utilizan los siguientes parámetros de optimización resumidos en la Tabla 4.6. Donde V_0 y τ son coeficientes de velocidad inicial y tiempo en un paso del robot, quien inicia su trayectoria en $\vec{X}_0 \in \mathbb{R}^2$. Estos coeficientes permiten dar un paso mayor al generado por el gradiente manteniendo su dirección. Además se buscó que se genera un número de iteraciones que permitiera a ruta calculada no tener cambio brusco, con la finalidad de que al momento de implementarlo en un robot móvil sea más sencillo emplear un controlador para seguir dicha trayectoria. La Tabla 4.7 muestra los resultados obtenidos a

Tabla 4.6. Parámetros usados para el ajuste de los algoritmos de optimización

Levenberg Marquard			Dog-Leg			
λ	V_0	τ	η	Δ_0	V_0	τ
0.001	2	1	0.2	1	2	1

partir de diferentes posiciones iniciales del robot dado los obstáculos y la meta como se muestra en la Fig. 4.6. De manera práctica y para asignar unidades métricas, se define un entorno de

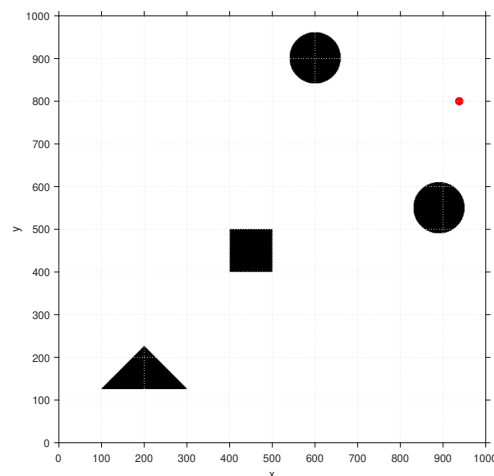


Figura 4.6. Mapa generado con cuatro obstáculos fijos. Con un punto rojo se muestra la meta la cual el robot debe alcanzar.

$1000 \times 1000 \text{ cm}^2$, con la meta en $\vec{X}_g = [940, 800]$. Es importante señalar que, por construcción del método basado en potenciales eléctricos, físicamente, la proximidad de dos cargas de signo

opuesto (el robot y la meta) podría generar una singularidad por lo cual se tiende a una región equipotencial muy cercana. De esta manera, es posible definir una función de error que nos determine qué tan alejado queda el robot de la meta. La función de error se define como la distancia euclidiana entre ambas posiciones. De acuerdo con los resultados de la Tabla 4.7,

Tabla 4.7. Resultados numéricos para llegar al vector objetivo dado por $\vec{X}_g = [940, 800]$.

Levenberg Marquardt			Dog-Leg		
\vec{X}_0	Iter	Error[cm]	\vec{X}_0	Iter	Error[cm]
[36, 220]	166	3.6	[36, 220]	143	2.6
[428,27]	150	3.2	[428, 27]	116	2.2
[150,850]	100	3.4	[150,850]	63	1.2

el método de Dog-Leg requiere un menor número de iteraciones y alcanza un menor error en comparación con el método de Levenberg-Marquardt. En la Fig. 4.7 se observa la trayectoria obtenida con el algoritmo Levenberg-Marquardt.

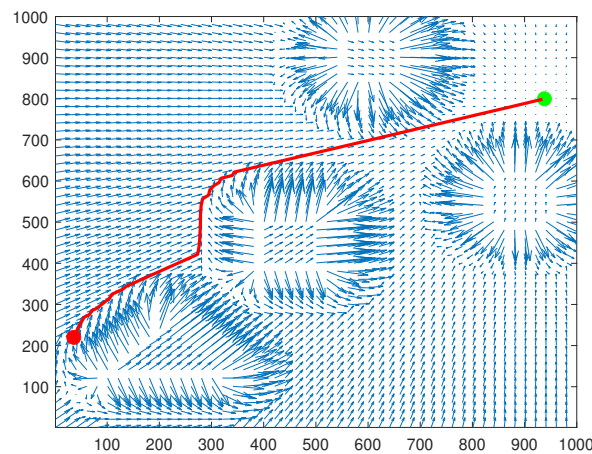


Figura 4.7. Ruta generada mediante Potenciales Eléctrico y Levenberg-Marquardt.

El robot logra llegar de $\vec{X}_0 = [36, 220]$ a $\vec{X}_g = [938, 800]$ en 166 iteraciones, con un error de 3.6 cm. Por su parte, en la Fig. 4.8 se observa la trayectoria obtenida con el algoritmo Dog-Leg para la misma posición inicial del robot, y la misma meta. En cada iteración se dibuja una región de confianza y cumpliendo el objetivo en 143 iteraciones con un error de 2.3 cm.

Sin embargo, ambos métodos presentan problemas ante la presencia de mínimos locales, ocasionados cuando la meta se encuentra en la misma dirección que el campo de repulsión del obstáculo, en la Figura 4.9a se presenta el escenario al cual se someterán los métodos, el punto azul representa el punto de partida y el punto rojo es la meta, en la Figura 4.9b y 4.9c se

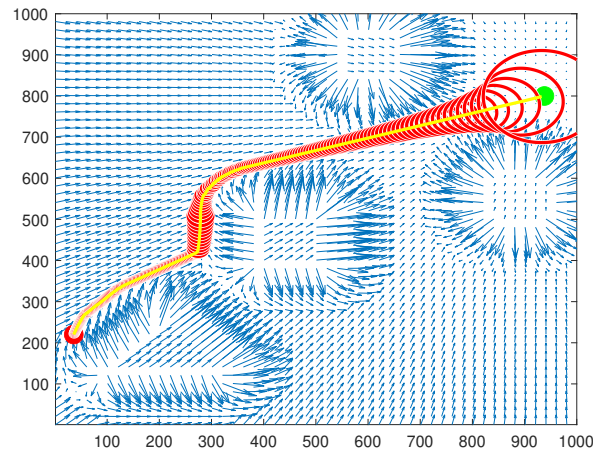
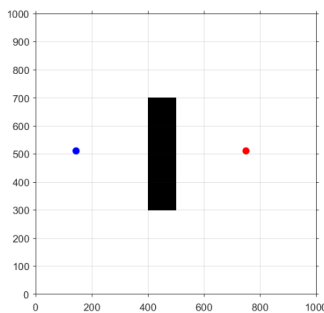
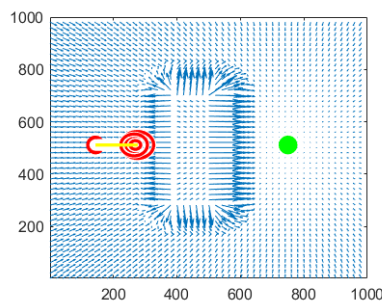


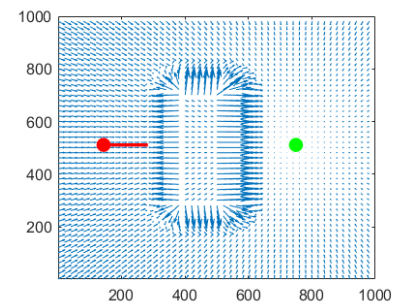
Figura 4.8. Ruta generada mediante Potenciales Eléctrico y Dog-Leg. Las regiones de confianza son resaltadas en color rojo.



(a) Mapa generado



(b) metodo de dog-leg



(c) metodo de Levenberg-Marquardt

Figura 4.9. Problema de mínimos locales con el método de Dog-Leg y Levenberg-Marquardt.

observa al método de dog-leg y Levenberg-Marquardt ante este problema, en ambos caso no se consigue el objetivo, una solución práctica seria dar una fuerza virtual al robot para tratar de sacarlo del mínimo local, pero esto no nos garantiza que la ruta sea la óptima o si pueda llegar a chocar con el obstáculo.

Para garantizar conseguir una solución que nos permita tener la ruta más corta y segura, se utilizarán métodos metaheurísticos, los cuales se describirán en el siguiente capítulo.

CAPÍTULO 5

Métodos Metaheurísticos

5.1	Introducción	57
5.2	Optimización por enjambres de partículas (PSO)	58
5.3	Evolución Diferencial (ED)	61
5.4	Cuckoo search (CS)	63
5.5	Algoritmo de búsqueda gravitacional (GSA)	65
5.6	Funciones de Prueba	70
5.6.1	Validación del Algoritmo Metaheurístico PSO	72
5.6.2	Validación del Algoritmo Metaheurístico ED	73
5.6.3	Validación del Algoritmo Metaheurístico CS	76
5.6.4	Validación del Algoritmo Metaheurístico GAS	78
5.6.5	Comparación de los Algoritmos PSO, ED, CS, GAS	79
5.7	Integración APF con el algoritmo CS	82
5.7.1	Validación de método APF+CS	87

5.1. Introducción

Por lo general los métodos de optimización clásicos son deterministas, estos métodos de optimización utilizan información del gradiente, y por ello llevan el nombre de métodos basados en el gradiente. Un ejemplo de los métodos deterministas o basados en el gradiente es el conocido método Newton-Raphson, dicho método utiliza el gradiente como parámetro para el cálculo del tamaño de paso entre iteraciones, utilizando los valores de las funciones y sus derivadas, funciona bien para problemas con complejidad media. Sin embargo, si hay alguna discontinuidad en la función objetivo, no lograra conseguir una solución debido a la inestabilidad producida por su Hessiana. Ante esta situación, se prefiere un método no basado en el gradiente, ya que métodos no utilizan ninguna derivada, sino solo los valores de la función.

Para los métodos de optimización estocásticos, en general tenemos dos tipos: heurísticos y metaheurísticos, aunque su diferencia es pequeña. En términos generales, heurístico significa “encontrar” o “descubrir”. Estos métodos por lo general dan soluciones problemas de optimización complejos, pero no hay garantía de que se alcancen soluciones óptimas.

Los métodos de optimización metaheurísticos son una evolución de los métodos heurísticos, meta significa “más allá” o “nivel superior”, y suelen tener un mejor rendimiento que los heurísticos simples. Debido a que todos los métodos metaheurísticos utilizan cierta compensación de aleatoriedad y búsqueda local. Cabe destacar que no existen definiciones exactas de heurística y metaheurística en la literatura; algunos utilizan “heurística” y “metaheurística” indistintamente. Sin embargo, las tendencias recientes denominan metaheurísticos a todos los algoritmos estocásticos con aleatorización y búsqueda local. Al incluir la aleatoriedad en el método facilita la forma de pasar de la búsqueda local a la búsqueda a escala global. Por lo tanto, casi todos los métodos metaheurísticos pretenden ser adecuados para la optimización global.

El objetivo de estos métodos es encontrar una buena solución factible en un tiempo aceptable. No hay garantía de que se puedan encontrar las mejores soluciones, e incluso no sabemos si un algoritmo funcionará. La idea es tener un algoritmo eficiente y práctico que funcione la mayor parte del tiempo y sea capaz de producir soluciones de buena calidad.

En el presente capítulo se abordarán métodos de optimización metaheurísticos tales como: Optimización por enjambres de partículas (PSO), Evolución Diferencial (ED), Cuckoo search (CS) y Algoritmo de búsqueda gravitacional (GSA), sus aplicaciones ante funciones de referencia complejas y con discontinuidades, así como la fusión de métodos de optimización

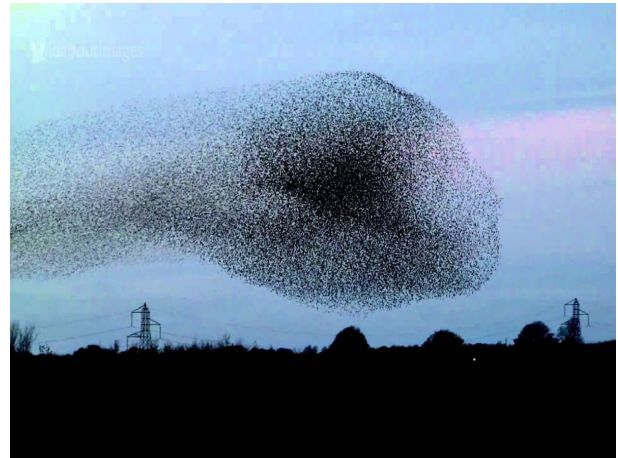
metaheurísticos con el método APF para la planificación de trayectorias en robots móviles.

5.2. Optimización por enjambres de partículas (PSO)

La optimización por enjambre de partículas (PSO) fue desarrollada por Kennedy y Eberhart [Kennedy and Eberhart, 1995], fundamentando su método con en el comportamiento de los enjambres, como los peces y las aves en la naturaleza (ver Figura 5.1). Por este motivo, PSO es un tema que se le ha prestado mucha atención por los investigadores con el paso de los años. El algoritmo PSO es aplicado en diversas áreas de la optimización, la inteligencia computacional y las aplicaciones de diseño y programación. Existen de variantes de PSO [Mahapatra et al., 2019; Lodhi et al., 2018], y también son cada vez más populares los algoritmos híbridos que combinan PSO con otros algoritmos existentes.



(a) Enjambre de peces



(b) Enjambre de aves

Figura 5.1. Comportamiento de enjambres de peces y aves ©.

Muchos algoritmos, como los algoritmos de colonias de hormigas, utilizan el comportamiento llamado inteligencia de enjambre. La optimización por enjambre de partículas puede tener algunas similitudes con los algoritmos genéticos y los algoritmos de hormigas, pero es mucho más sencillo debido a que no utiliza operadores de mutación/cruzamiento ni feromonas. En su lugar, utiliza la aleatoriedad y la comunicación global entre las partículas del enjambre. En este sentido, también es más fácil de implementar, ya que no hay codificación ni decodificación de los parámetros en cadenas binarias como en los algoritmos genéticos.

Este algoritmo busca en el espacio de una función objetivo ajustando las trayectorias de los agentes individuales, denominados partículas, formando vectores posicionales de forma casi estocástica. El movimiento de una partícula de enjambre consta de dos componentes principales: un componente estocástico y un componente determinista. Cada partícula es atraída hacia la posición del mejor global actual \mathbf{g}^* y su propia mejor ubicación \mathbf{x}_i^* en el tiempo de compilación, mientras que al mismo tiempo tiene una tendencia a moverse al azar.

Cuando una partícula encuentra una ubicación que es mejor que cualquiera de las ubicaciones encontradas anteriormente, se actualiza como el nuevo mejor actual para la partícula i . Hay un mejor actual para todas las partículas n en cualquier momento t durante las iteraciones. El objetivo es encontrar la mejor solución global entre todas las mejores soluciones actuales hasta que el objetivo deje de mejorar o después de un cierto número de iteraciones. El movimiento de las partículas se representa esquemáticamente en la Figura 5.2 donde \mathbf{x}_i^* es la mejor solución actual para la partícula i , y $\mathbf{g}^* \approx \min \{f(\mathbf{x}_i)\}$ para $i = 1, 2, \dots, n$, es el mejor global actual.

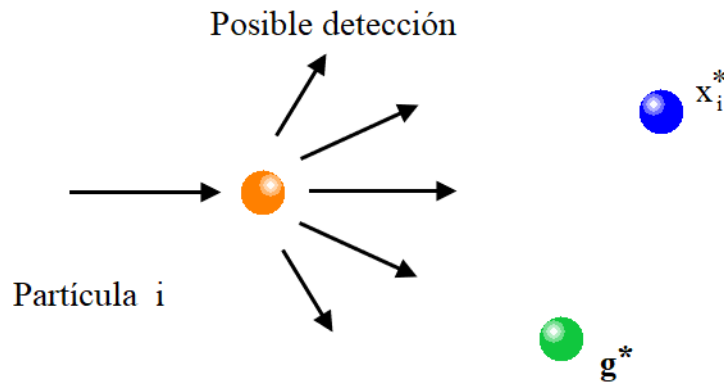


Figura 5.2. Representación esquemática del movimiento de una partícula en PSO, moviéndose hacia el mejor global \mathbf{g}^* y el mejor actual \mathbf{x}_i^* para cada partícula i .

Los pasos esenciales de la optimización del enjambre de partículas se pueden resumir como el pseudocódigo 6. Donde \mathbf{x}_i y \mathbf{v}_i son los vectores de posición y la velocidad de la partícula i , respectivamente. El nuevo vector de velocidad se determina por la siguiente forma,

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \alpha \epsilon_1 \odot [\mathbf{g}^* - \mathbf{x}_i^t] + \beta \epsilon_2 \odot [\mathbf{x}_i^* - \mathbf{x}_i^t], \quad (5.1)$$

donde ϵ_1 y ϵ_2 son dos vectores aleatorios, y cada entrada toma los valores entre 0 y 1. El

producto de Hadamard de dos matrices $\mathbf{u} \odot \mathbf{v}$ se define como el producto por entradas, es decir $[\mathbf{u} \odot \mathbf{v}]_{ij} = u_{ij}v_{ij}$. Los parámetros α y β son los parámetros de aprendizaje o las constantes de aceleración, que pueden tomarse típicamente como, por ejemplo, $\alpha \approx \beta \approx 0.5$.

Las ubicaciones iniciales de todas las partículas deben distribuirse de forma relativamente uniforme para que puedan muestrear la mayoría de las regiones, lo que es especialmente importante para los problemas multimodales. La velocidad inicial de una partícula puede tomarse como cero, es decir, $\mathbf{v}_i^{t=0} = 0$. La nueva posición puede entonces actualizarse mediante

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \quad (5.2)$$

Aunque \mathbf{v}_i puede ser cualquier valor, suele estar acotado en algún rango $[0, \mathbf{v}_{max}]$.

Bajo los lineamientos mencionados anteriormente, el algoritmo PSO es detallado en el pseudocódigo 6

Algoritmo 6 Algoritmo PSO.

Input : $f(\mathbf{x})$, $\mathbf{x} = (x_1, \dots, x_n)^\top$, i

Output: Coordenadas \mathbf{x}_i^* y \mathbf{g}^*

Inicialización de la posición x_i y velocidad v_i de n partículas.

$\mathbf{g}^* \approx \min \{f(\mathbf{x}_i)\}$

$t \leftarrow 0$

while *Criterio de parada* **do**

$t = t + 1$

for *Bucle de n partículas y d dimensiones* **do**

$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \alpha \epsilon_1 \odot [\mathbf{g}^* - \mathbf{x}_i^t] + \beta \epsilon_2 \odot [\mathbf{x}_i^* - \mathbf{x}_i^t]$

$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}$

 Evaluar la función objetivo en las nuevas ubicaciones de \mathbf{x}_i^{t+1}

 Encontrar el mejor Para cada partícula \mathbf{x}_i^*

end

 Encontrar el mejor Global \mathbf{g}^*

end

5.3. Evolución Diferencial (ED)

La evolución diferencial es un algoritmo de optimización pequeño y simple basado de un proceso de evolución bio-inspirado y naturalmente complejo [Price and Price, 2015]. El algoritmo fue propuesto por primera vez en 1997 por Storn y Price y descrito con mayor detalle en [Storn and Price, 1997].

La primera etapa del algoritmo es la mutación. La mutación es el proceso que evita la pérdida de diversidad, algorítmicamente está relacionado con el producto de bits que han convergido a un cierto valor de toda la población, y por lo tanto no pueden ser recuperados por el operador de recombinación. Para cada vector de parámetro, se seleccionan otros tres vectores \mathbf{x}_{r1n}^t , \mathbf{x}_{r2n}^t y \mathbf{x}_{r3n}^t al azar y se suma la diferencia ponderada de dos de los vectores al tercero.

Donde \mathbf{v}_n^{t+1} es llamado el *Donor vector* (vector donante) y el factor de mutación F generalmente se toma entre 0 y 1.

$$\mathbf{v}_n^{t+1} = \mathbf{x}_{r1n}^t + F(\mathbf{x}_{r2n}^t - \mathbf{x}_{r3n}^t) \quad n = 1, 2, 3, \dots, N \quad (5.3)$$

La segunda etapa del algoritmo es la recombinación, la cuales es el principal operador genético, representa la reproducción sexual, este opera sobre dos cromosomas a la vez para generar dos descendientes, los cuales se combinan y generan nuevas características provenientes de los cromosomas padres.

Un vector de prueba $u_{n,i}^{t+1}$ se desarrolla a partir del vector objetivo o *elemento poblacional*, $\mathbf{x}_{n,i}^t$, y el *Donor vector*, $\mathbf{v}_{n,i}^{t+1}$

$$\mathbf{u}_{n,i}^{t+1} = \begin{cases} \mathbf{v}_{n,i}^{t+1} & \text{si } rand_i \leq C_p \text{ ó } i = I_{\text{rand}}, \quad i = 1, \dots, D, \\ \mathbf{x}_{n,i}^t & \text{si } rand_i > C_p \text{ y } i \neq I_{\text{rand}} \quad n = 1, \dots, N \end{cases} \quad (5.4)$$

I_{rand} es un número entero aleatorio entre $[1, D]$, y C_p es la probabilidad de recombinación. Después de conocer la aptitud de cada cromosoma, se procede a determinar los cromosomas que serán cruzados en la próxima generación. Los individuos con mejor aptitud serán seleccionados.

El vector objetivo $\mathbf{x}_{n,i}^t$ se compara con el vector de prueba $u_{n,i}^{t+1}$ y se selecciona el que

tiene el valor de la función de costo más bajo (mejor *fitting*) para la siguiente generación.

$$x_n^{t+1} = \begin{cases} u_{n,i}^{t+1} & \text{si } f(u_n^{t+1}) < f(x_n^t) \\ x_n^t & \text{otro caso.} \end{cases} \quad (5.5)$$

Para todo $n = 1, 2, 3, \dots, N$. Todas las soluciones de la población tienen las mismas posibilidades de ser seleccionadas como progenitoras independientemente de su valor de aptitud. Si el padre es aún mejor, se retiene en la población.

El procedimiento iterativo puede finalizar cuando se cumple un criterio de paro, que puede ser una solución adecuada o calcula con el error cuadrático medio. En algunos casos, no es fácil determinar una solución aceptable. En algunas investigaciones, la convergencia del algoritmo se determina cuando los resultados permanecen constantes durante un número fijo de generaciones (estancamiento) o se puede considerar un criterio de parada, cuando se alcanza el número máximo de generaciones, lo que ocurra primero [Varadarajan and Swarup, 2008]. El algoritmo de ED se resume en el pseudocódigo 7.

Algoritmo 7 Algoritmo Evolución Diferencial.

```

X ← [x1, x2, ..., xi] ∈ ℝN×D                                ▷ Población inicial
for k = 1 to G do                                            ▷ Bucle de generaciones
    Etapa de Mutacion N, D, F, xi                                ▷ Mutación
    ri ← Irand(Z) | Z ∈ {1, 2, ..., N}  ∀i ∈ [1, 3]
    uG+1 = xr1G + F(xr2G − xr3G),  i ≠ ri ≠ rj  ∀i, j ∈ [1, N]    ▷ Vector donante
    Etapa de Recombinacion
    N, D, ui, xi                                            ▷ Recombinación
    Irand ∼ Irand(Z) | Z ∈ {1, 2, ..., D}                    ▷ Urand(a, b) ∼ U[a, b]
    vij ←  $\begin{cases} u_{ij} & U_{rand_{ij}} \leq CR \vee j == I_{rand} \\ x_{ij} & U_{rand_{ij}} \neq CR \wedge j \neq I_{rand} \end{cases}$   i ∈ [1, N], j ∈ [1, D]    ▷ Vectores de prueba
    Etapa de Seleccion
    N, D, vi, xi                                            ▷ Mejores candidatos
    If f(vi) < f(xi) then xi = vi end if
    if (Stop Criteria) then BreakFor() end if                ▷ Convergencia anticipada
end

```

5.4. Cuckoo search (CS)

La búsqueda del *cuckoo* (CS) es un algoritmo metaheurísticos inspirados en la naturaleza, desarrollado por Xin-She Yang [Yang and Suash Deb, 2009]. CS se basa en el comportamiento parasitario de las crías de algunas especies de *cuckoo* (ver Figura 5.3). Este algoritmo se ve potenciado por los vuelos de Lévy, que sustituyen las caminatas aleatorias isotrópicas. Estudios recientes demuestran que el CS es más eficiente que el PSO y los algoritmos genéticos [Bradley, 2010].



(a) *cuckoo* recién nacido

(b) Ave alimentando a un *cuckoo* invasor

Figura 5.3. Comportamiento de las Aves *cuckoo* ©.

Los *cuckoo* son aves fascinantes, no solo por los bellos sonidos que emiten, sino también por su agresiva estrategia de reproducción. Algunas especies, como los *cuckoo ani Guira*, ponen sus huevos en nidos comunales, en algunas ocasiones desplazan los huevos de otra especie en el nido para aumentar la probabilidad de eclosión de los suyos. Un buen número de especies practican el parasitismo de cría obligatorio poniendo sus huevos en los nidos de otras aves huésped (a menudo de otras especies).

Hay tres tipos básicos de parasitismo de cría: el parasitismo de cría intraespecífico, la cría cooperativa y la toma de nidos. Algunas aves hospedadoras pueden detectar a los intrusos y entablar un conflicto directo con los *cuckoo*. Si un ave huésped descubre que los huevos no son suyos, se deshará de estos huevos ajenos o simplemente abandonará su nido y construirá uno nuevo en otro lugar. Algunas especies de *cuckoo*s, como la *Tapera*, han evolucionado de tal manera que las hembras de los *cuckoo* parásitos pigmentan con el color y el patrón sus huevos

de tal manera que otras especies anfitrionas no los detecten. Esto reduce la probabilidad de que sus huevos sean abandonados y aumenta así su reproductividad.

Los *cuckoo* s parásitos suelen elegir un nido en el que el ave huésped acaba de poner sus propios huevos. En general, los huevos del *cuckoo* eclosionan ligeramente antes que los de su anfitrión. Una vez que el primer polluelo de *cuckoo* sale del cascarón, la primera acción instintiva que llevará a cabo será la de desalojar los huevos del anfitrión impulsando los huevos fuera del nido, lo que aumenta la cuota de comida proporcionada por su pájaro anfitrión. Los estudios también demuestran que el polluelo de *cuckoo* también puede imitar la llamada de los polluelos del anfitrión para acceder a más oportunidades de alimentación. Para realizar el algoritmo de búsqueda de *cuckoo*, se establecen tres reglas básicas:

- Cada *cuckoo* pone un huevo a la vez y lo deposita en un nido elegido al azar.
- Los mejores nidos con huevos de alta calidad pasarán a las siguientes generaciones.
- El número de nidos anfitriones disponibles es fijo, y el huevo puesto por un *cuckoo* es descubierto por el pájaro anfitrión con una probabilidad $p_a \in [0, 1]$. En este caso, el pájaro anfitrión puede deshacerse del huevo o simplemente abandonar el nido y construir uno completamente nuevo.

Visto de forma computacional, podemos utilizar las siguientes analogías, cada huevo en un nido representa una solución, y cada *cuckoo* puede poner solo un huevo, el objetivo es utilizar las nuevas y mejores soluciones para reemplazar una solución no tan buena en los nidos. Al generar nuevas soluciones $\mathbf{x}^{(t+1)}$ para un *cuckoo*_{*i*} cualquiera, se realiza un vuelo de Lévy,

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \alpha \oplus \text{Levy}(\lambda) \tag{5.6}$$

donde $\alpha > 0$ es la constante de escalamiento, la cual define el tamaño de paso. En la mayoría de los casos, se puede utilizar $\alpha = 1$. La ecuación 5.6 representa una ecuación estocástica de una caminata aleatoria. En general, una caminata aleatoria es una cadena de Markov cuyo siguiente estado solo depende de la ubicación actual (el primer término de la ecuación anterior) y de la probabilidad de transición (el segundo término).

Para implantar esta caminata aleatoria es necesario conocer la distribución de Lévy, la cual se define como,

$$\text{Lévy} \sim u = t^{-\lambda}, \quad (1 < \lambda \leq 3), \tag{5.7}$$

dicha distribución presenta una varianza y media infinita.

Algunas de las nuevas soluciones deben ser generadas por caminata aleatoria de Lévy alrededor de la mejor solución obtenida hasta el momento, esto acelerará la búsqueda local. Sin embargo, una parte de las nuevas soluciones debe ser generada por la aleatorización del campo lejano y cuyas ubicaciones deben estar lo suficientemente lejos de la mejor solución actual, esto asegurará que el sistema no quede atrapado en un óptimo local.

Considerando las tres reglas establecidas y con lo descrito en las ecuaciones anteriores, la búsqueda de *cuckoo* puede resumirse como el pseudocódigo 8, que se muestra continuación.

Algoritmo 8 Algoritmo de búsqueda de Cuckoo Search

Input $f(\mathbf{x})$, $\mathbf{x} = (x_1, \dots, x_n)^\top$, i

Generación la posición inicial de n nidos hospedadores x_i .

while *Criterio de parada* **do**

 Obtener un Cucu al azar.

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \alpha \oplus \text{Levy}(\lambda)$$

 Seleccionar el mejor f_i

 Elegir un nido entre los n (por ejemplo un J)

if $f_i > f_j$ **then**

 | Reemplazar j como nueva solución

end

 Se abandona una fracción P_a de los nidos no buenos y se construyen/generan nuevas soluciones y se mantienen las mejores.

 Orederar las mejores soluciones

 Transmitir a las nuevas generaciones.

end

5.5. Algoritmo de búsqueda gravitacional (GSA)

La gravitación es una tendencia que presentan las masas a acelerarse entre sí. La gravedad se encuentra presente en todas partes, presentando una característica única, ningún objeto o partícula se encuentra excepto de ella, esto la hace diferente a cualquier otro tipo de fuerza presente en la naturaleza.

La forma en que la fuerza gravitacional de Newton actúa entre las partículas depende de la distancia entre ellas. En la ley de gravedad de Newton cada partícula atrae a todas las demás con una fuerza que es directamente proporcional a al producto de las masas e inversamente proporcional al cuadrado de la distancia entre ellas:

$$F = G \frac{M_1 M_2}{R^2}, \quad (5.8)$$

donde F es la magnitud de la fuerza gravitacional, G es la constante de gravitacional, M_1 y M_2 son las masas de las partículas y R es la distancia entre las dos partículas. La segunda ley de Newton nos indica que cuando se aplica una fuerza, una partícula, su aceleración, depende solo de la fuerza y su masa,

$$a = \frac{F}{M}, \quad (5.9)$$

Entonces existe una fuerza de atracción de gravedad entre todas las partículas del universo donde el efecto de la partícula más grande y más cercana es mayor. Un aumento en la distancia entre dos partículas significa disminuir la fuerza de gravedad entre ellas.

Además, debido al efecto de la disminución de la gravedad, el valor real de la “constante gravitacional” depende de la edad real del universo por lo que podemos definirla como:

$$G(t) = G(t_0)e^{-\alpha t_0/t}, \quad (5.10)$$

donde $G(t)$ es el valor de la constante gravitacional en el tiempo t . $G(t_0)$ es el valor de la constante gravitacional en el primer intervalo cuántico cósmico de tiempo t_0 .

El algoritmo de Búsqueda gravitacional (GSA), fue propuesto por [Rashedi et al., 2009b], el cual plantean que cada agente se considera un objeto y su aptitud se mide por su masa, es decir el agente con mejor aptitud será el que presente una mayor fuerza de atracción (ver Figura 5.4). Todos estos agentes son atraídos entre sí por la fuerza gravitacional, provocando un movimiento en cada uno de los agentes hacia los que tengan mayor masa. Es decir, todas las masas de los agentes cooperan entre sí para generar movimientos a través de la fuerza gravitacional.

La posición del agente corresponde a una solución del problema, y sus masas gravitacionales se determinarán mediante una función de costo. Es decir, cada agente con masa m es una potencial solución al problema. Una característica propia del algoritmo es la de auto-ajusta de forma adecuada las masas gravitacionales de cada agente dependiendo de si es una solución mejor o no. A medida que el algoritmo evoluciona, las masas deberían ser

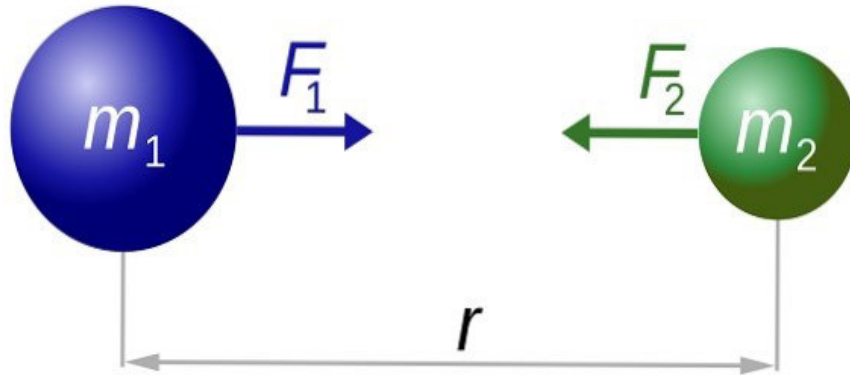


Figura 5.4. Fuerza de atracción entre masas ©.

atraídas por los agentes con mayor masa, dicho de otra forma, el agente con mejor aptitud.

La velocidad actual de cualquier masa es igual a la suma de la fracción de su velocidad anterior y la variación de la velocidad actual. El cambio de velocidad o aceleración de cualquier masa es igual a la fuerza ejercida por el sistema dividida por la masa inercial.

Ahora, consideremos un sistema con N agentes (masas). Definimos la posición del agente i como,

$$X_i = (x_i^1, \dots, x_i^d, \dots, x_i^n) \quad \forall i = 1, 2, \dots, N \quad (5.11)$$

Donde x_i^d representa la posición del agente i en la dirección (dimensión) d . En un momento dado t , definimos la fuerza que actúa sobre la masa i desde la masa j como,

$$F_{ij}^d(t) = G(t) \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \epsilon} (x_j^d(t) - x_i^d(t)) \quad (5.12)$$

donde M_{aj} es la masa gravitatoria activa relacionada con el agente j , M_{pi} es la masa gravitatoria pasiva relacionada con el agente i , $G(t)$ es la constante gravitatoria en el tiempo t , ϵ es una pequeña constante, y $R_{ij}(t)$ es la distancia euclidiana entre el agente i y j y se define como,

$$R_{ij}(t) = \|X_i(t) - X_j(t)\|_2 \quad (5.13)$$

Para dar una característica estocástica, se asume que la fuerza total que actúa sobre el agente i en una dirección d y es la suma ponderada aleatoria de los componentes de las fuerzas

ejercidas por otros agentes:

$$F_i^d(t) = \sum_{j=1, j \neq i}^N \text{rand}() F_{ij}^d(t), \quad (5.14)$$

donde $\text{rand}()$ es un número aleatorio en el intervalo $[0, 1]$. Por tanto, de acuerdo con la ley del movimiento, la aceleración del agente i en el tiempo t , y en la dirección $a_j^d(t)$ está dada como

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)}, \quad (5.15)$$

donde M_{ii} es la masa inercial del agente i .

Además, la velocidad siguiente de un agente se considera como la fracción de su velocidad actual sumada a su aceleración. Por lo tanto, su posición y velocidad podrían calcularse como,

$$v_i^d(t+1) = \text{rand}() \times v_i^d(t) + a_i^d(t) \quad (5.16)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (5.17)$$

Donde $\text{rand}()$ es una variable aleatoria uniforme en el intervalo $[0, 1]$. Este número aleatorio se utiliza en GSA para dar una característica estocástica a la búsqueda.

La constante gravitacional, G , se inicializa al inicio del algoritmo y su valor será reducido con el paso del tiempo, para controlar la precisión de la búsqueda. En otras palabras, G es una función del valor inicial (G_0) y dependiente del tiempo (t),

$$G(t) = G(G_0, t) \quad (5.18)$$

Las masas gravitacionales y de inercia se calculan evaluando la función de objetivo. Una masa más pesada se traduce como un agente más eficiente. Esto significa que los mejores agentes tienen mayores fuerzas de atracción y se mueven más lentamente. Asumiendo la igualdad de la gravedad y la masa inercial, actualizamos la masa gravitatoria e inercial mediante las siguientes ecuaciones:

$$M_{ai} = M_{pi} = M_{ii} = Mi, \quad \forall \quad i = 1, 2, \dots, N \quad (5.19)$$

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \quad (5.20)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^N m_j(t)}, \quad (5.21)$$

donde $fit_i(t)$ representa el valor de fitness del agente i en el tiempo t , $worst(t)$ y $best(t)$ se definen como,

$$best(t) = \min_{j \in \{1, \dots, N\}} fit_j(t) \quad (5.22)$$

$$worst(t) = \max_{j \in \{1, \dots, N\}} fit_j(t) \quad (5.23)$$

Finalmente, el algoritmo de búsqueda gravitacional, se puede describir en su forma estándar como pseudocódigo 9.

Algoritmo 9 Algoritmo de búsqueda Gravitacional

Input $f(\mathbf{x})$, $\mathbf{X} = (x_1, \dots, x_n)^\top, i$

Generación la posición inicial de $x_i(t)$ de forma aleatoria.

while *Criterio de parada* **do**

 Evaluar la función objetivo $f(x_i(t))$ para cada agente en la población $X(t)$.

 seleccionar el mejor y el peor agente de la población $X(t)$.

 Actualice la constante gravitacional G

for $i=0:i < N$ **do**

for $j=i+1:j < N$ **do**

 Calcule la acción de la fuerza sobre el agente i a partir del agente j , ver Ecuación 5.12

end

 Calcule la fuerza total que actúa sobre el agente i del agente j , ver Ecuación 5.14

 Calcule la masa inercial M_i , ver Ecuación 5.21

 Calcule la aceleración del agente i , ver Ecuación 5.15

 Actualice la velocidad del agente i , ver Ecuación 5.16

 Actualice la posición del agente i , ver Ecuación 5.17

end

$t = t + 1$

end

Produce la mejor solución

Devuelve la mejor solución

5.6. Funciones de Prueba

Para validar los métodos de optimización metaheurísticos propuestos, se sometieron a pruebas de rendimiento que incluían encontrar el mínimo global en las funciones no convencionales de alta complejidad (*funciones benchmark*) [Wikipedia, 2020]. Este tipo de funciones presentan un gran reto para los algoritmos y nos permiten identificar y comparar el desempeño de los métodos propuestos.

Los algoritmos serán evaluados con 9 funciones de prueba descritas brevemente, donde se mencionan sus rangos de búsqueda, su formulación matemática y sus mínimos globales teóricos, ver Tabla 5.1.

Tabla 5.1. Funciones de referencia para evaluación de algoritmos Metaheurísticos

No. Función	Formula	Mínimo global	Dominio de búsqueda
F_1 Rastrigin	$f(x) = An + \sum_{i=1}^n [x_i^2 - A \cos(2\pi x_i)]$ where: $A = 10$	$f(0, 0) = 0$	$-5.12 \leq x_i \leq 5.12$
F_2 Ackley	$f(x, y) = -20 \exp \left\{ -0.2 \sqrt{0.5(x^2 + y^2)} \right\} - \exp \{0.5(\cos 2\pi x + \cos 2\pi y)\} + e + 20$	$f(0, 0) = 0$	$-5 \leq x_i \leq 5$
F_3 Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$f(1, 1) = 0$	$-\infty \leq x_i \leq \infty$
F_4 Bukin N.6	$f(x, y) = 100\sqrt{ y - 0.01x^2 } + 0.01 x + 10 $	$f(-10, 1) = 0$	$-15 \leq x \leq -5,$ $-3 \leq y \leq 3$
F_5 N.13	Lévi $f(x, y) = \sin^2 3\pi x + (x - 1)^2(1 + \sin^2 3\pi y) + (y - 1)^2(1 + \sin^2 2\pi y)$	$f(1, 1) = 0$	$-10 \leq x, y \leq 10$
F_6 Cross-in-tray	$f(x, y) = -0.0001 \left[\left \sin x \sin y \exp \left\{ \left 100 - \frac{\sqrt{x^2 + y^2}}{\pi} \right \right\} \right + 1 \right]^{0.1}$	$f(\pm 1.34941, \pm 1.34941) = -2.06261$	$-10 \leq x, y \leq 10$
F_7 Eggholder	$f(x, y) = -(y + 47) \sin \sqrt{\left \frac{x}{2} + (y + 47) \right } - x \sin \sqrt{ x - (y + 47) }$	$f(512, 404.2319) = -959.6407$	$-512 \leq x, y \leq 512$
F_8 Hölder	$f(x, y) = - \left \sin x \cos y \exp \left\{ \left 1 - \frac{\sqrt{x^2 + y^2}}{\pi} \right \right\} \right $	$f(\pm 8.05502, \pm 9.66459) = -19.2085$	$-10 \leq x, y \leq 10$
F_9 N. 2	Schaffer $f(x, y) = 0.5 + \frac{\sin^2(x^2 - y^2) - 0.5}{[1 + 0.001(x^2 + y^2)]^2}$	$f(0, 0) = 0$	$-100 \leq x, y \leq 100$

Además, algunas funciones muestran expresiones trigonométricas y exponenciales con una gran cantidad de máximos y mínimos, otras son demasiado extensas y con rangos de búsqueda más amplios.

Las Figuras 5.5, se observan las funciones de prueba las cuales serán sometidas a cada uno de los algoritmos metaheurísticos planteados en este capítulo, donde podemos detallar el grado de complejidad de dichas funciones, además de incluir mínimos locales que pondrán a prueba los algoritmos propuestos.

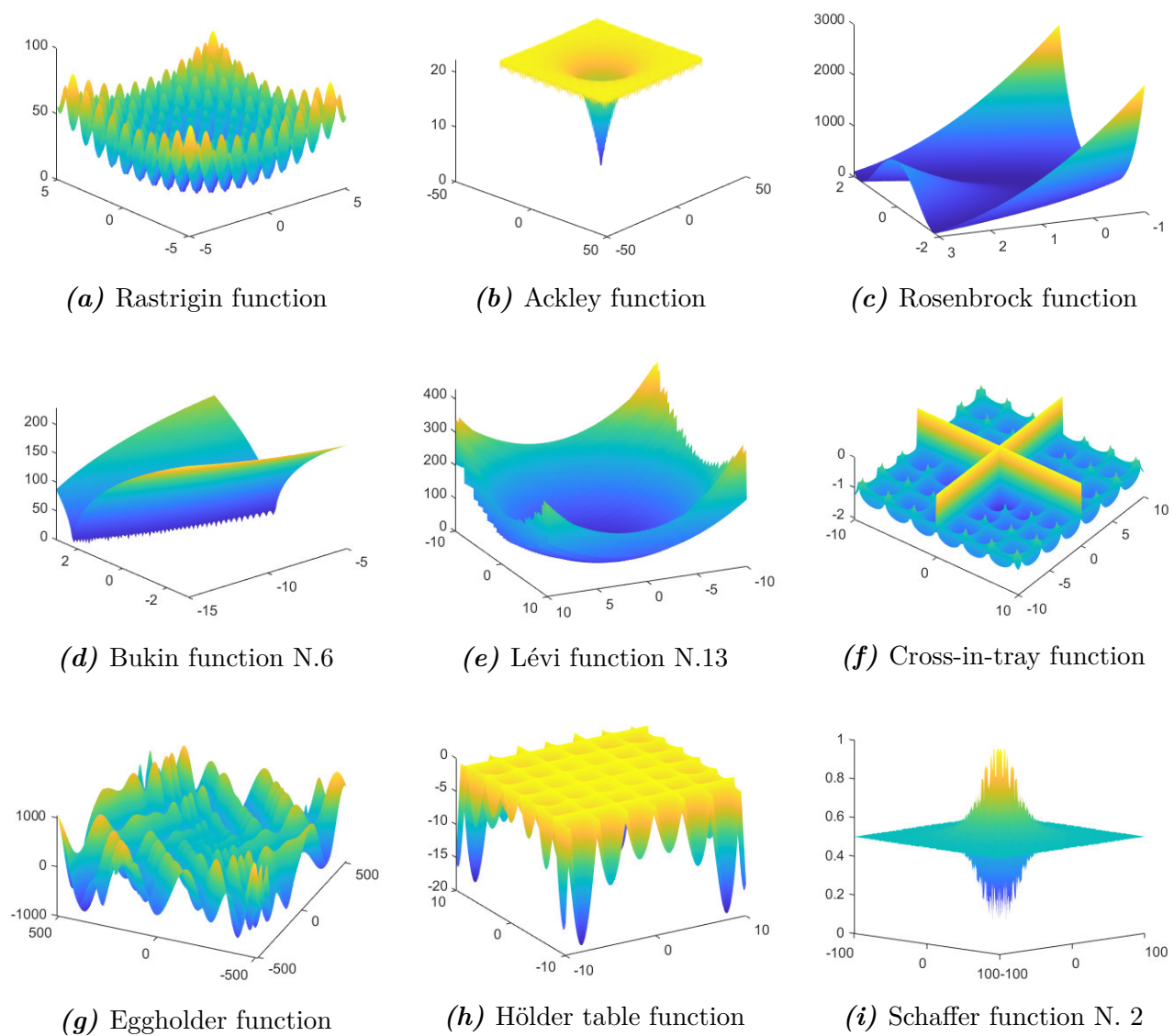


Figura 5.5. Funciones de prueba para validación de algoritmos de optimización Metaheurística.

Se evaluarán pruebas de desempeño, que dependerán del número de iteraciones, error cuadrático medio (RMSE), tiempo de ejecución del algoritmo y prueba de estabilidad (mínimo conseguido) al repetir el experimento 100 veces.

5.6.1. Validación del Algoritmo Metaheurístico PSO

El algoritmo PSO, se puso a prueba con las funciones de referencia descritas en la Tabla 5.1, se seleccionaron específicamente estas funciones, debido a su grado de complejidad, como también la existencia de mínimos locales.

La Tabla 5.2 muestra los resultados numéricos del algoritmo metaheurístico PSO, sometido a las funciones de prueba. El objetivo de esta prueba, es encontrar los mínimos

Tabla 5.2. Resultados obtenidos por el algoritmo PSO para las funciones de referencia 5.1.

Función No.	Óptimo F(x,y)	Mínimo encontrado	PSO		
			RMSE	iteraciones	Tiempo [seg]
F_1	0	5.4315E-6	2.9719E-11	50	0.0025
F_2	0	3.2583E-5	1.0617E-09	60	0.0045
F_3	0	2.25843E-5	5.1005E-10	40	0.0018
F_4	0	4.2344E-5	1.7930E-09	30	0.0015
F_5	0	3.3594E-5	1.1286E-09	50	0.0028
F_6	-2.06261	-2.062588	4.8400E-10	60	0.0038
F_7	-959.64	-959.6389	1.2100E-06	30	0.0019
F_8	-19.2085	-19.20785	4.2250E-07	25	0.0012
F_9	0	3.662E-4	1.3410E-07	30	0.0021

de cada función de prueba con la mejor exactitud posible y cuantificar el tiempo que se toma calculando la mejor solución.

En la Figura 5.6 se observa como el algoritmo converge a una solución ante cada función de prueba, consiguiendo el mínimo en cada una de estas funciones. Se seleccionaron las funciones de prueba *Hölder table function* y *Ackley function*, para observar el comportamiento del enjambre de partículas (ver Figura 5.7), el enjambre sigue las reglas propuestas por el algoritmo. Al final de las iteraciones, si se consiguió un mínimo global, el enjambre se posicionara en una sola posición.

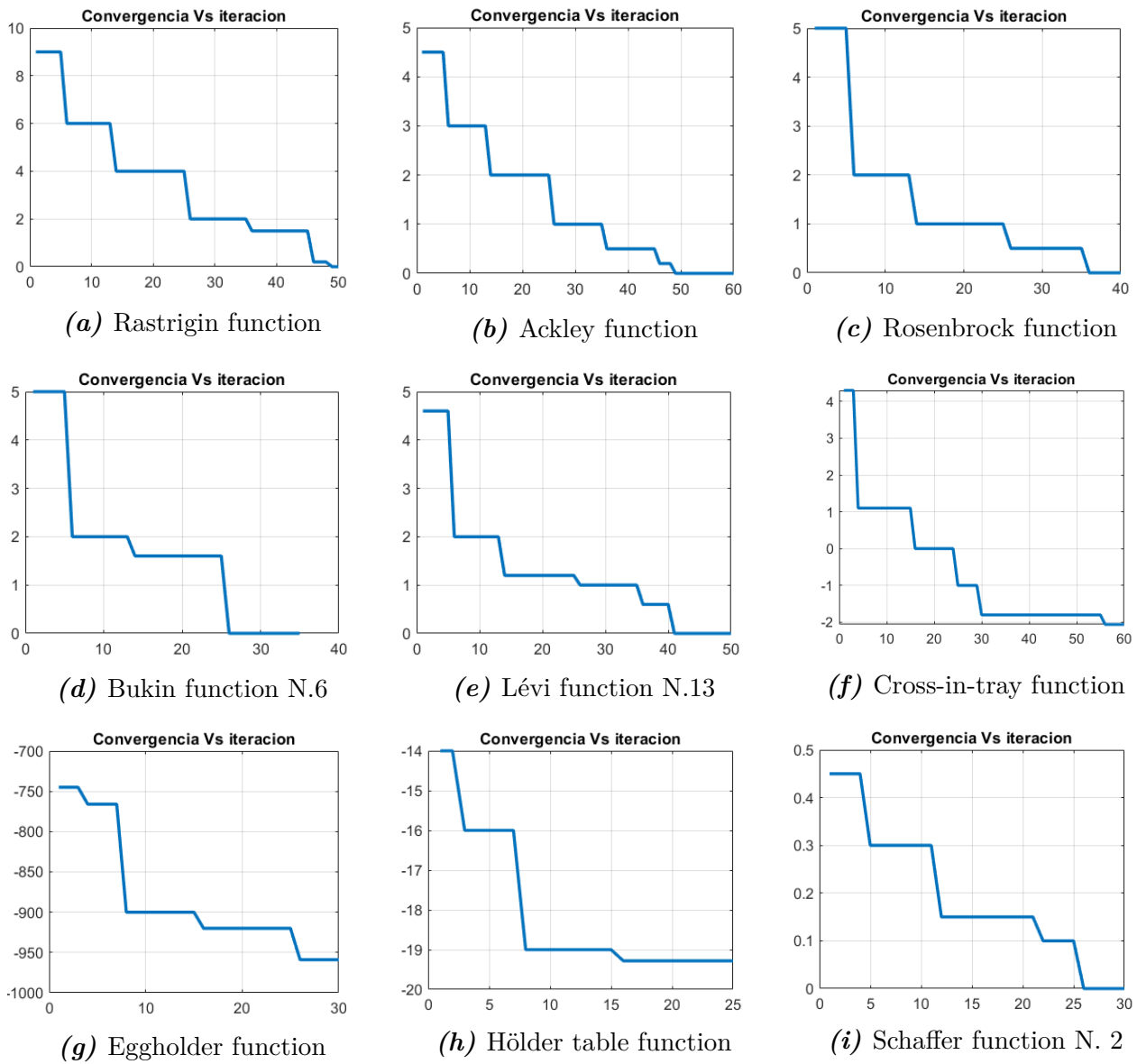


Figura 5.6. Convergencia VS Iteración con el algoritmo PSO.

5.6.2. Validación del Algoritmo Metaheurístico ED

La implementación de la evolución diferencial (ED) es relativamente sencilla, en comparación con los algoritmos genéticos. Si se utiliza un software basado en vectores/matrices, como Matlab, la implementación es más sencilla.

Para evaluar la convergencia, estabilidad y eficiencia del algoritmo ED, se puso a prueba con las funciones de referencia descritas en la Tabla 5.1, se seleccionaron específicamente estas funciones, ya que presentan un gran reto para los algoritmos debido a que son funciones no

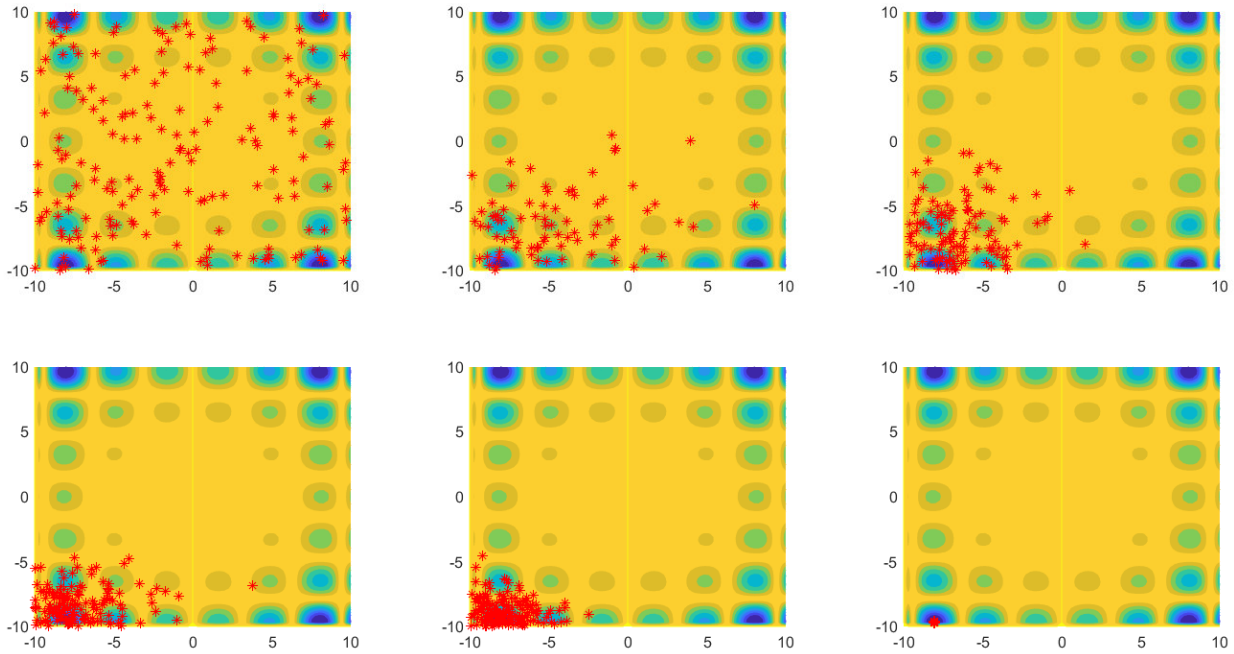


Figura 5.7. Comportamiento del enjambre de partículas ante la función de prueba *Hölder table function*.

lineales y complejas.

La Tabla 5.3 muestra los resultados numéricos del algoritmo metaheurístico ED, sometido a las funciones de prueba. El objetivo de este experimento, es encontrar los mínimos

Tabla 5.3. Resultados obtenidos por el algoritmo ED para las funciones de referencia 5.1.

Función No.	Óptimo $F(x,y)$	Mínimo encontrado	ED		
			RMSE	iteraciones	Tiempo [seg]
F_1	0	4.4575E-5	1.9869E-09	100	0.0011
F_2	0	8.2361E-5	6.7833E-09	65	0.00086
F_3	0	4.24523E-4	1.8022E-07	54	0.00076
F_4	0	8.8744E-5	7.8755E-09	46	0.00057
F_5	0	5.95874E-5	3.5507E-09	63	0.00084
F_6	-2.06261	-2.0582	1.9360E-05	30	0.00024
F_7	-959.64	-959.6384	2.5600E-06	36	0.00033
F_8	-19.2085	-19.19775	1.1556E-04	30	0.00023
F_9	0	4.985E-4	2.4850E-07	40	0.00046

de cada función de prueba, con la mejor exactitud posible y cuantificar el tiempo que se toma calculando la mejor solución. En la Figura 5.8 se observa como el algoritmo converge a una solución ante cada función de prueba, consiguiendo el mínimo en cada una de estas funciones.

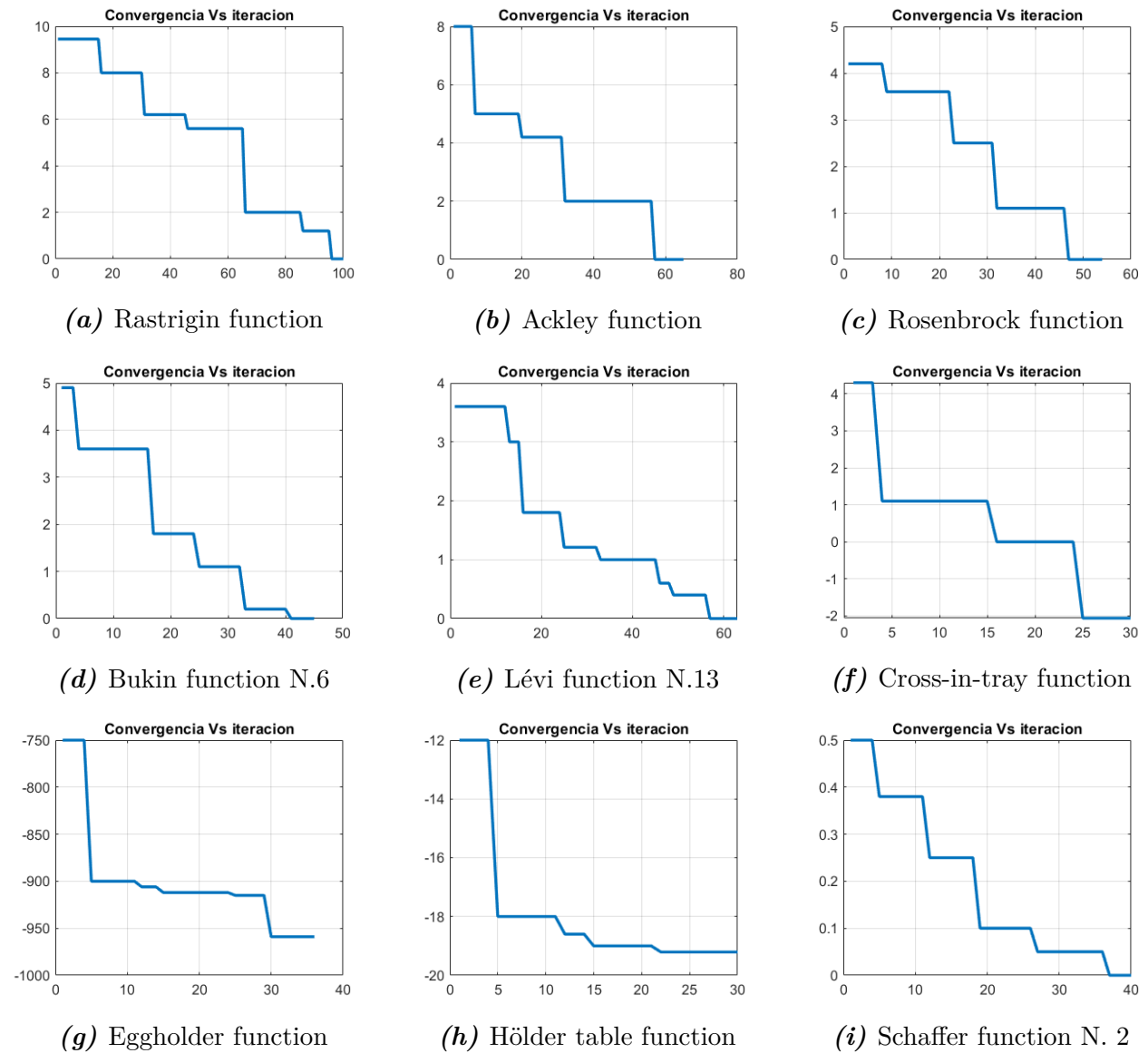


Figura 5.8. Convergencia VS Iteración con el algoritmo ED.

5.6.3. Validación del Algoritmo Metaheurístico CS

Para demostrar la eficiencia del algoritmo CS, se realizaron pruebas con las funciones de referencia descritas en la Tabla 5.1, estas funciones presentan mínimos locales (ver Figura 5.5g, 5.5h), lo cual pondrán a prueba el algoritmo para encontrar el mínimo global.

La Tabla 5.4 muestra los resultados numéricos del algoritmo metaheurístico CS, sometido a las funciones de referencia, consiguiendo el mínimo global en cada una de ellas. En la Figura 5.9 se observa como el algoritmo converge a una solución para cada función de prueba.

Tabla 5.4. Resultados obtenidos por el algoritmo CS para las funciones de referencia 5.1.

Función No.	Óptimo F(x,y)	Minimo encontrado	CS		
			RMSE	iteraciones	Tiempo [seg]
F_1	0	1.25845E-7	1.5837e-14	46	0.0053
F_2	0	6.10423E-7	3.7262e-13	42	0.0056
F_3	0	3.25059E-6	1.0566e-11	30	0.0042
F_4	0	5.56701E-6	3.0992e-11	42	0.0050
F_5	0	3.40372E-6	1.1585e-11	36	0.0045
F_6	-2.06261	-2.062531	6.2410e-09	52	0.0064
F_7	-959.6407	-959.6406627	1.3913e-09	28	0.0036
F_8	-19.2085	-19.20792	3.3640e-07	30	0.0038
F_9	0	5.05358E-7	2.5539e-13	26	0.0031

En la Figura 5.10 se puede apreciar, como el algoritmo encuentra los mejores nidos (mejor mínimo encontrado), a medida que el algoritmo evoluciona, la información de estos nidos se transmite entre generaciones de *cuckoo* s, el criterio de parada para este algoritmo consiste en que un porcentaje alto de *cuckoo* se ubique en una región dada, es decir que la manada se aloje alrededor del mejor nido encontrado.

En la figura los * negros representan huevos *cuckoo* alojados en nidos aleatorios, el * verde representa la mejor solución encontrada en un tiempo t .

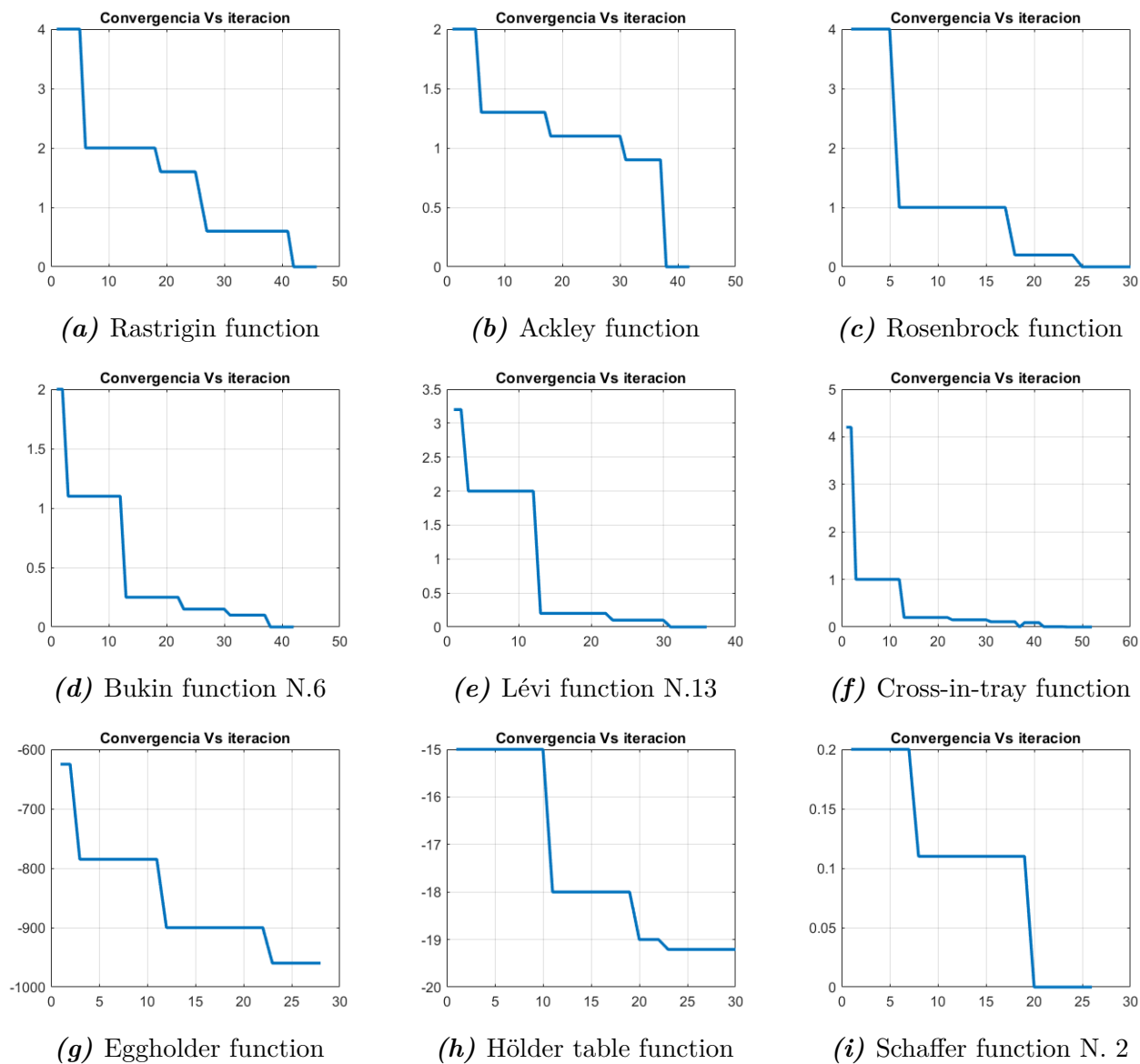


Figura 5.9. Convergencia Vs Iteración con el algoritmo CS.

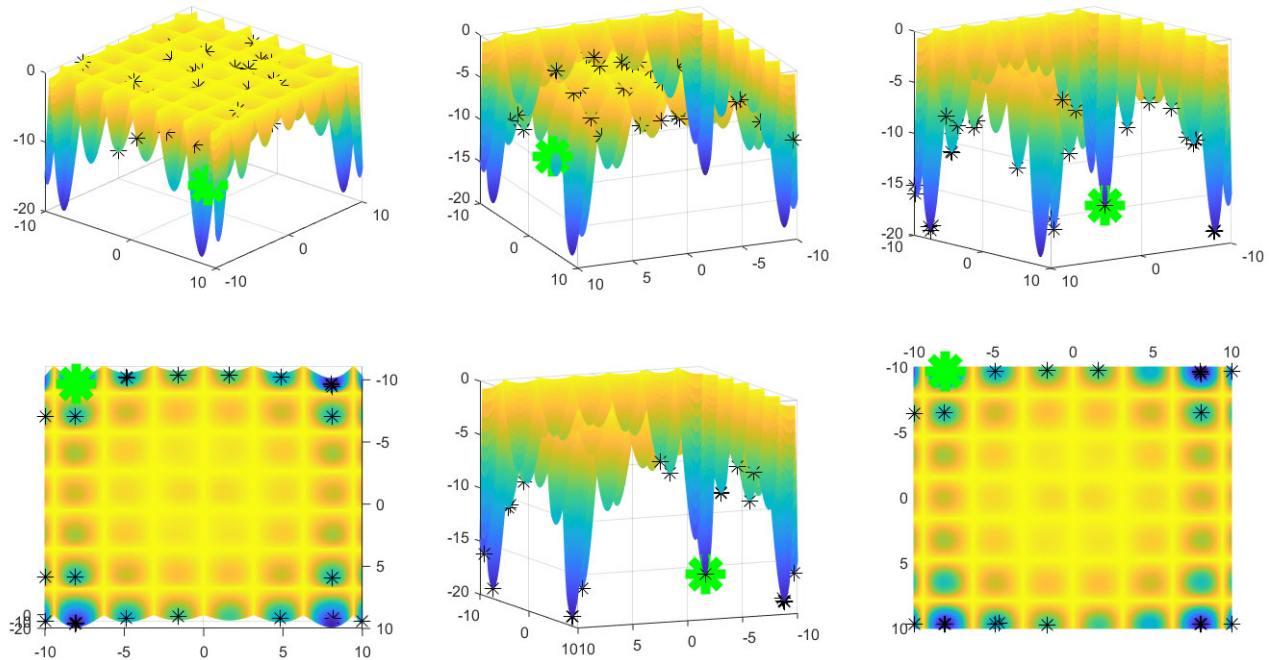


Figura 5.10. Comportamiento de las aves *cuckoo* en la función de prueba *Hölder table function*.

5.6.4. Validación del Algoritmo Metaheurístico GAS

Siguiendo el Pseudocódigo 9, es relativamente sencillo implementar el algoritmo GAS en cualquier lenguaje de programación, para facilitar los cálculos y obtener tiempos de ejecución mucho más rápidos, se sugiere implementar el algoritmo en software con paquetes vectoriales y matriciales como lo es Matlab.

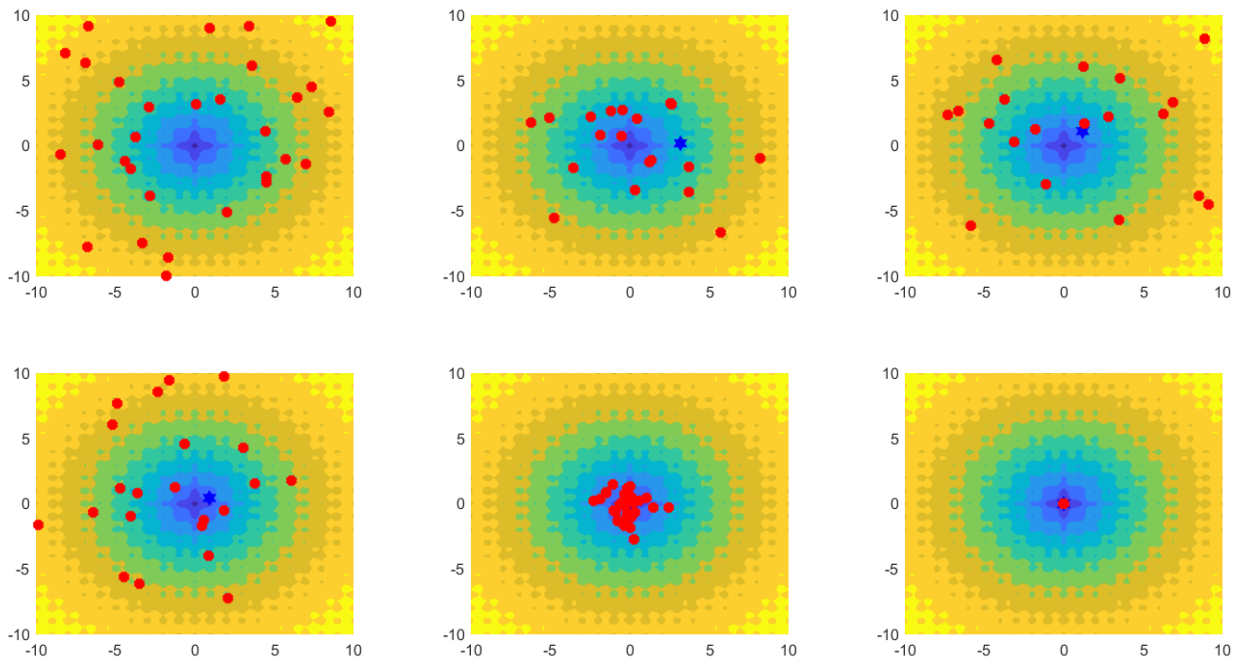
Una vez definido los lineamentos del algoritmo GAS, se procede a validar el algoritmo utilizando funciones de prueba con soluciones analíticas o conocidas (ver Tabla 5.1).

La Tabla 5.5 muestra los resultados numéricos del algoritmo metaheurístico GAS, sometido a las funciones de referencia, consiguiendo el mínimo global en cada una de ellas. En la Figura 5.12 se observa como el algoritmo converge a una solución para cada función de prueba.

En la figura 5.11 podemos observar de como se mueven los agentes dependiendo de su masa inercial y de la fuerza de atracción, los * representan los agentes, mientras que él * representa el agente con mayor masa.

Tabla 5.5. Resultados obtenidos por el algoritmo GAS para las funciones de referencia 5.1.

Función No.	Óptimo $F(x,y)$	Mínimo encontrado	GAS		
			RMSE	iteraciones	Tiempo [seg]
F_1	0	3.84585E-4	1.4791e-07	58	0.056
F_2	0	5.58901E-4	3.1237e-07	45	0.048
F_3	0	2.9544E-4	8.7285e-08	59	0.059
F_4	0	1.96702E-4	3.8692e-08	72	0.068
F_5	0	6.5483E-4	4.2880e-07	49	0.044
F_6	-2.06261	-2.0624942	1.3410e-08	73	0.070
F_7	-959.6407	-959.63969	1.0161e-06	88	0.083
F_8	-19.2085	-19.20785	4.2250e-07	30	0.026
F_9	0	2.6527E-4	7.0368e-08	24	0.018

**Figura 5.11.** Comportamiento del algoritmo GAS ante la función de prueba *Ackley function*.

5.6.5. Comparación de los Algoritmos PSO, ED, CS, GAS

Para realizar una comparación, que nos permitan establecer que algoritmo tiene el mejor desempeño, se establecieron los factores de ajuste (*tuning*) para cada algoritmo de optimización, los cuales deben de ser revisados cuidadosamente para obtener el máximo rendimiento de cada

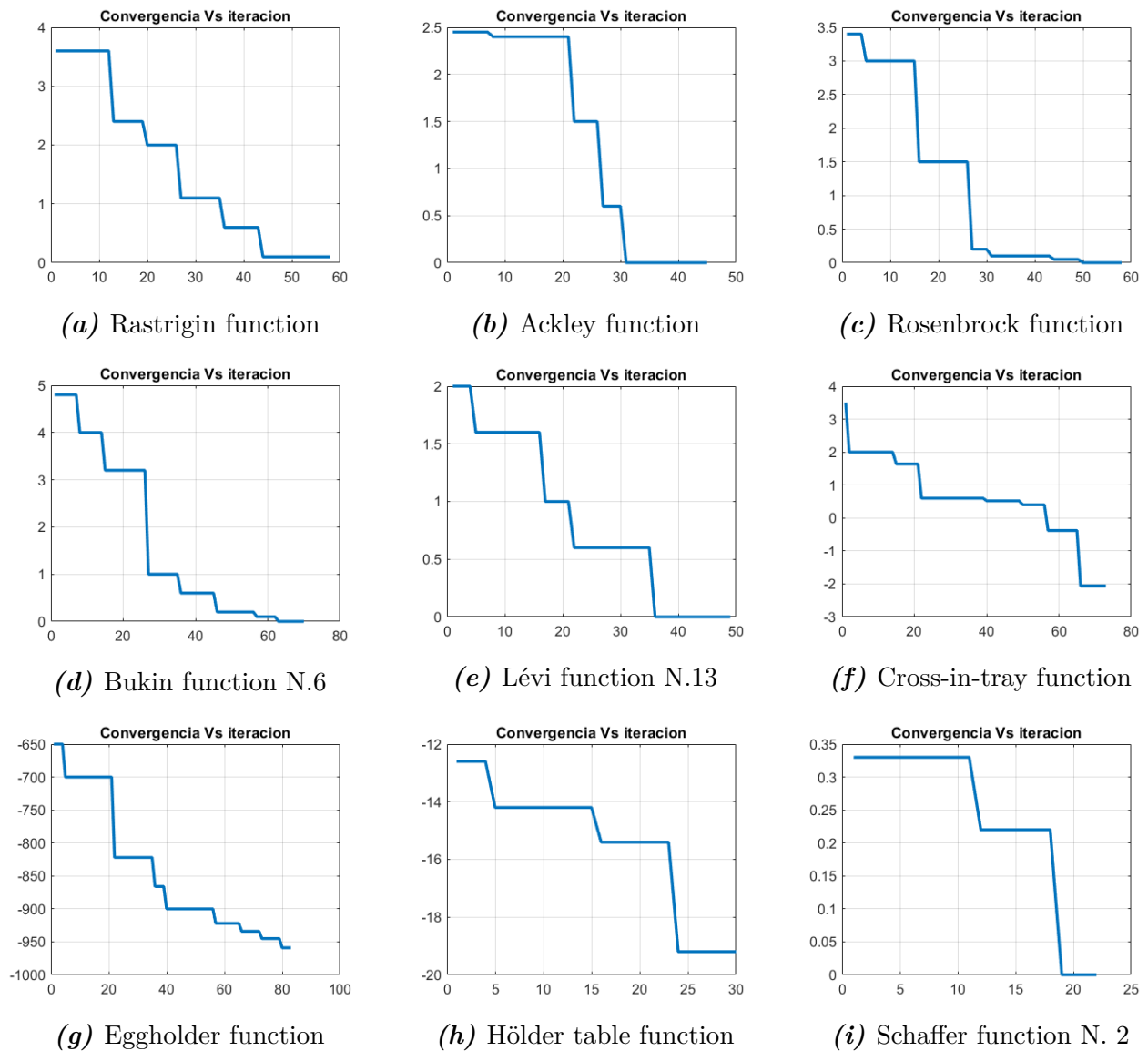


Figura 5.12. Convergencia Vs Iteración con el algoritmo GAS.

algoritmo tanto en tiempo de ejecución como en exactitud.

Los factores de ajuste son propios de cada método de optimización y son seleccionados como se muestra en la Tabla 5.6. En el algoritmo PSO se selecciona $c_1 = c_2 = 0.45$; el algoritmo de DE, se requiere definir una probabilidad de combinación $C_p = 0.2$ y una probabilidad de mutación $F = 0.1$; El algoritmo CS $P_a = 0.25$ y finalmente el algoritmo GAS $v_0 = 0.1, G_0 = 100, \alpha = 10$. Es importante resaltar que los parámetros de ajuste pueden ser diferentes para otro tipo de aplicación.

Para homologar las condiciones comunes de operación, en todos los métodos se considera

Algoritmo	Parámetro de ajuste
PSO	$C_1 = C_2 = 0.45$
ED	$C_p = 0.2, F = 0.1$
CS	$P_a = 0.25$
GAS	$v_0 = 0.1, G_0 = 100, \alpha = 10.$

Tabla 5.6. Factores de ajuste para los algoritmos de optimización

una población, enjambre o grupo inicial de $n = 40$, con un máximo de $N = 1000$ generaciones y sin condición de paro anticipada, además de que se repetirá el experimento para cada función 100 veces. Los parámetros a considerar para la comparación de rendimiento, serán: mínimo encontrado y tiempo de ejecución.

En las Figuras 5.13, se puede observar la convergencia de cada algoritmo ante las 100 repeticiones, esto se repitió para cada función de prueba. Podemos observar que el algoritmo CS supera a los algoritmos PSO, ED y GAS para esta prueba. Las principales razones son:

- Un equilibrio entre aleatoriedad y selección.
- Menor número de parámetros de control.

Como en el caso de cualquier algoritmo metaheurístico, un buen equilibrio entre la búsqueda local y la exploración eficiente de todo el espacio de búsqueda suele conducir a un algoritmo más eficiente. Por otra parte, solo hay dos parámetros en este algoritmo, el tamaño de la población n , y p_a . Una vez que n es definida, el parámetro p_a es el encargado de controlar esencialmente la aleatoriedad y la búsqueda local. Pocos parámetros hacen que un algoritmo sea menos complejo y, por tanto, más genérico.

En las Figuras 5.14 podemos observar la comparación del tiempo de ejecución de cada algoritmo con el mejor criterio de parada que se ajustara a cada uno de ellos. El algoritmo ED, demostró ser el más rápido, debido a sus componentes altamente matriciales, por otro lado los algoritmos PSO y CS demostraron gran estabilidad y un buen tiempo de ejecución, el algoritmo GAS fue el más lento, esto se debe a que propiedad de exploración al pasar el tiempo disminuye notablemente, haciéndolo más lento.

Observando los resultados obtenidos y haciendo una comparación entre tiempo de ejecución y mínimo encontrado, se seleccionará el algoritmo CS como el encargado para optimizar el método APF, esta integración se describe en la siguiente sección.

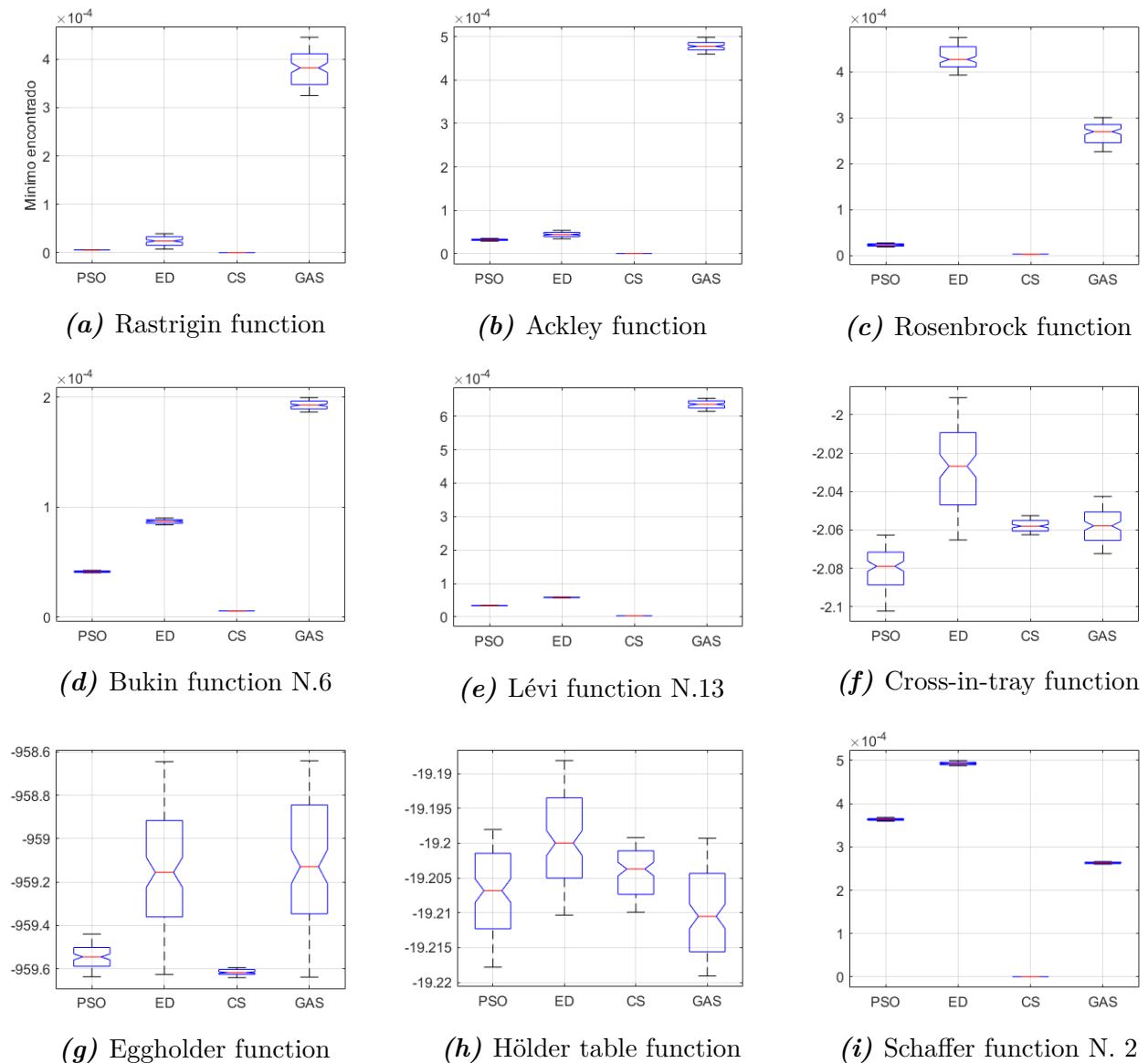


Figura 5.13. Comparación de convergencia de los algoritmos PSO, ED, CS, GAS.

5.7. Integración APF con el algoritmo CS

EL principal problema del método APF, es la presencia de los mínimos locales, estos mínimos locales son producidos por la interacción de la fuerza de atracción y repulsión, estas fuerzas al ir en la misma dirección, pero en sentido opuesto, ocasiona que la carga puntual (robot) queda atrapada y no logre alcanzar la meta 5.15.

Una solución a este problema, consiste en alterar las líneas de fuerzas generadas, es decir

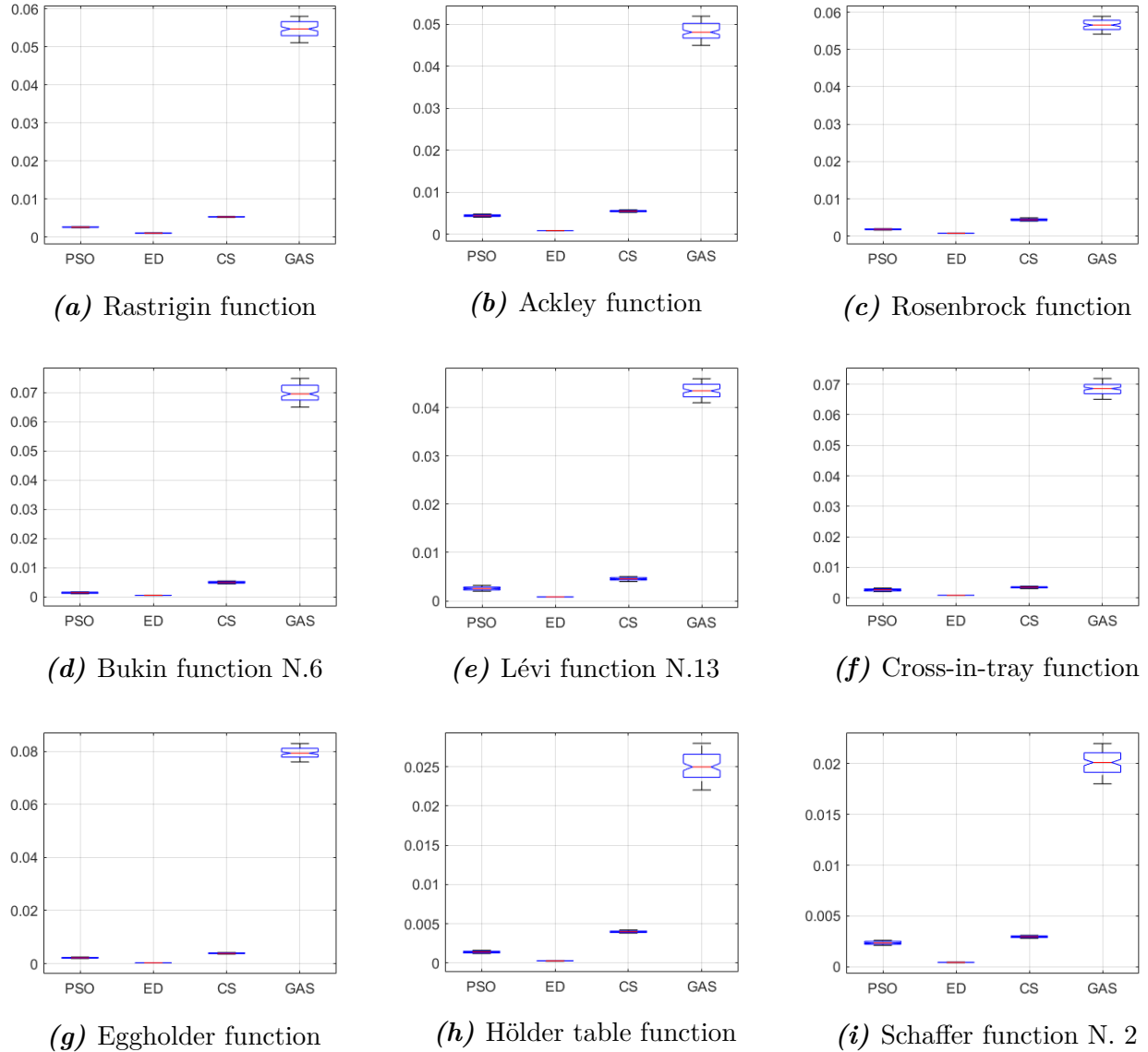


Figura 5.14. Comparación del tiempo de ejecución de los algoritmos PSO, ED, CS, GAS.

que no coincidan en dirección, la fuerza de repulsión y la fuerza de atracción. Si observamos el mapa como una rejilla de puntos, donde cada punto representa una fuerza, esta fuerza se puede descomponer en una que va en dirección al eje X y otra en dirección al eje Y. Esta descomposición de fuerzas se pueden representar de la siguiente forma:

$$F_x = F_{x_att} + F_{x_rep} \quad (5.24)$$

$$F_y = F_{y_att} + F_{y_rep} \quad (5.25)$$

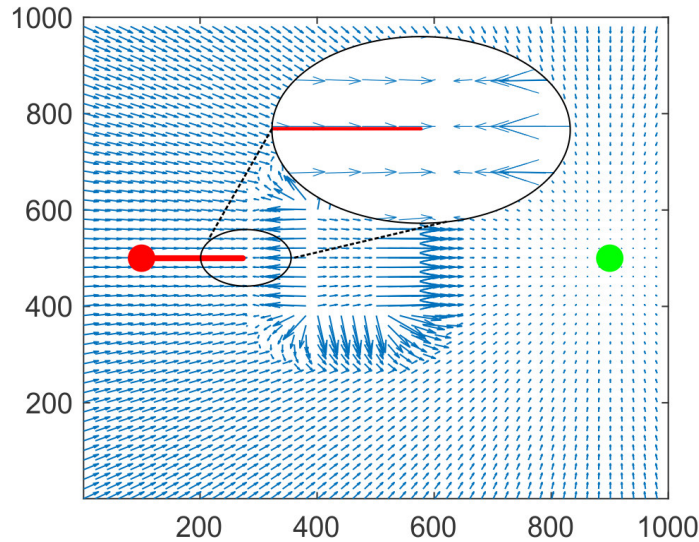


Figura 5.15. Problema de mínimo local APF

Las Ecuaciones 5.24 ,5.25, representan la fuerza total del sistema, con sus componentes tanto en X como en Y. Una manera de hacer que estas fuerzas presenten una distorsión en su dirección es la siguiente:

$$F_x = F_{x_att} + F_{x_rep} + \alpha F_{y_rep} \quad (5.26)$$

$$F_y = F_{y_att} + F_{y_rep} + \beta F_{x_rep} \quad (5.27)$$

Como resultado de aplicar los cambios a las fuerzas descritas en las Ecuaciones 5.26, 5.27, se pueden obtener los resultados mostrados en la Imagen 5.16, donde Los vectores del campo repulsivo tienen una nueva dirección, diferente a la fuerza del campo de atracción, esto ocasionará que el robot no quede atascado y logre evadir el obstáculo sin problema.

Una vez resuelto el problema de mínimos locales, se crea otro nuevo inconveniente con el método propuesto, el cual consiste en seleccionar los valores de α y β adecuados para que la ruta sea la más corta y eficiente (Ver figura 5.17). En la Figura 5.17a se seleccionó un valor de $\alpha = 1.2$ y $\beta = -1.3$, podemos observar que se obtiene una solución al problema propuesto,

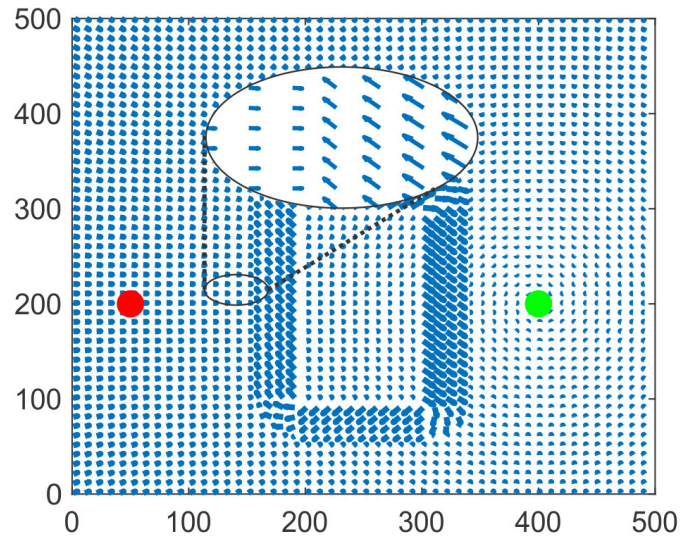


Figura 5.16. Distorsión de las líneas de fuerza del método APF

pero la ruta no es la más corta, por otro lado en la Figura 5.17b se seleccionó un valor de $\alpha = -1.2$ y $\beta = 1.3$ obteniendo como resultado una más corta. Para solucionar este problema, se recurre a realizar una optimización mediante el algoritmo metaheurístico CS.

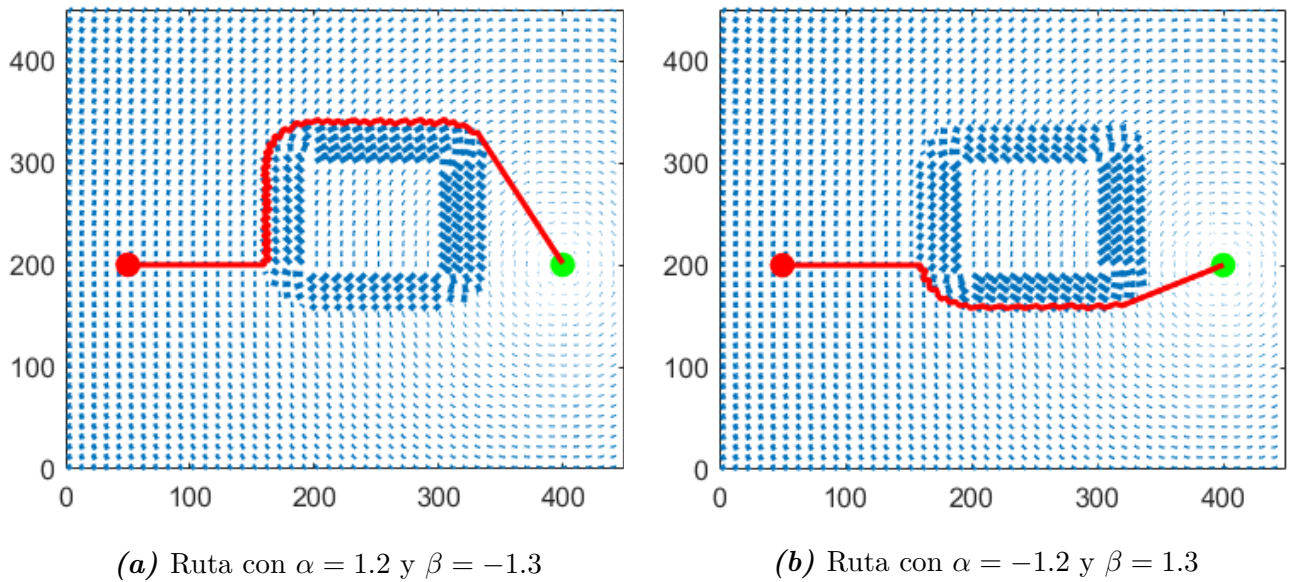


Figura 5.17. Rutas generadas a partir de la modificación de las fuerzas del método APF

Considerando el problema descrito, se seleccionó como función objetivo el tamaño de la ruta generada, la cual se define como:

$$F_{fit} = \|X_{path}(end) - X_{obj}\|_2 + \|X_{path}\|_2 \quad (5.28)$$

Donde el primer termino de la Ecuación 5.28 representa la distancia euclidiana entre el último punto de la ruta calculada y la meta, el segundo término representa la distancia euclidiana de la ruta.

Una vez definida la función objetivo (Ver Ecuación 5.28) y los parámetros a optimizar (α y β), se procede a detallar los pasos necesarios para realizar la integración con el algoritmo CS:

1. Definir los parámetros de CS (P_a) y los rangos de operación de las variables α y β . Los parámetros α y β , mediante un experimento estocástico se determinó que sus rangos de operación deben estar entre $[-10, 10]$ para obtener resultados óptimos.
2. Generar aleatoriamente los parámetros a optimizar dentro de los rangos propuestos.
3. Aplicar el algoritmo APF basado en los factores generados aleatoriamente, una vez minimizado el APF, se procede a evaluar la función objetivo (Ver Ecuación 5.28).
4. Actualizar los factores globales del algoritmo CS.
5. Repetir los pasos 2, 3, 4 para el número de Nidos seleccionado.
6. Obtener los mejores valores de α y β .
7. Aplicar el APF con base a los valores actualizados de α y β y calcular la función objetivo.
8. Después de encontrar los valores globales de cada parámetro como el mejor valor, la ruta generada debe corresponder a la mejor ruta calculada.

Finalmente, la optimización del método APF con el algoritmo CS se puede resumir en el diagrama de flujo mostrado a continuación.

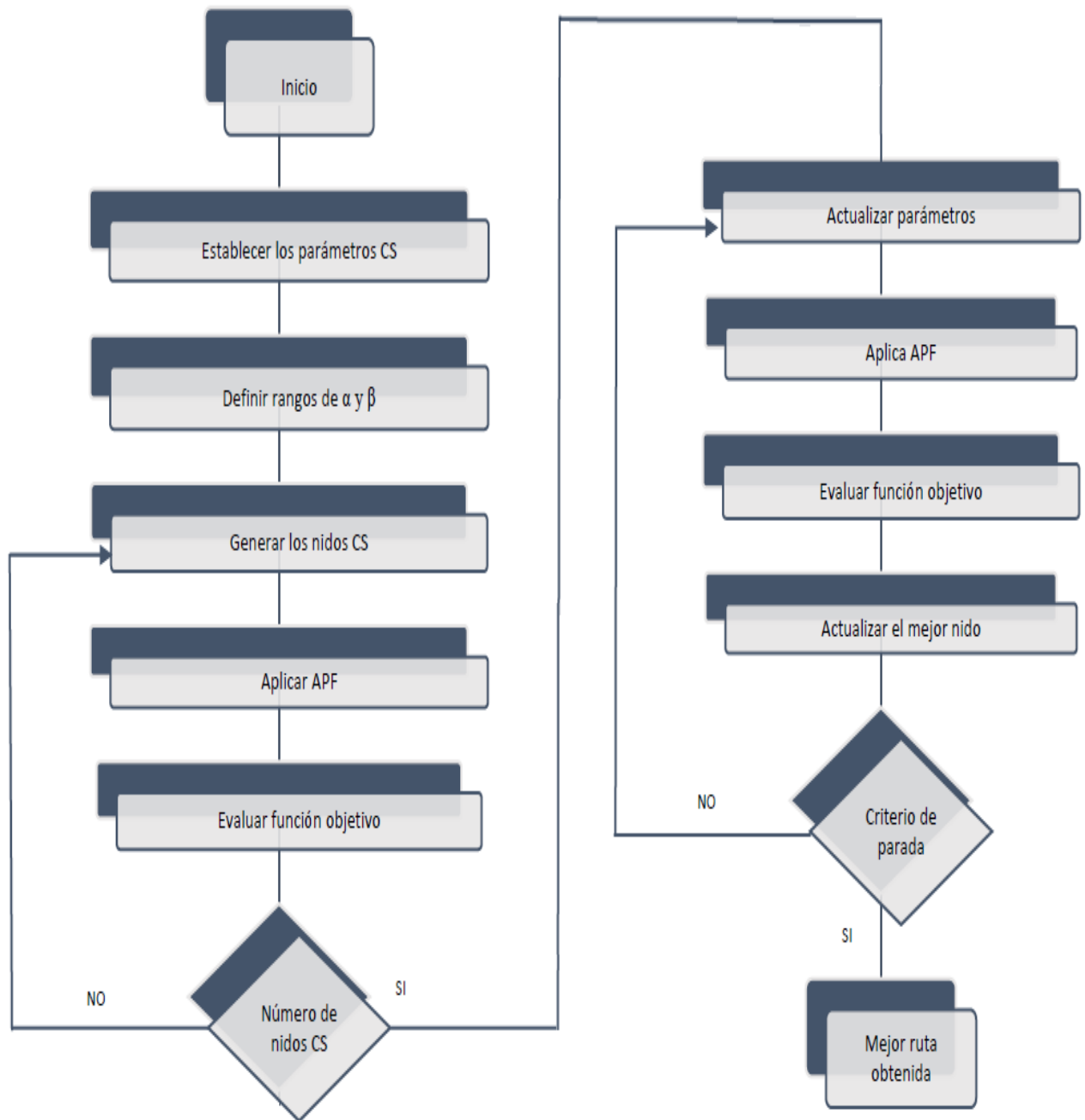


Figura 5.18. Diagrama de flujo del método APF+CS

5.7.1. Validación de método APF+CS

Se sometió el método APF+CS a escenarios con cierto grado de complejidad, el objetivo del experimento es encontrar la ruta más corta que lleve a la carga puntal (robot) de un punto de partida a la meta, en las Figuras 5.19, el punto rojo representa el punto de partida, mientras

que la meta es descrito por el punto verde.

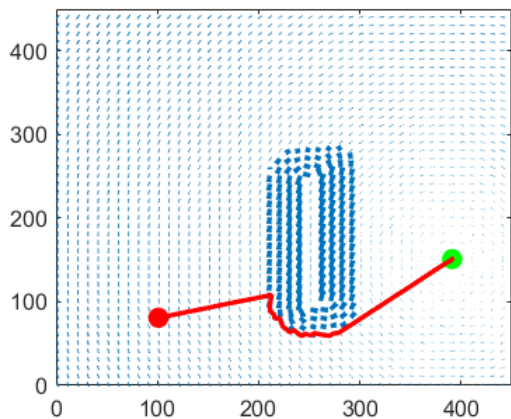
Se seleccionaron diferentes puntos de inicio y de meta, así como diferente posiciones de los obstáculos, para comprobar la estabilidad del algoritmo. Uno de los principales retos es el escenario 5.19d, este escenario en forma de C puede ocasionar que la carga puntal (robot) quede atascado, pero el algoritmo es capaz de evitar este obstáculo con los parámetros de α y β .

Otro de los retos más estudiados en la literatura es el visto en la Figura 5.19f, donde se ubica a la carga puntal dentro de la C formada por el obstáculo, en este escenario sin la ayuda de la optimización CS, no se lograría salir del mínimo local generado en el centro del obstáculo.

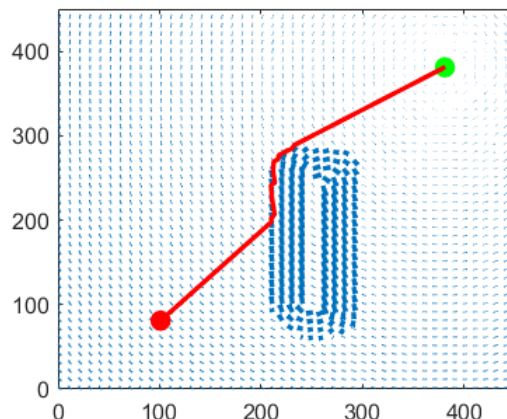
En la Tabla 5.7 se detalla las coordenadas obtenidas por el método APF+CS, así como su error cuadrático medio con respecto a la meta. Es importante resaltar que el método de APF no permite acercarse en su totalidad a la meta, debido a que la distancia euclidiana entre la carga y la meta puede ocasionar una singularidad al ser cero.

Tabla 5.7. Resultados obtenidos por el método APF+CS con los escenarios de la Figura 5.19.

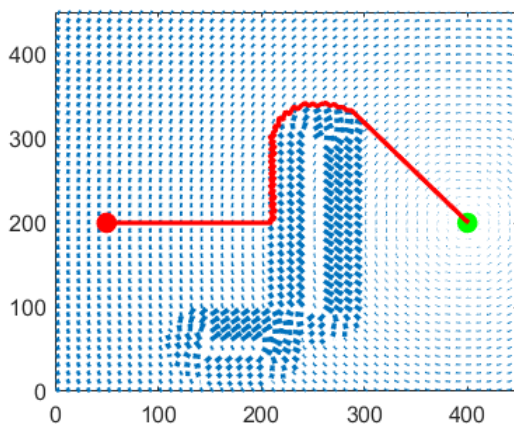
Escenario	Meta	Meta	APF+CS
No.	(X, Y)	encontrada	RMSE
5.19a	[150, 400]	[148, 398]	4
5.19b	[381, 381]	[378, 380]	5
5.19c	[400, 200]	[398, 199]	2.5
5.19d	[400, 200]	[397, 198]	6.5
5.19e	[400, 200]	[398, 199]	2.5
5.19f	[400, 130]	[399, 130]	0.5



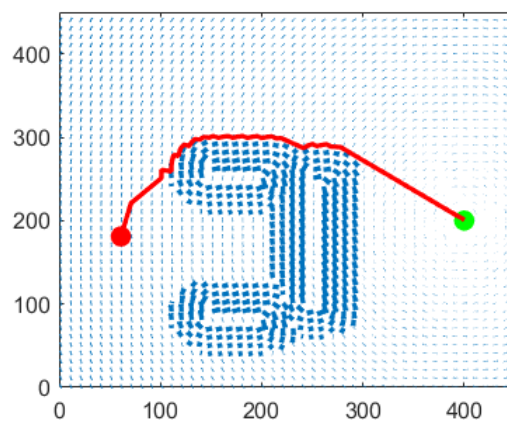
(a) Ruta con $\alpha = -3.5$ y $\beta = 4.6$



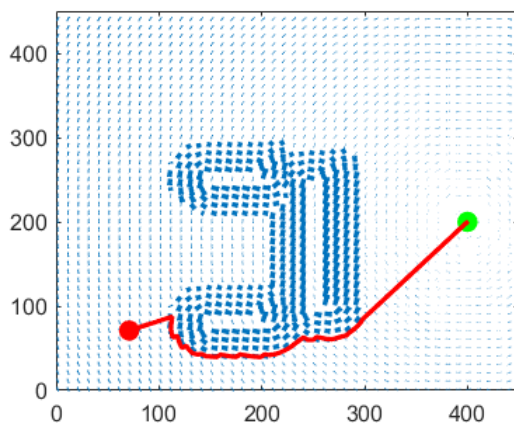
(b) Ruta con $\alpha = 3.4$ y $\beta = -4.4$



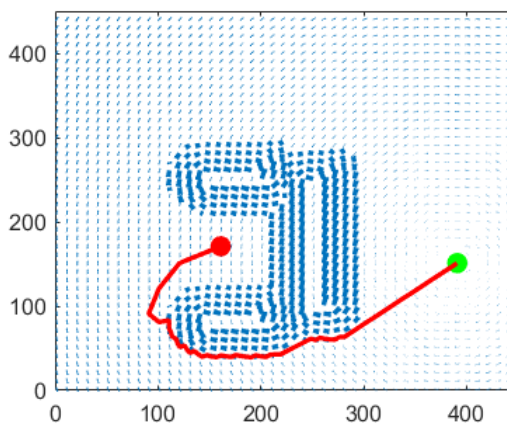
(c) Ruta con $\alpha = -5.3$ y $\beta = 2.2$



(d) Ruta con $\alpha = -8.2$ y $\beta = 2.45$



(e) Ruta con $\alpha = 4.4$ y $\beta = -6.77$



(f) Ruta con $\alpha = 9.25$ y $\beta = -3.33$

Figura 5.19. Optimización del método APF mediante el algoritmo CS

CAPÍTULO 6

Conclusiones

En este trabajo, se presenta un análisis detallado de los fundamentos matemáticos de las leyes de interacción de cargas electrostáticas para aplicaciones en robótica móvil, en específico en la planificación de trayectorias. Esta metodología plantea la solución de la planificación de trayectorias de un robot que evoluciona en un ambiente conocido representado por un mapa de potenciales. Ciertas depresiones pueden ocurrir por efecto de la interacción global de fuerzas y el robot puede ser atraído hacia un mínimo local, lo cual puede ser solventado variando dinámicamente los parámetros del optimizador. Inicialmente, para encontrar la mejor solución, se compararon cuatro técnicas de optimización numérica determinística o dependientes del gradiente, las se seleccionaron como los mejores, la técnica de Levenberg-Marquardt y el método de Dog-Leg para la minimización de la función objetivo que permite una evolución hacia la meta. Los resultados numéricos demuestran que la planificación de trayectorias mediante potenciales eléctricos y técnicas de optimización numérica obtienen rutas continuas, y permiten la evasión dinámica de obstáculos. Además, se pueden aplicar un sin número de obstáculos como de potenciales se disponga, y el método aun en casos extremos se desempeña adecuadamente. Aunque estos algoritmos presentan una mayor exploración y explotación de la función a minimizar, la solución puede estar comprometida al caer en mínimos locales. Para solucionar este problema se recurrieron a los algoritmos metaheurísticos PSO, ED, CS, GAS,

para seleccionar que algoritmo se desempeñaba de mejor forma, se sometieron a las funciones de prueba, el resultado de experimento demuestra que el algoritmo CS fue el más estable y rápido. Al integrar el método APF y CS y someterlo a escenarios donde se generaban mínimos locales, la metodología propuesta consiguió obtener una solución a demás de generar la ruta más corta. Esta metodología permitió explorar escenarios complejos como los estudiados en el Capítulo 5, obteniendo en cada uno de ellos soluciones óptimas.

Trabajos futuros

Con respecto a la investigación desarrollada hasta el momento, se proponen los siguientes temas como posibles trabajos futuros:

- Realizar curvas suavizadas a partir de la ruta generada por el método APF+CS.
- Implementar un controlador robusto para que se incluya el modelo real de un robot y así seguir la ruta generada.

Publicaciones

1. *Path planning for mobile robots based on optimal interaction of electrostatic fields..*

Autores: Daniel Fernando Zambrano-Gutierrez, Emmanuel Ovalle-Magallanes, Horacio Rostro-Gonzalez, Juan Gabriel Avina-Cervantes

Congreso Nacional de Control Automático 2021.

2. *Boost Converter Controlled by a PID Tuned with Metaheuristics Optimization..*

Autores: José Manuel Vido Ramirez, Daniel Fernando Zambrano-Gutierrez, Juan Gabriel Avina-Cervantes

Congreso Nacional de Control Automático 2021.

Referencias

- ASİL, U. and USLU, E. (2020). Potential field path planning and potential field path tracking. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–4.
- Baquela, E. G. and Redchuk, A. (2013). *Optimización Matemática con R. Volumen I: Introducción al modelado y resolución de problemas*. Bubok Publishing.
- Barraquand, J., Langlois, B., and Latombe, J.-C. (1991). Numerical potential field techniques for robot path planning. In *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pages 1012–1017 vol.2.
- Bing, H., Gang, L., Jiang, G., Hong, W., Nan, N., and Yan, L. (2011). A route planning method based on improved artificial potential field algorithm. *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 550–554.
- Bradley, D. (2010). Novel 'cuckoo search algorithm' beats particle swarm optimization in engineering design.
- Candeloro, M., Lekkas, A. M., Sørensen, A. J., and Fossen, T. I. (2013). Continuous Curvature Path Planning using Voronoi diagrams and Fermat's spirals. *IFAC Proceedings Volumes*, 46(33):132–137.
- Cao, J., Li, Y., Zhao, S., and Bi, X. (2016). Genetic-algorithm-based global path planning for auv. In *2016 9th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 79–82.

- Caselli, S., Reggiani, M., and Rocchi, R. (2001). Heuristic methods for randomized path planning in potential fields. In *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515)*, pages 426–431.
- Castillo, O., Neyoy, H., Soria, J., Melin, P., and Valdez, F. (2015). A new approach for dynamic fuzzy logic parameter tuning in Ant Colony Optimization and its application in fuzzy control of a mobile robot. *Applied Soft Computing*, 28:150–159.
- CAUCHY, A. (1847). Methode generale pour la resolution des systemes d’equations simultanees. *C.R. Acad. Sci. Paris*, 25:536–538.
- Chauhan, V. K., Dahiya, K., and Sharma, A. (2018). Trust Region Levenberg-Marquardt Method for Linear SVM. In *2017 9th International Conference on Advances in Pattern Recognition, ICAPR 2017*, pages 380–385, India. IEEE.
- Chelouah, R. and Siarry, P. (2000). Tabu Search applied to global optimization. *European Journal of Operational Research*, 123(2):256–270.
- Chengqing, L., Ang, M., Krishnan, H., and Yong, L. S. (2000). Virtual obstacle concept for local-minimum-recovery in potential-field based navigation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 983–988 vol.2.
- Das, P. K. ; Jena, P. K. (2020). Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators. *Applied Soft Computing*, 92:106312.
- Dashkevich, A., Rosokha, S., and Vorontsova, D. (2020). Simulation tool for the drone trajectory planning based on genetic algorithm approach. In *2020 IEEE KhPI Week on Advanced Technology (KhPIWeek)*, pages 387–390.
- Devol, G. (1961). Dynastat memory aids unimate robot. *Electrical Engineering*, 80(4):319–320.
- Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., and Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137:106040.
- Evans, J., Krishnamurthy, B., Pong, W., Croston, R., Weiman, C., and Engelberger, G. (1989). HelpMate™: A robotic materials transport system. *Robotics and Autonomous Systems*, 5(3):251–256.

- Gao, M., Ding, P., and Yang, Y. (2015). Time-optimal trajectory planning of industrial robots based on particle swarm optimization. In *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, pages 1934–1939.
- García, D. A., Bravo, F., Cuesta, F., and Ollero, A. (2010). Planificación de Trayectorias con el Algoritmo RRT. Aplicación a Robots No Holónomos. *Revista iberoamericana de automatica e informatica industrial (RIAI)*, ISSN 1697-7912, Vol. 3, N^o. 3, 2006, pags. 56-67, 3.
- Ha, Q. P., Yen, L., and Balaguer, C. (2019). Robotic autonomous systems for earthmoving in military applications. *Automation in Construction*, 107:102934.
- Hacene, N. and Mendil, B. (2013). Autonomous Navigation and Obstacle Avoidance for a Wheeled Mobile Robots: A Hybrid Approach. *International Journal of Computer Applications*, 81:34–37.
- Hwang, Y. K. and Ahuja, N. (1992). Gross motion planning a survey. *ACM Computing Surveys*, 24(3):219–291.
- Jianguo, W., Biao, D., Guijuan, M., Jianwu, B., and Xuedong, Y. (2012). Path Planning of Mobile Robot Based on Improving Genetic Algorithm. In Jiang, L., editor, *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19–20, 2011, Melbourne, Australia*, Advances in Intelligent and Soft Computing, pages 535–542, Berlin, Heidelberg. Springer.
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.
- Kennedy, J. and Spears, W. (1998). Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 78–83.
- Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, China.
- Khatib, O. (1986). The Potential Field Approach And Operational Space Formulation In Robot Control. In Narendra, K. S., editor, *Adaptive and Learning Systems: Theory and Applications*, pages 367–377. Springer US, Boston, MA.

- LaValle, S. M. (2006). *Frontmatter*, pages i–vi. Cambridge University Press.
- Lee, L.-f. (2004). *Decentralized Motion Planning Within an Artificial Potential Framework (APF) for Cooperative Payload Transport by Multi-Robot Collectives*. PhD thesis, State University of New York.
- Lee, M. C. and Park, M. G. (2003). Artificial potential field based path planning for mobile robots using a virtual obstacle concept. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, volume 2, pages 735–740 vol.2.
- Li, H. (2020). Robotic path planning strategy based on improved artificial potential field. In *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, pages 67–71.
- Lin, X., Wang, Z.-Q., and Chen, X.-Y. (2020). Path planning with improved artificial potential field method based on decision tree. In *2020 27th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, pages 1–5.
- Lingelbach, F. (2004). Path planning for mobile manipulation using probabilistic cell decomposition. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2807–2812 vol.3.
- Lodhi, V., Chakravarty, D., and Mitra, P. (2018). A study of pso and its variants for fractional abundance estimation in hyperspectral data. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 197–201.
- Lozano-Perez (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120.
- Mahapatra, S., Badi, M., and Raj, S. (2019). Implementation of pso, it’s variants and hybrid gwo-pso for improving reactive power planning. In *2019 Global Conference for Advancement in Technology (GCAT)*, pages 1–6.
- Marquardt, D. W. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441. Publisher: Society for Industrial and Applied Mathematics.

- Masehian, E. and Sedighzadeh, D. (2007). Classic and Heuristic Approaches in Robot Motion Planning - A Chronological Review. *World Academy of Science, Engineering and Technology*, 29.
- Mitchell, J. S. B. (1988). An algorithmic approach to some problems in terrain navigation. *Artificial Intelligence*, 37(1):171–201.
- Paterega, I. (2011). Artificial evolution mechanisms in robot navigation. In *2011 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, pages 281–286.
- Patle, B. K., Babu L, G., Pandey, A., Parhi, D. R. K., and Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4):582–606.
- Price, K. V. and Price, K. V. (2015). Differential evolution. *Natural Computing Series*, 28:83–93.
- Pugh, J. and Martinoli, A. (2006). Multi-robot learning with particle swarm optimization. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, AAMAS '06*, pages 441–448, New York, NY, USA. Association for Computing Machinery.
- Qin, Y.-Q., Sun, D.-B., Li, N., and Cen, Y.-G. (2004). Path planning for mobile robot using the particle swarm optimization with mutation operator. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 4, pages 2473–2478 vol.4.
- Qingzhen, Z., Cunjia, L., Bo, Y., and Zhang, R. (2007). Reentry trajectory planning optimization based on ant colony algorithm. In *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1064–1068.
- Rashedi, E., Nezamabadi-pour, H., and Saryazdi, S. (2009a). GSA: A Gravitational Search Algorithm. *Information Sciences*, 179(13):2232–2248.
- Rashedi, E., Nezamabadi-pour, H., and Saryazdi, S. (2009b). Gsa: A gravitational search algorithm. *Information Sciences*, 179(13):2232–2248. Special Section on High Order Fuzzy Sets.
- Ratnaweera, A., Halgamuge, S., and Watson, H. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3):240–255.

- Rogoff, M. (1988). Electronic charts-at the heart of the 21st century navigation. In *IEEE PLANS '88., Position Location and Navigation Symposium, Record. 'Navigation into the 21st Century'*, pages 88–94.
- Rosa, E. A. d. l. S. d. l. (2004). Heuristica para la generacion de configuraciones en pasajes estrechos aplicada al problema de los clavos.
- Santiago, R. M. C., De Ocampo, A. L., Ubando, A. T., Bandala, A. A., and Dadios, E. P. (2017). Path planning for mobile robots using genetic algorithm and probabilistic roadmap. In *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pages 1–5.
- Sarkar, R., Barman, D., and Chowdhury, N. (2020). Domain knowledge based genetic algorithms for mobile robot path planning having single and multiple targets. *Journal of King Saud University - Computer and Information Sciences*.
- Seyyedhasani, H., Peng, C., Jang, W.-j., and Vougioukas, S. G. (2020). Collaboration of human pickers and crop-transporting robots during harvesting – Part I: Model and simulator development. *Computers and Electronics in Agriculture*, 172:105324.
- Sharir, M. (1989). Algorithmic motion planning in robotics. *Computer*, 22(3):9–19.
- Shibata, M. and Ohnishi, K. (1992). Collision detection and distance calculation for safe path planning based on mathematical programming methods. In *Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation*, pages 832–837 vol.2.
- Siegwart, R., Nourbakhsh, I., and Scaramuzza, D. (2004). *Introduction to Autonomous Mobile Robots*. Bradford Books.
- Stacey, A., Jancic, M., and Grundy, I. (2003). Particle swarm optimization with mutation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 2, pages 1425–1430 Vol.2.
- Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.
- Sugihara, K. and Smith, J. (1997). Genetic algorithms for adaptive motion planning of an autonomous mobile robot. In *Proceedings 1997 IEEE International Symposium*

- on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 138–143.
- Varadarajan, M. and Swarup, K. S. (2008). Differential evolutionary algorithm for optimal reactive power dispatch. *International Journal of Electrical Power and Energy Systems*, 30(8):435–441.
- Wahab, M. N. A., Nefti-Meziani, S., and Atyabi, A. (2020). A comparative review on mobile robot path planning: Classical or meta-heuristic methods. *Annual Reviews in Control*, 50:233–252.
- Weigl, M., Siemiątkowska, B., Sikorski, K. A., and Borkowski, A. (1993). Grid-based mapping for autonomous mobile robot. *Robotics and Autonomous Systems*, 11(1):13–21.
- Wikipedia (2020). Test functions for optimization.
- Yang, X. and Suash Deb (2009). Cuckoo Search via Lévy flights. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 210–214.
- Zeqing, Y., Libing, L., Zhihong, T., and Weiling, L. (2008). Application of adaptive genetic algorithm in flexible inspection path planning. In *2008 27th Chinese Control Conference*, pages 75–80.
- Zhu, H., Wang, J., and Li, J. (2013). A novel potential field method for path planning of mobile robot. In *2013 25th Chinese Control and Decision Conference (CCDC)*, pages 2811–2814.
- Zou, A.-M., Hou, Z.-G., Fu, S.-Y., and Tan, M. (2006). Neural Networks for Mobile Robot Navigation: A Survey. In Wang, J., Yi, Z., Zurada, J. M., Lu, B.-L., and Yin, H., editors, *Advances in Neural Networks - ISNN 2006*, Lecture Notes in Computer Science, pages 1218–1226, Berlin, Heidelberg. Springer.