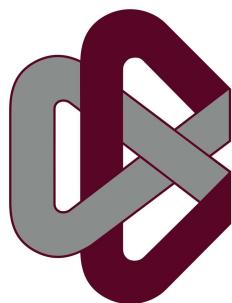


ÁNGEL ARTURO ROJAS GARCÍA

**TOPOLOGÍAS DE OPTIMIZACIÓN POR
ENJAMBRE DE PARTÍCULAS BASADAS
EN INFORMACIÓN MUTUA**



CIMAT

Centro de Investigación en Matemáticas A.C.
Departamento de Ciencias de la Computación

Tesis:

**TOPOLOGÍAS DE OPTIMIZACIÓN POR
ENJAMBRE DE PARTÍCULAS BASADAS
EN INFORMACIÓN MUTUA**

**Que para obtener el grado de:
MAESTRO EN CIENCIAS CON ESPECIALIDAD EN
COMPUTACIÓN Y MATEMÁTICAS INDUSTRIALES**

**Presenta:
ÁNGEL ARTURO ROJAS GARCÍA**

**Director de tesis:
DR. ARTURO HERNÁNDEZ AGUIRRE**

Guanajuato, Guanajuato a 9 de abril del 2015

Yo he preferido hablar de cosas imposibles,
porque de lo posible se sabe demasiado.

Silvio Rodríguez

Resumen

Optimización por enjambre de partículas (PSO, por sus siglas en inglés) es una algoritmo de optimización estocástica perteneciente a los algoritmos de inteligencia de enjambre. La principal característica de estos algoritmos es la comunicación que existe entre los miembros del enjambre (partículas), con el objetivo de resolver un problema en común, en este caso un problema de optimización. Esto se logra organizando al enjambre en una topología que le indica a cada partícula con quién tendrá comunicación directa durante la evolución del algoritmo.

Estudios han demostrado la influencia de la conectividad de las topologías en el desempeño de PSO, así como la importancia de cambiarlas con el paso de las generaciones. Pese a esto, en la actualidad la mayoría de las implementaciones de este se hacen con topologías que son definidas desde el inicio, sin tomar en cuenta información que las partículas pudieran ofrecer y, además, manteniéndolas estáticas con el paso de las generaciones del algoritmo. Tomando como base los estudios mencionados, se han propuesto distintas formas de crear topologías dinámicas de las partículas, las cuales van desde ideas simples como el incremento de la conectividad de las partículas y vecindarios aleatorios, hasta conceptos más complejos, como clustering para encontrar partículas cercanas y triangulación de Delaunay.

Debido la importancia de tener vecindarios dinámicos que utilicen la información que puedan proporcionar las partículas, en la presente tesis se propone una topología dinámica que conecta a las partículas con base en su información mutua, considerando que haciéndolo de esta forma se encontrarán dependencias entre estas. Para lograr esto, se utilizarán dos modelos gráficos que maximizan la información mutua, en este caso, entre las partículas del enjambre: la cadena MIMIC y el árbol Chow-Liu. Resultados obtenidos en varias funciones de prueba demuestran una convergencia más rápida al óptimo global (menor número de evaluaciones de función), la cual es más notable en funciones de alta dimensionalidad.

Agradecimientos

Quiero agradecer a la vida por poner a las personas indicadas en mi camino, porqué cada una de ellas ha sido parte fundamental para la obtención de cada una de mis metas y logros.

Primeramente, quiero agradecer a Ángela por ser el amor de mi vida y por apoyarme en cada uno de mis proyectos, pero sobre todo, por traer a nuestras vidas a Angelito, que ha llenado de alegrías la vida de los dos.

No puedo dejar de lado a mi familia en Veracruz, mi hermana Mayra, mi papá Arturo, mi abuelita Victoria, mi sobrino Yotzer y especialmente a mi mamá Benita, por hacer todo lo que estuvo a su alcance para darnos a mi hermana y a mí la posibilidad de estudiar y ser las personas que hoy somos. Quiero agradecerles porqué aun en la lejanía han sabido apoyarme y amarme.

A mi asesor, el Dr. Arturo Hernández Aguirre, por su apoyo y comprensión durante mi estancia en la maestría y en la realización de esta tesis, así como por sus conocimientos transmitidos.

A mis sinodales Ivvan Valdez y Carlos Segura, por el tiempo brindado para realizar las revisiones y observaciones de la presente tesis.

A cada uno de los investigadores del Departamento de Ciencias de la Computación, por ser una fuente enorme de conocimientos e inspiración, ya que tanto dentro de las aulas y fuera de estas aprendí mucho de cada uno ellos.

A la comunidad CIMAT-DEMAT por el ambiente agradable que se siente en sus edificios. A Lolita, Janet y Lalo; por apoyarme y guiarme en cada trámite que me fue necesario realizar.

Quiero agradecer a cada uno de mis compañeros de generación por hacer más amena mi estancia en el CIMAT, por el apoyo y conocimiento que com-

partimos. A Marcela, Cristóbal y Rafa por cada momento agradable dentro y fuera de los salones y cubículos del CIMAT. A Mario por apoyarme en los momentos más difíciles de mi estancia en Guanajuato. De igual forma, a cada compañero de otras generaciones y otros programas del CIMAT que me brindaron su amistad.

Quiero agradecer especialmente a Nacho, por su ayuda en las muchas dudas que me surgían durante el desarrollo de esta tesis, así como a Iván por su amistad y apoyo brindado en una de mis estancias en Guanajuato.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) y al Centro de Investigación en Matemáticas A.C. (CIMAT) por el apoyo económico aportado para la realización de estos estudios de posgrado.

Finalmente, quiero agradecer a los familiares y amigos que no se mencionan en estas líneas, pero que han sido una parte importante en distintos momentos de mi vida.

Índice

1	Introducción	1
2	Optimización y computación evolutiva	5
2.1	Optimización	5
2.1.1	Conceptos básicos	5
2.2	Algoritmos de optimización	9
2.2.1	Conceptos básicos	9
2.2.2	Clasificación	14
2.3	Computación evolutiva	16
2.3.1	Breve historia	17
2.3.2	Algoritmos evolutivos	19
2.3.3	Inteligencia de enjambre	26
3	Optimización por enjambre de partículas (PSO)	28
3.1	Introducción	28
3.2	Versión canónica	29
3.3	Parámetros	32
3.3.1	Coeficientes de aceleración	33
3.3.2	Factor de inercia	33
3.3.3	Velocidad máxima	35
3.3.4	Factor de restricción	36
3.3.5	Tamaño del enjambre	37
3.4	Topologías	37
3.4.1	Topologías estáticas	38
3.4.2	Topologías dinámicas	41
4	Información mutua y modelos gráficos	52
4.1	Información mutua	52
4.1.1	Variables aleatorias discretas	53
4.1.2	Variables aleatorias continuas	54
4.2	Modelos gráficos	59

4.2.1	Conceptos básicos	59
4.2.2	Modelo de cadena (MIMIC)	64
4.2.3	Modelo de árbol (Chow-Liu)	65
5	Construcción de la topología utilizando información mutua	67
5.1	Justificación	67
5.2	Descripción del algoritmo	68
5.2.1	Una topología distinta para cada dimensión	69
5.2.2	VARIABLES con dominio continuo	71
5.2.3	Parámetros	72
5.2.4	Modelos gráficos utilizados	73
5.3	Algoritmo propuesto	74
5.3.1	Complejidad del algoritmo	77
6	Experimentos y resultados	79
6.1	Funciones de prueba	79
6.2	Parámetros	81
6.2.1	Factor de inercia	81
6.2.2	Topología inicial	83
6.2.3	Frecuencia de actualización	85
6.2.4	Historial de las partículas	87
6.2.5	Modelos gráficos utilizados	88
6.3	Resultados obtenidos en las funciones de prueba	89
6.4	Comparación de algoritmos	103
6.5	Impacto del uso de la información mutua	109
7	Conclusiones y trabajo a futuro	112
Bibliografía		114
Anexos		124
A	Resultados obtenidos actualizando las topologías cada 50 generaciones con Chow-Liu	124
B	Resultados obtenidos construyendo las topologías con MIMIC	
	128	
C	Comparación de los resultados obtenidos con distintas frecuencia de actualización y formas de construir las topologías	135

D Otros experimentos	140
D.1 Evolución de los árboles construidos y movimiento de las partículas 140	
D.2 Evolución de los árboles construidos y movimiento de las partículas por pares conectados	147
D.3 Conectividad de los árboles generados	170
D.4 Información mutua de los vecindarios	170
D.5 Evolución de las distribuciones	178
D.6 Información mutua utilizando MIMIC	193

Índice de figuras

2.1	Representación gráfica de un problema de minimización global	8
2.2	Gráfica de la función Griewank en 2 dimensiones, un ejemplo de una función multimodal	13
2.3	Taxonomía de los algoritmos de optimización global	15
2.4	Línea del tiempo de algunos algoritmos de la computación evolutiva	19
2.5	Ciclo básico de los algoritmos evolutivos	22
2.6	La familia de los algoritmos evolutivos	26
3.1	Evolución a través del tiempo de la versión canónica de PSO	31
3.2	Influencias de la mejor experiencia personal y la mejor expe- riencia social sobre una partícula	34
3.3	Distintos tipos de topología utilizadas en PSO	39
3.4	Construcción de la topología en Hierarchical D-LPSO	42
3.5	Construcción de la topología en PSO-ITC	46
4.1	Ejemplo de un grafo dirigido y de un grafo no dirigido	60
4.2	Ejemplo de un grafo que contiene un lazo	61
4.3	Ejemplo de un grafo conectado y de un grafo no conectado	61
4.4	Ejemplo de un árbol	62
4.5	Ejemplo de un árbol de expansión	62
4.6	Ejemplo de un grafo completo	63
4.7	Ejemplo de una cadena	64
4.8	Ejemplo de una matriz de adyacencia	64
5.1	Distribución de las partículas en el espacio de búsqueda de la función esfera	70
5.2	Construcción de las topologías en PSO-MI	75

6.1	Evolución de la media de la aptitud y del mejor elemento del enjambre en la función Esfera en dimensión 80 con distintos tiempos de actualización de la topología	86
6.2	Comparación del número de evaluaciones de función requeridas por cada algoritmo para alcanzar el óptimo (parte 1) . . .	104
6.3	Comparación del número de evaluaciones de función requeridas por cada algoritmo para alcanzar el óptimo (parte 2) . . .	105
6.4	Comparación del porcentaje de éxito de cada algoritmo . . .	106
6.5	Comparación de la media de la función de aptitud alcanzada por cada algoritmo	107
6.6	Evolución de la media de la población y del mejor individuo encontrado para los algoritmos comparados durante las primeras 2000 generaciones para la función Esfera en dimensión 100 . .	108
C.1	Comparación del número de evaluaciones de función requeridas por cada combinación algoritmo-frecuencia de actualización para alcanzar el óptimo (parte 1)	136
C.2	Comparación del número de evaluaciones de función requeridas por cada combinación algoritmo-frecuencia de actualización para alcanzar el óptimo (parte 2)	137
C.3	Comparación de la media de la función de aptitud alcanzada por cada combinación algoritmo-frecuencia de actualización .	138
C.4	Comparación del porcentaje de éxito de cada combinación algoritmo-frecuencia de actualización	139
D.1	Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 1, parte 1 . .	141
D.2	Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 1, parte 2 . .	142
D.3	Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 2, parte 1 . .	143
D.4	Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 2, parte 2 . .	144
D.5	Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 3, parte 1 . .	145
D.6	Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 3, parte 2 . .	146
D.7	Movimiento de las partículas antes de la construcción del primer árbol, separadas por pares de partículas conectadas	148
D.8	Primer árbol de partículas construido	149

D.9	Movimiento de las partículas antes de la construcción del segundo árbol, separadas por pares de partículas conectadas	150
D.10	Segundo árbol de partículas construido	151
D.11	Movimiento de las partículas antes de la construcción del tercer árbol, separadas por pares de partículas conectadas	152
D.12	Tercer árbol de partículas construido	153
D.13	Movimiento de las partículas antes de la construcción del cuarto árbol, separadas por pares de partículas conectadas	154
D.14	Cuarto árbol de partículas construido	155
D.15	Movimiento de las partículas antes de la construcción del quinto árbol, separadas por pares de partículas conectadas	156
D.16	Quinto árbol de partículas construido	157
D.17	Movimiento de las partículas antes de la construcción del sexto árbol, separadas por pares de partículas conectadas	158
D.18	Sexto árbol de partículas construido	159
D.19	Movimiento de las partículas antes de la construcción del séptimo árbol, separadas por pares de partículas conectadas	160
D.20	Séptimo árbol de partículas construido	161
D.21	Movimiento de las partículas antes de la construcción del octavo árbol, separadas por pares de partículas conectadas	162
D.22	Octavo árbol de partículas construido	163
D.23	Movimiento de las partículas antes de la construcción del noveno árbol, separadas por pares de partículas conectadas	164
D.24	Noveno árbol de partículas construido	165
D.25	Movimiento de las partículas antes de la construcción del décimo árbol, separadas por pares de partículas conectadas	166
D.26	Décimo árbol de partículas construido	167
D.27	Movimiento de las partículas antes de la construcción del undécimo árbol, separadas por pares de partículas conectadas	168
D.28	Undécimo árbol de partículas construido	169
D.29	Máxima conectividad de los vecindarios construidos con Chow-Liu	
	172	
D.30	Información mutua de los vecindarios construidos con Chow-Liu.	
	Factor de inercia $\omega = 0.578766$	173
D.31	Información mutua de los vecindarios construidos con Chow-Liu.	
	Factor de inercia $\omega = 0.7298$	174
D.32	Información mutua de la topología todos conectados. Factor de inercia $\omega = 0.578766$	175
D.33	Información mutua de la topología todos conectados. Factor de inercia $\omega = 0.7298$	176

D.34 Información mutua de la topología todos conectados. Factor de inercia $\omega = 0.8$	177
D.35 Evolución de los árboles al resolver la función esfera en dimensión 30 con 10 partículas (parte 1)	179
D.36 Evolución de los árboles al resolver la función esfera en dimensión 30 con 10 partículas (parte 2)	180
D.37 Distribución de las partículas en el espacio de búsqueda de la generación 1 a la 100	181
D.38 Distribución de las partículas en el espacio de búsqueda de la generación 101 a la 200	182
D.39 Distribución de las partículas en el espacio de búsqueda de la generación 201 a la 300	183
D.40 Distribución de las partículas en el espacio de búsqueda de la generación 301 a la 400	184
D.41 Distribución de las partículas en el espacio de búsqueda de la generación 401 a la 500	185
D.42 Distribución de las partículas en el espacio de búsqueda de la generación 501 a la 600	186
D.43 Distribución de las partículas en el espacio de búsqueda de la generación 601 a la 700	187
D.44 Distribución de las partículas en el espacio de búsqueda de la generación 701 a la 800	188
D.45 Distribución de las partículas en el espacio de búsqueda de la generación 801 a la 900	189
D.46 Distribución de las partículas en el espacio de búsqueda de la generación 901 a la 1000	190
D.47 Distribución de las partículas en el espacio de búsqueda de la generación 1001 a la 1100	191
D.48 Distribución de las partículas en el espacio de búsqueda de la generación 1101 a la 1200	192
D.49 Información mutua en la topología anillo y cadena MIMIC utilizando una frecuencia de actualización de 100 generaciones	194
D.50 Información mutua en la topología anillo y cadena MIMIC utilizando una frecuencia de actualización de 50 generaciones	195

Índice de tablas

6.1	Funciones de prueba utilizadas para evaluar el desempeño del algoritmo propuesto	80
6.2	Clasificación de las funciones de prueba con base en su modalidad	81
6.3	Resultados obtenidos al utilizar distintos factores de inercia	82
6.4	Resultados obtenidos al iniciar el algoritmo propuesto con topología todos conectados y con topología todos desconectados	83
6.5	Resultados obtenidos al utilizar las 100 posiciones previas y todas las posiciones visitadas por las partículas en las funciones Esfera y Rastrigin en dimensión 30	88
6.6	Resultados obtenidos en las funciones f_1, f_2, f_3, f_4, f_5 y f_6 en dimensión 30	91
6.7	Resultados obtenidos en las funciones $f_7, f_8, f_9, f_{10}, f_{11}$ y f_{12} en dimensión 30	92
6.8	Resultados obtenidos en las funciones $f_{13}, f_{14}, f_{15}, f_{16}$ y f_{17} en dimensión 30	93
6.9	Resultados obtenidos en las funciones f_1, f_2, f_3, f_4, f_5 y f_6 en dimensión 50	94
6.10	Resultados obtenidos en las funciones $f_7, f_8, f_9, f_{10}, f_{11}$ y f_{12} en dimensión 50	95
6.11	Resultados obtenidos en las funciones $f_{13}, f_{14}, f_{15}, f_{16}$ y f_{17} en dimensión 50	96
6.12	Resultados obtenidos en las funciones f_1, f_2, f_3, f_4, f_5 y f_6 en dimensión 80	97
6.13	Resultados obtenidos en las funciones $f_7, f_8, f_9, f_{10}, f_{11}$ y f_{12} en dimensión 80	98
6.14	Resultados obtenidos en las funciones $f_{13}, f_{14}, f_{15}, f_{16}$ y f_{17} en dimensión 80	99

6.15	Resultados obtenidos en las funciones f_1, f_2, f_3, f_4, f_5 y f_6 en dimensión 100	100
6.16	Resultados obtenidos en las funciones $f_7, f_8, f_9, f_{10}, f_{11}$ y f_{12} en dimensión 100	101
6.17	Resultados obtenidos en las funciones $f_{13}, f_{14}, f_{15}, f_{16}$ y f_{17} en dimensión 100	102
6.18	Resultados obtenidos con topologías de anillo, anillo aleatorio y anillo MIMIC en la función Esfera	110
6.19	Resultados obtenidos con topologías de anillo, anillo aleatorio y anillo MIMIC en la función Griewank	111
A.1	Resultados obtenidos en las funciones f_1, f_2, f_3, f_4, f_5 y f_6 actualizando las topologías cada 50 generaciones con Chow-Liu	125
A.2	Resultados obtenidos en las funciones $f_7, f_8, f_9, f_{10}, f_{11}$ y f_{12} actualizando las topologías cada 50 generaciones con Chow-Liu	126
A.3	Resultados obtenidos en las funciones $f_{13}, f_{14}, f_{15}, f_{16}$ y f_{17} actualizando las topologías cada 50 generaciones con Chow-Liu	127
B.1	Resultados obtenidos en las funciones f_1, f_2, f_3, f_4, f_5 y f_6 actualizando las topologías cada 50 generaciones con MIMIC	129
B.2	Resultados obtenidos en las funciones $f_7, f_8, f_9, f_{10}, f_{11}$ y f_{12} actualizando las topologías cada 50 generaciones con MIMIC	130
B.3	Resultados obtenidos en las funciones $f_{13}, f_{14}, f_{15}, f_{16}$ y f_{17} actualizando las topologías cada 50 generaciones con MIMIC	131
B.4	Resultados obtenidos en las funciones f_1, f_2, f_3, f_4, f_5 y f_6 actualizando las topologías cada 100 generaciones con MIMIC	132
B.5	Resultados obtenidos en las funciones $f_7, f_8, f_9, f_{10}, f_{11}$ y f_{12} actualizando las topologías cada 100 generaciones con MIMIC	133
B.6	Resultados obtenidos en las funciones $f_{13}, f_{14}, f_{15}, f_{16}$ y f_{17} actualizando las topologías cada 100 generaciones con MIMIC	134

Índice de algoritmos

1	Algoritmo iterativo general	14
2	Algoritmo evolutivo estándar	23
3	Versión canónica de PSO	32
4	Algoritmo Hierarchical D-LPSO	43
5	Módulo ITC. PSO-ITC.	45
6	Algoritmo para generar ejemplares. PSO-ITC.	47
7	Algoritmo de la estrategia de aprendizaje basada en elitismo (EBLS). PSO-ITC.	48
8	Generar ejemplar (Operador NS). PSO-ITC.	49
9	Operador NS. PSO-ITC.	50
10	Algoritmo PSO-ITC.	51
11	Algoritmo de Kruskal	63
12	Algoritmo MIMIC	65
13	Algoritmo Chow-Liu	66
14	Algoritmo PSO-MI	76

Capítulo 1

Introducción

Optimización por enjambre de partículas (Particle Swarm Optimization, PSO) es un algoritmo de optimización perteneciente a los algoritmos de inteligencia de enjambre. La característica principal de ese tipo de algoritmos es la interacción social que existe entre los integrantes del enjambre, aprovechando esta capacidad de “comunicación” para resolver algún problema en común. Inspirándose en la forma en la que buscan maíz algunas bandadas de aves y en un trabajo previo de Frank Heppner y Ulf Grenander [HG90]; Kennedy y Eberhart, en un intento de producir inteligencia computacional utilizando analogías simples de interacción social en vez de usar solamente habilidades cognitivas individuales, propusieron PSO en 1995 [KE95], un algoritmo que a la postre se convirtió en un poderoso y ampliamente utilizado método de optimización. PSO consiste en colocar un cierto número de “partículas” en el espacio de búsqueda de la función a optimizar. Cada partícula explorará el espacio de búsqueda basando su movimiento en la propia experiencia que ha obtenido de la función (influencia personal) y en la experiencia que han obtenido otras partículas (influencia social). Debido a que las partículas guían su movimiento en dirección de la mejor posición (personal y social) respecto a la función de aptitud, el algoritmo eventualmente convergerá, aunque no siempre a un mínimo global. Para lograr la interacción entre las partículas, la población se organiza en una estructura de comunicación llamada topología. Esta funciona como un tipo de “red social” en la que las partículas se conectan, generalmente, mediante aristas no dirigidas, lo cual implica que dos partículas conectadas entre sí son vecinas bidireccionalmente, esto es, si la partícula i es vecina de la partícula j , entonces la partícula j también es vecina de la partícula i . Esta estructura indica a las partículas cual es su vecindario y con que otras partículas compartir información.

Con esta información se puede deducir que el uso de una topología adecuada conllevará a un buen desempeño de PSO. Generalmente la asignación de la topología se realiza al inicio del algoritmo, sin ningún tipo de información sobre las partículas y sin poder determinar si esta ayudará a obtener

un desempeño adecuado del algoritmo. Intentos por construir topologías basadas en alguna característica en común de las partículas fueron ideadas con el paso del tiempo. Además se comenzaron a tomar en cuenta las ventajas que tenía el construir estructuras que cambiaran conforme el algoritmo avanzara, observando que el tener topologías dinámicas ayudaba a introducir diversidad en las partículas [MMWP05]. Algunos trabajos demostraron que la conectividad de las topologías está directamente relacionada con el comportamiento de exploración y explotación de las partículas [MN04], llegando a la conclusión de que topologías con poca conectividad favorecen la exploración, mientras que topologías con mucha conectividad favorecen la explotación [Ken99, KM02].

El tipo de trabajos mencionados en el párrafo anterior comenzaron a generar la inquietud de preguntarse si las topologías utilizadas para el algoritmo eran benéficas para el desempeño de este, y si el hecho de mantenerlas fijas durante todo el desarrollo de este era conveniente. Debido a esta inquietud comenzaron a surgir propuestas de topologías que cambiaban con el paso de las generaciones. A este tipo de topologías se les llamó topologías dinámicas. Uno de los primeros trabajos realizados en los que se utilizaban topologías dinámicas fue realizado por Suganthan en 1999, en donde iniciaba con una topología con muy poca conectividad y progresivamente agregaba aristas, para transitar de la etapa de exploración a la etapa de explotación con el paso del tiempo [Sug99]. En el 2000, Kennedy propone un algoritmo en el que se realizaba clustering para encontrar partículas cercanas, para posteriormente asignarles una topología todo conectado a cada cluster encontrado [Ken00]. En el 2004 Mohais propone crear topologías dinámicas aleatorias en las que en cada iteración se quita una arista a alguna partícula con más de un vecino para después agregar este nodo a alguna partícula que no esté completamente conectada [MWP04]. En el 2005, consciente de las deficiencias que llevaba el tener topologías estáticas, Liang propone un PSO con una topología multienjambre dinámica [LS05]. En el 2008, Lane propone un estudio en el que se explora el uso de vecindarios espacialmente significativos para PSO usando triangulación de Delaunay [LEG08]. En ese mismo año Hamdan propone un método en el que son usados varios vecindarios simultáneamente, específicamente estrella, anillo y Von Neumann , evaluando para cada partícula con cual de estos se obtiene una mejor función de aptitud y escogiendo esta como la topología de la partícula [Ham08]. En el 2009, en su Frankenstein PSO, Montes de Oca propone una topología en la que comienza con todas las partículas conectadas para poco a poco ir decreciendo la conectividad hasta finalizar con una topología de anillo [OSBD09]. También en el 2009, Cui propone un algoritmo en el que las partículas interactúan directamente con su vecino más cercano [CCC09]. En el 2011 Ghosh

y otros proponen un algoritmo en el que utilizan una topología de árbol que cambia en cada generación, intercambiando las posiciones del padre y el hijo en la estructura en caso de que la función de aptitud del hijo sea mejor que la del padre [GZDA11]. Por último Lim, en el 2014, propone una topología en la que se incrementa la conectividad linealmente con el paso de las generaciones [LI14].

Pese a los trabajos en los que se demuestra la importancia de la conectividad de los vecindarios de PSO y pese al buen desempeño que en general demuestran las topologías dinámicas, es muy común hoy en día encontrar el uso de topologías estáticas. Esto nos lleva a preguntarnos qué tan conveniente es seguir utilizando este tipo de topologías. Seguramente esto dependerá de la función que queramos optimizar, pero creemos que se puede obtener un mejor desempeño si se utilizan topologías dinámicas, sin importar el tipo de función. Entonces, la pregunta obvia ahora es cómo construir las topologías de tal forma que ayude a optimizar el problema en cuestión de una forma más eficiente. La respuesta a esta pregunta probablemente se encuentra en la información que nos puedan proporcionar las propias partículas, pero ahora debemos buscar que información de estas nos ayudará a cumplir con este propósito. Como se puede ver en los trabajos mencionados con anterioridad, se han ideado distintas formas de conectar las partículas utilizando, principalmente, su posición en el espacio de búsqueda y la conectividad de los vecindarios. Probablemente existe otro tipo de información que nos puede servir para conectar las partículas. En nuestro caso proponemos buscar dependencias entre las partículas y conectarlas con base en estas. Para cumplir con este objetivo utilizamos la información mutua entre las partículas. Pese a la importancia mencionada que tienen las topologías de la población de PSO, consideramos que no existen suficientes propuestas para la construcción dinámica de vecindarios de las partículas, en contraste con la cantidad de trabajos relacionados con otros parámetros del algoritmo. Debido a esto creemos pertinente el hecho de proponer una nueva forma de construir la topología de las partículas.

Otro tópico en el que consideramos que no hay suficientes trabajos es en el análisis de la influencia de las topologías en el desempeño de PSO. Entonces, el hecho de introducir topologías construidas basándose en la información mutua podría dar a pie a futuros análisis de estas tomando en cuenta esta y otras medidas de dependencia, para tratar de comprender y ampliar el conocimiento existente de los vecindarios de PSO y poder determinar qué topologías pueden ser las mejores.

En la presente tesis se propone una topología dinámica en la que se maximiza la suma de las informaciones mutuas entre las partículas, con la finalidad de obtener vecindarios que comuniquen partículas con alta dependencia entre

ellas. Para lograr esto, se crea un modelo gráfico utilizando la información de las posiciones que ha visitado cada partícula en cada dimensión cada determinado número de generaciones. Los modelos gráficos utilizados para la construcción de los vecindarios son la cadena MIMIC (acrónimo de Mutual Information Maximizing Input Clustering) y el árbol Chow-Liu.

Capítulo 2

Optimización y computación evolutiva

En el presente capítulo se presentan los principales conceptos de la optimización y de la computación evolutiva. Comenzamos presentando que es la optimización y sus conceptos básicos. Posteriormente se presenta que son los algoritmos de optimización, sus conceptos básicos y como se clasifican estos. Finalmente se aborda la computación evolutiva, de la que se presenta una breve historia y los algoritmos que abarca esta.

2.1 Optimización

Hablando en términos simples, **optimizar** significa escoger el mejor elemento de un conjunto de elementos posibles. Basándonos en esta acepción podemos darnos cuenta que utilizamos la optimización en muchos aspectos de nuestra vida cotidiana. Escoger la ruta más corta de nuestro trabajo a nuestra casa, acomodar la mayor cantidad de pertenencias posibles en la maleta antes de un viaje y buscar los productos más económicos en el supermercado son algunos ejemplos de esto. Pero la optimización no se limita a actividades de nuestra vida diaria, sino que también es muy importante en problemas más complejos en los que se requieren, de igual forma, procedimientos más complejos para resolverlos. A lo largo de los años se han ideado distintos procedimientos para resolver estos problemas, siendo las matemáticas la piedra angular en la que se cimientan estos métodos. Dicho esto, comenzaremos por fundamentar matemáticamente la optimización.

2.1.1 Conceptos básicos

Consideremos la función $f(x_1, x_2) = x_1^2 + x_2^2$, queremos encontrar los valores de x_1 y x_2 para el que esta toma el valor más pequeño. Es obvio que

en el punto $(0, 0)$ la función vale 0, el **mínimo**. Retomando la definición presentada al principio de esta sección, $(0, 0)$ es el mejor elemento de todos los posibles que puede tomar la función (su **dominio**) para cumplir con el objetivo de hacerla lo más pequeña posible, es decir, optimizamos la función. A un problema de este tipo se le conoce como un problema de **minimización**. De una forma similar al caso anterior, consideremos la función $f(x_1, x_2) = -x_1^2 - x_2^2$, de la cual queremos encontrar su valor más grande. Claramente podemos observar que en el mismo punto $(0, 0)$ la función toma su **máximo** valor posible, nuevamente 0, por lo que el punto $(0, 0)$ es el mejor elemento de todos, es decir, el **óptimo**. A un problema de este tipo se le conoce como un problema de **maximización**.

Los ejemplos mencionados en el párrafo anterior corresponden a problemas de **optimización sin restricciones**, en los que la función puede tomar cualquier valor en su dominio. Ahora supongamos que queremos encontrar el óptimo de las funciones, pero además queremos que $x_1 + x_2 \geq 2$. Se puede demostrar que para ambos casos el óptimo es el punto $(1, 1)$. Problemas de este tipo son llamados problemas de **optimización con restricciones**, en los que los elementos del dominio de la función son condicionados a cumplir con alguna o algunas restricciones.

Definición 2.1.1 [Spi65] *Un **n-espacio euclíadiano** \mathbb{R}^n es el conjunto de todas las n -tuplas (x^1, \dots, x^n) de números reales x^i . Un elemento de \mathbb{R}^n es conocido como un **punto** en \mathbb{R}^n .*

Definición 2.1.2 [Spi65] *Una **función de \mathbb{R}^n a \mathbb{R}** (también conocida como **función de n variables**) es una regla que asocia a cada punto en \mathbb{R}^n un valor en \mathbb{R} . El valor al que la función asocia el punto x se denota $f(x)$. La notación que se utiliza para representar una función de n variables es $f : \mathbb{R}^n \rightarrow \mathbb{R}$ y esto significa que $f(x) \in \mathbb{R}$ es definida por $x \in \mathbb{R}^n$.*

Un punto importante a considerar es el dominio de la función. Como se puede observar, los puntos que puede tomar esta al estar definida en un n -espacio euclíadiano son infinitos y la tarea de encontrar el mejor de estos es exhaustiva. Una forma de simplificar este trabajo es delimitando los valores que puede tomar la función a un subespacio de \mathbb{R}^n . A este subespacio se le conoce como el **espacio de búsqueda de la función** (\mathbb{S}). Ahora el dominio de esta es el espacio de búsqueda en vez de \mathbb{R}^n , por lo que es “más pequeño”. En términos prácticos, lo que se hace es agregar restricciones de desigualdad a cada elemento x^i de la n -tupla (x^1, \dots, x^n) y ahora se deben cumplir las restricciones $l_{inf} \leq x^i \leq l_{sup}$ para $i = 1, \dots, n$, en donde $l_{inf}, l_{sup} \in \mathbb{R}$. Vale la pena aclarar que, pese a que se agregan restricciones para delimitar el

espacio de búsqueda de la función, el problema sigue siendo un problema de optimización sin restricciones, ya que se puede asumir que estas no tienen efecto en la solución óptima y se pueden utilizar los métodos que se plantean para resolver este tipo de problemas.

En las definiciones 2.1.3, 2.1.4 y 2.1.5 se presentan los espacios sobre los que se define un problema de optimización.

Definición 2.1.3 [Wei08] *El espacio problema \mathbb{X} de un cierto problema de optimización es el conjunto que contiene todos los elementos x que pueden ser una solución al problema.*

Definición 2.1.4 [Wei08] *El espacio de búsqueda \mathbb{S} de un cierto problema de optimización es el conjunto de elementos s en los que las operaciones de búsqueda de un algoritmo de optimización trabaja.*

Definición 2.1.5 [Wei08] *El espacio objetivo \mathbb{O} es el espacio abarcado por los codominios de las funciones objetivo.*

Un ejemplo en el que se puede observar la diferencia entre el espacio problema y el espacio de búsqueda es el algoritmo genético. Debido a que en este algoritmo las operaciones de búsqueda (definición 2.2.1) operan con representaciones discretas de los números reales, es necesario hacer una transformación de los puntos con dominio continuo a representaciones en cadenas discretas de estos. Entonces, el conjunto de todos los puntos en el dominio de la función es el espacio problema, mientras que las representaciones en cadena de estas (sobre los que se realizan las operaciones de mutación, cruce, etc.) constituyen el espacio de búsqueda.

Debido a que sólo nos enfocaremos en minimizar una sola función de n variables (nuestra función objetivo), el espacio objetivo definido en 2.1.5 se reduce al codominio de esta función. También para este caso el espacio de búsqueda es el mismo que el espacio problema, ya que PSO trabaja con puntos en \mathbb{R}^n .

Una vez definidos estos espacios, en la definición 2.1.6 se presenta formalmente que es un problema de optimización.

Definición 2.1.6 [Wei08] *Un problema de optimización está definido por una tetratupla $(\mathbb{X}, F, \mathbb{S}, Op)$, especificando el espacio problema \mathbb{X} , la función objetivo F , el espacio de búsqueda \mathbb{S} y el conjunto de operaciones de búsqueda Op .*

En la definición 2.2.1 de la siguiente sección se aborda el conjunto de operaciones de búsqueda.

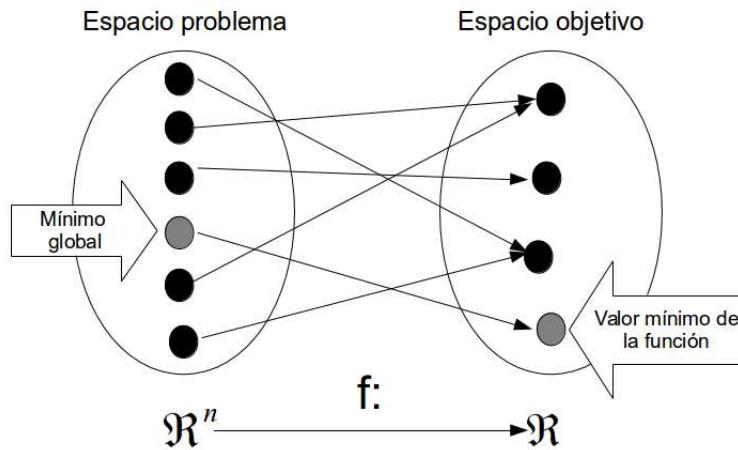


Figura 2.1: Representación gráfica de un problema de minimización global

En la presente tesis solamente se abordaran problemas de minimización sin restricciones para resolver funciones de n variables. En la definición 2.1.7 se define formalmente un problema de este tipo [NW99]. En la figura 2.1 se muestra gráficamente un problema de minimización global.

Definición 2.1.7 *Un problema de minimización sin restricciones se define matemáticamente como:*

$$\hat{x} = \arg \min_x f(x)$$

En donde $x \in \mathbb{R}^n$ es un vector real con $n \geq 1$ componentes, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ y \hat{x} es el mínimo global en \mathbb{S} .

Definición 2.1.8 [NW99] *Un punto \hat{x} es un mínimo global si $f(\hat{x}) \leq f(x) \forall x \in \mathbb{S}$.*

Definición 2.1.9 [NW99] *Un punto \check{x} es un mínimo local si para un vecindario \mathbb{N} de \check{x} se cumple que $f(\check{x}) \leq f(x) \forall x \in \mathbb{N}$.*

De la definición 2.1.7, se puede concluir que el objetivo de un algoritmo de optimización sin restricciones, en este caso de minimización, es encontrar la solución \hat{x} del problema (o al menos un punto bastante cercano a este). Es muy común que los algoritmos se estanquen en mínimos locales. En la siguiente sección se presenta qué es un algoritmo de optimización y cómo se clasifican estos.

2.2 Algoritmos de optimización

Como ya se mencionó en la sección anterior, un algoritmo de optimización es aquel que tiene por objetivo encontrar el óptimo, que para el caso de la presente tesis es el mínimo de una función de n variables. Es probable que en un buen curso de optimización se enseñen formas en las cuales se pueden resolver problemas de este tipo con el simple uso de papel y lápiz. Comúnmente los problemas que se resuelven son problemas simples, con una solución a la cual es fácil llegar y en dimensiones pequeñas, dicho de otra forma, problemas a modo. Pero es prácticamente un hecho que los problemas que se presentan en la vida real distan mucho de ser así y en lugar de esto podemos encontrarnos con problemas en donde es difícil llegar al óptimo global y en dimensiones muy grandes, en los que el uso de papel y lápiz ya no es tan conveniente y a la vez puede volverse tedioso. Es entonces que resolverlos computacionalmente se vuelve una solución factible y es aquí en donde los algoritmos de optimización toman vital importancia. A continuación se describen algunos conceptos básicos de los algoritmos de optimización.

2.2.1 Conceptos básicos

Como ya se ha mencionado, un algoritmo de optimización tiene la finalidad de encontrar el óptimo de un problema de optimización. Para lograr este fin este debe realizar una serie de pasos que nos aseguren que el algoritmo encontrará el óptimo, ya sea local o global. En los algoritmos de optimización iterativos se pretende encontrar una aproximación más cercana al óptimo en cada iteración (o al menos igual de cercana), por lo que se deben tener operaciones que “acerquen” al óptimo en cada iteración, esto es, que el mejor elemento que encuentre el algoritmo en cada repetición sea mejor (o al menos igual) que el que encontró en repeticiones anteriores. Para cumplir con este objetivo los algoritmos de optimización utilizan operaciones de búsqueda, las cuales se presentan en la definición 2.2.1.

Definición 2.2.1 [Wei08] *Las operaciones de búsqueda son todas aquellas operaciones que pueden ser usadas por un algoritmo de optimización con la finalidad de explorar el espacio de búsqueda \mathbb{S} .*

Cabe aclarar que pese a que el objetivo de las operaciones de búsqueda es encontrar un óptimo (y por ende, acercarnos a este), estas no siempre lograrán esto, ya que en alguna iteración podría ocurrir que el elemento encontrado no sea mejor que el de la iteración anterior (para algún individuo

en particular), pero eventualmente estas operaciones deben acercarnos al óptimo.

Supongamos que queremos pintar nuestra casa con un cierto tono de “azul cielo”, pero sólo tenemos pinturas de color “azul fuerte” y blanco. Nuestro problema de optimización es encontrar la cantidad de cada pintura que debemos combinar para obtener el tono que deseamos. El espacio de búsqueda son las distintas cantidades de pintura “azul fuerte” y blanco que se pueden utilizar por separado para realizar las mezclas, en tanto que el espacio objetivo son los distintos tonos de azul que se pueden obtener al mezclar estas cantidades de pintura. Es claro que el óptimo del problema son las cantidades de pintura que necesitamos mezclar para obtener el tono “azul cielo” que deseamos. Ahora bien, las operaciones de búsqueda son todas aquellas que nos permiten acercarnos al óptimo, entonces podemos decir que las operaciones de búsqueda que utilizaremos son agregar pintura “azul fuerte” y agregar pintura blanca. Probablemente esto se hará poco a poco (iterativamente) hasta obtener el tono deseado.

Definición 2.2.2 [Wei08] *El paisaje del problema* $\Phi : \mathbb{X} \times \mathbb{N} \mapsto [0, 1] \subset \mathbb{R}^+$ *mapea todos los puntos* x *en un espacio problema* \mathbb{X} *a la función de probabilidad acumulativa de alcanzar estos hasta (inclusivamente) la* τ -*ésima evaluación de una solución candidata. Esto es:*

$$\Phi(x, \tau) = P(x \text{ sea visitado hasta la } \tau - \text{ésima evaluación individual})$$

$$\forall x \in \mathbb{X}, \tau \in \mathbb{N}$$

En donde $\mathbb{X} \times \mathbb{N}$ es el producto cartesiano entre los puntos del espacio problema y el número de evaluaciones de soluciones candidatas. Esto es, si llamamos τ al número de evaluaciones de soluciones candidatas, entonces $\tau \in \{1, 2, \dots\}$.

Para tratar de ejemplificar el paisaje del problema utilizaremos algunos conceptos que se darán a conocer más adelante, estos conceptos son el de exploración y explotación de los algoritmos evolutivos. En términos generales, la fase de exploración del algoritmo consiste en tener puntos distribuidos “uniformemente” en el espacio de búsqueda del problema con la finalidad de encontrar potenciales óptimos. Una vez que el algoritmo detecta una región promisoria, en donde con alta probabilidad se encuentra un óptimo, este pasa a su fase de explotación, en la que los puntos se concentran en esta zona con el objetivo de acelerar la llegada al óptimo. Como se puede ver, en las primeras generaciones la probabilidad de seleccionar el óptimo es igual a la

probabilidad de seleccionar cualquier otro punto, pero conforme el algoritmo avanza a su fase de explotación los puntos se aglomeran en torno a este, por lo que crece la probabilidad de visitarlo. Es claro observar que la función a optimizar es parte fundamental del paisaje del problema, ya que con base en esta se seleccionarán los puntos a visitar. Debemos aclarar que algunos investigadores del área utilizan el término paisaje del problema para la función como tal, mientras que la definición utilizada en la presente tesis se refiere más a una distribución de probabilidad cuyo valor esperado se aproxima al óptimo global conforme las evaluaciones individuales aumentan.

Una vez dadas a conocer las definiciones 2.2.1 y 2.2.2 se puede dar una definición formal de un algoritmo de optimización, la cual se presenta en la definición 2.2.3.

Definición 2.2.3 [Wei08] *Un algoritmo de optimización es una transformación $(\mathbb{X}, F, \mathbb{S}, Op) \mapsto \Phi$ de un problema de optimización $(\mathbb{X}, F, \mathbb{S}, Op)$ a un paisaje del problema Φ .*

Los algoritmos de optimización tienen por objetivo encontrar el óptimo global, pero en muchas ocasiones esto no ocurre y se encuentran óptimos locales, lo cual en la mayoría de las ocasiones no es deseable. Es por esto que se han creado algoritmos que aumentan la probabilidad de obtener el óptimo global. Estos algoritmos son llamados *algoritmos de optimización global* los cuales se definen en la definición 2.2.4.

Definición 2.2.4 [Wei08] *Un algoritmo de optimización global es un algoritmo de optimización que emplea medidas que previenen la convergencia en óptimos locales e incrementa la probabilidad de encontrar un óptimo global.*

Se debe mencionar que es prácticamente imposible encontrar el valor exacto del óptimo, por lo que se opta por detener el algoritmo cuando se encuentra un valor aproximado de este. En algunas otras ocasiones puede ocurrir una convergencia muy lenta o en su defecto, ni siquiera se converge a alguna región cercana a este. En este caso se asigna una cierta cantidad de recursos computacionales, y cuando estos se exceden el algoritmo se detiene. Para este efecto, se cuenta con una función que tiene acceso a toda la información del proceso de optimización para determinar en qué momento se detendrá el algoritmo.

Definición 2.2.5 [Wei08] *El criterio de paro es una función que toma valores del conjunto $\{\text{true}, \text{false}\}$ y que cuando esta es evaluada con `true` el proceso de optimización se detendrá y regresará los resultados que ha obtenido.*

Algunos de los criterios de paro que pueden ser utilizados para decidir si detener o no un problema son:

- Cuando se considere que el algoritmo de optimización ha excedido el tiempo de cómputo.
- Cuando se ha rebasado el límite de iteraciones o evaluaciones individuales.
- Cuando no se encuentran soluciones mejores del óptimo en un determinado número de iteraciones.
- Cuando se considere que se ha obtenido una aproximación satisfactoria del óptimo.
- Una combinación de los criterios anteriores.

Definición 2.2.6 [Wei08] *Se dice que un algoritmo de optimización converge cuando obtiene soluciones de un “pequeño” subconjunto del espacio problema, posiblemente ignorando mejores soluciones en otras áreas.*

Un problema que se presenta en muchas ocasiones es que no sabemos si en la región a la que han convergido los individuos se encuentra un óptimo local o un óptimo global, por lo que no se puede determinar cuando detener el algoritmo. Si se observa que las partículas se encuentran en una región donde existe un óptimo local lo mejor es buscar en otras partes del espacio de búsqueda, tratando de encontrar mejores zonas de la función. Por el contrario, si hemos llegado a la región en la que se encuentra el óptimo global lo mejor es intensificar la búsqueda en esta zona, con el objetivo de obtener una mejor aproximación de este. El hecho de ignorar que tipo de óptimo hemos encontrado dificulta la elección de cual de estas estrategias seguir. Este problema es más notable en funciones multimodales. En la figura 2.2 se muestra un ejemplo de este tipo de funciones.

Definición 2.2.7 [Wei08] *Una función objetivo F es multimodal si tiene múltiples óptimos locales o globales.*

Vale la pena mencionar que la presencia de múltiples óptimos globales no es tan problemática como si lo puede ser la presencia de múltiples óptimos locales. Esto debido a que en el primer caso es suficiente con encontrar uno de los varios óptimos globales para considerar que se tuvo éxito. Pero en el segundo caso aumenta considerablemente la probabilidad de que el algoritmo obtenga como resultado un óptimo local, lo cual conlleva al problema de la convergencia prematura del algoritmo de optimización.

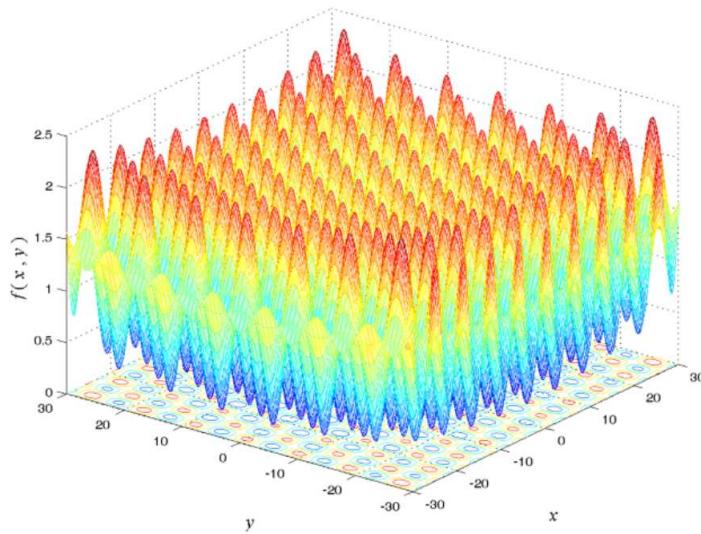


Figura 2.2: Gráfica de la función Griewank en 2 dimensiones, un ejemplo de una función multimodal

Definición 2.2.8 [Wei08] *Se dice que un algoritmo de optimización tiene convergencia prematura a un óptimo local cuando ya no le es posible explorar otras partes del espacio de búsqueda distintas a las que esta explotando y además existe otra región que contiene una mejor solución a la que actualmente explota.*

Por último, hacemos énfasis en el hecho de que muchos algoritmos de optimización son iterativos. Un algoritmo iterativo es aquel que realiza la mayor parte de su trabajo en una repetición cíclica de un ciclo principal. En la presente tesis, PSO siempre iniciará en $t = 0$, el valor $t \in \mathbb{N}_0$ se refiere a la iteración actual y $t + 1$ a la siguiente iteración.

Definición 2.2.9 [Wei08] *Una iteración es una repetición de una secuencia específica de instrucciones dentro de un algoritmo.*

En el algoritmo 1 se presenta un caso general de un algoritmo iterativo.

Algoritmo 1 Algoritmo iterativo general

Entrada: El criterio de paro, los datos necesarios para el funcionamiento del algoritmo y el contador de las iteraciones (t)

- 1: $t \leftarrow 0$
 - 2: Inicializar los datos del algoritmo
 - 3: **mientras** Criterio de paro == false **hacer**
 - 4: Realizar una iteración
 - 5: $t \leftarrow t + 1$
 - 6: **fin mientras**
-

2.2.2 Clasificación

Basándose en el método de operación de los algoritmos de optimización, existen dos grandes ramas en las que se pueden agrupar estos: los determinísticos y los estocásticos. En la presente sección se da a conocer de forma muy general en que consiste cada una de estas categorías y algunos ejemplos de cada una de ellas. En la figura 2.3 [Wei08] se muestra la taxonomía de los algoritmos de optimización global.

Algoritmos de optimización determinísticos

Un suceso determinista es aquel que da lugar a un resultado seguro. Podemos afirmar con certeza que, partiendo de las mismas condiciones iniciales que algún suceso anterior llegaremos al mismo resultado. Un algoritmo determinístico es aquel en el que una cierta solución inicial nos conducirá siempre a la misma solución final [Tal09]. Es obvio pensar que este tipo de algoritmos es bastante útil cuando se conoce el tipo de solución y se tiene una función a optimizar sencilla, algo que no es siempre conveniente cuando la función a optimizar es multimodal, por razones que se darán a conocer más adelante.

Algunos algoritmos de optimización determinística basan su búsqueda en seguir la dirección del gradiente. Esto se debe a que el hecho de moverse en esta dirección asegura un máximo incremento (o máximo descenso en la dirección contraria, para el caso de la minimización).

Definición 2.2.10 [Wei08] *Un gradiente de un campo escalar $f : \mathbb{R}^n \mapsto \mathbb{R}$ es un campo vectorial cuyos puntos están en la dirección del más grande incremento del campo escalar. El gradiente se denota por ∇f o $\text{grad}(f)$.*

Algoritmos de optimización estocásticos

Una vez descritos los algoritmos de optimización determinísticos veremos el otro lado de la moneda de la optimización global, los algoritmos estocásticos.

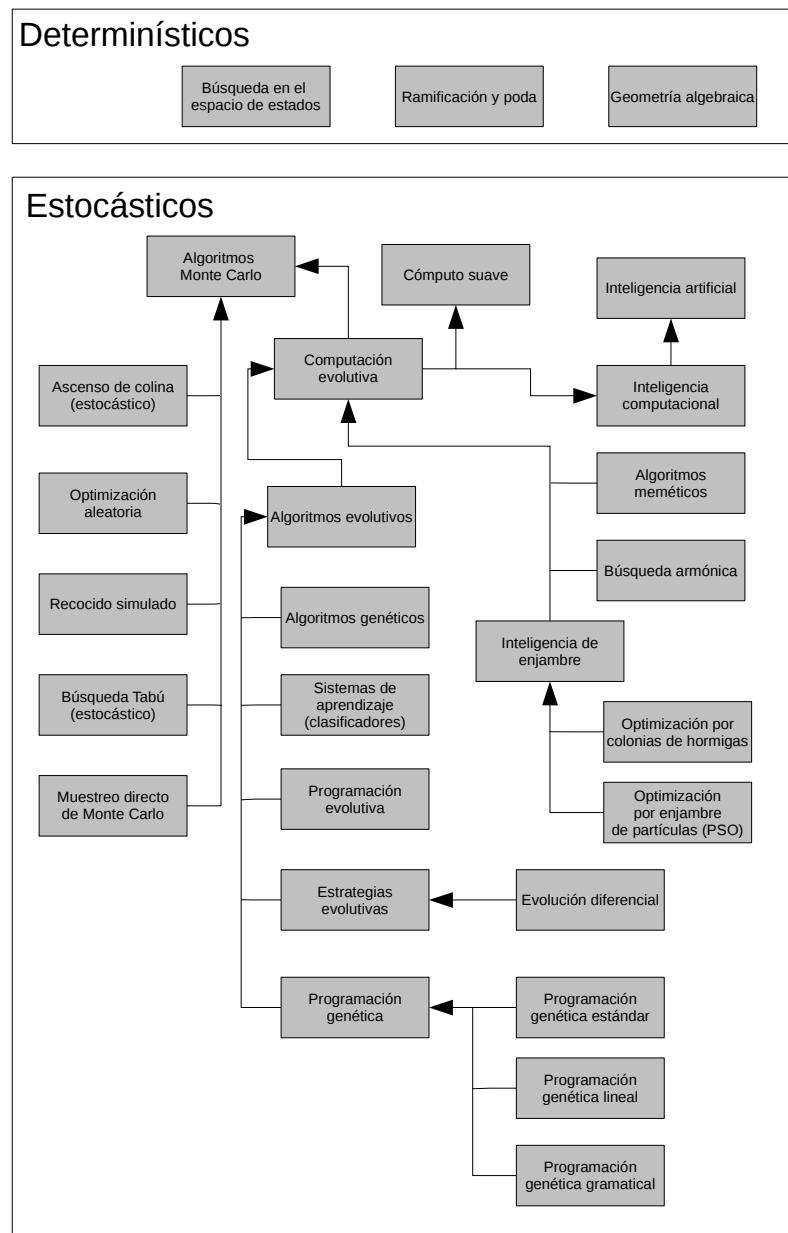


Figura 2.3: Taxonomía de los algoritmos de optimización global

Partiendo de la idea de que los algoritmos estocásticos son lo contrario a los determinísticos podemos decir que estos son aquellos en los que no sabemos a ciencia cierta como irá avanzando nuestro algoritmo, aunque si debemos estar seguros de que nos estamos acercando al óptimo. Si repetimos un experimento con condiciones similares a otro no podemos estar seguros del resultado que se obtendrá, de hecho, podemos afirmar que el resultado obtenido será diferente. En este tipo de algoritmos se crean iterativamente muestras de posibles soluciones en el espacio de búsqueda y entonces, es prácticamente imposible obtener la misma secuencia de muestras en dos experimentos distintos.

Supongamos la función Griewank mostrada en la figura 2.2. El mínimo global de la función se encuentra en el punto $(0, 0)$, pero tiene muchos mínimos locales. El mínimo al que llegará un algoritmo de optimización basado en gradiente en esta función dependerá de en donde se inicie la búsqueda, por lo que la probabilidad de encontrar el mínimo global es pequeña, ya que sólo se llegará a este si se inicia en un pequeño vecindario del punto $(0, 0)$. Si se inicia en un punto fuera de este vecindario se llegará al mínimo local más cercano. Esto no es conveniente ya que lo mejor es llegar al punto $(0, 0)$. Para resolver esta situación existen estrategias que consisten en crear una muestra de puntos desde los cuales iniciará el algoritmo con el objetivo de que desde alguno de ellos se alcance el mínimo global. Esta es de hecho una implementación estocástica de un algoritmo determinístico. En el ejemplo dado es preferible utilizar los algoritmos estocásticos, los cuales pese a la incertidumbre que pueden generar tienen un mejor desempeño que los algoritmos de optimización determinísticos. En general, para la elección del algoritmo a utilizar se deben tomar en cuenta las características del problema y la eficacia y eficiencia de este.

Hacemos mención que algunas metaheurísticas pueden ser estocásticas o determinísticas. Esto depende del diseño de estas. Por ejemplo, en el caso de la búsqueda Tabú, si los vecinos se consideran en orden aleatorio, entonces esta es estocástico (como se considera en la figura 2.3), en caso contrario es un algoritmo determinístico.

2.3 Computación evolutiva

Existe dentro de la naturaleza un principio muy importante para la supervivencia de las especies, la adaptación. Cada especie tiene que adaptarse a las complicaciones que le presenta su entorno para sobrevivir, si no lo hace estará destinada a desaparecer. El origen de esta capacidad está en la evolución y es producida por la selección natural, que favorece la perpetuación de los

individuos más adaptados a su entorno. La selección natural (o que se produce sin intervención del hombre) no es propiamente una selección. Nosotros podemos seleccionar para la reproducción los seres que más nos interesan, por ejemplo las vacas que producen más leche. En cambio, en la naturaleza no existe una inteligencia exterior que determine la dirección de la evolución, sin embargo la evolución se produce. Es entonces que las técnicas utilizadas por la naturaleza se presentan como procesos dignos a imitar, asumiendo que utilizando estos métodos y aplicándoles una selección guiada por una inteligencia exterior se lograrán mejores resultados. La selección natural es en sí un proceso de optimización. Como se mencionó con anterioridad, optimizar consiste en seleccionar a los mejores individuos. La naturaleza, al seleccionar a los individuos más aptos a su entorno, está realizando una optimización de las especies. Entonces podemos emular la selección natural para resolver problemas de optimización que nos sean necesarios. Un resultado evidente de la evolución es la capacidad de organización que presentan algunas especies. Es un hecho que existen especies que necesitan trabajar en conjunto para satisfacer ciertas necesidades y que de no hacerlo así, es decir haciéndolo individualmente, no sobrevivirían a su entorno. Las abejas son un ejemplo de esto ya que para buscar el néctar de las flores necesitan organizarse. Otro ejemplo de esto son pequeñas especies de peces que individualmente son vulnerables a ser devorados por especies más grandes, pero se agrupan en grandes cantidades para parecer una especie de mayor tamaño y así evitar esto. Desde mediados del siglo pasado se han ideado algoritmos de optimización que toman como base las ideas expuestas en este párrafo, y siguen apareciendo propuestas basadas en estas. El área de la computación que agrupa estos algoritmos es la computación evolutiva. Esta área interpreta a la naturaleza como una inmensa máquina de resolver problemas y trata de encontrar el origen de dicha potencialidad para utilizarla en la resolución de una gran variedad de problemas. En la presente sección se presentan los dos grandes grupos en los que se divide la computación evolutiva: los algoritmos evolutivos y la inteligencia de enjambre.

2.3.1 Breve historia

Si bien es cierto que no es sencillo dar una fecha exacta de los orígenes de la computación evolutiva, esta puede ser fechada a finales de la década de 1950. Uno de los primeros usos de un proceso evolutivo para resolver problemas computacionales es descrito en los trabajos de Friedberg [Fri58] [FDN59], en los que se realiza uno de los primeros trabajos en aprendizaje máquina y se describe el uso de un algoritmo evolutivo para encontrar un programa que calcula una cierta función de entrada - salida. Durante el mismo pe-

riodo, Box desarrolló la operación evolutiva (EVOP) en donde se utilizaba una técnica evolutiva para el diseño y análisis de experimentos industriales [Box57]. Posteriormente Bremermann presenta uno de los primeros intentos de aplicar evolución simulada a problemas de optimización numérica [Bre62]. A mediados de los 60's se establecieron las bases de los tres principales casos de los algoritmos evolutivos al presentarse las raíces de la programación evolutiva por Lawrence Fogel [Fog62] [Fog64], el desarrollo de los algoritmos genéticos por Holland en la Universidad de Michigan [Hol62] [Hol75] [HR77] y de las estrategias evolutivas desarrolladas por Rechenberg ([Rec73] y [Rec94]), Schwefel ([Sch75] y [Sch93]) y Bienert [Fog98]. Estos trabajos motivaron el aumento de publicaciones y conferencias en esta área de la computación.

Debieron pasar algunos años para que surgieran nuevas propuestas a las ya mencionadas con anterioridad. En 1986, J. Doyne Farmer y otros propusieron un algoritmo basado en el sistema inmune al que llamaron Artificial Immune System (AIS) [FPP86]. En 1992, John R. Koza propone la programación genética en la que se pretende desarrollar automáticamente programas de computadora que realicen una tarea definida por el usuario, así como optimizar funciones que toman individuos con forma de árbol [Koz92]. Posteriormente, en 1997, al intentar resolver el polinomio de Chebychev, Rainer Storn y Kenneth Price concibieron la idea de utilizar las diferencias entre vectores para perturbar el vector población, naciendo así la evolución diferencial [SP97].

En otro rubro de la computación evolutiva, en fechas más recientes se propusieron los algoritmos de la inteligencia de enjambre. Uno de los primeros algoritmos que se propuso fue Stochastic Diffusion Search (SDS) en 1989, cuando J. M. Bishop presenta un algoritmo basado en la idea de un grupo de personas que van a una ciudad desconocida por ellos y deben encontrar el mejor lugar para comer de entre los muchos restaurantes que hay en esa ciudad [Bis89]. En 1992, Marco Dorigo propone Ant Colony Optimization (ACO), inspirándose en el proceso que realizan las hormigas para buscar comida [Dor99]. En 1995 James Kennedy y Russel C. Eberhart proponen PSO, uno de los algoritmos de inteligencia de enjambre más utilizados, ideándose en la forma en la que se organizan socialmente las manadas de animales; como lo son las parvadas de aves, los bancos de peces y algunas otras especies [KE95]. En el 2002 Kevin M. Passino propuso Bacterial Foraging Optimization (BFO), un algoritmo basado en el comportamiento quimiotáctico de la bacteria *E. Coli* [Pas02]. Uno de los tipos de algoritmos con un mayor número de propuestas son los algoritmos basados en el comportamiento de las abejas, que en su mayoría se inspiran en la forma en la que estas buscan néctar en las flores. Dentro de este tipo de algoritmos se encuentran Bee

Colony Optimization (BCO) propuesto por P. Lucic y D. Teodorovic en el 2001 [LT], Artificial Bee Colony (ABC) de Dervis Karaboga [Kar05] y el algoritmo Bees de D. T. Pham y otros [PGK⁺06], propuestos en el 2005, Bee Nectar Search Optimization (BNSO) propuesto por Wilfredo Alfonso y otros en el 2007 [AMLC07] y Artificial Bee Hive Algorithm (ABHA) propuesto por Mario A. Muñoz y otros en el 2009 [MLC09].

En el 2001 Zong Woo Geem propone Harmony Search, un algoritmo inspirado en proceso de improvisación que tienen los músicos al componer música [GKL01]. Este algoritmo no esta clasificado dentro de los algoritmos evolutivos ni dentro de los algoritmos de inteligencia de enjambre, pero si dentro de la computación evolutiva. En la figura 2.4 se presenta una línea del tiempo de algunos algoritmos de optimización en la computación evolutiva.

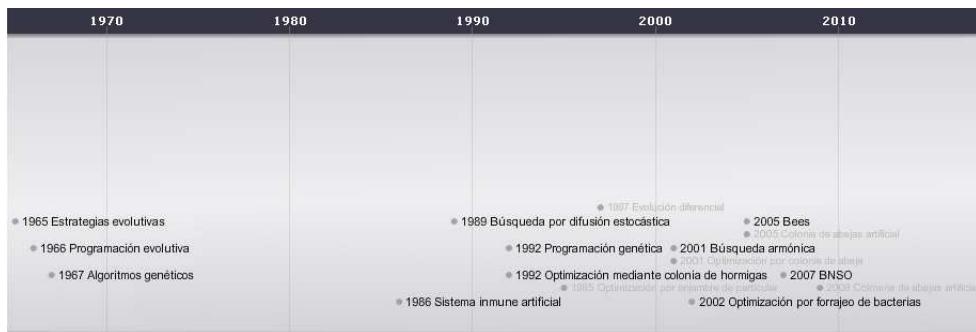


Figura 2.4: Línea del tiempo de algunos algoritmos de la computación evolutiva

2.3.2 Algoritmos evolutivos

Dentro de los algoritmos de optimización estocásticos se encuentran los algoritmos evolutivos. Estos algoritmos están inspirados en la teoría biológica de la evolución de las especies de Charles Darwin, en la que se propone que sólo sobrevivirán los individuos más aptos, generando individuos que se adaptan mejor a su entorno y optimizando las aptitudes de las especies. En el entorno de la optimización, esto se traduce como la selección de los mejores elementos (los más pequeños para el caso de la minimización) para ser parte de la siguiente iteración. En general, la computación evolutiva ha utilizado ideas que se basan en el comportamiento de la naturaleza para resolver problemas, como lo son la teoría de la evolución mencionada en los algoritmos evolutivos, la forma en la que se conectan las neuronas en el proceso cognitivo en las redes neuronales y el comportamiento social de algunos conjuntos de animales en la inteligencia de enjambres, por mencionar algunos ejemplos. En todos los

casos mencionados se tratan de emular las estrategias utilizadas por la naturaleza para resolver “sus problemas”. No podemos negar que la evolución da resultado a mejores individuos con el paso del tiempo, que conforme un ser humano crece aprende de sus experiencias y que las abejas recolectan miel de forma óptima gracias a su organización social. Con base en este tipo de resolución eficaz de problemas naturales, la computación evolutiva pretende resolver problemas complejos de optimización.

Conceptos básicos

En la presente sección se abordan los conceptos más importantes que se utilizan en los algoritmos evolutivos. Los algoritmos evolutivos combinan elementos estocásticos, adaptación y aprendizaje con la finalidad de tener esquemas inteligentes de optimización. Debido a que estos algoritmos son iterativos, estos esquemas se adaptan a su entorno (la función a optimizar, en nuestro caso) en cada iteración, utilizando la información recopilada en cada paso con tal de obtener mejores soluciones al problema de optimización. La parte del algoritmo encargada de “analizar” la información recopilada y que “decide” como actuar en la siguiente iteración son las heurísticas.

Definición 2.3.1 [Wei08] *Una **heurística** es una parte de un algoritmo de optimización que usa la información recopilada por el algoritmo y que ayuda a decidir cual de las soluciones candidatas debe ser examinada inmediatamente o como debe ser producido el siguiente individuo.*

Definición 2.3.2 [Wei08] *Una **metaheurística** es un método heurístico que resuelve una clase muy general de problemas. Esta combina funciones objetivo o heurísticas de una forma abstracta y eficiente.*

Una vez definidas las heurísticas y las metaheurísticas, en la definición 2.3.3 se presenta que son los algoritmos evolutivos.

Definición 2.3.3 [Wei08] *Los **algoritmos evolutivos** son algoritmos de optimización metaheurísticos y basados en población que están inspirados en mecanismos biológicos tales como la mutación, crusa, selección natural y supervivencia de los más aptos.*

Los algoritmos evolutivos tienen tres características principales:

- Basados en población.
- Orientados a la aptitud.

- Guiados por variaciones.

Como ya se mencionó, los algoritmos evolutivos son basados en población, esto es, trabajan con una muestra del espacio de búsqueda en la que el algoritmo realizará sus operaciones de búsqueda y que, paulatinamente, deberán hacer que esta población converja a un óptimo (generalmente). A cada elemento de la población se le conoce como individuo.

Definición 2.3.4 [Wei08] *Un **individuo** p es una tupla (p_s, p_x) de un elemento p_s en el espacio de búsqueda \mathbb{S} y su correspondiente elemento p_x en el espacio problema \mathbb{X} .*

Se aclara nuevamente que, para el caso de la presente tesis, el espacio de búsqueda es el mismo que el espacio problema, por lo que el individuo puede ser considerado como un solo elemento p en lugar de ser considerado una tupla (p_s, p_x) .

Definición 2.3.5 [Wei08] *La **población** es el conjunto de individuos utilizado durante un proceso de optimización.*

Ahora bien, para determinar cuales son los individuos más aptos debe existir una función que nos indique que tan útil son cada uno de estos como una posible solución. Esta función “calificará” al individuo con un valor basándose en criterios que sean útiles para el buen funcionamiento del algoritmo. Es claro observar que estos criterios deben ser seleccionados de tal forma que la función califique con un mejor valor a aquellos individuos que tengan una mejor adaptación al entorno del problema de optimización. Para el caso de la presente tesis, este valor de aptitud es dado directamente por la función en n variables que se desea optimizar.

Definición 2.3.6 [Wei08] *El **valor de aptitud** $v(x) \in \mathbb{V}$ de un elemento x del espacio problema \mathbb{X} corresponde a su utilidad como una solución o su prioridad en el siguiente paso del problema de optimización, en donde \mathbb{V} es el conjunto que contiene todos los posibles valores de aptitud que puede asignar la función de aptitud v .*

Como se puede ver en la definición 2.3.3 los algoritmos evolutivos están inspirados en mecanismos biológicos tales como la mutación, crusa, selección natural y supervivencia de los más aptos. Estos mecanismos son los que guían la búsqueda del óptimo. La forma en la que los algoritmos evolutivos funcionan de acuerdo con estos mecanismos se muestran en el siguiente esquema:

1. Se inicializa aleatoriamente una población de individuos p .
2. Los valores de las funciones objetivo son computadas para cada solución candidata en la población.
3. Con los valores de las funciones objetivo se determina la utilidad de las diferentes características de las soluciones candidatas y se le asigna un valor de aptitud $v(p)$ a cada una de ellas.
4. Se realiza un proceso de selección de las soluciones candidatas en las que los individuos con mejores aptitudes tengan una mayor probabilidad de “heredar” sus características a la siguiente generación.
5. Se realiza la fase de reproducción en la que se crea la descendencia utilizando las operaciones de reproducción. Esta descendencia se integra a la población.
6. Si se cumple el criterio de paro se detiene el proceso de optimización, si no se regresa al paso 2.

Un bosquejo de este esquema se ilustra en la figura 2.5.

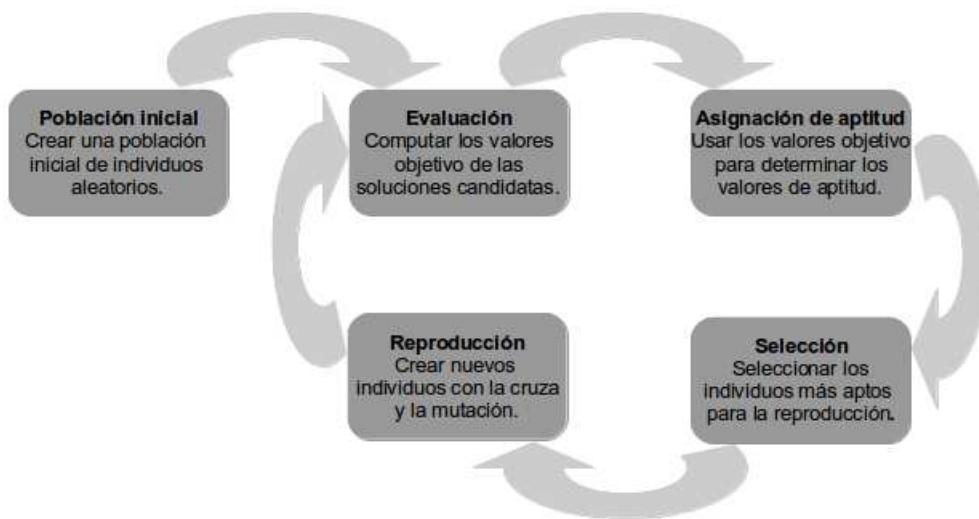


Figura 2.5: Ciclo básico de los algoritmos evolutivos

En el algoritmo 2 se muestra un algoritmo evolutivo estándar.

Todos estos pasos que siguen los algoritmos evolutivos deben asegurarnos que habrá convergencia al óptimo. Debido a que generalmente no se sabe en

Algoritmo 2 Algoritmo evolutivo estándar

Entrada: La función a optimizar y los parámetros del algoritmo.

Salida: El valor de la aproximación del óptimo encontrado.

- 1: Inicializar la población.
 - 2: Evaluar la población.
 - 3: **mientras** Criterio de paro == false **hacer**
 - 4: Aplicar selección.
 - 5: Aplicar crusa.
 - 6: Aplicar mutación.
 - 7: Evaluar los individuos generados.
 - 8: Actualizar la población.
 - 9: **fin mientras**
-

donde está el óptimo, primero el algoritmo debe tratar de “visitar” partes diversas de la función para encontrar zonas en las que se encuentren soluciones prometedoras. Una vez que se encontró una posible solución prometedora, el algoritmo debe intensificar la búsqueda en torno a esta para mejorar esta solución encontrada. Estas etapas del algoritmo evolutivo se conocen como exploración y explotación, respectivamente.

Definición 2.3.7 [Wei08] *La exploración, en términos de optimización, significa buscar nuevos puntos en áreas del espacio de búsqueda en las cuales aun no se ha investigado.*

Definición 2.3.8 [Wei08] *La explotación, en términos de optimización, significa intentar mejorar las soluciones conocidas hasta el momento realizando pequeños cambios que generarán nuevos individuos muy cercanos a estas.*

Como es de sospecharse, si un algoritmo pasa la mayor parte de las iteraciones en su fase de exploración tardará más tiempo en llegar a un óptimo o incluso no llegar a ninguno. Es conveniente usar la exploración con mayor frecuencia en funciones multimodales, ya que nos interesa explorar la mayoría de los óptimos para tratar de identificar cual tiene la mayor probabilidad de ser un óptimo global. Si se usa mayormente la fase de explotación en este tipo de funciones es muy probable que se tenga una convergencia prematura a un óptimo local. Por el contrario, si se utiliza más la fase de explotación, el algoritmo tendrá una convergencia más rápida, por lo que en problemas unimodales es mejor utilizar mayoritariamente esta fase, ya que no es necesario explorar partes diversas de la función, debido a que sólo necesitamos acercarnos al único óptimo. Como en la mayoría de los problemas no se conoce

a ciencia cierta como es la función a optimizar, se debe encontrar un balance adecuado entre la exploración y la explotación de tal forma que se lleguen a soluciones aceptables. Este balance se logra utilizando una combinación apropiada de los parámetros del algoritmo. Se debe aclarar que aunque una cierta combinación de parámetros funcione para un determinado problema, esto no implica que esta misma combinación funcionará para la mayoría de las funciones. Es este precisamente uno de los retos más importantes a los que se enfrentan los investigadores en el área de la computación evolutiva.

Un concepto relacionado con la exploración y explotación de los algoritmos evolutivos es la **diversidad**. En el contexto de la optimización evolutiva, la diversidad es una medida que indica que tan distribuidos están los individuos en la función a optimizar. La exploración permite que la diversidad de la población se incremente, mientras que la explotación favorece a la pérdida de esta. Esto no es deseable ya que la pérdida de la diversidad conlleva a la convergencia prematura a óptimos locales (cuando los individuos se encuentran en regiones en donde existe un óptimo local).

Con el objetivo de garantizar la convergencia y basándose en la idea de que con el paso de las generaciones los individuos mejorarán su aptitud, algunos algoritmos evolutivos utilizan la estrategia de reemplazar completamente la población de una iteración a otra, extinguiendo a todos los individuos actuales (los padres) para que los individuos generados por estos a través de los mecanismos biológicos mencionados (los hijos) sean los que tomen parte en el proceso de optimización de la siguiente generación. Otra forma que emplean los algoritmos evolutivos en su propósito de alcanzar el óptimo es mantener en la población al mejor individuo encontrado hasta el momento. La ventaja de utilizar esta estrategia es que hay convergencia asegurada, aunque muy probablemente esta sea a un óptimo local.

Definición 2.3.9 [Wei08] *Se dice que un algoritmo evolutivo es **generacional** cuando la generación $t + 1$ sólo contiene la descendencia (los hijos) de la generación t , es decir, ningún individuo de la generación t (los padres) es seleccionado para formar parte de la generación $t + 1$.*

Definición 2.3.10 [Wei08] *Se dice que un algoritmo evolutivo es **elitista** cuando este se asegura de que al menos una copia del mejor individuo de la generación actual se propague a la siguiente generación.*

Es obvio mencionar que si un algoritmo evolutivo es elitista, este no puede ser generacional, aunque es importante mencionar que existen algunos algoritmos que son generacionales y que manejan cierto porcentaje de elitismo. Por ejemplo, si un algoritmo es generacional con un 10% de elitismo, el 10%

de los individuos (los mejores) de la generación anterior pasarán a la siguiente generación, mientras que el 90% restante serán individuos nuevos.

Clasificación

Con base en su búsqueda especial y sus espacios problema, los algoritmos evolutivos se clasifican en cinco grupos, los cuales se enlistan a continuación:

- **Algoritmos genéticos.** A este grupo pertenecen los algoritmos evolutivos que tienen cadenas de bits como espacio de búsqueda \mathbb{S} .
- **Estrategias evolutivas.** En este grupo están incluidos los algoritmos evolutivos que exploran el espacio de vectores reales $\mathbb{X} \subseteq \mathbb{R}^n$.
- **Programación genética.** Incluye a los algoritmos evolutivos que desarrollan programas, algoritmos y similares; así como los algoritmos evolutivos que toman individuos con formas de árboles.
- **Sistemas de aprendizaje y clasificadores.** Son aproximaciones de aprendizaje que asignan un valor de salida a un conjunto de valores de entrada dados. Internamente usan un algoritmo genético para encontrar las nuevas reglas para realizar este mapeo.
- **Programación evolutiva.** Es una aproximación evolutiva que trata las instancias de los genomas como especies diferentes antes que como individuos. Con el paso del tiempo esta se ha ido mezclando con la programación genética y los otros algoritmos evolutivos.

En la figura 2.6 se ilustran las diferentes familias de los algoritmos evolutivos. Sin importar a que grupo pertenezca cada algoritmo, todos tienen como base el esquema básico presentado en el algoritmo 2, realizando algunas mejoras y extensiones. Las características distintivas de estos algoritmos son:

- El tamaño de la población o el número de poblaciones utilizadas.
- El método de selección de individuos para reproducción.
- La forma en la que la descendencia es incluida en la población.

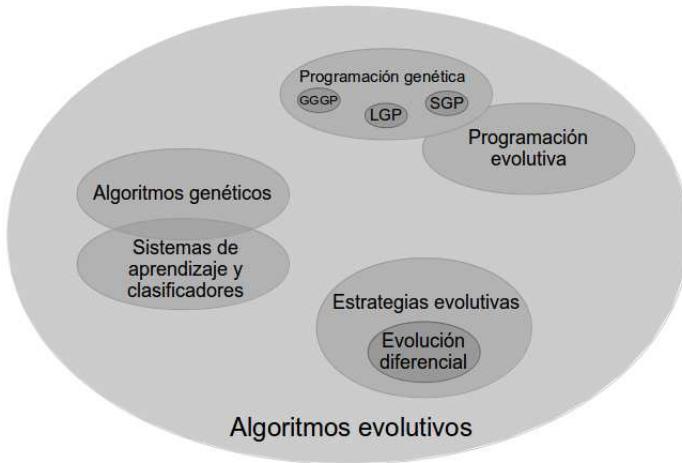


Figura 2.6: La familia de los algoritmos evolutivos

2.3.3 Inteligencia de enjambre

Como se ha visto hasta el momento, la computación se ha inspirado en muchas ocasiones de procesos naturales para resolver problemas difíciles, obteniendo de estas ideas resultados satisfactorios, siendo un ejemplo de esto los algoritmos evolutivos tratados en la sección anterior. Una idea más reciente de aplicaciones de este tipo utilizada para resolver problemas complejos de optimización es la inteligencia de enjambre. Basándose en la interacción social de pequeños grupos de “agentes” para alcanzar un cierto objetivo, esta área de la inteligencia artificial se enfoca en encontrar el óptimo de un problema de optimización por medio de la comunicación entre los integrantes del enjambre. Para alcanzar esta meta, la inteligencia de enjambre estudia el comportamiento colectivo y las propiedades emergentes de sistemas con estructura social compleja, auto-organizada y descentralizada. Ejemplos de “sociedades” de este tipo son los bancos de peces, las parvadas de pájaros, las colonias de hormigas y otras manadas de animales que tienen increíbles capacidades de auto-organización que generan comportamientos colectivos que no pueden ser descritos analizando cada integrante por separado, sino que cada uno de estos se ve influenciado por los demás integrantes del conjunto al que pertenece, esto es, existe una influencia social sobre él. En el entorno de la optimización global, la inteligencia de enjambres apareció en 1989 con un conjunto de algoritmos para controlar enjambres de robots [BW89]. Si bien los algoritmos de inteligencia de enjambre no están considerados dentro de los algoritmos evolutivos debido a las diferencias existentes en la filosofía y operaciones de estos, cuando se habla de inteligencia de enjambre suele

hacerse a la par de los algoritmos evolutivos ya que presentan similaridades como lo son el hecho de que son estocásticos y poblacionales, las áreas de aplicación y la audiencia científica que se interesó inicialmente en estas propuestas y que cobijó los trabajos relacionados con la inteligencia de enjambre en sesiones especiales de algoritmos evolutivos, además de publicarlos en revistas científicas bajo el tópico de computación evolutiva. A mediados de la década de los 90's la inteligencia de enjambre fue considerada dentro de la computación evolutiva.

A continuación se enlistan algunos de los más importantes algoritmos de inteligencia de enjambre:

- **Optimización mediante colonia de hormigas (Ant Colony Optimization, ACO).** Este algoritmo se inspira en el comportamiento de las hormigas en la búsqueda de comida.
- **Búsqueda por difusión estocástica (Stochastic Diffusion Search, SDS).** Este algoritmo está basado en el denominado juego del restaurante, en el cual un grupo de personas deben encontrar un lugar para comer en una ciudad desconocida.
- **Optimización por enjambre de partículas (Particle Swarm Optimization, PSO).** Se basa en los conceptos y reglas que rigen poblaciones socialmente organizadas en la naturaleza, tales como parvadas de pájaros, bancos de peces y otras manadas de animales.
- **Optimización por forrajeo de bacterias (Bacterial Foraging Optimization, BFO).** Está basado en el comportamiento quimiotáctico de la E.Coli.
- **Colonias de abejas.** Se basa en el proceso de búsqueda de néctar en las flores por parte de las abejas melíferas. Este tipo de algoritmos no cuenta con un nivel de agrupamiento. Dentro de estos algoritmos se encuentran la colonia artificial de abejas (Artificial Bee Colony, ABC), la optimización por colonia de abejas (Bee Colony Optimization, BCO), el algoritmo Bees, la optimización por búsqueda de néctar por abejas (Bee Nectar Search Optimization, BNSO) y el algoritmo de colmena de abejas artificiales (Artificial Bee Hive Algorithm, ABHA).

Los conceptos básicos de este tipo de algoritmos son prácticamente los mismos que los algoritmos evolutivos, por lo que esos conceptos son válidos también para la inteligencia de enjambre. En el capítulo 3 se analizará a detalle el algoritmo principal de esta tesis, optimización por enjambre de partículas, un caso de este tipo de algoritmos.

Capítulo 3

Optimización por enjambre de partículas (PSO)

En el presente capítulo se presentan los conceptos más importantes de optimización por enjambre de partículas. Comenzamos presentando la versión canónica de PSO y las características principales de esta. Posteriormente se describen a detalle los parámetros con los que funciona el algoritmo. Finalmente, se mencionan las principales topologías que se utilizan para la comunicación de las partículas en PSO.

3.1 Introducción

Desde el principio de los tiempos el ser humano ha tenido que satisfacer sus necesidades. Para cumplir con este objetivo el hombre se organizó en grupos para poder sortear de mejor forma las adversidades que se le pudieran presentar. Inicialmente este se estructuró en grupos pequeños en su etapa de nómada, para posteriormente formar conjuntos más grandes en su etapa sedentaria, permitiendo así la formación de grandes civilizaciones que conllevaron al dominio del hombre sobre las demás especies. Pero este comportamiento social no sólo se presenta en el ser humano. Especies más pequeñas han empleado esta práctica para resolver los problemas que la naturaleza les puede presentar. Esta forma de organización ha permitido que estas especies puedan obtener alimento, sobrevivir a ataques de especies más grandes, entre muchos otros beneficios. Este comportamiento social es un resultado directo del proceso de la evolución de las especies y de la selección natural. Es claro advertir que los individuos de estas especies que se agrupan en “manadas” tienen mayor probabilidad de sobrevivir que aquellos que no lo hacen, por lo que la selección natural desechará a los miembros que trabajen individualmente, dando paso a que estos desarrollen mecanismos de comunicación que les permitan organizarse de mejor forma. Como ya se mencionó con anteri-

oridad, resolver un problema de optimización consiste en encontrar el valor óptimo de este, esto es, el mejor elemento bajo un cierto criterio establecido. Dada esta premisa podemos considerar que las especies, al resolver los problemas que les presenta la naturaleza, están resolviendo en sí un problema de optimización, ya que al conseguir alimento o al lograr sobrevivir al ataque de una especie de mayor tamaño, retomando los ejemplos mencionados con anterioridad en el párrafo, están resolviendo de manera satisfactoria, y quizás la mejor de todas las posibles, este problema. El hombre ha resuelto a lo largo de toda su historia problemas de optimización, pero debido al auge científico y tecnológico presentado en los últimos años estos problemas son cada vez más complejos, por lo que se han tenido que idear mejores métodos para resolverlos. Afortunadamente el incremento en recursos computacionales permiten hoy en día crear algoritmos que resuelvan de forma rápida y eficaz estos complejos problemas de optimización. Un ejemplo digno a imitar para resolver problemas de optimización es precisamente el de la naturaleza. Basándose en la forma en la que algunas especies resuelven en conjunto problemas complejos, surge una gama de algoritmos dentro de la computación evolutiva: la inteligencia de enjambre. Dentro de esta se encuentra optimización por enjambre de partículas (PSO), un algoritmo que originalmente se inspiró en un conjunto de aves que buscaban maíz y cuyo objetivo es producir inteligencia computacional utilizando interacciones sociales en lugar de utilizar solamente habilidades cognitivas individuales.

En el presente trabajo se considera que PSO pertenece a la computación evolutiva, aunque existe una discusión al respecto. Algunos investigadores del área lo consideran así (Eberhart, por ejemplo [EK95]), mientras que otros investigadores consideran que PSO no pertenece a la computación evolutiva (Talbi [Tal09], Fogel [Fog98], Coello y Segura [CSM16], entre otros). En el caso de la presente tesis se está considerando a PSO dentro de la computación evolutiva, debido a que se considera que la capacidad de comunicación de las partículas puede verse como un proceso evolutivo y de adaptación. De igual forma, en PSO se pueden observar una “especie” de mutación en la influencia de la mejor posición que ha visitado una partícula, una “especie” de cruza en la influencia social de las partículas y una “especie” de selección determinada por la topología de las partículas. Por estas razones en esta tesis se considera a PSO como un algoritmo perteneciente a la computación evolutiva.

3.2 Versión canónica

La versión canónica de optimización por enjambre de partículas fue propuesta por James Kennedy y Russell Eberhart en [KE95]. Su idea principal

era producir inteligencia computacional utilizando analogías simples de interacción social en vez de usar solamente habilidades cognitivas individuales. Las simulaciones realizadas por ellos y desarrolladas en [KE95], [EK95] y [ESD96] fueron influenciadas por el trabajo presentado por Frank Heppner y Ulf Grenander en [HG90]. A la postre, este método se convirtió en un poderoso y ampliamente utilizado método de optimización.

Optimización por enjambre de partículas consiste en colocar un cierto número de “partículas” en el espacio de búsqueda de la función a optimizar. Cada partícula explorará el espacio de búsqueda basando su movimiento en la propia experiencia de la función (**influencia personal**) y en la experiencia de las demás partículas (**influencia social**), utilizando su posición actual, la mejor experiencia personal y la mejor experiencia de todo el enjambre, agregando algunas perturbaciones aleatorias. Para simular este comportamiento, cada partícula consta de tres vectores n -dimensionales, en donde n es la dimensión del espacio de búsqueda. Denotaremos por \vec{x}_i a la posición actual de la partícula i , \vec{p}_i a la mejor experiencia personal de la partícula i y \vec{v}_i la velocidad actual de la partícula i . En cada iteración, cada partícula actualiza su velocidad \vec{v}_i y este valor se suma a la posición actual \vec{x}_i , provocando que esta se mueva a su nueva posición en el espacio. Posteriormente, se evalúa la función de aptitud de la partícula para comparar si esta posición es mejor que la que se encuentra almacenada en \vec{p}_i , y si es así, se almacena \vec{x}_i en \vec{p}_i . El vector \vec{p}_i también es conocido como el p_{best} (personal best) de i . Este proceso se realiza iterativamente hasta que se cumpla algún criterio de paro y el algoritmo se detenga. Debido a que PSO guía su movimiento en la dirección de las mejores posiciones (personal y social), este eventualmente convergerá, aunque no se puede asegurar que sea a un mínimo global. Se hace énfasis en el hecho de que el algoritmo basa su funcionamiento en la interacción entre las partículas, ya que si se utilizara únicamente una partícula para resolver un problema, esta no podría hacerlo por si sola. Esta es en sí la característica principal de cualquier algoritmo perteneciente a la inteligencia de enjambre. Ahora, la pregunta a resolver es como se producen estas interacciones entre las partículas. Para lograr esto, la población es organizada en una estructura de comunicación, comúnmente llamada **topología**. Esta funciona como un tipo de “red social” en la que las partículas se conectan por pares mediante aristas no dirigidas, lo cual implica que las partículas son vecinas bidireccionalmente, esto es, que si i es vecina de j entonces j es vecina de i . Entonces, las partículas compartirán información sobre cual es el mejor punto que han visitado solamente con su **vecindario**, esto es, todas las partículas con las que está conectada en la topología. Se denotará al mejor valor social del vecindario de la partícula i como \vec{g}_i . Es claro observar que el funcionamiento de PSO es fuertemente afectado por la topología con la que

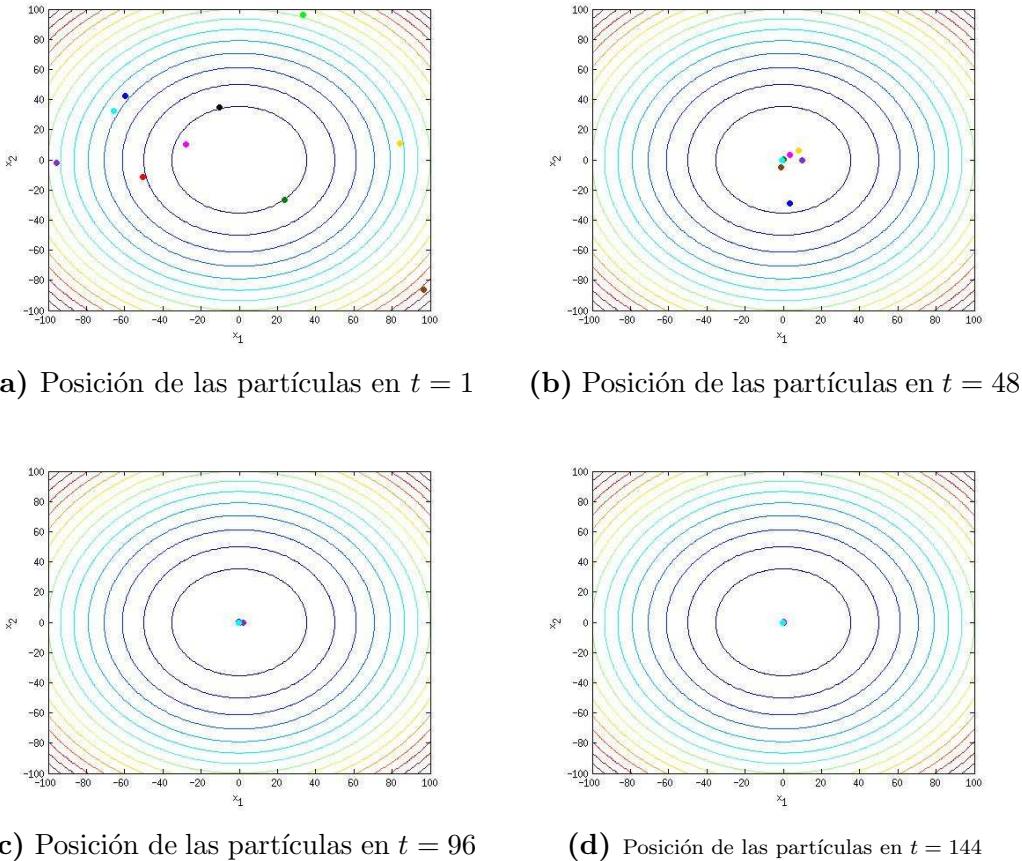


Figura 3.1: Evolución a través del tiempo de la versión canónica de PSO

se conectan las partículas. Más adelante se discutirán a detalle las topologías en PSO. En la versión canónica del algoritmo, se propuso conectar a todas las partículas entre si, es decir, el vecindario de cada partícula son todas las demás partículas que integran la población. Esta topología es comúnmente llamada **totalmente conectada** y es una de las más utilizadas. En este caso se utiliza el mejor valor de todo el enjambre para guiar la búsqueda social. A este valor lo llamaremos $\vec{g_{best}}$. En el algoritmo 3 se presenta la versión canónica de PSO. En la figura 3.1 se muestra como converge esta versión de PSO en la función esfera en dos dimensiones $f(x_1, x_2) = x_1^2 + x_2^2$, en el espacio de búsqueda $x_1 \in [-100, 100]$ y $x_2 \in [-100, 100]$, usando 10 partículas, factor de inercia $\omega = 0.7298$ y coeficientes de aceleración $c_1 = c_2 = 1.49618$.

Los criterios de paro que se pueden utilizar en el algoritmo 3 pueden ser algunos de los descritos en la sección 2.2.1. El símbolo \otimes denota la

Algoritmo 3 Versión canónica de PSO

Entrada: La función a optimizar, el espacio de búsqueda, c_1 , c_2 , número de partículas que integrarán la población, criterios de paro.

Salida: La aproximación del mínimo obtenido.

- 1: $t \leftarrow 1$
- 2: Inicialización aleatoria de la posición y la velocidad de las partículas dentro del espacio de búsqueda.
- 3: Guardar el valor de la posición obtenida en el p_i de cada partícula y el valor de aptitud (la función a optimizar) de cada partícula.
- 4: **mientras** No se cumpla algún criterio de paro **hacer**
- 5: Actualizar la velocidad y posición de la partícula i usando las ecuaciones 3.1 y 3.2.

$$\vec{v}_i^{t+1} = \vec{v}_i^t + c_1 \vec{r}_{1i} \otimes (\vec{p}_i^t - \vec{x}_i^t) + c_2 \vec{r}_{2i} \otimes (\vec{g}_{best} - \vec{x}_i^t) \quad (3.1)$$

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1} \quad (3.2)$$

- 6: Evaluar la función de aptitud de cada partícula.
 - 7: Actualizar el \vec{p}_i de cada partícula y buscar el \vec{g}_{best} del enjambre.
 - 8: $t \leftarrow t + 1$
 - 9: **fin mientras**
-

multiplicación elemento por elemento de dos vectores. Los vectores r_1 y r_2 son dos vectores aleatorios n -dimensionales uniformemente distribuidos en el intervalo $[0, 1]$. Los parámetros del algoritmo c_1 y c_2 (coeficientes de aceleración) serán tratados a detalle en la siguiente sección. Por último, es importante mencionar que en la versión canónica se limitaban las velocidades dentro del rango $[-V_{max}, +V_{max}]$, con la finalidad de que las partículas no tuvieran movimientos muy bruscos y evitar que pudiesen salir del espacio de búsqueda.

3.3 Parámetros

El funcionamiento de optimización por enjambre de partículas está determinado por un pequeño número de parámetros. Para obtener un buen desempeño del algoritmo, es importante encontrar un conjunto de valores que ayuden a alcanzar el óptimo de una forma eficiente. En la presente sección se abordan a detalle los parámetros con los que funciona optimización por enjambre de partículas, así como que valores de estos, en general, son los que

generan la obtención de mejores resultados del algoritmo.

3.3.1 Coeficientes de aceleración

Con la finalidad de guiar la búsqueda del óptimo, cada partícula se ve influenciada por su experiencia personal y la experiencia social de su vecindario. Un punto significativo del funcionamiento del algoritmo es determinar que tanto influye cada una de estas experiencias en las partículas. Esta cantidad está determinada por los coeficientes de aceleración, cuya importancia en el algoritmo es indicarle a cada partícula que proporción de cada experiencia tomar para guiar su movimiento. Asumiendo que c_1 es el coeficiente de aceleración en la dirección de la mejor experiencia personal y c_2 el coeficiente de aceleración en la dirección de la mejor experiencia social del vecindario, es obvio observar que si asignamos $c_1 > c_2$ estamos dándole un mayor peso a la experiencia personal, mientras que hacer $c_1 < c_2$ implica preferir guiar la búsqueda en dirección de la mejor experiencia social del vecindario. En los primeros trabajos se prefirió tener un balance entre estas dos, tratando de encontrar un equilibrio entre la exploración y la explotación del algoritmo, por lo que se propuso hacer $c_1 = c_2 = 2$. Otra observación importante que debe mencionarse es que utilizar coeficientes de aceleración grandes implica movimientos muy abruptos de las partículas que incrementan el riesgo de no converger en una cierta región, mientras que valores pequeños de los coeficientes de aceleración implican movimientos lentos que podrían hacer que el algoritmo demorara mucho en converger a algún mínimo, o incluso no converger. En la literatura se ha propuesto variar los coeficientes de aceleración en dependencia directa con la generación en la que se encuentra el algoritmo, por ejemplo en [BM09] y coeficientes de aceleración adaptables como los propuestos en [ZXZC07]. En la figura 3.2 se muestra la influencia de la experiencia personal y la experiencia social, las cuales son afectadas directamente por los coeficientes de aceleración.

3.3.2 Factor de inercia

Uno de los problemas que se presentaron en la versión canónica de PSO fue el del poco control que se tenía con la velocidad de las partículas. Como se mencionó, para evitar movimientos descontrolados de estas debido a velocidades muy grandes, se limitaba su velocidad a no superar una velocidad máxima. El hecho de agregar esta restricción involucraba tener un parámetro más para el algoritmo, que además era muy importante para el funcionamiento de este. Una solución a este problema fue propuesta por Shi y Eberhart en [SE98], en

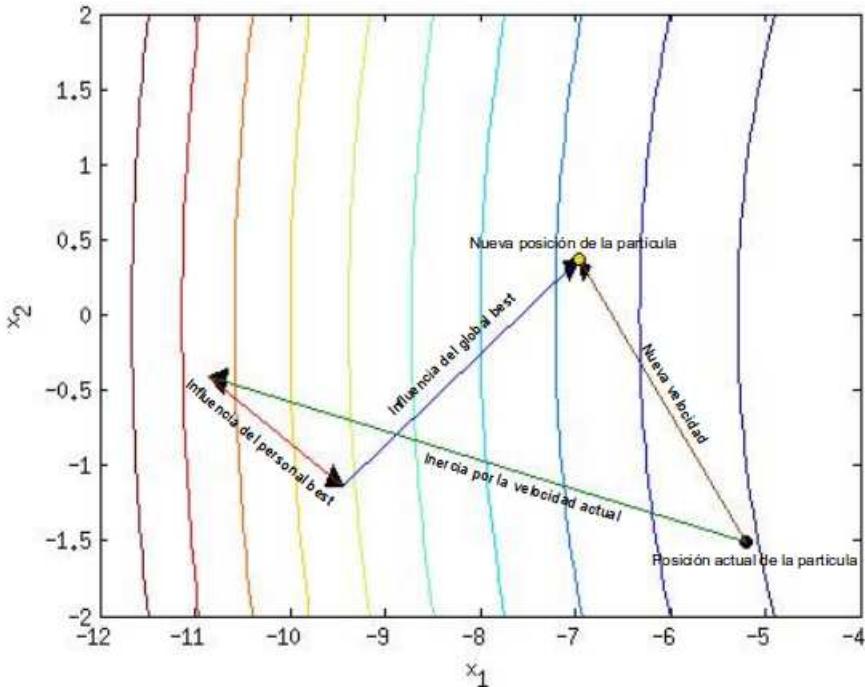


Figura 3.2: Influencias de la mejor experiencia personal y la mejor experiencia social sobre una partícula

donde proponen actualizar la velocidad de las partículas usando la ecuación 3.3:

$$\vec{v}_i^{t+1} = \omega \vec{v}_i^t + c_1 r_{1i}^t \otimes (\vec{p}_i^t - \vec{x}_i^t) + c_2 r_{2i}^t \otimes (\vec{g}_{best} - \vec{x}_i^t) \quad (3.3)$$

En donde ω fue llamado “factor de inercia”. Considerando el incremento en la velocidad:

$$\Delta v = \vec{v}_i^{t+1} - \vec{v}_i^t = \omega \vec{v}_i^t + c_1 r_{1i}^t \otimes (\vec{p}_i^t - \vec{x}_i^t) + c_2 r_{2i}^t \otimes (\vec{g}_{best} - \vec{x}_i^t) - \vec{v}_i^t = \vec{f}_i^t - (1 - \omega) \vec{v}_i^t$$

En donde $\vec{f}_i^t = c_1 r_{1i}^t \otimes (\vec{p}_i^t - \vec{x}_i^t) + c_2 r_{2i}^t \otimes (\vec{g}_{best} - \vec{x}_i^t)$ puede ser visto como una fuerza externa que actúa sobre una partícula cambiando su velocidad. Como se puede observar, el término $1 - \omega$ puede ser visto como un coeficiente de fricción y entonces ω puede ser interpretado como la fluidez del medio en el que las partículas se mueven. Basándose en esta idea, en trabajos iniciales en los que se utilizaba el factor de inercia se utilizaban valores relativamente grandes para beneficiar la fase de exploración del algoritmo y valores relativamente pequeños para favorecer la etapa de explotación. Este comportamiento deseado del algoritmo, de iniciar en la etapa de exploración

y transitar a la etapa de explotación hacia las iteraciones finales fue emulado por Eberhart y Shi en [ES00], en donde realizan pruebas iniciando con un factor de inercia de 0.9 y haciendo que este fuera disminuyendo gradualmente con el paso de las generaciones hasta alcanzar un valor final de 0.4. Otra estrategia utilizada es usar un factor de inercia con un componente aleatorio, en vez de un decremento en el tiempo, como lo proponen Eberhart y Shi en [ES01] en el que utilizan $\omega = U(0.5, 1)$. También existen estudios en los que se han obtenido buenos resultados incrementando el factor de inercia, como se demuestra en [ZMZQ03].

Lo que es un hecho es que con una selección apropiada de ω y de los coeficientes de aceleración, c_1 y c_2 , se puede generar un comportamiento estable de PSO. Frans van den Bergh y Andries Engelbrecht realizaron un estudio en [BE06] en el que analizan el comportamiento de las partículas para ciertas combinaciones de ω , c_1 y c_2 en la topología todos conectados, encontrando que los valores con los que PSO con esta topología obtenía un mejor comportamiento en distintas funciones de prueba son $\omega = 0.7298$ y $c_1 = c_2 = 1.49618$.

3.3.3 Velocidad máxima

Como se mencionó con anterioridad, en la versión canónica de PSO no había un control en la velocidad de las partículas, por lo que estas podían tener movimientos indeseados. Para resolver este problema, se planteó limitar esta a un valor máximo permitido, por lo que si una partícula alcanzaba una velocidad mayor a esta, la velocidad de la partícula se reasignaba a la velocidad máxima o dicho más formalmente, se limitaba el valor de \vec{v}_i al intervalo $[-V_{max}, V_{max}]$, en donde V_{max} es la velocidad máxima permitida. Pese a que hasta la fecha se han ideado adecuaciones al algoritmo que le han restado importancia a la velocidad máxima, para algunas versiones de PSO la velocidad máxima sigue siendo muy importante, por lo que no está de más presentar un breve análisis de esta. Como cualquier parámetro del algoritmo, el valor de la velocidad máxima afecta el desempeño de PSO. Valores muy grandes de V_{max} generan movimientos erráticos de las partículas provocando una fuerte atracción hacia el \vec{g}_{best} , provocando que las partículas no realicen suficiente exploración del espacio de búsqueda y aumentando el riesgo estancarse en óptimos locales. Además, una velocidad máxima muy grande podría provocar que las partículas se salgan del espacio de búsqueda, lo cual es catastrófico para el algoritmo. Por otro lado, si se asigna un valor pequeño a V_{max} , se evitan los problemas mencionados para un valor muy grande de V_{max} , pero las partículas se moverán muy lentamente, generando una lenta convergencia del algoritmo, lo cual podría hacer que se agoten

los recursos computacionales asignados. Un estudio hecho por Fan y Shi en [FS01] muestra que un cambio dinámico adecuado en V_{max} genera un buen rendimiento de PSO. Un problema que se puede presentar es si el espacio de búsqueda tiene diferentes tamaños en distintas dimensiones, ya que la asignación de una velocidad máxima en una dimensión podría ser mala para otra dimensión en la que el espacio de búsqueda es más pequeño (o más grande). Para evitar este problema Abido propone en [Abi01] asignar el valor de V_{max} usando la ecuación 3.4.

$$V_{max} = \frac{X_{max} - X_{min}}{N_I} \quad (3.4)$$

en donde N_I es el número de intervalos seleccionados, X_{max} y X_{min} son los valores máximos y mínimos, respectivamente, del espacio de búsqueda, todo esto para la dimensión d . Con esto se obtienen distintas velocidades máximas para cada dimensión, con base en el tamaño del espacio de búsqueda de cada una de ellas.

3.3.4 Factor de restricción

En los primeros trabajos de PSO se pudo observar el comportamiento desordenado que se podía presentar en las velocidades de las partículas, aunque sin llegar a comprender a qué se debía esto. Conforme avanzaban las iteraciones del algoritmo se observa un crecimiento desmedido de la velocidad a niveles inaceptables. Una solución propuesta para lidiar con este problema fue el de limitar las velocidades de las partículas a una velocidad máxima, como se mencionó en la sección anterior. En [Ken98], James Kennedy observó que el comportamiento de las partículas presentaba mayor regularidad cuando $c_1 + c_2 \in (0, 4)$. Con base en esta observación propuso una estrategia en la que se utilizaba un parámetro al que llamó “factor de restricción”. Este parámetro controlaba la velocidad de convergencia de las partículas y se convirtió en un método que prevenía la explosión, aseguraba convergencia y eliminaba el parámetro V_{max} . En [CK02], Maurice Clerc y James Kennedy propusieron una forma de incorporar el factor de restricción en PSO. Esto lo lograron actualizando la velocidad de las partículas utilizando la ecuación 3.5.

$$\vec{v}_i^{t+1} = \chi [\vec{v}_i^t + c_1 \vec{r}_{1i}^t \otimes (\vec{p}_i^t - \vec{x}_i^t) + c_2 \vec{r}_{2i}^t \otimes (\vec{l}_i^t - \vec{x}_i^t)] \quad (3.5)$$

en donde $c = c_1 + c_2$ y $\chi = \frac{2}{c-2+\sqrt{c^2-4c}}$.

Comúnmente, cuando se utiliza este factor se asigna $c_1 = c_2 = 2.05$ y $c = 4.1$. Con esta asignación se obtiene un valor de $\chi = 0.7298$. En sí,

PSO con restricción es algebraicamente equivalente a un PSO con factor de inercia. Esta equivalencia se obtiene haciendo $\omega = \chi$, $c_1 = \chi c_1$ y $c_2 = \chi c_2$. Con estas transformaciones, se obtendría un PSO con inercia con parámetros $\omega = 0.7298$ y $c_1 = c_2 = 1.49618$, parámetros con los que PSO con topología todos conectados tiene un buen desempeño en una gran cantidad de funciones de prueba, como se demuestra en [BE06].

3.3.5 Tamaño del enjambre

El tamaño de la población tiene un papel muy importante en el rendimiento de los algoritmos de optimización estocásticos. Poblaciones pequeñas favorecerán la búsqueda local, generando así convergencia en óptimos locales. Por el contrario, poblaciones muy grandes incrementan el costo computacional del algoritmo, propiciando una convergencia más lenta, aunque probablemente al óptimo global. Entonces, es importante asignar un tamaño apropiado de la población para el problema que se desea resolver. Uno de los aspectos a considerar al momento de hacer esta asignación es la dimensión del problema a optimizar. Es conveniente asignar una población de mayor tamaño en problemas de dimensión grande, aunque esta no puede ser considerada como una regla general. Algunos trabajos en los que se implementan estrategias para modificar el tamaño de la población en búsqueda de un mejor rendimiento del algoritmo han sido propuestos, aunque en menor cantidad a los enfocados a otros parámetros. Leong y Yen proponen en [LY06] un PSO con población dinámica. En este, la cantidad de las partículas es calculada en cada generación, aunque el proceso para calcular esta es complejo. En el 2009, DeBao y ChunXia proponen un método en el que el tamaño de la población se modifica con base en la diversidad de esta [DC09]. Otra estrategia propuesta fue la de manejar un número variable de partículas basándose en el estado actual de la búsqueda y en sí existen algunas partículas redundantes, tratando de evitar estancamiento en mínimos locales. Esta idea fue presentada por Sheng-Ta Hsieh et. al. en [HSLT09]. Algunos valores que se utilizan como tamaño del enjambre son 30, 50 y 100, cuyo valor puede ser seleccionado con base en el tipo de problema que se quiere resolver.

3.4 Topologías

La comunicación entre las partículas es una de las partes más importantes en el funcionamiento de PSO. Con el objetivo de moverse a zonas promisorias de la función de aptitud, la partícula debe considerar la mejor experiencia personal y la mejor experiencia de su vecindario. Para la primera consideración

la partícula sólo debe de ser capaz de “recordar” la mejor posición que ha visitado. Para la segunda, debe ser capaz de saber la mejor posición de las partículas de las que es vecina. Entonces, cada partícula debe saber quienes son su vecinas. Esta relación entre partículas está dada por la **topología** de las partículas, una estructura en la que se indican que partículas tienen comunicación directa entre sí. En general, existen dos filosofías en la asignación de la topología de PSO. En la primera, esta se asigna al principio del algoritmo y no cambia durante todo el proceso de búsqueda del óptimo. En la segunda, la topología se actualiza cada determinado tiempo, con base en ciertos criterios establecidos. A continuación se presentan a detalle cada una de estas formas de manejar la topología de las partículas.

3.4.1 Topologías estáticas

Originalmente, optimización por enjambre de partículas constaba de un conjunto de partículas que se comunicaban entre ellas para buscar el óptimo de una función. Todas las partículas tenían comunicación entre sí, por lo que estas seguían a aquella que había encontrado la mejor posición en algún momento de la búsqueda. A esta partícula se le llamó $\vec{g_{best}}$. Es obvio pensar que el hecho de que las partículas tomen información del mejor punto encontrado favorecerá a la explotación y que el algoritmo tendrá una convergencia más rápida, pero con el riesgo mencionado con anterioridad de estancarse en óptimos locales. Entonces, en trabajos iniciales se pudo observar que esta topología era conveniente para problemas unimodales, pero que en problemas multimodales no era conveniente utilizarla. Por lo tanto, en trabajos posteriores se idearon nuevas formas de comunicación entre las partículas, es decir, nuevas topologías. Debido a que en las nuevas topologías propuestas las partículas ya no tenían comunicación con todas las demás partículas, si no que solamente se comunicaban con un subconjunto de estas (las que estuvieran en su vecindario local), al mejor elemento social de cada partícula se le llamó $\vec{l_{best}^i}$ (el local best de i). Posteriormente se realizaron estudios en los que se analizaba la influencia de las topologías en el desempeño del algoritmo. Rui Mendes y José Neves demuestran en [MN04] que la conectividad de las topologías está directamente relacionada con el comportamiento de exploración y explotación de las partículas. James Kennedy y Rui Mendes, en [Ken99, KM02], llegan a la conclusión de que topologías con poca conectividad favorecen a la exploración, mientras que topologías con mucha conectividad favorecen a la explotación del algoritmo. Un punto importante a mencionar es que generalmente las topologías se asignan al principio del algoritmo y estas no cambian durante todo el proceso de búsqueda. A este tipo

de topologías que se mantienen igual durante todo el algoritmo se les conoce como **topologías estáticas**. A continuación se presentan las topologías estáticas más comunes. En la figura 3.3 se muestran gráficamente estas topologías, en donde el árbol mostrado es un árbol binario y la topología aleatoria tiene vecindarios de tamaño tres.

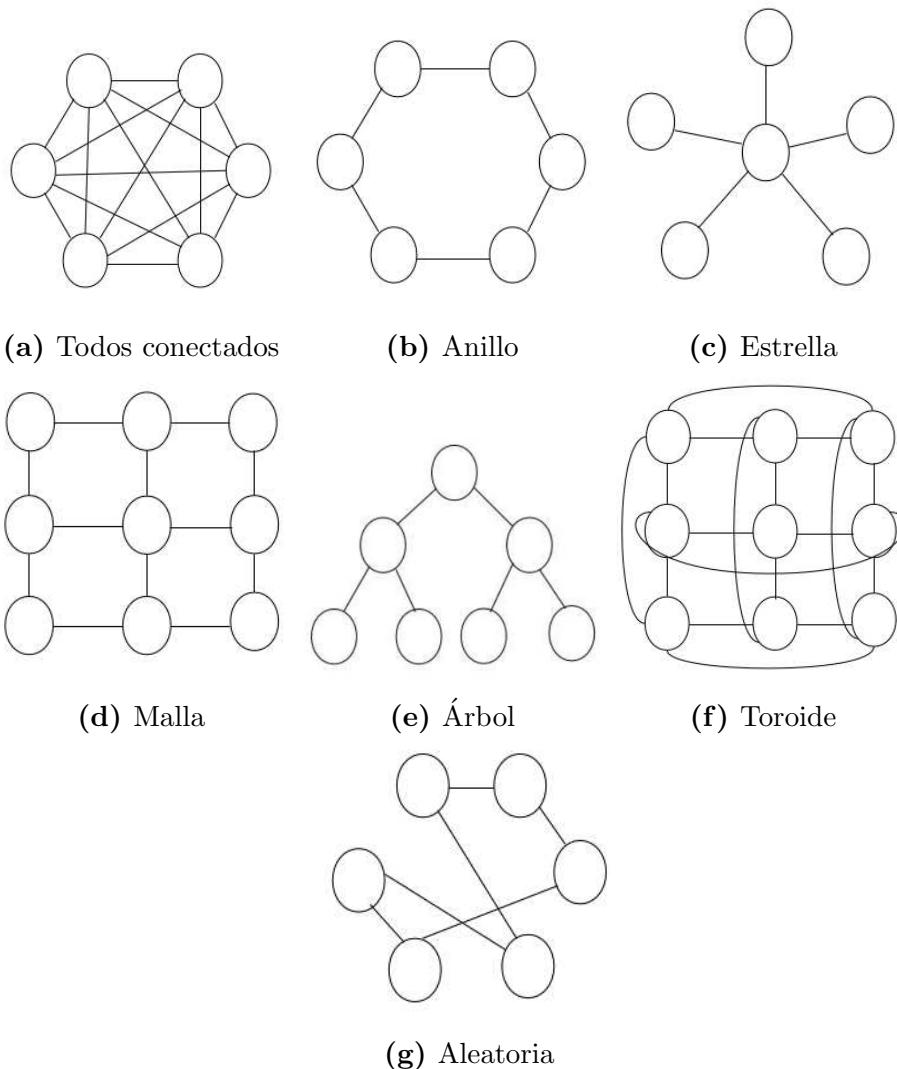


Figura 3.3: Distintos tipos de topología utilizadas en PSO

- **Todos conectados.** En esta topología cada nodo tiene comunicación con todos los demás. Esto significa que todas las partículas tienen comunicación entre sí. Como se mencionó, esta fue la topología propuesta en el PSO canónico. Esta versión de PSO es conocida como *gbest*.

- **Anillo.** En esta topología la partícula con índice k se conecta con las partículas con índice $k - 1$ y $k + 1$ (salvo las partículas 1 y p , que se conectan entre sí, en donde p es la cantidad de partículas). A diferencia de la topología todos conectados, cada partícula sólo será afectada por sus vecinos inmediatos. Esta versión fue conocida como *lbest*.
- **Estrella.** En esta topología la información pasa a través de una sola partícula ya que un nodo central se conecta con todas las demás partículas. Las partículas entonces toman información del nodo central y la “siguen”. Si la partícula central es la mejor, entonces el comportamiento de esta topología es similar al de la topología todos conectados.
- **Malla.** En esta topología, los nodos son organizados en un arreglo rectangular y se conectan con las partículas que se encuentran al norte, sur, este y oeste de ellas (excepto cuando no existan partículas en esas posiciones, como en las esquinas y en los extremos). Entonces, cada partícula tendrá un máximo de cuatro vecinos y un mínimo de dos, dependiendo de la posición en la que se encuentren en la malla. A esta topología también se le conoce como **Von Neumann**.
- **Árbol.** También es conocida como topología jerárquica. En esta topología se tiene un nodo raíz que es conectado a uno o más individuos. El nodo raíz es el primer nivel de la jerarquía, mientras que los nodos que fueron conectados a este forman parte del segundo nivel. Posteriormente los nodos del segundo nivel se conectan con otros nodos, los cuales ahora son parte del tercer nivel. De esta manera se siguen conectando nodos, hasta completar todas las conexiones entre partículas, evitando dejar alguna partícula desconectada. Normalmente las partículas se organizan en un árbol binario, esto es, en una estructura de árbol en la que los nodos de un nivel se conectan con a lo más dos nodos del siguiente nivel.
- **Toroidal.** La topología toroidal es similar a la topología de malla, salvo que en esta todos los nodos están conectados con cuatro vecinos, esto significa que incluso los nodos de las esquinas y de los extremos también tienen conexión con cuatro nodos.
- **Aleatoria.** En esta topología se seleccionan aleatoriamente las partículas que tendrán comunicación entre si. Por lo regular, para cada partícula se selecciona un cierto número de vecinos aleatorios. El número de vecinos que tendrá cada nodo es fijado al inicio del algoritmo.

3.4.2 Topologías dinámicas

Si bien en las versiones iniciales de PSO se utilizaban topologías que se asignaban al principio del algoritmo y estas se mantenían fijas durante el desarrollo de este, con el paso del tiempo se comenzó a discutir la necesidad de utilizar otras topologías para poder resolver problemas que con los vecindarios convencionales no se podían resolver. Un punto que se comenzó a discutir fue el de la forma en la que se construían los vecindarios, ya que esto generalmente se realizaba sin utilizar alguna información que pudiesen proporcionar las partículas. Los estudios iniciales de Suganthan [Sug99] demostraron que la topología *lbest* favorecía la etapa de exploración del algoritmo, mientras que la topología *gbest* favorecía la etapa de explotación, provocando una convergencia más rápida de las partículas. Un comportamiento deseado de las partículas es que realicen exploración en las generaciones iniciales y que conforme el algoritmo avance estas vayan adoptando un comportamiento explotativo. Este comportamiento deseado, en conjunto con los estudios de Suganthan, inspiraron el uso de una de las primeras topologías dinámicas. En su trabajo Suganthan propone tener topologías con poca conectividad en las generaciones iniciales de PSO y que conforme avance el algoritmo la conectividad de las partículas incremente, con el propósito de pasar de la etapa de exploración a la etapa de explotación gradualmente. Topologías como esta, en la que la estructura del vecindario cambia durante el desarrollo del algoritmo, son conocidas como **topologías dinámicas**. En muchos trabajos se han propuesto topologías de este tipo, con base en ideas como la distancia euclíadiana, conexiones aleatorias, agrupamiento de partículas, diagramas de Voronoi, entre muchas otras. En el capítulo 1 se mencionan algunos trabajos en los que se proponen topologías dinámicas. A continuación se tratan a detalle dos propuestas de PSO con topología dinámica, los cuales se utilizan para comparar con nuestro algoritmo.

Vecindario jerárquico dinámico basado en optimización por enjambre de partículas (Hierarchical D-LPSO)

Esta versión de PSO fue propuesta por Ghosh, Zafar, Das y Abraham en el 2011 [GZDA11]. En esta, las partículas son organizadas en un árbol que definirá su estructura de vecindario. El número de hijos de cada nodo en el árbol es fijo, salvo los nodos en el último nivel de la jerarquía, los cuales podrían tener menos. Cada partícula se comparará con su padre en la estructura y, en caso de ser mejor que este, intercambiarán su posición. Esto provocará que la partícula más apta esté en el nodo raíz. Este cambio en la posición de las partículas ayudará a preservar la diversidad en las partículas,

además, el hecho de que la mejor partícula se encuentre en la raíz del árbol de alguna forma influenciará en la mejora de la aptitud de las demás partículas. Para crear los vecindarios, se seleccionan aleatoriamente conjuntos de a lo más tres partículas del mismo nivel de la jerarquía. En el algoritmo 4 se presenta esta variante de PSO. En la figura 3.4 se ilustra la construcción de la topología del algoritmo Hierarchical D-LPSO.

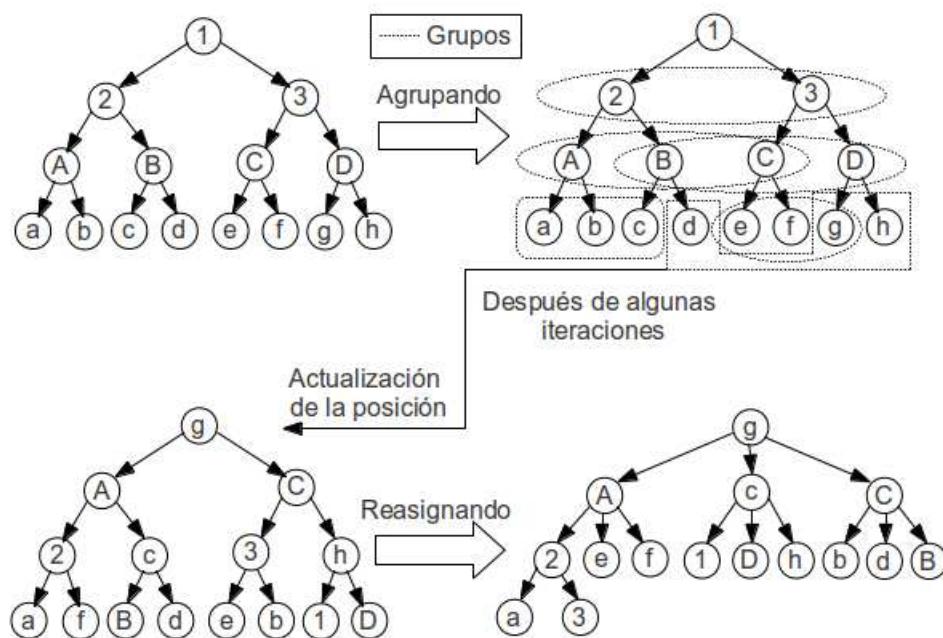


Figura 3.4: Construcción de la topología en Hierarchical D-LPSO

Algoritmo 4 Algoritmo Hierarchical D-LPSO

Entrada: La función a optimizar, el espacio de búsqueda, c_1 , c_2 , número de partículas que integrarán la población, criterios de paro, iteraciones máximas permitidas T .

Salida: La aproximación del mínimo obtenido.

- 1: Inicialización aleatoria de la posición y la velocidad de las partículas dentro del espacio de búsqueda.
 - 2: $n \leftarrow 2$
 - 3: $t \leftarrow 1$
 - 4: **mientras** no se cumpla algún criterio de paro **hacer**
 - 5: **si** $mod(t, T/5) = 0$ **entonces**
 - 6: $n \leftarrow n + 1$
 - 7: **fin si**
 - 8: La población es dividida en niveles. El primer nivel contiene una partícula, el siguiente nivel contiene, a lo más, n elementos. El N -ésimo nivel puede contener un máximo de n^N partículas.
 - 9: Cada partícula es asignada aleatoriamente a un parente del nivel previo, excepto la partícula del primer nivel.
 - 10: Evaluar la función de aptitud de cada partícula.
 - 11: Aplicar D-LPSO a cada nivel, excepto al primero.
 - 12: Comparar la aptitud de cada partícula con la de su parente. Si la aptitud del hijo es mejor que la del parente intercambiar la posición.
 - 13: Actualizar los vectores \vec{p}_i y \vec{l}_i de cada partícula .
 - 14: Actualizar la velocidad y posición de la partícula i usando las ecuaciones 3.5 y 3.2.
 - 15: $t \leftarrow t + 1$
 - 16: **fin mientras**
-

Optimización por enjambre de partículas con incremento en la conectividad de la topología (PSO-ITC)

Esta modificación de PSO fue propuesta por Wei Hong Lim y Nor Ashidi Mat Isa en el 2014 [LI14]. En esta propuesta, la conectividad de la topología aumenta dinámicamente conforme el proceso de búsqueda avanza, con el propósito de tener un mejor control entre exploración y la explotación. Para lograr esto, cada partícula es conectada con un vecino seleccionado aleatoriamente. Conforme el proceso de búsqueda avanza, la conectividad de la partícula se incrementa hasta terminar con un vecindario todos conectados. Matemáticamente, la conectividad de cada partícula es incrementada utilizando la ecuación 3.6.

$$TC_i = \lfloor TC_{min} + TC_{max}[(k - 1)/(FE_{max} - 1)] \rfloor \quad (3.6)$$

En donde TC_i es la conectividad actual de la partícula i ; TC_{min} y TC_{max} representa la conectividad mínima y máxima, respectivamente, $TC_{min} = 1$, $TC_{max} = S - 1$ y S es el número de partículas; k es la actual evaluación de función (FEs) consumida por el algoritmo y FE_{max} es el número máximo de evaluaciones de función permitidas. Se debe observar que al utilizar la ecuación 3.6 no se está yendo de TC_{min} a TC_{max} . Esto se debe a que en la última evaluación de función el valor de TC_i será $TC_{min} + TC_{max}$, por lo que TC_{max} no puede ser visto como una conectividad máxima a la que se pretende llegar.

En cada iteración el TC_i de cada partícula es incrementado en ΔTC_i , lo que implica seleccionar ΔTC_i nuevos vecinos de la población. Es importante mencionar que las conexiones entre los vecinos es de forma unidireccional, lo que significa que si la partícula i es vecina de la partícula j , no necesariamente la j es vecina de la i . Para evitar estancamiento en el enjambre, se agrega un mecanismo para cambiar los vecinos existentes. Si la partícula i no ayuda a mejorar la aptitud de la mejor experiencia del enjambre durante z iteraciones entonces el mecanismo es realizado, reasignando TC_i nuevos vecinos aleatoriamente. Además, para ayudar a la partícula g a escapar del óptimo local se perturba la dimensión d de esta utilizando la ecuación 3.7.

$$P_{g,d}^{per} = r_3 P_{g,d} + (1 - r_3)(P_{x,d} - P_{y,d}) \quad (3.7)$$

En donde $p_{g,d}^{per}$ es la dimensión d perturbada de la partícula g , r_3 es un número aleatorio uniformemente distribuido en el rango $[0, 1]$, $P_{x,d}$ y $P_{y,d}$ son las mejores posiciones personales de dos partículas seleccionadas aleatoriamente de la población. La nueva partícula obtenida con la perturbación sólo reemplazará a la partícula actual si tiene una mejor aptitud que esta. El proceso para la construcción de la topología se muestra en el algoritmo 5. En la figura 3.5 se ilustra el proceso de construcción de la topología.

Además de las modificaciones en la topología, se propone un entorno de aprendizaje. Este entorno generará un ejemplar cognitivo y un ejemplar social. Como se puede observar en la ecuación 3.5, el término $\vec{p}_i^t - \vec{x}_i^t$ es el componente cognitivo, mientras que el término $\vec{l}_i^t - \vec{x}_i^t$ es el componente social y este siempre tiene un valor de aptitud menor o igual que el componente cognitivo (para el caso de la minimización). Con base en este hecho se generan el ejemplar cognitivo $c_{exp,i}$ y el ejemplar social $s_{exp,i}$ de la partícula i . Para lograr esto, los vecinos de i (inclusive esta) se ordenan conforme a su aptitud (de mejor a peor), y aquellos que estén dentro del primer cuartil ($neighbor_upper_i$) se utilizarán para obtener $s_{exp,i}$, mientras que con los

Algoritmo 5 Módulo ITC. PSO-ITC.

Entrada: Contador de fallas (fc_i), posición de la mejor partícula del enjambre (P_g), aptitud de la mejor partícula del enjambre ($f(P_g)$), evaluaciones de función consumidas (k), conectividad de la topología de la partícula i (TC_i), vecindario de la partícula i ($neighbor_i$).

Salida: Ejemplares cognitivo y social ($c_{exp,i}$ y $s_{exp,i}$). Actualización de P_g , $f(P_g)$, TC_i , $neighbor_i$ y k .

```

1:  $TC_{old} \leftarrow TC_i$ .
2: Calcular el nuevo valor de  $TC_i$  usando la ecuación 3.6.
3: si  $TC_i \neq TC_{old}$  entonces
4:   /* Incrementar la conectividad de la topología de la partícula */
5:    $\Delta TC_i \leftarrow TC_i - TC_{old}$ .
6:   Seleccionar aleatoriamente  $\Delta TC_i$  miembros de la población y actualizar  $neighbor_i$ .
7:   Obtener  $c_{exp,i}$  y  $s_{exp,i}$  utilizando el algoritmo 6.
8:   Obtener el valor de aptitud de  $c_{exp,i}$  y  $s_{exp,i}$ .
9:   Actualizar  $P_g$  y  $f(P_g)$  si  $c_{exp,i}$  o  $s_{exp,i}$  tienen mejor aptitud.
10:   $k \leftarrow k + 2$ .
11: si no
12:  si  $fc_i > z$  entonces
13:    /* Activar el mecanismo de cambio de vecindario */
14:    Seleccionar aleatoriamente  $TC_i$  partículas como nuevos vecinos y actualizar  $neighbor_i$ .
15:    Realizar la perturbación en  $P_g$  usando la ecuación 3.7.
16:    Actualizar  $P_g$  y  $f(P_g)$  si  $P_g^{per}$  tiene una mejor aptitud.
17:     $k \leftarrow k + 1$ .
18:    Obtener  $c_{exp,i}$  y  $s_{exp,i}$  utilizando el algoritmo 6.
19:    Obtener el valor de aptitud de  $c_{exp,i}$  y  $s_{exp,i}$ .
20:    Actualizar  $P_g$  y  $f(P_g)$  si  $c_{exp,i}$  o  $s_{exp,i}$  tienen mejor aptitud.
21:     $k \leftarrow k + 2$ .
22:  fin si
23: fin si
```

vecinos restantes ($neighbor_lower_i$) se obtendrá $c_{exp,i}$. Para obtenerlos se aplicará una selección por ruleta a $neighbor_upper_i$ y a $neighbor_lower_i$, respectivamente, en donde a cada elemento de estos conjuntos se le asigna un peso W_k utilizando la ecuación 3.8.

$$W_k = \frac{f_{max} - f(P_k)}{f_{max} - f_{min}}, \forall k \in [1, K] \quad (3.8)$$

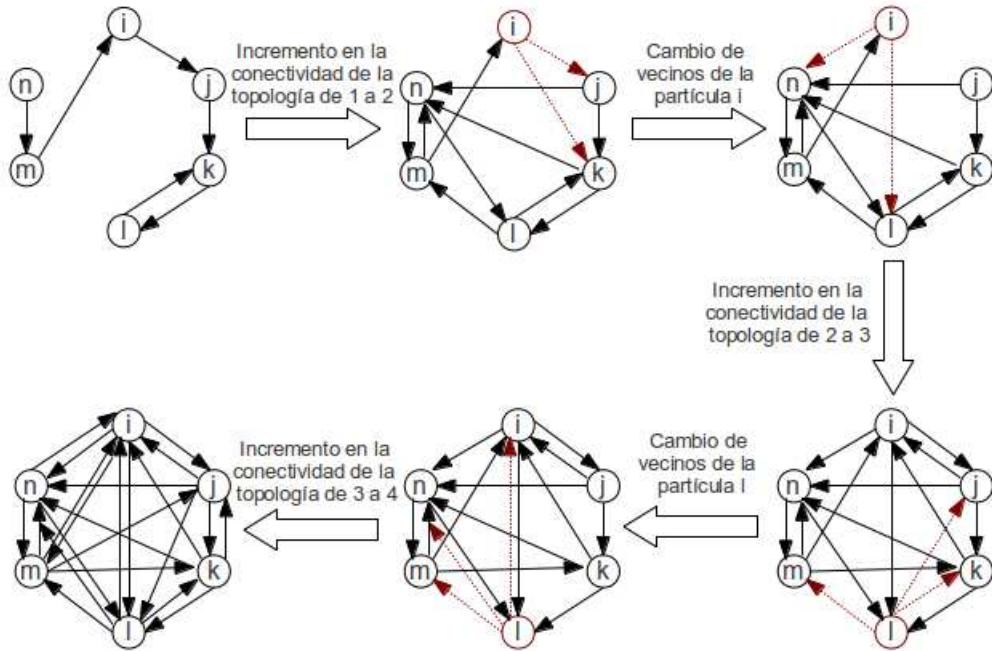


Figura 3.5: Construcción de la topología en PSO-ITC

En donde f_{max} y f_{min} son el peor y el mejor valor de aptitud de los elementos de $neighbor_upper_i$ y $neighbor_lower_i$, respectivamente, y K representa el tamaño de cada uno de estos. El procedimiento para generar los ejemplares se presenta en el algoritmo 6.

También se propone un nuevo mecanismo para actualizar la velocidad. Esta se actualizará utilizando el ejemplar cognitivo $c_{exp,i}$ y el mejor personal best del enjambre (P_g). Debido a que el ejemplar $c_{exp,i}$ es seleccionado utilizando un mecanismo probabilístico pueden ocurrir dos casos. El primero es que el ejemplar cognitivo tenga un mejor valor de aptitud que la partícula i . Para este caso, es conveniente que la partícula i sea atraída por $c_{exp,i}$ con el objetivo de mejorar el valor de aptitud de esta. El segundo caso es que la partícula i tenga un mejor valor de aptitud que $c_{exp,i}$. Para este caso no es conveniente guiar la partícula i hacia $c_{exp,i}$, por lo que se repele esta hacia zonas no exploradas del espacio de búsqueda. La velocidad de la partícula i es actualizada utilizando la ecuación 3.9.

$$V_i = \begin{cases} \omega V_i + c_1 r_4 (c_{exp,i} - X_i) + c_2 r_5 (P_g - X_i), & f(c_{exp,i}) < f(P_g) \\ \omega V_i - c_1 r_6 (c_{exp,i} - X_i) + c_2 r_7 (P_g - X_i) & \text{de otra forma} \end{cases} \quad (3.9)$$

En donde r_4, r_5, r_6 y r_7 son números aleatorios uniformemente distribui-

Algoritmo 6 Algoritmo para generar ejemplares. PSO-ITC.

Entrada: Vecindario de la partícula i ($neighbor_i$).

Salida: Ejemplares cognitivo y social ($c_{exp,i}$ y $s_{exp,i}$).

- 1: Ordenar los elementos de $neighbor_i$ conforme a su aptitud personal.
- 2: Asignar los miembros con aptitud personal dentro del primer cuartil a $neighbor_upper_i$.
- 3: Asignar los miembros restantes a $neighbor_lower_i$.
- 4: /* Generar $s_{exp,i}$ */
- 5: **para** cada miembro k en $neighbor_upper_i$ **hacer**
- 6: Calcular W_k usando la ecuación 3.8.
- 7: **fin para**
- 8: Seleccionar aleatoriamente una dimensión d_r .
- 9: **para** cada dimensión d **hacer**
- 10: **si** $d \neq d_r$ **entonces**
- 11: Realizar selección por ruleta basada en el W_k de cada miembro de $neighbor_upper_i$.
- 12: $s_{exp,i}(d) \leftarrow$ el d -ésimo componente del elemento seleccionado de $neighbor_upper_i$.
- 13: **si no**
- 14: Seleccionar aleatoriamente un miembro de $neighbor_upper_i$.
- 15: $s_{exp,i}(d_r) \leftarrow$ el d_r -ésimo componente del elemento seleccionado de $neighbor_upper_i$.
- 16: **fin si**
- 17: **fin para**
- 18: /* Generar $c_{exp,i}$ */
- 19: **para** cada miembro k en $neighbor_lower_i$ **hacer**
- 20: Calcular W_k usando la ecuación 3.8.
- 21: **fin para**
- 22: **para** cada dimensión d **hacer**
- 23: **si** $d \neq d_r$ **entonces**
- 24: Realizar selección por ruleta basada en el W_k de cada miembro de $neighbor_lower_i$.
- 25: $c_{exp,i}(d) \leftarrow$ el d -ésimo componente del elemento seleccionado de $neighbor_lower_i$.
- 26: **si no**
- 27: Seleccionar aleatoriamente un miembro de $neighbor_lower_i$.
- 28: $c_{exp,i}(d_r) \leftarrow$ el d_r -ésimo componente del elemento seleccionado de $neighbor_lower_i$.
- 29: **fin si**
- 30: **fin para**

dos en el intervalo $[0, 1]$. Si al actualizar la posición de la partícula i se obtiene que esta mejoró su valor de aptitud, entonces se utiliza una estrategia de aprendizaje elitista, con la finalidad de extraer información útil de la partícula i para tratar de mejorar la aptitud de la partícula g . Esto se hará sustituyendo los valores de cada dimensión de la partícula i en la dimensión correspondiente de la partícula g y manteniendo este cambio sólo si este implicó una mejora en la aptitud de g . En el algoritmo 7 se muestra el procedimiento para realizar esto.

Algoritmo 7 Algoritmo de la estrategia de aprendizaje basada en elitismo (EBLS). PSO-ITC.

Entrada: Posición del personal best de la partícula i (P_i), posición del global best (P_g), aptitud del global best ($f(P_g)$), evaluaciones de función consumidas (k).

Salida: Actualización de P_g , $f(P_g)$ y k .

```

1: para cada dimensión  $d$  hacer
2:    $P_{g,temp} \leftarrow P_g$  y  $f(P_{g,temp}) \leftarrow f(P_g)$ .
3:    $P_{g,temp}(d) \leftarrow P_i(d)$ .
4:   Obtener la función de aptitud de  $P_{g,temp}$ .
5:   si  $f(P_{g,temp}) < f(P_g)$  entonces
6:      $P_g(d) \leftarrow P_{g,temp}(d)$ .
7:   fin si
8:    $k \leftarrow k + 1$ .
9: fin para
```

Ahora bien, la velocidad que se utiliza no asegura que la aptitud de la partícula i mejorará, por lo que para manejar este problema se propone un operador de vecindario, el cual se activa cuando esto ocurre. En este operador, los ejemplares $c_{exp,i}$ y $s_{exp,i}$ se excluyen del proceso y los ejemplares generados por las demás partículas se almacenan en los arreglos $c_{candidate,i} = [c_{exp,1}, c_{exp,2}, \dots, c_{exp,i-1}, c_{exp,i+1}, \dots, c_{exp,S}]$ y $s_{candidate,i} = [s_{exp,1}, s_{exp,2}, \dots, s_{exp,i-1}, s_{exp,i+1}, \dots, s_{exp,S}]$, en donde S es el número de partículas. De estos arreglos, se seleccionan dos partículas guías ($c_{guide,i}$ y $s_{guide,i}$) utilizando una selección por ruleta. De estas partículas guías se obtiene un ejemplar $o_{exp,i}$ seleccionando aleatoriamente para cada dimensión un elemento de alguna de las dos guías. El procedimiento para obtener este ejemplar se muestra en el algoritmo 8.

De la misma forma que el ejemplar $c_{exp,i}$, la aptitud de $o_{exp,i}$ podría ser peor que la de la partícula i , entonces se utiliza una estrategia similar para atraer o repeler a esta hacia $o_{exp,i}$, según convenga. Específicamente cada partícula ajusta su P_i utilizando la ecuación 3.10.

Algoritmo 8 Generar ejemplar (Operador NS). PSO-ITC.

Entrada: Todos los ejemplares generados por la población (c_{exp_all} y s_{exp_all}), índice de la partícula i , posición del global best (P_g), aptitud del global best ($f(P_g)$), evaluaciones de función consumidas (k).

Salida: Ejemplar $o_{exp,i}$, valor actualizado de P_g , $f(P_g)$ y k .

- 1: Excluir $c_{exp,i}$ y $s_{exp,i}$ de c_{exp_all} y s_{exp_all} respectivamente.
 - 2: Almacenar los ejemplares no excluidos en $c_{candidate,i}$ y $s_{candidate,i}$ respectivamente.
 - 3: /* Seleccionar $s_{guide,i}$ */
 - 4: **para** cada elemento k en $s_{candidate,i}$ **hacer**
 - 5: Calcular W_k usando la ecuación 3.8.
 - 6: **fin para**
 - 7: Realizar selección por ruleta basada en el W_k de cada miembro de $s_{candidate,i}$ para seleccionar $s_{guide,i}$.
 - 8: /* Seleccionar $c_{guide,i}$ */
 - 9: **para** cada elemento k en $c_{candidate,i}$ **hacer**
 - 10: Calcular W_k usando la ecuación 3.8.
 - 11: **fin para**
 - 12: Realizar selección por ruleta basada en el W_k de cada miembro de $c_{candidate,i}$ para seleccionar $c_{guide,i}$.
 - 13: /* Obtener $o_{exp,i}$ */
 - 14: **para** cada dimensión d **hacer**
 - 15: **si** $rand < 0.5$ **entonces**
 - 16: $o_{exp,i}(d) \leftarrow s_{guide,i}(d)$.
 - 17: **si no**
 - 18: $o_{exp,i}(d) \leftarrow c_{guide,i}(d)$.
 - 19: **fin si**
 - 20: **fin para**
 - 21: Obtener la función de aptitud de $o_{exp,i}$.
 - 22: Actualizar P_g y $f(P_g)$ si $o_{exp,i}$ tiene mejor aptitud.
 - 23: $k \leftarrow k + 1$.
-

$$P_{i,temp} = \begin{cases} P_i + cr_8(o_{exp,i} - P_i), & f(o_{exp,i}) < f(P_i) \\ P_i - cr_9(o_{exp,i} - P_i) & \text{de otra forma} \end{cases} \quad (3.10)$$

En donde $P_{i,temp}$ es la experiencia cognitiva ajustada de la partícula i y r_8 y r_9 son dos números aleatorios uniformemente distribuidos en el intervalo $[0, 1]$. La aptitud de $P_{i,temp}$ se comparará con la de P_i y P_g y en caso de que sea mejor $P_{i,temp}$ reemplazará a P_i y a P_g . En caso de que esto suceda se

aplicará el algoritmo 7 para tratar de mejorar P_g utilizando la información de $P_{i,temp}$. En el algoritmo 9 se presenta como realizar este procedimiento.

Una vez descrita cada parte de PSO-ITC, este se presenta en el algoritmo 10.

Algoritmo 9 Operador NS. PSO-ITC.

Entrada: Todos los ejemplares generados por la población (c_{exp_all} y s_{exp_all}), índice de la partícula i , posición del personal best de la partícula i (P_i), posición del global best (P_g), aptitud del global best ($f(P_g)$), evaluaciones de función consumidas (k).

Salida: Actualización de P_g , $f(P_g)$ y k .

- 1: Obtener $o_{exp,i}$ utilizando el algoritmo 8.
 - 2: Calcular el $P_{i,temp}$ de la partícula i usando la ecuación 3.10.
 - 3: Obtener la función de aptitud de $P_{i,temp}$.
 - 4: $k \leftarrow k + 1$.
 - 5: $previous_position_i \leftarrow P_i$ y $previous_fitness_i \leftarrow f(P_i)$.
 - 6: Actualizar P_i , P_g , $f(P_i)$ y $f(P_g)$.
 - 7: **si** $f(P_i) < previous_fitness_i$ y $P_g \neq P_i$ **entonces**
 - 8: Aplicar el algoritmo 7.
 - 9: **fin si**
-

Algoritmo 10 Algoritmo PSO-ITC.

Entrada: Tamaño de la población (S), dimensionalidad del espacio problema (D), función objetivo (F), dominio de inicialización (RG), tolerancia (ϵ) y el número máximo de evaluaciones de función permitidas (FE_{max}).

Salida: La aproximación del mínimo obtenido.

```

1: Generar aleatoriamente el enjambre inicial y obtener los parámetros de
   cada partícula.
2:  $k \leftarrow 0$ .
3: para cada partícula  $i$  hacer
4:    $TC_i \leftarrow 1$  y  $fc_i \leftarrow 0$ .
5:   Inicializar  $neighbor_i$  seleccionando aleatoriamente una partícula de la
      población restante.
6: fin para
7: mientras  $k < FE_{max}$  hacer
8:   para cada partícula  $i$  hacer
9:     Realizar el algoritmo 5.
10:    Actualizar la velocidad y posición de la partícula  $i$  utilizando las
        ecuaciones 3.9 y 3.2 respectivamente.
11:    Obtener la función de aptitud de la partícula  $i$ .
12:     $k \leftarrow k + 1$ .
13:     $previous\_position_i \leftarrow P_i$  y  $previous\_fitness_i \leftarrow f(P_i)$ .
14:    Actualizar  $P_i$ ,  $P_g$ ,  $f(P_i)$  y  $f(P_g)$ .
15:    si  $P_g$  es mejorado entonces
16:       $fc_i \leftarrow 0$ .
17:    si no
18:       $fc_i \leftarrow fc_i + 1$ .
19:    fin si
20:    si  $f(P_i) < previous\_fitness_i$  y  $P_g \neq P_i$  entonces
21:      Realizar el algoritmo 7.
22:    si no
23:      Realizar el algoritmo 9.
24:    fin si
25:  fin para
26: fin mientras
```

Capítulo 4

Información mutua y modelos gráficos

En el presente capítulo se presentan los conceptos básicos que se utilizarán para la construcción de la topología de PSO que maximiza la información mutua entre las partículas. Primero, se define la información mutua y como se calcula esta para variables aleatorias discretas y para variables aleatorias continuas. A continuación se abordan los conceptos básicos de modelos gráficos, principalmente de grafos no dirigidos. Finalmente, se presentan las dos estructuras que maximizan la información mutua que se utilizan para construir los vecindarios de las partículas: la cadena MIMIC y el árbol Chow-Liu.

4.1 Información mutua

Antes de dar una idea general de la información mutua, comenzaremos por definir qué es un experimento (definición 4.1.1) y qué es el espacio muestral (definición 4.1.2), las cuales son necesarias para definir qué es una variable aleatoria (definición 4.1.3). Esto debido a que la información mutua es una función que recibe como argumento variables aleatorias.

Definición 4.1.1 [Sti99] *Una actividad o procedimiento que da lugar a un conjunto bien definido de resultados es llamado **experimento**.*

Definición 4.1.2 [Sti99] *El conjunto de todos los posibles resultados de un experimento es llamado **espacio muestral** y se denota como Ω .*

Definición 4.1.3 [Sti99] *Una **variable aleatoria** es una función que mapea los elementos en el espacio muestral Ω a valores reales.*

Una vez definido qué es una variable aleatoria, procederemos a hablar sobre la información mutua. La información mutua es una medida no lineal

que se utiliza para medir correlaciones lineales y no lineales. Esta es una medida natural de la dependencia entre dos variables aleatorias y que sirve de igual forma para obtener la estructura de dependencia de más de dos variables aleatorias. La información mutua entre las variables aleatorias X y Y se representa como $I(X, Y)$ y mide la reducción de la incertidumbre en X cuando Y es conocida. Esta fue introducida por Shannon en 1948 [Sha48] y tiene las siguientes propiedades:

1. Es simétrica. $I(X, Y) = I(Y, X)$
2. Es no negativa. $I(X, Y) \geq 0$

Dado que las variables aleatorias pueden ser discretas o continuas, la información mutua se puede calcular para cualquiera de estos dos casos. En las secciones 4.1.1 y 4.1.2 se explica, respectivamente, como se realiza esto para cada caso.

4.1.1 Variables aleatorias discretas

Cuando tenemos una variable aleatoria que toma valores de un dominio discreto decimos que esta es una variable aleatoria discreta. Para este caso, consideremos la divergencia Kullback-Leibler para distribuciones de probabilidad discretas P y Q , la cual está dada por la ecuación 4.1.

$$\sum_i P(i) \log \frac{P(i)}{Q(i)} \quad (4.1)$$

En donde i son todos los posibles valores que puede tomar las distribuciones P y Q . Ahora, recordemos que dos variables aleatorias X y Y son independientes si se cumple que $P(X, Y) = P(X)P(Y)$. Considerando que la divergencia Kullback-Leibler mide la similitud que existe entre dos distribuciones de probabilidad, suena lógico pensar que si obtenemos el valor de esta divergencia entre la distribución conjunta $P(X, Y)$ y la multiplicación de las distribuciones marginales $P(X)P(Y)$, de cierto modo estamos obteniendo una medida de dependencia entre las variables aleatorias. Esta idea es tomada para obtener la información mutua entre las variables aleatorias X y Y . Tomemos como la distribución P a $P(X, Y)$ y como la distribución Q a $P(X)P(Y)$ y sustituimos en la ecuación 4.1, obteniendo así la ecuación 4.2.

$$I(X, Y) = \sum_{x \in \Omega_X} \sum_{y \in \Omega_Y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \quad (4.2)$$

La ecuación 4.2 es la información mutua entre las variables aleatorias discretas X y Y . Si tomamos la información mutua precisamente como la información que comparten un par de variables aleatorias, es obvio asumir que si dos variables aleatorias comparten mucha información entonces estas son dependientes, y por el contrario, si no comparten información entonces son independientes. Entonces, valores altos de información mutua implica dependencia entre variables aleatorias, mientras que valores bajos de esta implica independencia entre estas variables. Esto queda aun más claro si consideramos que las variables aleatorias X y Y son independientes. En este caso, sabemos que $P(X, Y) = P(X)P(Y)$ y entonces el término $\log\{P(X, Y)/[P(X)P(Y)]\}$ de la ecuación 4.2 es 0 y, por lo tanto, la información mutua es 0. Es importante mencionar que entonces la información mutua es una medida que puede ser utilizada para obtener modelos gráficos que representan dependencias entre variables.

Por último, podría haber ocasiones en las que el cálculo de la información mutua conlleve a obtener el valor de 0 $\log(0)$, el cual está indefinido. Para lidiar con este caso, se toma el valor del $\lim_{x \rightarrow 0} x \log(x) = 0$ en lugar del cálculo indefinido del logaritmo.

4.1.2 Variables aleatorias continuas

Ahora consideremos el caso de una variable aleatoria continua, es decir, de una variable aleatoria que toma valores de un dominio continuo. En este caso, para obtener la divergencia Kullback-Leibler y la información mutua de dos variables aleatorias se utilizan las ecuaciones 4.1 y 4.2, respectivamente, pero cambiando las sumas por integrales en el dominio de cada variable. El problema de aplicar directamente estas ecuaciones para las variables aleatorias continuas es que en la mayoría de las ocasiones no existe una función primitiva en la que se evite la integral. Entonces, para resolver esta se tienen que realizar aproximaciones numéricas que involucran un mayor costo computacional, lo cual para algunas aplicaciones es aceptable, pero para muchas otras no, por lo que en estos casos es deseable tener una ecuación directa para obtener la información mutua de dos variables aleatorias con dominio continuo. Una forma de lograr esto es asumiendo que $X \sim \mathcal{N}(\mu_x, \sigma_x)$ y $Y \sim \mathcal{N}(\mu_y, \sigma_y)$, así como $(X, Y) \sim \mathcal{N}(\mu, K)$, en donde $\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$ es el vector de medias y $K = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$ es la matriz de covarianzas. Antes que nada, definamos la entropía de una variable aleatoria continua X como se muestra en la ecuación 4.3 y la entropía conjunta de las variables aleatorias

(X, Y) como se muestra en la ecuación 4.4.

$$H(X) = - \int_{\Omega_X} P(x) \log P(x) dx \quad (4.3)$$

$$H(X, Y) = - \int_{\Omega_X} \int_{\Omega_Y} P(x, y) \log P(x, y) dx dy \quad (4.4)$$

Ahora bien, utilizando la ecuación 4.2, cambiando las sumas por integrales y utilizando propiedades de logaritmos tenemos que:

$$I(X, Y) = \int_{\Omega_X} \int_{\Omega_Y} P(x, y) [\log P(x, y) - \log P(x) - \log P(y)] dx dy$$

Separando los términos obtenemos:

$$\begin{aligned} I(X, Y) &= \int_{\Omega_X} \int_{\Omega_Y} P(x, y) \log P(x, y) dx dy - \int_{\Omega_X} \log P(x) dx \underbrace{\int_{\Omega_Y} P(x, y) dy}_{P(x)} \\ &\quad - \int_{\Omega_Y} \log P(y) dy \underbrace{\int_{\Omega_X} P(x, y) dx}_{P(y)} \end{aligned}$$

De donde después de marginalizar y sustituir las ecuaciones 4.3 y 4.4, se obtiene la ecuación 4.5, que es una expresión de la información mutua basada en la entropía.

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \quad (4.5)$$

Una vez definida la ecuación 4.5, procederemos a obtener la entropía de la variable aleatoria $X \sim \mathcal{N}(\mu_x, \sigma_x)$, cuya función de densidad está dada por la ecuación 4.6.

$$P(x) = \frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x-\mu_x}{\sigma_x} \right)^2} \quad (4.6)$$

Sustituyendo la ecuación 4.6 en la ecuación 4.3 obtenemos:

$$H(X) = - \int_{\Omega_X} P(x) \log \left[\frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x-\mu_x}{\sigma_x} \right)^2} \right] dx$$

Aplicando las propiedades de los logaritmos, desarrollando y separando términos se obtiene:

$$\begin{aligned}
H(X) = & \log \sqrt{2\pi\sigma_x^2} \underbrace{\int_{\Omega_X} P(x)dx}_{1} + \frac{1}{2\sigma_x^2} \underbrace{\int_{\Omega_X} x^2 P(x)dx}_{E(X^2)} \\
& - \frac{\mu_x}{\sigma_x^2} \underbrace{\int_{\Omega_X} xP(x)dx}_{E(X)} + \frac{\mu_x^2}{2\sigma_x^2} \underbrace{\int_{\Omega_X} P(x)dx}_{1}
\end{aligned}$$

Considerando que para una distribución normal el primer y segundo momento son $E(X) = \mu_x$ y $E(X^2) = \sigma_x^2 + \mu_x^2$, respectivamente, procederemos a sustituir estos valores, con lo que se obtiene:

$$H(X) = \log \sqrt{2\pi\sigma_x^2} + \frac{1}{2} + \underbrace{\frac{1}{\sigma_x^2} \left(\frac{\mu_x^2}{2} - \mu_x^2 + \frac{\mu_x^2}{2} \right)}_0$$

Que después de aplicar las propiedades de los logaritmos, se obtiene la ecuación 4.7, que es la entropía de la variable aleatoria $X \sim \mathcal{N}(\mu_x, \sigma_x)$ (o Y , si cambiamos μ_x y σ_x por μ_y y σ_y , respectivamente).

$$H(X) = \frac{1}{2} \log (2\pi e \sigma_x^2) \quad (4.7)$$

Consideremos un conjunto de variables aleatorias $\vec{X} = \{X_1, X_2, \dots, X_n\}$ que tienen una distribución normal multivariada con media μ y matriz de covarianza K , cuya función de densidad está dada por la ecuación 4.8.

$$P(\vec{x}) = -\frac{1}{(\sqrt{2\pi})^n |K|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{x}-\mu)^T K^{-1}(\vec{x}-\mu)} \quad (4.8)$$

En la ecuación 4.9 se muestra la entropía del conjunto de variables aleatorias $\vec{X} = \{X_1, X_2, \dots, X_n\}$.

$$H(\vec{X}) = - \int_{\Omega_{\vec{X}}} P(\vec{x}) \log P(\vec{x}) d\vec{x} \quad (4.9)$$

Sustituyendo, la ecuación 4.8 en la ecuación 4.9 y aplicando las propiedades de los logaritmos, obtenemos:

$$H(\vec{X}) = \frac{1}{2} \int_{\Omega_{\vec{X}}} P(\vec{x}) [(\vec{x} - \mu)^T K^{-1}(\vec{x} - \mu)] d\vec{x} + \frac{1}{2} \log [(2\pi)^n |K|] \underbrace{\int_{\Omega_{\vec{X}}} P(\vec{x}) d\vec{x}}_1$$

Que es lo mismo que:

$$H(\vec{X}) = \frac{1}{2}E \left[(\vec{X} - \mu)^T K^{-1} (\vec{X} - \mu) \right] + \frac{1}{2} \log [(2\pi)^n |K|]$$

Si consideramos que $A^T = (\vec{X} - \mu)^T K^{-1}$, entonces:

$$A_i^T = \sum_{j=1}^n (X_j - \mu_j)(K^{-1})_{ji}$$

y

$$(\vec{X} - \mu)^T K^{-1} (\vec{X} - \mu) = \sum_{i=1}^n \sum_{j=1}^n (X_j - \mu_j)(X_i - \mu_i)(K^{-1})_{ji}$$

Sustituyendo esto se obtiene:

$$H(\vec{X}) = \frac{1}{2}E \left[\sum_{i=1}^n \sum_{j=1}^n (X_j - \mu_j)(X_i - \mu_i)(K^{-1})_{ji} \right] + \frac{1}{2} \log [(2\pi)^n |K|]$$

Que es lo mismo que:

$$H(\vec{X}) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n E[(X_j - \mu_j)(X_i - \mu_i)](K^{-1})_{ji} + \frac{1}{2} \log [(2\pi)^n |K|]$$

Por definición, sabemos que $E[(X_i - \mu_i)(X_j - \mu_j)] = K_{ij}$ y entonces:

$$H(\vec{X}) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n K_{ji}(K^{-1})_{ji} + \frac{1}{2} \log [(2\pi)^n |K|]$$

En donde:

$$K_{ji}(K^{-1})_{ji} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Por lo tanto:

$$H(\vec{X}) = \frac{n}{2} + \frac{1}{2} \log [(2\pi)^n |K|]$$

Al aplicar las propiedades de los logaritmos se obtiene la ecuación 4.10, que es la entropía del conjunto de variables aleatorias $(X_1, X_2, \dots, X_n) \sim \mathcal{N}(\mu, K)$.

$$H(\vec{X}) = \frac{1}{2} \log[(2\pi e)^n |K|] \quad (4.10)$$

Para el caso particular de las variables aleatorias $(X, Y) \sim \mathcal{N}(\mu, K)$, con $\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$ y $K = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$, tenemos que $|K| = (1 - \rho^2)\sigma_x^2\sigma_y^2$ y, sustituyendo este valor en la ecuación 4.10, se obtiene:

$$H(X, Y) = \frac{1}{2} \log[(2\pi e)^2(1 - \rho^2)\sigma_x^2\sigma_y^2]$$

Sustituyendo este valor y $H(X) = \frac{1}{2} \log (2\pi e \sigma_x^2)$, $H(Y) = \frac{1}{2} \log (2\pi e \sigma_y^2)$ en la ecuación 4.5, obtenemos:

$$I(X, Y) = \frac{1}{2} \log (2\pi e \sigma_x^2) + \frac{1}{2} \log (2\pi e \sigma_y^2) - \frac{1}{2} \log[(2\pi e)^2(1 - \rho^2)\sigma_x^2\sigma_y^2]$$

Aplicando las propiedades de los logaritmos se obtiene:

$$\begin{aligned} I(X, Y) = & \frac{1}{2} \log (2\pi e) + \log \sigma_x + \frac{1}{2} \log (2\pi e) + \log \sigma_y \\ & - \log (2\pi e) - \frac{1}{2} \log (1 - \rho^2) - \log \sigma_x - \log \sigma_y \end{aligned}$$

De donde, reduciendo términos, se obtiene la ecuación 4.11, que es la información mutua entre las variables aleatorias X y Y , en donde $(X, Y) \sim \mathcal{N}(\mu, K)$.

$$I(X, Y) = -\frac{1}{2} \log[1 - \rho(X, Y)^2] \quad (4.11)$$

En donde $\rho(X, Y)$ es el coeficiente de correlación de Pearson, el cual se presenta en la ecuación 4.12.

$$\rho(X, Y) = \frac{\sigma(X, Y)}{\sigma(X)\sigma(Y)} \quad (4.12)$$

En donde $\sigma(X, Y)$ es la covarianza de (X, Y) , $\sigma(X)$ es la desviación estándar de X , $\sigma(Y)$ es la desviación estándar de Y y $\rho(X, Y) \in [0, 1]$.

Como se puede analizar en la ecuación 4.11, un valor de $\rho(X, Y)$ cercano a 1 indica una fuerte dependencia entre las variables X y Y , por lo que en este caso la información mutua debe ser alta. Se puede ver que, sustituyendo $\rho(X, Y) = 1$ en la ecuación 4.11, la información mutua tiende a infinito, por lo que esto se cumple. Contrario a esto, valores de $\rho(X, Y)$ cercanos a 0 implican independencia entre las variables aleatorias. Sustituyendo $\rho(X, Y) = 0$ en la

ecuación 4.11 se obtiene que $I(X, Y) = 0$. Con base en esto se puede observar que esta formulación de la información mutua también es válida para medir dependencias entre variables aleatorias.

4.2 Modelos gráficos

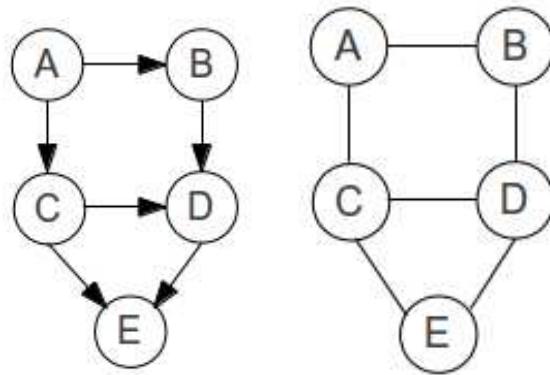
Una forma práctica de representar dependencias entre variables aleatorias es utilizando modelos gráficos. Estos integran teoría de grafos y teoría de probabilidad lo cual los hace una herramienta muy poderosa. Los modelos gráficos son ampliamente utilizados en varios campos, como los son el procesamiento de imágenes, teoría de control, modelos estadísticos entre muchos otros. Estos ayudan a explorar de forma sencilla una gran cantidad de modelos estadísticos, ya que se pueden observar visualmente relaciones de dependencia de muchas variables aleatorias. Con el paso del tiempo se han propuesto varios tipos de modelos gráficos y distintas formas de construirlos. Los modelos gráficos se construyen utilizando medidas que indican dependencia entre variables aleatorias, como lo es la información mutua.

4.2.1 Conceptos básicos

Debido a que para representar las relaciones entre las partículas se utilizaran modelos gráficos en los que se maximiza la información mutua, es necesario presentar a grandes rasgos algunos conceptos de teoría de grafos que se ocupan en la construcción de las topologías de PSO. Primeramente, en la definición 4.2.1 se menciona que es un grafo.

Definición 4.2.1 [Bar13] *Un grafo G consta de nodos (también llamados vértices) y aristas que unen a los nodos. Las aristas pueden ser dirigidas (que tienen alguna flecha en alguna dirección) o no dirigidas. Un grafo con todas las aristas dirigidas es llamado grafo dirigido, mientras que uno con todas las aristas no dirigidas es un grafo no dirigido.*

En la figura 4.1 se muestra un ejemplo de un grafo dirigido (figura 4.1a) y de un grafo no dirigido (figura 4.1b). Los círculos son los nodos, las flechas son aristas dirigidas y las líneas aristas no dirigidas. Una arista dirigida significa que solo se puede ir en una dirección, por ejemplo, en la figura 4.1a la arista de A a B implica que sólo se puede ir de A a B, mas no al revés. Por el contrario, una arista no dirigida implica que hay una conexión bidireccional, esto es, que se puede ir de A a B, así como de B a A. En la presente tesis sólo se utilizan grafos no dirigidos. En algunas ocasiones, las aristas de los



(a) Grafo dirigido (b) Grafo no dirigido

Figura 4.1: Ejemplo de un grafo dirigido y de un grafo no dirigido

grafos pueden tener “pesos”, esto es, se les puede asignar una cantidad que represente alguna medida entre los nodos. Un ejemplo de esto podría ser que el peso de la arista sea la distancia entre dos ciudades, en donde las ciudades son representadas por los nodos. En nuestro caso, las aristas se pesarán con la información mutua entre las variables aleatorias representadas por los nodos. En la definición 4.2.2 se presenta que es un vecindario.

Definición 4.2.2 [Bar13] Para un grafo no dirigido G , el **vecindario** de x , $ve(x)$ son aquellos nodos que están conectados a x .

Cabe mencionar que el concepto de vecindario en teoría de grafos es el mismo que se utiliza en PSO. Una de las estructuras que se utilizan en la topología propuesta son los árboles. Las características de los árboles es que estos no tienen lazos y son conectados. Los conceptos de lazo y de un grafo conectado se presentan en las definiciones 4.2.4 y 4.2.5. En la definición 4.2.6 se presenta que es un árbol.

Definición 4.2.3 [Bar13] Un **caminio** $A \mapsto B$ de un nodo A a un nodo B es una secuencia de nodos que conectan a A y a B . Esto es, un camino tiene la forma $A_0, A_1, \dots, A_{n-1}, A_n$, con $A_0 = A$ y $A_n = B$ y cada arista (A_{k-1}, A_k) , $k = 1, \dots, n$ existe en el grafo.

Definición 4.2.4 [Bar13] Un **lazo** es un camino que contiene más de dos nodos, sin importar la dirección, que empieza y termina en el mismo nodo.

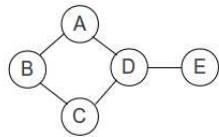
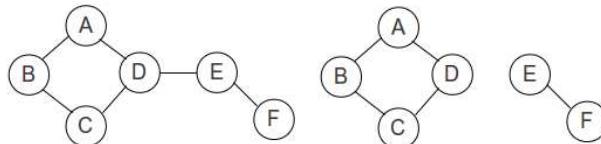


Figura 4.2: Ejemplo de un grafo que contiene un lazo



(a) Grafo conectado (b) Grafo no conectado

Figura 4.3: Ejemplo de un grafo conectado y de un grafo no conectado

En la figura 4.2 se muestra un grafo que contiene un lazo. En esta, el camino A-B-C-D-A, es un lazo ya que empieza y termina en el mismo nodo.

Definición 4.2.5 [Bar13] *Un grafo no dirigido es **conectado** si existe un camino entre cada par de nodos (en otras palabras si no hay nodos aislados).*

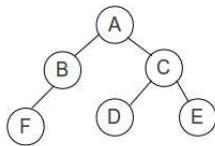
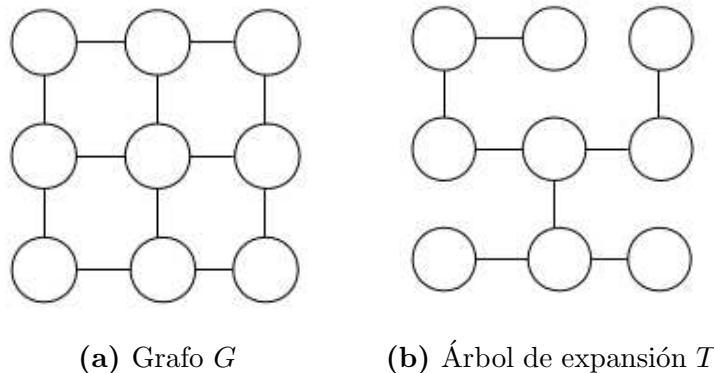
Para dejar claro el concepto de grafo conectado se muestra la figura 4.3. En la figura 4.3a, el grafo es conectado porque existe al menos un camino entre cualquier par de nodos. Por el contrario, el grafo de la figura 4.3b no es conectado ya que no existe camino de los nodos E y F hacia los nodos A, B, C y D.

Definición 4.2.6 [Bar13] *Un **árbol** es un grafo en el que sólo existe un camino de un nodo A a otro nodo B.*

En la figura 4.4 se muestra un ejemplo de árbol. Para obtener las estructuras que maximizan la información mutua se utilizan árboles de expansión, cuyo concepto se presenta en la definición 4.2.7.

Definición 4.2.7 [Bar13] *Un **árbol de expansión** de un grafo no dirigido G es un subconjunto de las aristas de G tales que forman un árbol que abarca todos los nodos de G .*

El árbol mostrado en la figura 4.5b es uno de los posibles árboles de expansión del grafo mostrado en la figura 4.5a. Es claro que se pueden

**Figura 4.4:** Ejemplo de un árbol**Figura 4.5:** Ejemplo de un árbol de expansión

generar muchos árboles de expansión de un grafo. Generalmente, cuando se tiene un grafo con aristas pesadas nos interesa encontrar el árbol de expansión con peso máximo (o en algunas ocasiones mínimo). El **árbol de expansión con peso máximo** de un grafo G es un árbol de expansión tal que la suma de todos los pesos en sus aristas es mayor que la suma de los pesos de las aristas de cualquier otro árbol de expansión que se pueda generar del grafo G . En el algoritmo 11 se presenta un algoritmo glotón para encontrar el árbol de expansión con peso máximo de un grafo G . Este algoritmo fue propuesto por Joseph Kruskal en [Kru56], por lo que es conocido como algoritmo de Kruskal.

Ahora bien, para nuestro caso nos interesa encontrar el árbol de expansión con máximo peso de un grafo completo con aristas pesadas con la información mutua entre las variables aleatorias representadas por los nodos. En la definición 4.2.8 se presenta que es un grafo completo. En la figura 4.6 se muestra un ejemplo de un grafo completo con seis nodos.

Definición 4.2.8 *Un grafo completo es un grafo en el que existe una arista entre cada par de nodos.*

Otro grafo utilizado en la construcción de las topologías de PSO son las cadenas. En la definición 4.2.9 se presenta que es una cadena.

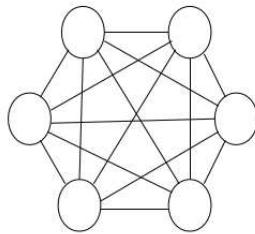


Figura 4.6: Ejemplo de un grafo completo

Algoritmo 11 Algoritmo de Kruskal

Entrada: Un grafo G con aristas pesadas con n nodos.

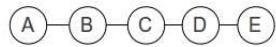
Salida: El árbol de expansión con peso máximo T correspondiente al grafo G .

- 1: Inicializar T como un grafo sin aristas.
 - 2: $aristas \leftarrow 0$
 - 3: Ordenar las aristas de G de mayor a menor conforme a su peso. A esta lista ordenada se le llamará *restantes*.
 - 4: **mientras** $aristas < n - 1$ **hacer**
 - 5: $mayor \leftarrow$ arista que se encuentre al inicio de *restantes*.
 - 6: Agregar *mayor* a T .
 - 7: **si** existe algún lazo en T **entonces**
 - 8: Eliminar *mayor* de T .
 - 9: **si no**
 - 10: $aristas \leftarrow aristas + 1$
 - 11: **fin si**
 - 12: Eliminar *mayor* de *restantes*.
 - 13: **fin mientras**
-

Definición 4.2.9 Una cadena es un grafo en el que los nodos son unidos de forma lineal. En otras palabras, si el grafo contiene los nodos A_1, A_2, \dots, A_n entonces las aristas del grafo son $\{A_1, A_2\}, \{A_2, A_3\}, \dots, \{A_{n-1}, A_n\}$.

La notación $\{A_i, A_j\}$ significa que existe una arista entre los nodos A_i y A_j . Como se puede observar, una cadena es también un árbol, pero debido a que es un modelo que se puede construir de una forma más simple es preferible tratarlo diferente a los árboles. En la figura 4.7 se presenta un ejemplo de una cadena.

Por último, existen muchas formas de representar un grafo para implementarlo computacionalmente. En la presente tesis se utilizan matrices de adyacencia para este fin. Una matriz de adyacencia A es una matriz en la

**Figura 4.7:** Ejemplo de una cadena

$$\begin{array}{ccccc} & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} \\ \text{A} & \left(\begin{array}{ccccc} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array} \right) \\ \text{B} & & & & & \\ \text{C} & & & & & \\ \text{D} & & & & & \\ \text{E} & & & & & \end{array}$$

Figura 4.8: Ejemplo de una matriz de adyacencia

que un valor de 1 en la posición A_{ij} indica que existe una arista que conecta a los nodos i y j . Opuesto a esto, $A_{ij} = 0$ significa que no existe arista entre los nodos i y j . En el caso de grafos no dirigidos, para indicar una arista entre los nodos i y j es necesario hacer $A_{ij} = A_{ji} = 1$. Esto implica que la matriz de adyacencia A para un grafo no dirigido es simétrica. En la figura 4.8 se muestra la matriz de adyacencia correspondiente al grafo mostrado en la figura 4.2.

4.2.2 Modelo de cadena (MIMIC)

Una vez descrito que es un modelo de cadena, nos interesa saber como construir este con base en las dependencias existentes entre un conjunto de variables aleatorias. Como se mencionó con anterioridad, la información mutua es una medida que nos indica dependencia entre variables aleatorias, por lo que puede ser utilizada para este propósito. Asumiendo que cada nodo en el modelo gráfico es una variable aleatoria y que las aristas entre estas serán pesadas con la información mutua entre ellas, se debe encontrar la cadena que involucre todos los nodos del grafo y que maximice la información mutua entre las variables. En 1996, Jeremy De Bonet, Charles Isbell y Paul Viola propusieron un algoritmo para encontrar el modelo de cadena que maximizaba la información mutua entre un conjunto de variables aleatorias [BJV96]. A este algoritmo lo llamaron MIMIC (Mutual Information Maximizing Input Clustering) y se muestra en el algoritmo 12.

Algoritmo 12 Algoritmo MIMIC

Entrada: Las muestras de tamaño N de las m variables aleatorias. $X_1 = (x_1^1, x_1^2, \dots, x_1^N), X_2 = (x_2^1, x_2^2, \dots, x_2^N), \dots, X_m = (x_m^1, x_m^2, \dots, x_m^N)$

Salida: El modelo de cadena T que maximiza la información mutua entre las m variables aleatorias.

- 1: Inicializar T como un grafo sin aristas.
 - 2: Obtener la información mutua entre cada par de variables aleatorias.
 - 3: Buscar el par de variables aleatorias con mayor información mutua y agregar una arista entre ellas.
 - 4: *restantes* \leftarrow las m variables aleatorias, excluyendo las dos que tuvieron la mayor información mutua.
 - 5: *extremos* \leftarrow las dos variables aleatorias que tuvieron la mayor información mutua.
 - 6: *aristas* $\leftarrow 1$
 - 7: **mientras** *aristas* $< m - 1$ **hacer**
 - 8: Encontrar la máxima información mutua entre una variable aleatoria de *restantes* y una variable aleatoria de *extremos*. A la variable aleatoria seleccionada de *restantes* le llamaremos r y a la variable aleatoria seleccionada de *extremos* le llamaremos e .
 - 9: Aregar una arista entre e y r en T .
 - 10: Aregar r a *extremos*.
 - 11: Eliminar r de *restantes*.
 - 12: Eliminar e de *extremos*.
 - 13: *aristas* \leftarrow *aristas* + 1
 - 14: **fin mientras**
-

4.2.3 Modelo de árbol (Chow-Liu)

El otro modelo que se utilizará para construir las topologías de PSO es el modelo de árbol. De igual forma que en el modelo de cadena, nos interesa encontrar un modelo de árbol que represente las relaciones de dependencia entre las variables aleatorias. De nueva cuenta, la información mutua será utilizada como una medida de la dependencia entre las variables aleatorias. En este caso, para encontrar el árbol con la mayor suma de informaciones mutuas, se deberá encontrar el árbol de expansión con máximo peso de un grafo completo con aristas pesadas, en las que el peso de las aristas será la información mutua entre las variables aleatorias representadas por los nodos. Una solución para realizar esto fue propuesta por Chow y Liu en 1968 [CL68], la cual se muestra en el algoritmo 13.

Algoritmo 13 Algoritmo Chow-Liu

Entrada: Las muestras de tamaño N de las m variables aleatorias. $X_1 = (x_1^1, x_1^2, \dots, x_1^N), X_2 = (x_2^1, x_2^2, \dots, x_2^N), \dots, X_m = (x_m^1, x_m^2, \dots, x_m^N)$

Salida: El modelo de árbol T que maximiza la información mutua entre las m variables aleatorias.

- 1: Construir un grafo completo G con aristas pesadas con la información mutua entre las variables aleatorias representadas por cada nodo.
 - 2: Obtener el árbol de expansión con máximo peso T correspondiente a G utilizando el algoritmo 11.
-

Debemos aclarar que en los algoritmos originales para construir ambos modelos, después de generar el grafo no dirigido, se selecciona un nodo como raíz y a partir de este se “cuelgan” los demás nodos, lo que generará un grafo dirigido a partir de este hacia los demás nodos. Debido a que en nuestra propuesta no nos interesa la dirección de las aristas, ya que en los vecindarios de las partículas las aristas son bidireccionales, omitimos el paso descrito en este párrafo, obteniendo como modelo final el grafo no dirigido correspondiente.

También es importante mencionar que cuando se construyen los árboles con Chow-Liu se está obteniendo el árbol óptimo (árbol de máxima expansión), mientras que al construir la cadena con MIMIC se obtiene una aproximación de este.

Capítulo 5

Construcción de la topología utilizando información mutua

En el presente capítulo se describe y presenta el algoritmo propuesto. Primero damos una justificación del porqué consideramos pertinente construir las topologías utilizando la información mutua entre las partículas. Posteriormente explicamos como construir los vecindarios de las partículas, abordando la problemática de la dimensionalidad y el dominio continuo de los problemas a resolver. Despues describimos los parámetros del algoritmo y los algoritmos que se utilizarán para construir los modelos gráficos. Finalmente presentamos el algoritmo optimización por enjambre de partículas basado en información mutua (Particle Swarm Optimization based on Mutual Information, PSO-MI).

5.1 Justificación

Como se mencionó en el capítulo 1, el desempeño de PSO está altamente relacionado con la forma en la que se conectan las partículas. Se ha demostrado que topologías con alta conectividad favorecen la etapa de explotación, por lo que se podría asumir que son una buena opción para funciones unimodales. Por otro lado, las topologías con poca conectividad hacen que las partículas tengan un comportamiento explorativo, por lo que se podría considerar apropiado que estas topologías son adecuadas para funciones multimodales. De la aseveración anterior se podría decir que si conocemos la función que deseamos optimizar, quizás podamos asignar una topología que nos ayude a encontrar el óptimo en un número de evaluaciones de función razonables. Esto no es del todo cierto, ya que aunque se conozca la función, la determinación de la conectividad no es tan trivial: ¿cuanto es mucha conectividad?, ¿cuanto es poca conectividad? y, ¿con base en que se podría determinar esto? Por ejemplo, podría pensarse que al asignar la topología todos

conectados para resolver la función Zakharov, que es una función unimodal, se obtendrán buenos resultados. Pero, como se puede ver en los resultados de las tablas 6.8, 6.11, 6.14 y 6.17 (utilizando una tolerancia $\epsilon = 10^{-10}$ y un máximo de 300000 evaluaciones de función como condiciones de paro y los parámetros mencionados en la sección 6.3), sólo en dimensión 30 se alcanzó el óptimo, obteniendo valores muy lejanos en dimensión 50, 80 y 100. De hecho esta es una de las principales problemáticas con la que se enfrenta PSO (y todos los algoritmos de optimización): la dimensionalidad. Para cuantificar la gravedad de esta problemática basta con ver los resultados obtenidos al resolver la función Esfera, una función ampliamente utilizada y conocida por ser sencilla de resolver, con PSO canónico (mismas condiciones de paro y parámetros que la función Zakharov). Aunque se alcanzó el óptimo global en dimensión 30, 50 y 80 (tablas 6.6, 6.9 y 6.12, respectivamente), el número de evaluaciones de función necesarias para llegar a este se incrementó notoriamente, de tal forma que 300,000 ya no bastaron para resolver la función en dimensión 100 (sólo un 23.333% de porcentaje de éxito, tabla 6.15). Estos casos son un buen ejemplo para hacer notar que no es suficiente conocer el tipo de función al momento de asignar la topología, mucho menos cuando se trata de funciones de alta dimensionalidad. Esta de más decir que la situación se complica cuando aun más cuando se desconoce como es la función a resolver. Si bien la selección de los parámetros adecuados para resolver un problema es importante, hacemos énfasis en la topología, ya que se han realizado muchos trabajos en los que se pretende dar a las partículas “inteligencia” para moverse en la función con el objetivo de llegar a un óptimo, pero creemos que no se ha estudiado lo suficiente para dotarle a las partículas “inteligencia” al momento de buscar conexiones con otras partículas, además, es necesario idear algoritmos que puedan lidiar con el problema de la dimensionalidad. De esta forma, consideramos que es válido conectar entre si a aquellas partículas que comparten una mayor cantidad de información. Para lograr esto se utilizará la información mutua, una medida de dependencia entre variables aleatorias ampliamente utilizada en la literatura que nos generará topologías con un alto nivel de dependencia entre las partículas.

5.2 Descripción del algoritmo

La idea detrás del algoritmo propuesto puede considerarse simple: “construir dinámicamente las topologías con base en la información mutua entre las partículas”. Esto quiere decir que el algoritmo realizará normalmente sus iteraciones y que cada determinado tiempo se utilizarán las posiciones en las que estuvo cada partícula para realizar el modelo gráfico de partículas

correspondiente (la topología). Pese a que la idea puede sonar simple, la implementación de este no es tan trivial como parece. En las secciones 5.2.1, 5.2.2, 5.2.3 y 5.2.4 se presenta a detalle como construir los vecindarios de optimización por enjambre de partículas.

5.2.1 Una topología distinta para cada dimensión

Imaginemos que estamos optimizando una cierta función en dimensión n con PSO, utilizando m partículas. En la primera iteración la partícula 1 se encuentra en la posición $\vec{P}_1^1 = (p_{11}^1, p_{12}^1, \dots, p_{1n}^1)$, la partícula 2 en la posición $\vec{P}_2^1 = (p_{21}^1, p_{22}^1, \dots, p_{2n}^1)$ y así sucesivamente hasta la partícula m , que se encontrará en la posición $\vec{P}_m^1 = (p_{m1}^1, p_{m2}^1, \dots, p_{mn}^1)$. Como queremos construir una topología para las partículas, debemos considerar a cada una de ellas como una variable aleatoria y tener una muestra de esta. Una primera idea podría ser obtener la información mutua entre las posiciones de cada partícula. Sin embargo, consideramos que esto no es del todo correcto, ya que en muchas ocasiones no existe relación alguna entre cada dimensión del problema, además que serían topologías locales que no tomarían en cuenta el conocimiento de que cada partícula ha obtenido de la función. Esta idea también implicaría construir los vecindarios en cada iteración, lo cual sería muy costoso computacionalmente.

La segunda aproximación, la cual consideramos más adecuada, es considerar a cada punto $\{\vec{P}_1^1, \vec{P}_2^1, \dots, \vec{P}_m^1\}$ como un elemento de una muestra cuya distribución de probabilidad es la función a optimizar. Debido a que en cada generación las partículas se mueven a mejores posiciones (generalmente), la probabilidad de que se obtenga un valor cercano a un óptimo aumenta conforme el algoritmo avanza, generando que esta distribución tenga altas densidades en los valores cercanos al óptimo, lo cual es deseable. Ahora consideremos que el historial de cada partícula es una muestra de esta distribución. Entonces, la partícula 1 habrá visitado, durante las primeras i iteraciones, las posiciones $\{\vec{P}_1^1, \vec{P}_1^2, \dots, \vec{P}_1^i\}$; las partícula 2 las posiciones $\{\vec{P}_2^1, \vec{P}_2^2, \dots, \vec{P}_2^i\}$ y así hasta la partícula m , que habrá visitado las posiciones $\{\vec{P}_m^1, \vec{P}_m^2, \dots, \vec{P}_m^i\}$. De esta forma, ahora las partículas pueden ser vistas como un vector aleatorio de dimensión n cuyos valores son tomados de la distribución de probabilidad definida por la función a optimizar. De esta forma, las topologías serán construidas tomando en cuenta el historial de las partículas, en otras palabras, tomando en cuenta el aprendizaje que las partículas han tenido sobre la función. Además, como se necesita tener muestras de un cierto tamaño, la topología se construiría cada determinado tiempo, por lo que el costo computacional se reduce.

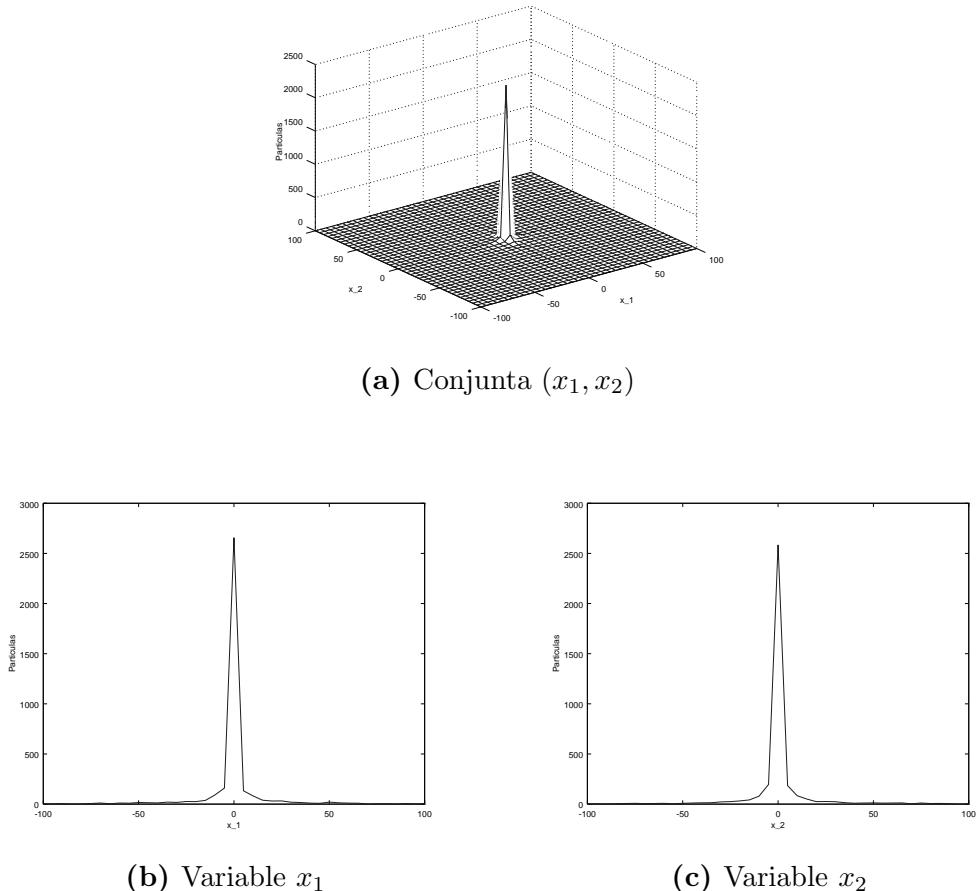


Figura 5.1: Distribución de las partículas en el espacio de búsqueda de la función esfera

Ahora bien, la problemática de utilizar esta aproximación es como obtener la información mutua entre las partículas: ¿cómo se calcula la información mutua entre dos vectores aleatorios? Antes de idear una forma de hacer esto, preferimos aprovechar que PSO optimiza la función dimensión por dimensión (ecuaciones 3.1 y 3.2), por lo que ahora, en vez de utilizar las posiciones de cada partícula como los elementos de la muestra, se tomará la posición para cada dimensión. En otras palabras, para las primeras i iteraciones, en la dimensión j la partícula 1 tendrá como muestra las posiciones $\{p_{1j}^1, p_{1j}^2, \dots, p_{1j}^i\}$, la partícula 2 las posiciones $\{p_{2j}^1, p_{2j}^2, \dots, p_{2j}^i\}$ y así hasta las partícula m , que tendrá como muestras las posiciones $\{p_{mj}^1, p_{mj}^2, \dots, p_{mj}^i\}$, de tal forma que ahora es posible obtener la información mutua entre las partículas, ya que cada partícula puede ser vista como una variable aleatoria. Esto se hará para cada dimensión j del problema, por lo que existirá una topología distinta

para cada dimensión. Esto es válido debido a que conforme PSO avanza, las partículas se acercan al valor correspondiente del óptimo para cada dimensión, por lo que se puede considerar que las partículas, en cada dimensión, toman valores de una distribución de probabilidad con altas densidades en la posición del óptimo de la dimensión correspondiente.

Para representar lo descrito en el párrafo anterior, se resolvió la función esfera en dimensión 2 utilizando PSO canónico con 30 partículas, $\omega = 0.7298$, $c_1 = c_2 = 1.49618$ y una tolerancia de 10^{-10} , con la finalidad de obtener la distribución de las partículas durante todo el proceso de búsqueda de PSO. En la figura 5.1a se muestra la distribución de las partículas en el espacio de búsqueda, observándose altas densidades en los valores cercanos al óptimo. Posteriormente, en las figuras 5.1b y 5.1c se obtuvo la distribución de las partículas para cada variable del problema por separado. En estas gráficas se puede ver que las distribuciones correspondientes a cada dimensión del problema también tienen altas densidades en su valor correspondiente del óptimo. Esta condición deseable de tener altas densidades en las proximidades del óptimo de cada variable del problema nos permite considerar a las partículas como muestras univariadas de cada dimensión del problema en vez de considerarlas como muestras multivariadas de una distribución conjunta definida por el número de variables de la función a optimizar. Por lo tanto, para nuestro algoritmo consideraremos distribuciones de las partículas como las mostradas en las figuras 5.1b y 5.1c, en lugar de una distribución como la de la figura 5.1a.

5.2.2 Variables con dominio continuo

Otra problemática que surge al utilizar la información mutua como medida para obtener dependencias entre las partículas es que los problemas a resolver generalmente son continuos. Entonces, de antemano, no es posible utilizar la ecuación 4.2, ya que esta se define para variables aleatorias discretas. Una posible solución sería “discretizar” el espacio de búsqueda de la función a optimizar, dividiendo el espacio de búsqueda en hipercubos de tamaño fijo y contando las ocurrencias de las partículas en estos. Si bien es cierto que esto es válido, el costo computacional se incrementaría exponencialmente, siendo un problema grave en problemas de alta dimensionalidad. Por lo tanto, consideramos que la mejor opción es utilizar la ecuación 4.11, la cual está definida para variables aleatorias continuas. Hay que recordar que para esta ecuación se está asumiendo que las variables aleatorias X y Y tienen una distribución normal, es decir, $X \sim \mathcal{N}(\mu_x, \sigma_x)$, $Y \sim \mathcal{N}(\mu_y, \sigma_y)$ y $(X, Y) \sim \mathcal{N}(\mu, K)$, con $\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$ y $K = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$. Si bien

es cierto que no existe evidencia para asumir esto (ni para no asumirlo), la asunción de normalidad es ampliamente utilizada. En nuestro caso, esto permite simplificar las ecuaciones utilizadas y así reducir la complejidad del cálculo de la información mutua.

5.2.3 Parámetros

Además de los parámetros del PSO canónico (factor de inercia, coeficientes de aceleración, tamaño del enjambre y velocidad máxima), para el algoritmo propuesto se necesitarán definir otros parámetros, todos relacionados con la construcción de la topología de las partículas. Estos parámetros se describen en las siguientes subsecciones.

Topología inicial

Debido a que se propone construir los vecindarios de las partículas utilizando modelos gráficos que representen las dependencias entre estas, necesitamos una cierta cantidad de datos para realizar esto. Entonces, surge la pregunta: ¿Qué datos se utilizarán para construir la topología en las primeras generaciones del algoritmo? Debido a que no se cuenta con datos para hacer esto, se ha optado por escoger *a priori* una topología para las primeras generaciones, antes de la construcción de las primeras topologías de las partículas utilizando un modelo gráfico. A este parámetro le llamaremos **topología inicial**. De hecho, como se verá en la sección 6.2.2, una adecuada selección de esta puede ayudar a un mejor desempeño del algoritmo, dependiendo de la función que se deseé optimizar. Esta topología inicial puede ser seleccionada de las topologías mostradas en la figura 3.3, o cualquier otra que se pueda considerar conveniente. Nos referiremos a este parámetro como T_0 .

Frecuencia de actualización

Este parámetro indica cada cuantas generaciones se construirá un nuevo modelo gráfico de las partículas. A este parámetro le llamaremos f . La importancia de este parámetro radica en que influye directamente en la complejidad computacional del algoritmo. Consideremos que queremos optimizar una función en dimensión n y que utilizamos m partículas para este objetivo. Supongamos que nos encontramos en la iteración $j + 1$, por lo que contamos con j posiciones que ha visitado cada partícula en el espacio de búsqueda. Entonces, para la dimensión i , la “muestra” de la partícula 1 son las posiciones $\{p_{1i}^1, p_{1i}^2, \dots, p_{1i}^j\}$, la de la partícula 2 las posiciones

$\{p_{2i}^1, p_{2i}^2, \dots, p_{2i}^j\}$ y así hasta la partícula m , cuya “muestra” serían las posiciones $\{p_{mi}^1, p_{mi}^2, \dots, p_{mi}^j\}$. Ahora bien, con estas muestras se construiría un modelo gráfico de las partículas (la topología) para la dimensión i . Esto mismo se debe hacer para cada una de las n dimensiones. Por lo tanto, si se decide actualizar las topologías en cada iteración se deben construir n modelos gráficos en cada iteración, lo cual es demasiado costoso. Además, si consideramos utilizar el mismo tamaño de muestra, estas no cambiarían mucho de una generación a otra. Retomando el ejemplo ya mencionado, en la generación $j + 2$ la partícula 1 en la dimensión i tendría por muestra los puntos $\{p_{1i}^2, p_{1i}^3, \dots, p_{1i}^{j+1}\}$, la misma que en la iteración $j + 1$, salvo que en lugar del punto p_{1i}^1 ahora estará el punto p_{1i}^{j+1} . Esto mismo se cumple para las demás partículas, por lo que es bastante probable que los modelos gráficos de una iteración a otra no cambien (o sean cambios mínimos). En el otro extremo, si se dejan pasar muchas iteraciones para actualizar las topologías se pueden perder los beneficios de tener vecindarios dinámicos, así como perder dependencias importantes entre las partículas.

Historial de las partículas

En la subsección anterior introdujimos, implícitamente, el **historial de las partículas**. Este parámetro consiste en cuantas posiciones previas se tomarán en cuenta para construir las topologías. A este parámetro le llamaremos h . Cuando en el ejemplo se menciona que se tomarán las j posiciones previas, lo que se está haciendo es asignar a h un valor de j . Para que nuestras topologías sean un modelo confiable se debe tener una cantidad suficiente de datos. En teoría, entre más datos se tengan el modelo será más certero, pero también se necesita una mayor cantidad de recursos computacionales [Dom12]. En primer lugar, nuestros datos para la construcción de las topologías son obtenidos directamente del algoritmo, por lo que estos estarán disponibles conforme el algoritmo avanza, por lo que es obvio que en las primeras generaciones no se dispondrá con una gran cantidad de datos. Además, hay que tomar en cuenta la complejidad del algoritmo. Utilizar una gran cantidad de datos será muy costoso, por lo que hay que buscar un equilibrio entre complejidad y desempeño. Esto sin dejar de lado que el utilizar una gran cantidad datos puede llevarnos a un caso de sobreentrenamiento [Dom12], lo cual no es deseado.

5.2.4 Modelos gráficos utilizados

Para construir las topologías de las partículas según nuestra propuesta, necesitamos utilizar modelos gráficos que representen dependencias entre las vari-

ables aleatorias (en este caso entre las partículas). Para lograr esto hemos decidido utilizar como medida de dependencia la información mutua. De esta forma, al construir los vecindarios estaremos obteniendo modelos que maximizan la información mutua entre las partículas. Hemos decidido utilizar dos algoritmos ampliamente utilizados en la literatura: Chow-Liu [CL68] y MIMIC [BJV96].

5.3 Algoritmo propuesto

Una vez descrito a detalle el algoritmo, procederemos a presentarlo. Debido a que para la construcción de las topologías de las partículas se utiliza como medida de dependencia la información mutua, hemos decidido llamarlo optimización por enjambre de partículas basado en información mutua (Particle Swarm Optimization based on Mutual Information, PSO-MI). Recordemos que para PSO-MI necesitamos asignar como valores de entrada el factor de inercia (ω), los coeficientes de aceleración (c_1, c_2), el tamaño del enjambre (S), la velocidad máxima (v_{max}), la topología inicial (T_0), la frecuencia de actualización (f) y el historial de las partículas (h). También debemos definir como se construirán los modelos gráficos que fungirán como topologías de las partículas. Una vez definido esto, corremos PSO utilizando durante f iteraciones utilizando T_0 como la topología de las partículas (que puede ser la misma para todas las dimensiones, o una diferente para cada dimensión). Una vez que han pasado f iteraciones, construimos los modelos gráficos correspondientes utilizando las h posiciones previas de las partículas. Estos modelos gráficos construidos serán la topología de las partículas durante las siguientes f generaciones, en donde se vuelven a actualizar los vecindarios de la misma forma. Este proceso se repetirá hasta que algún criterio de paro se cumpla.

Consideremos que han pasado f iteraciones y que se deben actualizar las topologías. Entonces, el historial de la partícula i serán las posiciones (las h posiciones previas) $\{P_i^1, P_i^2, \dots, P_i^h\}$, y para la dimensión j , la muestra de la partícula i serán las posiciones $\{P_{ij}^1, P_{ij}^2, \dots, P_{ij}^h\}$. De esta forma, para la dimensión j tomaremos las muestras de todas las partículas y construiremos el modelo gráfico correspondiente, obteniendo la topología de las partículas en la dimensión j . Esto se realizará para cada dimensión del problema. Este proceso puede ser visto en la figura 5.2. En esta se puede observar un conjunto de datos para cada dimensión, con los cuales se construirán los modelos gráficos correspondientes. En las filas de cada matriz se tienen las muestras de las m partícula en la dimensión correspondiente. El tamaño de cada muestra es h (iteraciones). PSO-MI se muestra en el algoritmo 14.

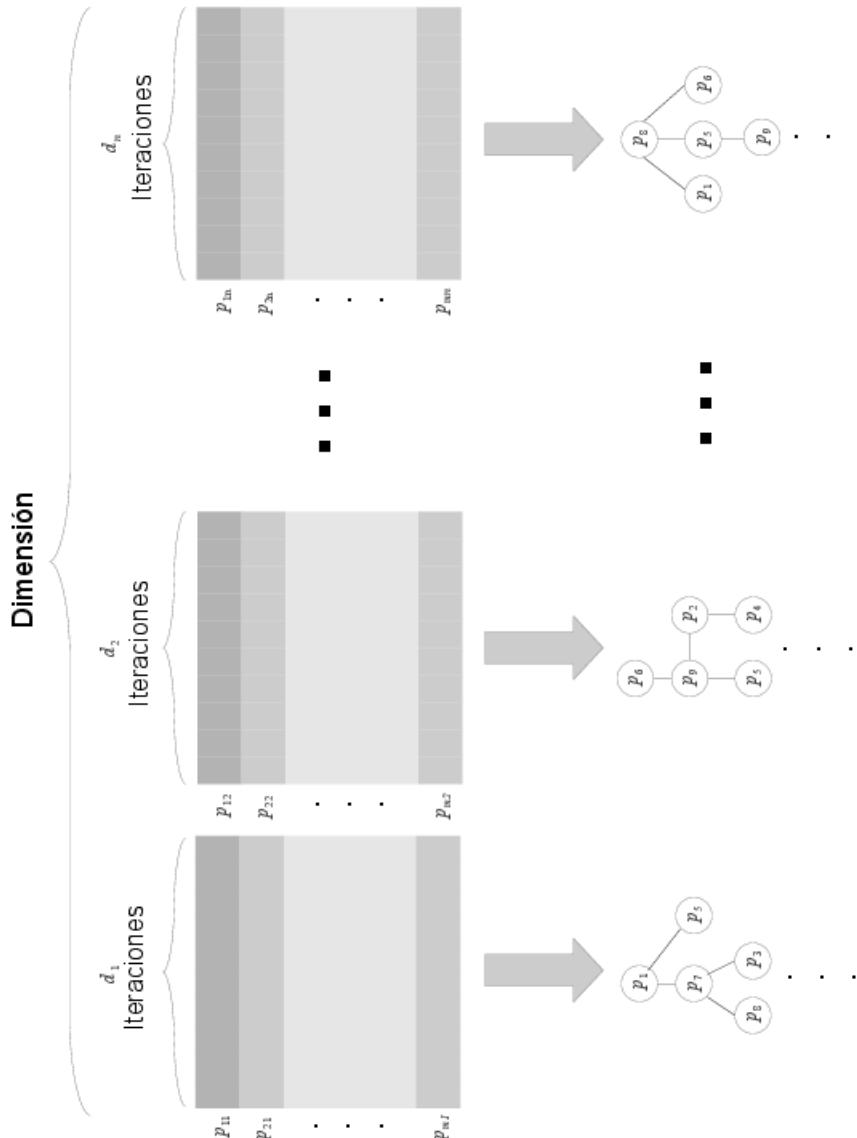


Figura 5.2: Construcción de las topologías en PSO-MI

Algoritmo 14 Algoritmo PSO-MI

Entrada: La función a optimizar, el espacio de búsqueda, criterios de paro, $\omega, c_1, c_2, S, v_{max}, T_0, f, h$ y cómo se construirán los modelos gráficos.

Salida: La aproximación del mínimo obtenida.

- 1: $t \leftarrow 1$
- 2: Inicialización aleatoria de la posición y la velocidad de las partículas dentro del espacio de búsqueda.
- 3: Realizar f iteraciones de PSO utilizando T_0 como topología. Almacenar las h posiciones previas de las partículas en $data$ y el personal best de cada partícula en p .
- 4: $t \leftarrow f$
- 5: **mientras** No se cumpla algún criterio de paro **hacer**
- 6: **si** $mod(t, f) = 0$ **entonces**
- 7: $T \leftarrow$ Actualización de las topologías para cada dimensión utilizando el modelo gráfico seleccionado, en nuestro caso con Chow-Liu (algoritmo 13) o MIMIC (algoritmo 12) utilizando las h posiciones previas de los datos almacenados en $data$.
- 8: **fin si**
- 9: **para** Cada partícula i **hacer**
- 10: **para** Cada dimensión j **hacer**
- 11: Actualizar la velocidad y posición de la partícula i en la dimensión j usando las ecuaciones 5.1 y 5.2.

$$v_{ij}^{t+1} = v_{ij}^t + c_1 r_1(p_{ij} - x_{ij}^t) + c_2 r_2(g_{ij} - x_{ij}^t) \quad (5.1)$$

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} \quad (5.2)$$

En donde g_{ij} es la posición en la dimensión j del mejor personal best de los vecinos de i en la dimensión j , los cuales son determinados por T , r_1 y r_2 son dos números aleatorios uniformemente distribuidos en el intervalo $[0, 1]$.

- 12: **fin para**
 - 13: Almacenar \vec{x}_i^{t+1} en $data$.
 - 14: Evaluar la función de aptitud de la partícula i .
 - 15: Actualizar \vec{p}_i .
 - 16: **fin para**
 - 17: $t \leftarrow t + 1$
 - 18: **fin mientras**
-

5.3.1 Complejidad del algoritmo

En la presente sección se analizará la complejidad algorítmica de PSO-MI. Primeramente, recordemos que para calcular la información mutua entre las variables aleatorias X y Y se utiliza la ecuación 4.11, en donde el coeficiente de correlación de Pearson se calcula utilizando la ecuación:

$$\rho(X, Y) = \frac{\sum_{i=1}^h (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^h (x_i - \mu_x)^2 \sum_{i=1}^h (y_i - \mu_y)^2}}$$

En donde x_i y y_i son los elementos de una muestra de tamaño h (historial de las partículas, para el caso de PSO-MI) de las variables aleatorias X y Y , respectivamente; $i \in \{1, \dots, h\}$, $\mu_x = \frac{1}{h} \sum_{i=1}^h x_i$ y $\mu_y = \frac{1}{h} \sum_{i=1}^h y_i$.

Debido a que todas las operaciones involucradas en las ecuaciones son operaciones simples (sumas, productos y raíz cuadrada), se puede asumir que estas son computadas en tiempo constante, por lo que la complejidad del cálculo de la información mutua es $O(h)$.

Ahora bien, si consideramos que para el problema de optimización se utilizan p partículas, se debe calcular $\frac{p(p-1)}{2}$ veces la información mutua (la información mutua entre todos los pares de partículas posibles). Además, como la complejidad de calcular una vez la información mutua es $O(h)$, entonces la complejidad de calcularla $\frac{p(p-1)}{2}$ veces es $O(hp^2)$.

Una vez computadas todas las aristas del grafo, se procede a obtener el árbol Chow-Liu utilizando el algoritmo de Kruskal. Es conocido que la complejidad del algoritmo Kruskal para encontrar el árbol de expansión mínima (máxima para el caso del árbol Chow-Liu) es $O(E \log V)$ [BB96], en donde E es el número de aristas en el grafo completo y V es el número de vértices (en nuestro caso $\frac{p(p-1)}{2}$ y p , respectivamente). Por lo tanto, la complejidad de este paso en PSO-MI es $O(p^2 \log p - p \log p)$, es decir $O(p^2 \log p)$.

Tomando en cuenta que en las pruebas realizadas se utilizan 30 partículas y valores del historial de las partículas de $h = 50$ y $h = 100$, entonces $h > \log p$ y la complejidad de construir una topología utilizando Chow-Liu es $O(hp^2 + p^2 \log p) = O(hp^2)$. De hecho, no es común utilizar poblaciones muy grandes, contrario a que si se necesita una suficiente cantidad de datos para construir los modelos gráficos. Supongamos el caso en el que $p = 100$ (un tamaño de población relativamente grande en PSO). En este caso, $\log_2 p \approx 7$, por lo que con cualquier $h \geq 7$ la complejidad de este paso sería $O(hp^2)$. Para

que se siga cumpliendo esto, si se usan 200 partículas se debe cumplir que $h \geq 8$, si se usan 500 partículas que $h \geq 9$, y así sucesivamente. Sin duda alguna, para tener modelos gráficos más certeros necesitamos una mayor cantidad de datos, probablemente más que el historial de partículas mencionado en el análisis anterior. Por esta razón consideramos que la complejidad de construir una topología de partículas usando Chow-Liu es $O(hp^2)$. Pero este es el caso únicamente para una dimensión. Si consideramos que queremos optimizar una función en dimensión n se deben construirán n árboles cada vez que se actualicen las topologías. Entonces, la complejidad de actualizar todas las topologías es $O(nhp^2)$.

Esta complejidad es la misma para el caso de MIMIC, ya que la información mutua se calcula con la misma ecuación que Chow-Liu, además que se debe calcular $\frac{p(p-1)}{2}$ veces la información mutua. Posteriormente, al momento de obtener la cadena MIMIC, el paso que mayor complejidad tiene es el de ordenar las aristas calculadas. Entonces, la complejidad de obtener la cadena MIMIC es $O(p^2 \log p)$ y la complejidad de actualizar las topologías es $O(nhp^2)$.

Si consideramos que el algoritmo tiene permitido realizar T iteraciones, entonces el número de veces que se actualizarán las topologías es $\frac{T}{f} - 1$ (no tiene caso actualizar las topologías en la última iteración). De esta forma, la complejidad algorítmica de construir todos los árboles (o cadenas) en el peor de los casos es $O(\frac{T}{f} nhp^2)$. A manera de ejemplificar esto, el peor caso que ocurrió en las pruebas realizadas es cuando el algoritmo utilizaba todas las evaluaciones de función permitidas (300000). Como el número de partículas que se utilizó en todos los casos fue 30, entonces $T = 10000$. Si utilizamos una frecuencia de actualización $f = 50$, entonces las topologías se actualizarán 199 veces. Además, cuando se optimizó una función en dimensión 100, el número de árboles que se construyó en este caso es 19900, cada uno con complejidad $O(hp^2)$.

Capítulo 6

Experimentos y resultados

En el presente capítulo se presentan las pruebas realizadas en algunas funciones de la literatura con la finalidad de medir el desempeño de PSO-MI. Primero, definimos cual es el conjunto de funciones en las que se realizarán las pruebas. Posteriormente, presentamos los experimentos realizados para asignar los parámetros adecuados para el algoritmo. También presentamos los resultados obtenidos en las funciones de prueba y una comparación con otros algoritmos que utilizan vecindarios dinámicos y con PSO canónico. Finalmente, mostramos el impacto benéfico de utilizar la información mutua para construir los vecindarios.

6.1 Funciones de prueba

Antes que nada, definiremos las funciones sobre las que se probará el desempeño del algoritmo propuesto. Para este fin se utilizaron un conjunto de 17 funciones ampliamente utilizadas en la literatura [JY13], [GBR06], [Jon75], [ZLJ04] y [LM85]. Estas se presentan en la tabla 6.1. Además de la expresión matemática de cada función se presenta su dominio y la función de aptitud del mínimo global correspondiente.

Se trató de buscar un equilibrio entre funciones unimodales y multimodales para poder evaluar de una forma más justa el desempeño del algoritmo propuesto. En la tabla 6.2 se separan las funciones de acuerdo a su modalidad. Debemos hacer mención aparte de la función Quartic with Noise (f_8) ya que esta no puede ser considerada ni unimodal ni multimodal, sino que es una función que es afectada por ruido aleatorio.

Función	Fórmula	Dimensión	Dominio	Mínimo global
Sphere	$f_1(x) = \sum_{i=1}^n x_i^2$	n	$-100 \leq x_i \leq 100$	0
Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	n	$-30 \leq x_i \leq 30$	0
Rastrigin	$f_3(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	n	$-5.12 \leq x_i \leq 5.12$	0
Griewank	$f_4(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	n	$-600 \leq x_i \leq 600$	0
Penalized 1	$f_5(x) = \frac{\pi}{n} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\}$ $+ \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	n	$-50 \leq x_i \leq 50$	0
Penalized 2	$f_6(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] \right.$ $\left. + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \right\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	n	$-50 \leq x_i \leq 50$	0
Ackley	$f_7(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + e$	n	$-32 \leq x_i \leq 32$	0
Quartic with noise	$f_8(x) = \sum_{i=1}^n i x_i^4 + \text{random}[0, 1]$	n	$-1.28 \leq x_i \leq 1.28$	0
Cigar	$f_9(x) = x_1^2 + \sum_{i=2}^n 10^6 x_i^2$	n	$-10 \leq x_i \leq 5$	0
Cigar Tablet	$f_{10}(x) = x_1^2 + \sum_{i=2}^{n-1} 10^4 x_i^2 + 10^8 x_n^2$	n	$-10 \leq x_i \leq 5$	0
Ellipsoid	$f_{11}(x) = \sum_{i=1}^n 10^{6(\frac{i-1}{n-1})} x_i^2$	n	$-10 \leq x_i \leq 5$	0
Different Powers	$f_{12}(x) = \sum_{i=1}^n x_i ^{2+10\frac{i-1}{n-1}}$	n	$-10 \leq x_i \leq 5$	0
Tablet	$f_{13}(x) = 10^6 x_1^2 + \sum_{i=2}^n x_i^2$	n	$-10 \leq x_i \leq 5$	0
Two Axes	$f_{14}(x) = \sum_{i=1}^{\lfloor n/2 \rfloor} 10^6 x_i^2 + \sum_{i=\lceil n/2 \rceil}^n x_i^2$	n	$-10 \leq x_i \leq 5$	0
Zakharov	$f_{15}(x) = \sum_{i=1}^n x_i^2 + (\sum_{i=1}^n 0.5ix_i)^2 + (\sum_{i=1}^n 0.5ix_i)^4$	n	$-5 \leq x_i \leq 10$	0
Levy	$f_{16}(x) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)]$ $+ (w_n - 1)^2 [1 + \sin^2(2\pi w_n)]$ $w_i = 1 + \frac{1}{4}(x_i - 1)$	n	$-10 \leq x_i \leq 10$	0
Salomon	$f_{17}(x) = 1 - \cos \left(2\pi \sqrt{\sum_{i=1}^n x_i^2} \right) + 0.1 \sqrt{\sum_{i=1}^n x_i^2}$	n	$-100 \leq x_i \leq 50$	0

Tabla 6.1: Funciones de prueba utilizadas para evaluar el desempeño del algoritmo propuesto

Funciones unimodales	Funciones multimodales
Esfera	Rosenbrock
Cigar Tablet	Rastrigin
Ellipsoid	Griewank
Different Powers	Penalized 1
Tablet	Penalized 2
Two Axes	Ackley
Zakharov	Levy
Cigar	Salomon

Tabla 6.2: Clasificación de las funciones de prueba con base en su modalidad

6.2 Parámetros

En la presente sección se muestran los valores seleccionados para los parámetros del algoritmo propuesto. Hacemos énfasis en que los valores de los coeficientes de aceleración ($c_1 = c_2 = 1.49618$) y el tamaño del enjambre ($S = 30$) se definieron *a priori*. Con base en estos valores asignados se realizaron pruebas para determinar que valores de los demás parámetros eran convenientes utilizar, tomando en cuenta el desempeño del algoritmo y el costo computacional de utilizar estos parámetros. Los resultados obtenidos y los valores seleccionados para las pruebas finales se muestran en las subsecciones 6.2.1, 6.2.2, 6.2.3 y 6.2.4.

6.2.1 Factor de inercia

Una vez determinados los coeficientes de aceleración y el tamaño del enjambre debemos seleccionar un valor adecuado del factor de inercia ω . Frans van den Bergh y Andries Engelbrecht en [BE06] demuestran que para que las partículas sigan una trayectoria convergente los coeficientes de aceleración y el factor de inercia seleccionados deben cumplir la desigualdad mostrada en la ecuación 6.1.

$$\omega > \frac{1}{2}(c_1 + c_2) - 1 \quad (6.1)$$

Con base en la ecuación 6.1 y en los coeficientes de aceleración seleccionados $c_1 = c_2 = 1.49618$, el factor de inercia que se utilice debe cumplir la desigualdad $\omega > 0.49618$. Para las pruebas se seleccionaron los valores

$\omega_1 = 0.578766$, $\omega_2 = 0.7298$ y $\omega_3 = 0.8$. Estos valores fueron probados en las funciones Esfera (f_1) y Ackley (f_7) en dimensión 30 con los valores de los coeficientes de aceleración y tamaño del enjambre mencionados con anterioridad. Como topología inicial se utilizó un vecindario totalmente desconectado y la topología se actualizó cada 100 generaciones utilizando el algoritmo Chow-Liu (algoritmo 13) y utilizando un historial de las partículas de 100 generaciones. Los criterios de paro fueron alcanzar el óptimo con una tolerancia $\epsilon = 10^{-10}$ o rebasar 300000 evaluaciones de función. Se realizaron 30 corridas de cada experimento. Los resultados obtenidos se muestran en la tabla 6.3.

Función	Estadístico reportado	Factor de inercia		
		0.578766	0.7298	0.8
Esfera	Valor del mínimo encontrado	Media	9.1401e-11	9.2315e-11
		Std	7.1177e-12	8.7492e-12
		Mediana	9.3162e-11	9.7098e-11
		Mínimo	7.3516e-11	6.4309e-11
		Máximo	9.9946e-11	9.9108e-11
	Evaluaciones de función	Media	33818	56229
		Std	1587.7	1212.3
	Porcentaje de éxito		100	100
				0
Ackley	Valor del mínimo encontrado	Media	9.6341e-11	9.7329e-11
		Std	4.1730e-12	1.8287e-12
		Mediana	9.7682e-11	9.7593e-11
		Mínimo	8.5067e-11	9.3167e-11
		Máximo	9.9921e-11	9.9981e-11
	Evaluaciones de función	Media	52741	97032
		Std	2400.5	30140
	Porcentaje de éxito		100	100
				0

Tabla 6.3: Resultados obtenidos al utilizar distintos factores de inercia

Como se puede observar en la tabla 6.3 los mejores resultados se obtienen para $\omega_1 = 0.578766$. Asumimos entonces que este factor de inercia funcionará de buena forma en funciones unimodales y multimodales, ya que en las pruebas para una función de cada tipo (Esfera y Ackley, respectivamente) se obtuvo una convergencia más rápida con este valor.

6.2.2 Topología inicial

Para construir la topología dinámica utilizando los algoritmos 12 o 13 se necesita una cierta cantidad de datos. Debido a que al inicio del algoritmo no contamos con estos (o no se tienen suficientes) no podemos utilizar la topología propuesta y es necesario iniciar con otra topología. El objetivo de este experimento es observar de que forma afecta la topología inicial en el desempeño del algoritmo. Para este efecto utilizaremos dos topologías: la topología todos conectados y una topología sin conectividad, es decir, una topología en la que todas las partículas están desconectadas. Se probará una función unimodal (Esfera, f_1) y una función multimodal (Griewank, f_4) en 30 dimensiones. Se realizarán 30 corridas del algoritmo 14 para cada función y para cada una de las topologías iniciales mencionadas. La actualización de la topología se hará cada 100 iteraciones utilizando el algoritmo Chow-Liu (algoritmo 13) y usando un historial de las partículas de 100 generaciones. Los parámetros utilizados para este experimento son $\omega = 0.578766$, $c_1 = c_2 = 1.49618$ y $S = 30$. El algoritmo se detendrá cuando se obtenga un valor del óptimo con una tolerancia $\epsilon = 10^{-10}$ o cuando se superen 300000 evaluaciones de función. Se realizaron 30 corridas para cada experimento. En la tabla 6.4 se muestran los resultados obtenidos.

Función	Estadístico reportado	Topología inicial	
		Todos conectados	Todos desconectados
Esfera	Valor del mínimo encontrado	Media	9.2862e-11
		Std	7.4535e-12
		Mediana	9.5501e-11
		Mínimo	7.0056e-11
		Máximo	9.9569e-11
	Evaluaciones de función	Media	27831
		Std	1889.9
	Porcentaje de éxito	100	100
Griewank	Valor del mínimo encontrado	Media	0.061294
		Std	0.12027
		Mediana	9.9122e-11
		Mínimo	8.2469e-11
		Máximo	0.54885
	Evaluaciones de función	Media	165280
		Std	128170
	Porcentaje de éxito	56.667	90

Tabla 6.4: Resultados obtenidos al iniciar el algoritmo propuesto con topología todos conectados y con topología todos desconectados

Como se puede observar en la tabla 6.4, al iniciar con la topología todos conectados se obtiene una convergencia más rápida en la función Esfera (27831 evaluaciones de función, contra 33818 con la topología todos desconectados), pero en la función Griewank se obtiene un mayor porcentaje de éxito con la topología todos desconectados (90%, contra 56.557% con la topología todos conectados). De cierto modo esto tiene lógica, ya que la alta conectividad en la topología todos conectados hará que las partículas tengan un comportamiento explotativo desde un principio, generando un mejor comportamiento en funciones unimodales. Por el contrario, la nula conectividad en la topología todos desconectados provocará que las partículas tengan un comportamiento totalmente explorativo, beneficiando la búsqueda en funciones multimodales. Con base en estos resultados se decidió iniciar con una topología desconectada para tratar de obtener un mayor porcentaje de éxito en funciones multimodales, asumiendo que, aunque en funciones unimodales la convergencia es un poco más lenta, estas eventualmente estas convergerán.

Recordemos que para actualizar la velocidad se utiliza la ecuación:

$$\vec{v}_i^{t+1} = \omega \vec{v}_i^t + c_1 \vec{r}_{1i}^t \otimes (\vec{p}_i^t - \vec{x}_i^t) + c_2 \vec{r}_{2i}^t \otimes (\vec{l}_i^t - \vec{x}_i^t)$$

En el caso de la topología todos desconectados, al no existir comunicación entre las partículas, la mejor experiencia del vecindario es la mejor experiencia personal de cada partícula, es decir $\vec{l}_i^t = \vec{p}_i^t$. Además, considerando que para nuestras pruebas $c_1 = c_2$, la ecuación de la velocidad se puede expresar como:

$$\vec{v}_i^{t+1} = \omega \vec{v}_i^t + c \vec{r}_i^t \otimes (\vec{p}_i^t - \vec{x}_i^t)$$

En donde $c = c_1 = c_2$ y \vec{r}_i^t es un vector aleatorio n -dimensional distribuido uniformemente en el intervalo $[0, 2]$. Consideremos el caso $t = 1$, en donde:

$$\vec{v}_i^2 = \omega \vec{v}_i^1 + c \vec{r}_i^1 \otimes (\vec{p}_i^1 - \vec{x}_i^1)$$

Debido a que la partícula sólo ha visitado una posición, la mejor experiencia personal es igual a la posición actual, es decir $\vec{x}_i^1 = \vec{p}_i^1$, por lo que la velocidad para $t = 1$ es:

$$\vec{v}_i^2 = \omega \vec{v}_i^1$$

Y entonces la posición de la partícula se actualiza utilizando la ecuación:

$$\vec{x}_i^2 = \vec{x}_i^1 + \omega \vec{v}_i^1$$

Esto implica que las partículas sólo se moverán con la inercia de su velocidad. Este comportamiento ocurrirá en las generaciones en las que las

partículas encuentren su mejor experiencia personal ya que, para este caso $\vec{x}_i^t = \vec{p}_i^t$.

6.2.3 Frecuencia de actualización

Otro factor a considerar es cada cuanto actualizar la topología propuesta. Suponemos que si se actualiza más veces esta se obtendrán mejores resultados, pero hay que considerar que se deben tener suficientes datos para la obtención de la información mutua entre las partículas. También se debe considerar el costo computacional de construir los árboles (o la cadena), ya que se obtendrá uno para cada dimensión de la función a optimizar, y esto en funciones de dimensión alta puede ser muy costoso. Para este experimento se probó la función Esfera (f_1) en dimensión 80. La topología se actualizó cada 50, 100, 200, 500 y 1000 generaciones. Los parámetros del algoritmo utilizados fueron $\omega = 0.578766$, $c_1 = c_2 = 1.49618$, $S = 30$ y $h = 50$, $h = 100$, $h = 200$, $h = 500$ y $h = 1000$; respectivamente. Se inició con una topología todos desconectados y el algoritmo se corrió durante 2000 generaciones para cada caso. En todas las pruebas se utilizó la misma configuración inicial, esto es, la misma posición y velocidad para cada partícula del enjambre. En la figura 6.1 se muestran los resultados obtenidos para este experimento.

Como se puede observar en la figura 6.1, al actualizar la topología cada 50 generaciones se obtiene una convergencia más rápida que para los demás tiempos de actualización. De hecho se puede ver que el tiempo de actualización influye directamente en la velocidad de convergencia del algoritmo, ya que al hacer esto más pronto se llega a mejores resultados. Además se observa la influencia positiva de la actualización; por ejemplo, al actualizar cada 500 y 1000 generaciones se nota un descenso en la aptitud de las partículas inmediatamente después de actualizar, pero después de un cierto número de iteraciones se nota un estancamiento de las partículas, del cual se “sale” al actualizar nuevamente la topología. Como se mencionó en el párrafo anterior se debe considerar el costo computacional de construir los árboles con mayor frecuencia. Pese a que actualizar estos cada 50 generaciones implica obtener mejores resultados, también implica un mayor costo y tiempo computacional. Tomando en cuenta esto, se decidió realizar pruebas actualizando la topología cada 50 y 100 generaciones. En la sección 6.3 se muestran los resultados obtenidos actualizando las topologías cada 100 generaciones. Para ver los resultados obtenidos en las funciones de prueba actualizando cada 50 iteraciones, puede consultar el anexo A.

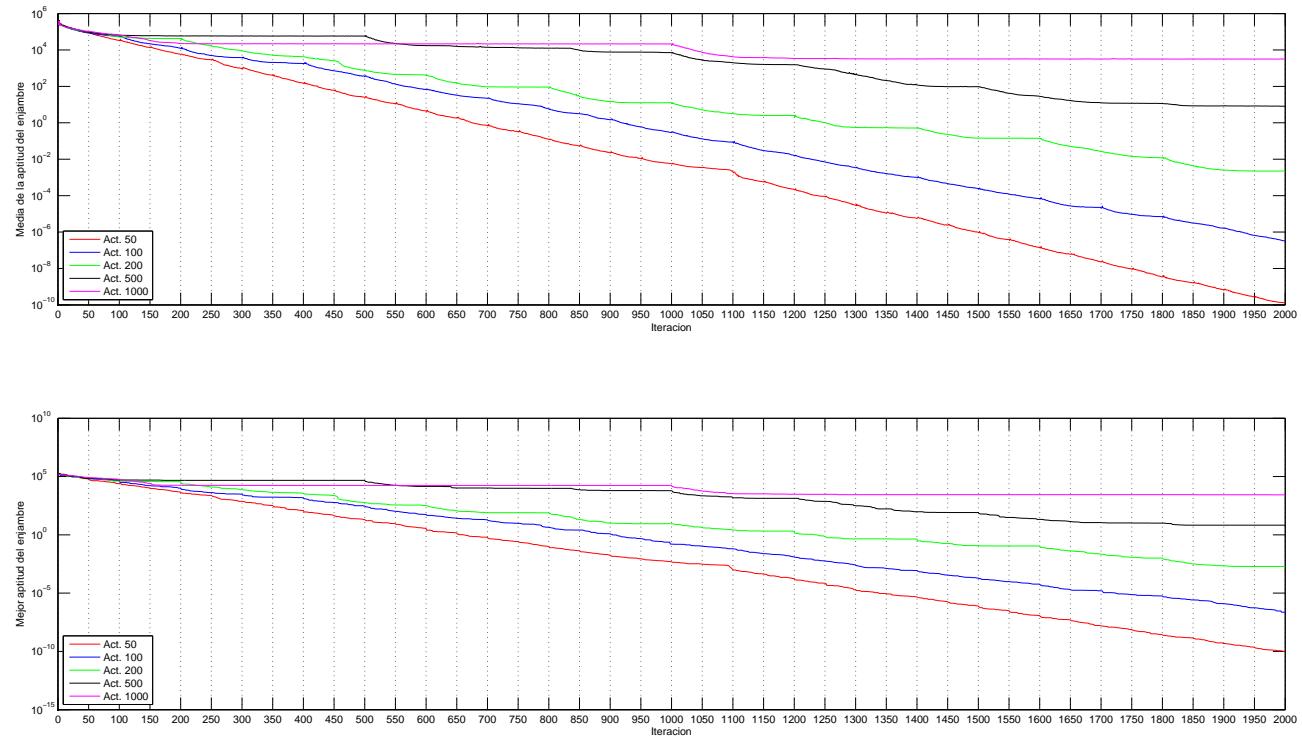


Figura 6.1: Evolución de la media de la aptitud y del mejor elemento del enjambre en la función Esfera en dimensión 80 con distintos tiempos de actualización de la topología

6.2.4 Historial de las partículas

En este experimento se pretende conocer que cantidad de datos es conveniente utilizar para la construcción de la topología. Como ya se mencionó con anterioridad, para la construcción del vecindario de cada partícula se utiliza la historia de las posiciones que “visitó” esta en un cierto número de generaciones previas. La cuestión a resolver es cuantas posiciones anteriores utilizar para actualizar la topología. Nuestra hipótesis es que utilizar pocas posiciones anteriores beneficiará la capacidad explotativa de las partículas, ya que se utiliza información local de las partículas debido a que estas habrán visitado menos lugares (y probablemente en una región pequeña del espacio de búsqueda) de la función objetivo en un corto tiempo. Por el contrario, al utilizar una mayor cantidad de datos de las posiciones previas de las partículas se estarán tomando en cuenta más áreas del espacio de búsqueda lo cual beneficiará el comportamiento explorativo de estas. Para demostrar esto se harán pruebas con una función unimodal (Esfera, f_1) y una función multimodal (Rastrigin, f_3) utilizando las 100 posiciones previas de cada partícula y utilizando todas las posiciones que visitó cada partícula durante el desarrollo del algoritmo. Las funciones se probaron en dimensión 30 con parámetros $\omega = 0.578766$, $c_1 = c_2 = 1.49618$ y $S = 30$. Como topología inicial se utilizó un vecindario con todas las partículas desconectadas y la topología se actualizó cada 100 generaciones utilizando el algoritmo Chow-Liu (algoritmo 13). Se realizaron 30 corridas de cada experimento y los criterios de paro utilizados fueron alcanzar una tolerancia $\epsilon = 10^{-10}$ o rebasar las 300000 evaluaciones de función. En la tabla 6.5 se presentan los resultados obtenidos para cada función probada para la cantidad de datos mencionadas.

Como se puede observar en los resultados presentados en la tabla 6.5, cuando se utilizó toda la historia de las partículas se obtuvo un menor porcentaje de éxito en la función Esfera que cuando se utilizaron solamente los 100 datos previos (66.667% en el primer caso contra el 100% en el segundo). Al ser la Esfera una función unimodal, se puede concluir que el utilizar todas las posiciones visitadas por las partículas generó un comportamiento explorativo en las partículas, lo que provocó una convergencia más lenta y por lo tanto en algunas ocasiones no se llegó al óptimo deseado en las evaluaciones de función permitidas. Por el contrario, en la función Rastrigin (multimodal), al utilizar toda la historia de las partículas se obtuvo un mejor valor de aptitud de la función (en promedio 11.65, contra 41.653 al utilizar sólo las 100 posiciones previas). También esto nos indica un comportamiento explorativo de las partículas, necesario para alcanzar zonas promisorias del espacio de búsqueda. Con la información obtenida podemos concluir que para funciones

unimodales es conveniente utilizar pocas posiciones previas de las partículas mientras que para funciones multimodales lo mejor es utilizar más datos del historial de estas (o toda la historia). Nuevamente hacemos énfasis en el costo computacional, ya que entre más datos se utilicen para construir la topología, este se incrementará. Por esta razón se decidió utilizar sólo los 100 datos previos de las posiciones de las partículas (en caso de que el tiempo de actualización sea 100, 50 datos previos si se utiliza una frecuencia de actualización de 50) en todas las funciones de prueba. Aun con esta elección, como se verá más adelante, se obtuvieron buenos resultados en varias funciones multimodales, pero se debe considerar este análisis para futuros experimentos.

Función	Estadístico reportado	Posiciones previas	
		Últimas 100	Todas
Esfera	Valor del mínimo encontrado	Media	9.1401e-11
		Std	7.1177e-12
		Mediana	9.3162e-11
		Mínimo	7.3516e-11
		Máximo	9.9946e-11
	Evaluaciones de función	Media	33818
		Std	1587.7
	Porcentaje de éxito	100	66.667
Rastrigin	Valor del mínimo encontrado	Media	41.653
		Std	14.758
		Mediana	42.035
		Mínimo	10.945
		Máximo	70.766
	Evaluaciones de función	Media	300000
		Std	0
	Porcentaje de éxito	0	0

Tabla 6.5: Resultados obtenidos al utilizar las 100 posiciones previas y todas las posiciones visitadas por las partículas en las funciones Esfera y Rastrigin en dimensión 30

6.2.5 Modelos gráficos utilizados

Para las pruebas mostradas en este capítulo, los vecindarios se construyeron utilizando el algoritmo Chow-Liu (algoritmo 13), es decir, las topologías de las partículas son árboles. También se realizaron pruebas en las funciones de la

tabla 6.1 utilizando modelos de cadena (algoritmo 12, MIMIC), y frecuencias de actualización de 50 y 100 generaciones. Estos resultados pueden ser vistos en el anexo B. Una comparación entre los resultados obtenidos con Chow-Liu y MIMIC se pueden observar en el anexo C.

6.3 Resultados obtenidos en las funciones de prueba

En esta sección se presentan los resultados obtenidos al aplicar el algoritmo propuesto, al que llamaremos PSO-MI (Particle Swarm Optimization based on Mutual Information), a las funciones de prueba definidas en la tabla 6.1. Para tener un punto de comparación con otras versiones de PSO propuestas en la literatura, se decidió probar las funciones con la versión canónica de PSO [KE95] y con dos propuestas que utilizan topologías dinámicas: Vecindario dinámico jerárquico basado en optimización por enjambre de partículas (Hierarchical D-LPSO) [GZDA11] y optimización por enjambre de partículas con incremento en la conectividad de la topología (PSO-ITC) [LI14].

Las funciones de la tabla 6.1 fueron probadas en dimensión 30, 50, 80 y 100. Para ser uniformes en la comparación de resultados, para todos los casos se utilizan 30 partículas ($S = 30$). Además, se utilizan los mismos criterios de paro, los cuales son alcanzar el óptimo con una tolerancia $\epsilon = 10^{-10}$ o rebasar 300000 evaluaciones de función. Se realizan 30 corridas para cada combinación función-algoritmo reportada.

Los parámetros utilizados para cada algoritmo son los siguientes:

- **PSO Canónico.** $\omega = 0.7298$ y $c_1 = c_2 = 1.49618$. [BE06]
- **Hierarchical D-LPSO.** $\omega = 0.578097$ y $c_1 = c_2 = 1.49445$. [GZDA11]
- **PSO-ITC.** $\omega_{inicial} = 0.9$, $\omega_{final} = 0.4$, $c_1 = c_2 = 2$, $z = 5$, $TC_{min} = 1$ y $TC_{max} = S - 1$. [LI14]
- **PSO-MI.** $\omega = 0.578766$, $c_1 = c_2 = 1.49618$, T_0 : todos desconectados, f : 100 generaciones, h : 100 posiciones previas y la construcción de las topologías se hace utilizando Chow-Liu (algoritmo 13). Estos parámetros fueron seleccionados por las razones explicadas en la sección 6.2.

Para cada combinación función-algoritmo se reporta la media, la desviación estándar (Std), la mediana, el mínimo y el máximo valor del óptimo encontrado de las 30 corridas realizadas. También se reporta la media y la

desviación estándar de las evaluaciones de función consumidas para alcanzar el óptimo, o en su defecto, las 300000 evaluaciones de función permitidas como máximo. Esto significa que, si el algoritmo no converge al óptimo conocido se le suma el máximo número de evaluaciones de función como una especie de penalización. Otro valor reportado es el porcentaje de éxito, el cual se obtiene dividiendo el número de veces que se alcanzó el óptimo (con una tolerancia $\epsilon = 10^{-10}$) entre el total de pruebas realizadas, en este caso 30. Los resultados obtenidos en dimensión 30 se reportan en las tablas 6.6 (f_1 a f_6), 6.7 (f_7 a f_{12}) y 6.8 (f_{13} a f_{17}); en dimensión 50 en las tablas 6.9 (f_1 a f_6), 6.10 (f_7 a f_{12}) y 6.11 (f_{13} a f_{17}); en dimensión 80 en las tablas 6.12 (f_1 a f_6), 6.13 (f_7 a f_{12}) y 6.14 (f_{13} a f_{17}) y en dimensión 100 en las tablas 6.15 (f_1 a f_6), 6.16 (f_7 a f_{12}) y 6.17 (f_{13} a f_{17}).

Función	Estadístico reportado	Algoritmo			
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI
f_1	Valor del mínimo encontrado	Media	9.4523e-11	9.2647e-11	8.5021e-11
		Std	5.0018e-12	6.7114e-12	1.4196e-11
		Mediana	9.5238e-11	9.4917e-11	8.6391e-11
		Mínimo	7.9328e-11	7.8473e-11	4.1146e-11
		Máximo	9.9599e-11	9.9975e-11	9.9893e-11
	Evaluaciones de función	Media	25527	31379	77758
		Std	1485.2	1348.5	1747.3
	Porcentaje de éxito		100	100	100
	Valor del mínimo encontrado	Media	5.8539	19.248	32.988
		Std	15.853	24.969	38.126
		Mediana	3.3482	10.203	11.741
		Mínimo	1.6711e-05	0.00261	0.041216
		Máximo	87.91	93.119	113.12
f_2	Evaluaciones de función	Media	300000	300000	300000
		Std	0	0	0
		Porcentaje de éxito	0	0	0
	Valor del mínimo encontrado	Media	70.708	63.947	8.3899e-11
		Std	18.816	13.992	1.8134e-11
		Mediana	68.652	65.667	9.0807e-11
		Mínimo	38.803	38.803	3.5243e-11
		Máximo	122.38	97.506	9.9988e-11
	Porcentaje de éxito	Media	300000	300000	76835
		Std	0	0	1670.2
		Porcentaje de éxito	0	0	100
f_3	Evaluaciones de función	Media	0.026901	0.017014	0.030324
		Std	0.03118	0.019417	0.029065
		Mediana	0.018463	0.0098573	0.023371
		Mínimo	8.5428e-11	9.1068e-11	6.4372e-11
		Máximo	0.11773	0.061295	0.12263
	Porcentaje de éxito	Media	199640	210580	247530
		Std	134150	120800	96735
		Porcentaje de éxito	36.667	36.667	23.333
	Valor del mínimo encontrado	Media	0.6585	0.024186	7.9854e-11
		Std	0.93565	0.10069	1.8992e-11
		Mediana	0.25914	9.6348e-11	8.4536e-11
		Mínimo	9.0963e-11	8.2972e-11	2.8458e-11
		Máximo	4.1802	0.51825	9.9525e-11
f_4	Porcentaje de éxito	Media	220230	73198	70211
		Std	123960	64304	2361.3
		Porcentaje de éxito	30	93.333	100
	Evaluaciones de función	Media	0.23634	0.0018312	8.117e-11
		Std	0.7198	0.0041648	1.9681e-11
		Mediana	9.9282e-11	9.6731e-11	9.0064e-11
		Mínimo	7.5948e-11	7.5864e-11	3.6044e-11
		Máximo	3.5975	0.010987	9.9777e-11
	Porcentaje de éxito	Media	156090	98079	77219
		Std	136940	94140	1939.6
		Porcentaje de éxito	53.333	83.333	100
f_5	Valor del mínimo encontrado	Media	0.61352	61352	61352
		Std	123960	64304	2361.3
		Mediana	0.25914	9.6348e-11	8.4536e-11
		Mínimo	9.0963e-11	8.2972e-11	2.8458e-11
		Máximo	4.1802	0.51825	9.9525e-11
	Porcentaje de éxito	Media	220230	73198	70211
		Std	123960	64304	2361.3
		Porcentaje de éxito	30	93.333	100
	Evaluaciones de función	Media	0.23634	0.0018312	8.117e-11
		Std	0.7198	0.0041648	1.9681e-11
		Mediana	9.9282e-11	9.6731e-11	9.0064e-11
		Mínimo	7.5948e-11	7.5864e-11	3.6044e-11
		Máximo	3.5975	0.010987	9.9777e-11
f_6	Porcentaje de éxito	Media	156090	98079	77219
		Std	136940	94140	1939.6
		Porcentaje de éxito	53.333	83.333	100
	Valor del mínimo encontrado	Media	0.23634	0.0018312	8.117e-11
		Std	0.7198	0.0041648	1.9681e-11
		Mediana	9.9282e-11	9.6731e-11	9.0064e-11
		Mínimo	7.5948e-11	7.5864e-11	3.6044e-11
		Máximo	3.5975	0.010987	9.9777e-11
	Evaluaciones de función	Media	156090	98079	77219
		Std	136940	94140	1939.6
		Porcentaje de éxito	53.333	83.333	100

Tabla 6.6: Resultados obtenidos en las funciones f_1 , f_2 , f_3 , f_4 , f_5 y f_6 en dimensión 30

Función	Estadístico reportado	Algoritmo			
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI
f_7	Valor del mínimo encontrado	Media	2.8603	1.0847	9.0042e-11
		Std	1.6736	0.92294	4.1730e-12
		Mediana	2.6188	1.2478	9.4801e-11
		Mínimo	9.9122e-11	9.3338e-11	4.5934e-11
		Máximo	7.2185	2.6584	9.9907e-11
	Evaluaciones de función	Media	291320	218020	114730
		Std	47537	111550	3485.6
	Porcentaje de éxito		3.3333	36.667	100
	Evaluaciones de función	Media	0.003029	0.004057	0.0042399
		Std	0.001247	0.0015664	0.0017329
		Mediana	0.0027416	0.0036402	0.0040736
		Mínimo	0.0012511	0.0018065	0.0013363
		Máximo	0.0067008	0.0076023	0.0080715
f_8	Valor del mínimo encontrado	Media	300000	300000	300000
		Std	0	0	0
		Mediana			
		Mínimo			
		Máximo			
	Evaluaciones de función	Media	300000	300000	300000
		Std	0	0	0
		Porcentaje de éxito			
	Evaluaciones de función	Media	9.4222e-11	9.3924e-11	9.0159e-11
		Std	3.4841e-12	5.1669e-12	1.307e-11
		Mediana	9.4144e-11	9.4964e-11	9.5048e-11
		Mínimo	8.6788e-11	7.963e-11	4.8148e-11
		Máximo	9.9405e-11	9.9835e-11	9.9982e-11
f_9	Valor del mínimo encontrado	Media	32942	39065	97523
		Std	2734.5	2657	1700
		Mediana			
		Mínimo			
		Máximo			
	Evaluaciones de función	Media			
		Std			
		Porcentaje de éxito	100	100	100
	Valor del mínimo encontrado	Media	9.3291e-11	9.2436e-11	8.5654e-11
		Std	6.7549e-12	7.4216e-12	1.4938e-11
		Mediana	9.5000e-11	9.5273e-11	9.1839e-11
		Mínimo	6.8599e-11	7.4862e-11	4.8214e-11
		Máximo	9.9810e-11	9.974e-11	9.9679e-11
f_{10}	Evaluaciones de función	Media	30268	35748	88495
		Std	2629.6	2016.7	2112.2
		Mediana			
		Mínimo			
		Máximo			
	Porcentaje de éxito	Media			
		Std			
		Porcentaje de éxito	100	100	100
f_{11}	Valor del mínimo encontrado	Media	9.4025e-11	9.5269e-11	8.2596e-11
		Std	4.5776e-12	3.9883e-12	1.6368e-11
		Mediana	9.4935e-11	9.5931e-11	8.7699e-11
		Mínimo	8.2358e-11	8.1951e-11	3.8121e-11
		Máximo	9.9826e-11	9.9907e-11	9.9987e-11
	Evaluaciones de función	Media	28854	33612	86814
		Std	1816.9	1441.5	1689.4
		Mediana			
	Porcentaje de éxito	Media			
		Std			
f_{12}	Valor del mínimo encontrado	Media	9.1578e-11	9.0249e-11	4.5515e-11
		Std	6.3117e-12	1.1116e-11	3.1544e-11
		Mediana	9.2804e-11	9.3169e-11	4.0037e-11
		Mínimo	7.7981e-11	5.3057e-11	5.2275e-13
		Máximo	9.9863e-11	9.9948e-11	9.9866e-11
	Evaluaciones de función	Media	15306	24408	58616
		Std	1350.5	2202.1	2131.8
		Mediana			
	Porcentaje de éxito	Media			
		Std			

Tabla 6.7: Resultados obtenidos en las funciones f_7 , f_8 , f_9 , f_{10} , f_{11} y f_{12} en dimensión 30

Función	Estadístico reportado	Algoritmo				
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI	
f_{13}	Valor del mínimo encontrado	Media	9.2182e-11	9.3615e-11	8.4362e-11	8.9246e-11
		Std	9.8034e-12	6.4085e-12	1.7357e-11	8.0705e-12
		Mediana	9.5961e-11	9.5583e-11	9.1084e-11	8.9872e-11
		Mínimo	5.1959e-11	7.1981e-11	3.1195e-11	6.8632e-11
		Máximo	9.9698e-11	9.9976e-11	9.9998e-11	9.9657e-11
	Evaluaciones de función	Media	24206	28148	67959	29092
		Std	1757.7	1768.6	1752.8	1666.8
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	9.4988e-11	9.415e-11	8.5876e-11	9.2828e-11
		Std	5.8284e-12	6.3487e-12	2.0341e-11	6.8286e-12
		Mediana	9.7143e-11	9.6152e-11	9.4814e-11	9.4794e-11
		Mínimo	7.1773e-11	6.8414e-11	1.8166e-11	7.1403e-11
		Máximo	9.9795e-11	9.989e-11	9.9819e-11	9.9815e-11
	Evaluaciones de función	Media	31384	34711	90529	35915
		Std	1794.8	2488.4	1761.7	2019.6
	Porcentaje de éxito		100	100	100	100
f_{15}	Valor del mínimo encontrado	Media	9.5458e-11	2.3718e-08	684.64	1.7827e-08
		Std	3.1527e-12	9.5015e-08	602.89	9.1173e-08
		Mediana	9.5737e-11	5.6451e-10	511.03	1.7998e-10
		Mínimo	8.8952e-11	8.663e-11	244.64	9.1995e-11
		Máximo	9.9846e-11	5.016e-07	2821.9	5.0039e-07
	Evaluaciones de función	Media	131110	298930	300000	289510
		Std	9519.9	2881	0	16733
	Porcentaje de éxito		100	20	0	46.667
	Valor del mínimo encontrado	Media	6.5341	1.3052	8.5891e-11	0.051401
		Std	3.4054	1.3493	1.5697e-11	0.14045
		Mediana	6.1358	0.90865	9.2553e-11	9.5561e-11
		Mínimo	1.1772	8.5858e-11	4.0351e-11	6.3066e-11
		Máximo	17.364	4.908	9.9807e-11	0.45432
	Evaluaciones de función	Media	300000	272150	67053	66758
		Std	0	72535	1699.1	93061
	Porcentaje de éxito		0	13.333	100	86.667
f_{17}	Valor del mínimo encontrado	Media	0.66321	0.19987	0.20687	0.25327
		Std	0.31347	2.9156e-17	0.04571	0.06819
		Mediana	0.54987	0.19987	0.19987	0.19987
		Mínimo	0.39987	0.19987	0.099873	0.19987
		Máximo	1.6999	0.19987	0.30969	0.39987
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
	Porcentaje de éxito		0	0	0	0

Tabla 6.8: Resultados obtenidos en las funciones f_{13} , f_{14} , f_{15} , f_{16} y f_{17} en dimensión 30

Función	Estadístico reportado	Algoritmo			
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI
f_1	Valor del mínimo encontrado	Media	9.6157e-11	9.5592e-11	8.5164e-11
		Std	3.9036e-12	5.8253e-12	1.5425e-11
		Mediana	9.7018e-11	9.8165e-11	9.0204e-11
		Mínimo	8.3996e-11	7.8182e-11	4.852e-11
		Máximo	9.9992e-11	9.9994e-11	9.9909e-11
	Evaluaciones de función	Media	67713	69082	115070
		Std	7712.5	3938.2	3420.1
	Porcentaje de éxito		100	100	100
	Valor del mínimo encontrado	Media	21.209	68.752	68.109
		Std	32.553	69.081	43.953
		Mediana	9.1837	42.228	79.654
		Mínimo	0.00071741	0.059805	3.6393e-06
		Máximo	148.76	233.85	146.3
f_2	Evaluaciones de función	Media	300000	300000	300000
		Std	0	0	0
		Porcentaje de éxito	0	0	0
	Valor del mínimo encontrado	Media	175.11	140.32	8.8285e-11
		Std	37.353	25.255	1.2883e-11
		Mediana	175.61	133.32	9.274e-11
		Mínimo	114.42	93.526	4.9454e-11
		Máximo	284.56	198.99	9.9988e-11
	Porcentaje de éxito	Media	300000	300000	109090
		Std	0	0	5702
		Porcentaje de éxito	0	0	100
f_3	Evaluaciones de función	Media	300000	300000	300000
		Std	0	0	0
		Porcentaje de éxito	0	0	100
	Valor del mínimo encontrado	Media	175.61	133.32	9.274e-11
		Std	37.353	25.255	1.2883e-11
		Mediana	175.11	140.32	8.8285e-11
		Mínimo	114.42	93.526	4.9454e-11
		Máximo	284.56	198.99	9.9988e-11
	Porcentaje de éxito	Media	300000	300000	109090
		Std	0	0	5702
		Porcentaje de éxito	0	0	100
f_4	Valor del mínimo encontrado	Media	0.19338	0.026496	0.01465
		Std	0.62890	0.041673	0.017268
		Mediana	0.053764	0.0086267	0.0098573
		Mínimo	6.6134e-11	8.4732e-11	4.4201e-11
		Máximo	3.4831	0.16705	0.06339
	Evaluaciones de función	Media	237120	231540	247530
		Std	106080	117580	91596
		Porcentaje de éxito	26.667	46.667	36.667
	Porcentaje de éxito	Media	0.13739	0.15377	8.4889e-11
		Std	0.38654	0.27044	1.5237e-11
		Mediana	0.18658	0.062201	9.0373e-11
		Mínimo	8.9502e-11	8.2143e-11	5.1206e-11
		Máximo	1.4352	1.375	9.9798e-11
f_5	Valor del mínimo encontrado	Media	257710	219190	100510
		Std	86126	100790	4965.6
		Evaluaciones de función	20	40	100
		Porcentaje de éxito	0	0	83.333
	Porcentaje de éxito	Media	0.31739	0.15377	8.4889e-11
		Std	0.38654	0.27044	1.5237e-11
		Mediana	0.18658	0.062201	9.0373e-11
		Mínimo	8.9502e-11	8.2143e-11	5.1206e-11
		Máximo	1.4352	1.375	9.9798e-11
f_6	Valor del mínimo encontrado	Media	0.51598	0.015421	8.8983e-11
		Std	1.001	0.043759	1.241e-11
		Mediana	0.016006	9.9798e-11	9.2955e-11
		Mínimo	9.3973e-11	8.7502e-11	5.321e-11
		Máximo	3.5975	0.22243	9.9959e-11
	Evaluaciones de función	Media	230320	196930	115170
		Std	100490	99014	6732.3
		Porcentaje de éxito	33.333	53.333	100
	Porcentaje de éxito	Media	0.51598	0.015421	8.8983e-11
		Std	1.001	0.043759	1.241e-11

Tabla 6.9: Resultados obtenidos en las funciones f_1 , f_2 , f_3 , f_4 , f_5 y f_6 en dimensión 50

Función	Estadístico reportado	Algoritmo			
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI
f_7	Valor del mínimo encontrado	Media	5.6092	2.7309	9.4506e-11
		Std	2.2230	0.847	6.6677e-12
		Mediana	4.7848	2.4506	9.6485e-11
		Mínimo	2.2599	1.2697	7.1116e-11
		Máximo	11.4815	5.0688	9.9939e-11
	Evaluaciones de función	Media	300000	300000	178230
		Std	0	0	4390.4
		Porcentaje de éxito	0	0	100
	Porcentaje de éxito	Media	0.015985	0.016075	0.01138
		Std	0.010776	0.0058698	0.0029532
		Mediana	0.011283	0.014832	0.011123
		Mínimo	0.0061107	0.0070618	0.0059095
		Máximo	0.043696	0.033946	0.017166
f_8	Valor del mínimo encontrado	Media	300000	300000	300000
		Std	0	0	0
		Mediana	0	0	0
		Mínimo	0	0	0
		Máximo	0	0	0
	Evaluaciones de función	Media	85159	83010	152780
		Std	7904.4	6063.1	3281.7
		Porcentaje de éxito	100	100	100
	Porcentaje de éxito	Media	9.5551e-11	9.6473e-11	9.0288e-11
		Std	5.6388e-12	4.8098e-12	1.1572e-11
		Mediana	9.7346e-11	9.8135e-11	9.4448e-11
		Mínimo	7.5253e-11	7.8702e-11	4.2217e-11
		Máximo	9.9930e-11	9.9812e-11	9.9995e-11
f_9	Valor del mínimo encontrado	Media	8.2166e-11	7.267e-11	5.3757e-11
		Std	4.7828e-12	6.4592e-12	1.3263e-11
		Mediana	9.8163e-11	9.7477e-11	9.1952e-11
		Mínimo	5.6093e-11	8.2282e-11	5.1328e-11
		Máximo	9.9915e-11	9.9655e-11	9.9833e-11
	Evaluaciones de función	Media	78873	75525	135490
		Std	9247.4	5912.6	3055.8
		Porcentaje de éxito	100	100	100
	Porcentaje de éxito	Media	9.4996e-11	9.4571e-11	9.104e-11
		Std	8.9971e-12	4.701e-12	1.231e-11
		Mediana	9.8405e-11	9.5799e-11	9.6578e-11
		Mínimo	5.6093e-11	8.2282e-11	5.1328e-11
		Máximo	9.9915e-11	9.9655e-11	9.9833e-11
f_{10}	Valor del mínimo encontrado	Media	77887	72319	132790
		Std	11286	4391.8	2983.5
		Mediana	100	100	100
		Mínimo	100	100	100
		Máximo	100	100	100
	Evaluaciones de función	Media	9.6402e-11	9.4819e-11	8.6897e-11
		Std	4.7828e-12	6.4592e-12	1.3263e-11
		Mediana	9.8163e-11	9.7477e-11	9.1952e-11
	Porcentaje de éxito	Mínimo	8.2166e-11	7.267e-11	5.3757e-11
		Máximo	9.9915e-11	9.9655e-11	9.9833e-11
f_{11}	Valor del mínimo encontrado	Media	77887	72319	132790
		Std	11286	4391.8	2983.5
		Mediana	100	100	100
		Mínimo	100	100	100
		Máximo	100	100	100
	Evaluaciones de función	Media	9.5138e-11	9.3555e-11	5.5433e-11
		Std	4.5679e-12	6.8387e-12	2.9918e-11
		Mediana	9.6120e-11	9.5946e-11	5.9909e-11
	Porcentaje de éxito	Mínimo	8.0677e-11	6.4988e-11	9.6943e-13
		Máximo	9.9780e-11	9.9788e-11	9.857e-11
f_{12}	Porcentaje de éxito	Media	32545	56332	87776
		Std	2452.4	4517.7	4526
		Mediana	100	100	100
		Mínimo	100	100	100
		Máximo	100	100	100
	Evaluaciones de función	Media	9.5138e-11	9.3555e-11	5.5433e-11
		Std	4.5679e-12	6.8387e-12	2.9918e-11
		Mediana	9.6120e-11	9.5946e-11	5.9909e-11
	Porcentaje de éxito	Mínimo	8.0677e-11	6.4988e-11	9.6943e-13
		Máximo	9.9780e-11	9.9788e-11	9.857e-11

Tabla 6.10: Resultados obtenidos en las funciones f_7 , f_8 , f_9 , f_{10} , f_{11} y f_{12} en dimensión 50

Función	Estadístico reportado	Algoritmo				
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI	
f_{13}	Valor del mínimo encontrado	Media	9.3898e-11	9.4759e-11	8.6992e-11	9.4407e-11
		Std	6.9822e-12	5.553e-12	1.4882e-11	4.4972e-12
		Mediana	9.6497e-11	9.651e-11	9.2349e-11	9.5752e-11
		Mínimo	7.3038e-11	7.5061e-11	4.1779e-11	8.4815e-11
		Máximo	9.9997e-11	9.9892e-11	9.9674e-11	9.9725e-11
	Evaluaciones de función	Media	61325	61748	98564	47045
		Std	9485.4	4619.9	2908.5	2289.1
	Porcentaje de éxito		100	100	100	100
f_{14}	Valor del mínimo encontrado	Media	9.2957e-11	9.5609e-11	8.2648e-11	9.5812e-11
		Std	8.8884e-12	5.356e-12	1.8548e-11	3.7961e-12
		Mediana	9.7069e-11	9.6688e-11	8.7023e-11	9.6342e-11
		Mínimo	6.656e-11	7.1159e-11	2.3795e-11	8.0771e-11
		Máximo	9.9951e-11	9.9989e-11	9.9536e-11	9.9927e-11
	Evaluaciones de función	Media	75525	73365	138840	56847
		Std	9318.9	6085.2	3639.8	2249.7
	Porcentaje de éxito		100	100	100	100
f_{15}	Valor del mínimo encontrado	Media	7.8798e-05	154.66	3008.1	0.29256
		Std	0.00020474	143.86	2350.4	0.49372
		Mediana	7.8688e-06	130.72	2149.2	0.13901
		Mínimo	2.8914e-07	11.079	589.19	0.0054277
		Máximo	0.00096457	703.99	8125.5	2.4507
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
	Porcentaje de éxito		0	0	0	0
f_{16}	Valor del mínimo encontrado	Media	26.159	5.9242	8.9665e-11	0.45365
		Std	11.164	4.2985	1.089e-11	0.53584
		Mediana	24.415	4.8612	9.3872e-11	0.45432
		Mínimo	8.9007	0.99818	5.8377e-11	8.8548e-11
		Máximo	56.131	23.544	9.9654e-11	2.1821
	Evaluaciones de función	Media	300000	300000	95014	225310
		Std	0	0	2749.5	116050
	Porcentaje de éxito		0	0	100	30
f_{17}	Valor del mínimo encontrado	Media	1.4099	0.37654	0.41896	0.50988
		Std	0.60989	0.072793	0.081797	0.12958
		Mediana	1.1999	0.39987	0.39987	0.49987
		Mínimo	0.69987	0.29987	0.29987	0.29987
		Máximo	3.6999	0.59987	0.69987	0.79988
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
	Porcentaje de éxito		0	0	0	0

Tabla 6.11: Resultados obtenidos en las funciones f_{13} , f_{14} , f_{15} , f_{16} y f_{17} en dimensión 50

Función	Estadístico reportado	Algoritmo				
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI	
f_1	Valor del mínimo encontrado	Media	9.5014e-11	9.7936e-11	9.118e-11	9.713e-11
		Std	5.228e-12	2.3052e-12	9.8003e-12	3.1957e-12
		Mediana	9.7081e-11	9.874e-11	9.5662e-11	9.8409e-11
		Mínimo	7.9577e-11	9.1373e-11	6.1185e-11	8.6384e-11
		Máximo	9.9993e-11	9.9998e-11	9.99e-11	9.9986e-11
	Evaluaciones de función	Media	184360	136140	184390	78942
		Std	20819	7191.6	7707.1	1831.5
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	97.545	128.35	108.97	129.21
		Std	68.413	54.057	54.773	61.084
		Mediana	90.346	116.6	97.175	131.7
		Mínimo	2.2121	45.415	0.34486	9.8624
		Máximo	330.63	301.52	230.52	254.08
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
f_3	Valor del mínimo encontrado	Media	333.11	296.84	0.033165	319.72
		Std	64.087	53.787	0.18165	50.503
		Mediana	317.89	284.06	9.2882e-11	323.82
		Mínimo	225.86	205.96	4.6384e-11	219.47
		Máximo	523.35	415.89	0.99496	423.76
	Evaluaciones de función	Media	300000	300000	181460	300000
		Std	0	0	30863	0
		Porcentaje de éxito	0	0	96.667	0
	Valor del mínimo encontrado	Media	0.38853	0.10075	0.017779	0.0018067
		Std	0.84167	0.17414	0.028036	0.0043363
		Mediana	0.10251	0.025805	0.0086267	9.7647e-11
		Mínimo	9.4425e-11	9.3186e-11	6.3539e-11	8.8481e-11
		Máximo	4.0486	0.7039	0.10057	0.01478
		Evaluaciones de función	285810	251180	245860	116460
f_5	Valor del mínimo encontrado	Std	38415	76135	63072	83555
		Porcentaje de éxito	13.333	30	43.333	83.333
	Valor del mínimo encontrado	Media	0.3792	0.22541	3.2171e-08	0.050929
		Std	0.58514	0.32917	1.7355e-07	0.1083
		Mediana	0.05831	0.038876	9.8131e-11	9.9144e-11
		Mínimo	9.3389e-11	7.9644e-11	6.9732e-11	8.4912e-11
		Máximo	2.5554	1.2552	9.5103e-07	0.51623
	Evaluaciones de función	Media	289190	259390	202430	173700
		Std	24467	51289	64615	97965
		Porcentaje de éxito	20	40	80	63.333
f_6	Valor del mínimo encontrado	Media	0.5825	0.14225	6.6704e-10	0.21401
		Std	0.87922	0.65676	1.8875e-09	0.71018
		Mediana	0.054935	0.010987	9.6027e-11	9.9393e-11
		Mínimo	6.4261e-11	8.9372e-11	7.1361e-11	8.6532e-11
		Máximo	3.5975	3.5975	9.5811e-09	3.5975
	Evaluaciones de función	Media	286740	251910	228380	183000
		Std	29916	56950	53979	97612
		Porcentaje de éxito	20	43.333	83.333	60

Tabla 6.12: Resultados obtenidos en las funciones f_1 , f_2 , f_3 , f_4 , f_5 y f_6 en dimensión 80

Función	Estadístico reportado	Algoritmo			
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI
f_7	Valor del mínimo encontrado	Media	11.383	5.3988	1.4368e-10
		Std	1.9274	1.4832	1.7672e-10
		Mediana	11.721	5.299	9.8441e-11
		Mínimo	7.9139	2.4522	7.761e-11
		Máximo	15.561	8.3554	8.5609e-10
	Evaluaciones de función	Media	300000	300000	290760
		Std	0	0	5455.2
		Porcentaje de éxito	0	0	90
	Porcentaje de éxito	Media	0.1396	0.052962	0.028947
		Std	0.28354	0.015859	0.010164
		Mediana	0.058004	0.049488	0.025963
		Mínimo	0.014638	0.032116	0.016061
		Máximo	1.5585	0.092766	0.066586
f_8	Valor del mínimo encontrado	Media	300000	300000	300000
		Std	0	0	0
		Mediana	0	0	0
		Mínimo	0	0	0
		Máximo	0	0	0
	Evaluaciones de función	Media	300000	300000	300000
		Std	0	0	0
		Porcentaje de éxito	0	0	0
	Porcentaje de éxito	Media	9.4595e-11	9.531e-11	9.1039e-11
		Std	1.1411e-11	1.073e-11	1.0163e-11
		Mediana	9.7153e-11	9.7507e-11	9.3993e-11
		Mínimo	4.8343e-11	4.052e-11	6.449e-11
		Máximo	1.1993e-10	9.9926e-11	9.9969e-11
f_9	Valor del mínimo encontrado	Evaluaciones de función	Media	236670	172590
		Std	29984	11834	5434.5
		Porcentaje de éxito	96.667	100	100
		Media	9.6138e-11	9.6184e-11	9.2882e-11
		Std	1.1797e-11	4.6081e-12	8.3482e-12
	Evaluaciones de función	Mediana	9.6368e-11	9.8085e-11	9.6299e-11
		Mínimo	7.3372e-11	8.2513e-11	7.053e-11
		Máximo	1.4962e-10	9.9995e-11	9.9998e-11
		Media	222300	161890	219330
		Std	28618	10452	4784.9
f_{10}	Porcentaje de éxito	Porcentaje de éxito	96.667	100	100
		Media	9.5904e-11	9.6847e-11	8.9434e-11
		Std	6.0603e-12	5.2144e-12	1.3669e-11
		Mediana	9.8978e-11	9.8682e-11	9.5e-11
		Mínimo	7.9552e-11	8.0194e-11	4.5126e-11
	Evaluaciones de función	Máximo	9.999e-11	9.9958e-11	9.9919e-11
		Media	213060	149770	216870
		Std	29248	9136.4	4215.8
		Porcentaje de éxito	100	100	100
		Media	9.5739e-11	9.6874e-11	5.5898e-11
f_{11}	Valor del mínimo encontrado	Std	6.5004e-12	4.7169e-12	3.2757e-11
		Mediana	9.7763e-11	9.8919e-11	6.03e-11
		Mínimo	6.6289e-11	7.8421e-11	1.7997e-12
		Máximo	9.9983e-11	9.9995e-11	9.9628e-11
	Evaluaciones de función	Evaluaciones de función	Media	77099	117700
		Std	8412.8	7278.4	6786
		Porcentaje de éxito	100	100	100
		Media	77099	117700	135550
		Std	8412.8	7278.4	2334.3
f_{12}	Porcentaje de éxito	Porcentaje de éxito	100	100	100
		Media	9.5739e-11	9.6874e-11	5.5898e-11
		Std	6.5004e-12	4.7169e-12	3.2757e-11
		Mediana	9.7763e-11	9.8919e-11	6.03e-11
		Mínimo	6.6289e-11	7.8421e-11	1.7997e-12
	Evaluaciones de función	Máximo	9.9983e-11	9.9995e-11	9.9628e-11
		Media	77099	117700	135550
		Std	8412.8	7278.4	2334.3
		Porcentaje de éxito	100	100	100

Tabla 6.13: Resultados obtenidos en las funciones f_7 , f_8 , f_9 , f_{10} , f_{11} y f_{12} en dimensión 80

Función	Estadístico reportado	Algoritmo				
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI	
f_{13}	Valor del mínimo encontrado	Media	9.3651e-11	9.791e-11	9.2255e-11	9.4871e-11
		Std	1.4173e-11	2.8395e-12	9.2618e-12	6.502e-12
		Mediana	9.8923e-11	9.9115e-11	9.5792e-11	9.678e-11
		Mínimo	2.5468e-11	8.9515e-11	5.9104e-11	6.802e-11
		Máximo	9.9974e-11	9.9974e-11	9.9926e-11	9.9986e-11
	Evaluaciones de función	Media	167990	123900	152990	71314
		Std	23831	7453.7	4934.9	2466.6
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	1.3482e-10	9.6294e-11	8.6895e-11	9.63e-11
		Std	2.4588e-10	4.194e-12	1.7439e-11	3.9019e-12
		Mediana	9.7819e-11	9.7849e-11	9.389e-11	9.6932e-11
		Mínimo	3.128e-11	8.2769e-11	3.1692e-11	8.2177e-11
		Máximo	1.4336e-09	9.9963e-11	9.9988e-11	9.9957e-11
	Evaluaciones de función	Media	216040	150280	226480	88087
		Std	27548	10045	6099.3	2773.5
		Porcentaje de éxito	96.667	100	100	100
f_{15}	Valor del mínimo encontrado	Media	178.8	5015	4866.7	72.179
		Std	105.55	2360.6	3571.1	27.93
		Mediana	166.11	4393.1	2641.5	69.176
		Mínimo	41.606	1997.8	1205.9	25.69
		Máximo	494.94	10121	14211	169
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
	Valor del mínimo encontrado	Media	77.631	48.032	9.032e-11	2.5126
		Std	15.678	14.355	1.1397e-11	1.5162
		Mediana	78.371	50.153	9.455e-11	2.2716
		Mínimo	49.215	24.813	5.918e-11	0.089528
		Máximo	120.21	77.136	9.9756e-11	5.9957
	Evaluaciones de función	Media	300000	300000	142960	300000
		Std	0	0	7936.4	0
		Porcentaje de éxito	0	0	100	0
f_{17}	Valor del mínimo encontrado	Media	4.7132	0.74987	0.95444	0.96321
		Std	1.7025	0.19957	0.13582	0.22664
		Mediana	4.6999	0.69987	0.90456	0.89987
		Mínimo	1.5999	0.49987	0.69987	0.69987
		Máximo	7.9999	1.4999	1.3	1.6999
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0

Tabla 6.14: Resultados obtenidos en las funciones f_{13} , f_{14} , f_{15} , f_{16} y f_{17} en dimensión 80

Función	Estadístico reportado	Algoritmo			
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI
f_1	Valor del mínimo encontrado	Media	1.2977e-08	9.6918e-11	9.3633e-11
		Std	4.4644e-08	3.0783e-12	2.2314e-11
		Mediana	8.3272e-10	9.7863e-11	9.372e-11
		Mínimo	7.0165e-11	8.7894e-11	6.2509e-11
		Máximo	2.4153e-07	9.9982e-11	1.9827e-10
	Evaluaciones de función	Media	295320	189880	245830
		Std	10980	8704.2	17249
		Porcentaje de éxito	23.333	100	96.667
f_2	Valor del mínimo encontrado	Media	178.05	189.86	8.9864e+07
		Std	70.063	67.028	2.3392e+08
		Mediana	173.59	167.7	196.67
		Mínimo	46.723	100.14	46.065
		Máximo	342.16	395.85	7.4664e+08
	Evaluaciones de función	Media	300000	300000	300000
		Std	0	0	0
		Porcentaje de éxito	0	0	0
f_3	Valor del mínimo encontrado	Media	464.11	406.46	8.0793e-05
		Std	69.615	63.354	0.00043098
		Mediana	459.17	407.43	9.7941e-11
		Mínimo	336.3	250.73	4.6612e-11
		Máximo	649.7	523.39	0.0023619
	Evaluaciones de función	Media	300000	300000	246380
		Std	0	0	34902
		Porcentaje de éxito	0	0	80
f_4	Valor del mínimo encontrado	Media	1.1738	0.047488	0.011055
		Std	3.095	0.079407	0.013813
		Mediana	0.20111	9.9867e-11	0.003698
		Mínimo	9.8474e-11	7.6451e-11	8.1135e-11
		Máximo	12.494	0.25244	0.044058
	Evaluaciones de función	Media	298330	243740	261090
		Std	6384.2	54571	40283
		Porcentaje de éxito	6.6667	53.333	50
f_5	Valor del mínimo encontrado	Media	0.44391	0.22779	3.0758e+08
		Std	0.60608	0.35927	5.7765e+08
		Mediana	0.21854	0.077742	4.3192e-06
		Mínimo	6.111e-10	9.365e-11	8.5322e-11
		Máximo	2.6094	1.4378	1.6603e+09
	Evaluaciones de función	Media	300000	286540	249350
		Std	0	25832	57574
		Porcentaje de éxito	0	26.667	46.667
f_6	Valor del mínimo encontrado	Media	0.90058	0.16155	2.2094e+08
		Std	1.2038	0.50732	7.5079e+08
		Mediana	0.11064	9.9834e-11	4.0365e-07
		Mínimo	3.8118e-08	8.5916e-11	5.2474e-11
		Máximo	3.6085	2.4975	2.9489e+09
	Evaluaciones de función	Media	300000	272030	274090
		Std	0	27434	36670
		Porcentaje de éxito	0	53.333	36.667

Tabla 6.15: Resultados obtenidos en las funciones f_1 , f_2 , f_3 , f_4 , f_5 y f_6 en dimensión 100

Función	Estadístico reportado	Algoritmo			
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI
f_7	Valor del mínimo encontrado	Media	13.925	7.9243	2.0596e-08
		Std	2.2936	2.3625	2.7712e-08
		Mediana	14.287	8.3843	7.3042e-09
		Mínimo	8.8891	4.6389	3.2147e-09
		Máximo	18.806	11.568	9.1455e-11
	Evaluaciones de función	Media	300000	300000	300000
		Std	0	0	75467
		Porcentaje de éxito	0	0	50
	Evaluaciones de función	Media	0.30915	0.085669	0.047993
		Std	0.36868	0.027299	0.013105
		Mediana	0.14911	0.078224	0.044249
		Mínimo	0.046934	0.035394	0.026423
		Máximo	1.3807	0.13752	0.071993
f_8	Valor del mínimo encontrado	Media	300000	300000	300000
		Std	0	0	0
		Porcentaje de éxito	0	0	0
		Media	0.30915	0.085669	0.074107
		Std	0.36868	0.027299	0.017076
	Evaluaciones de función	Mediana	0.14911	0.078224	0.044249
		Mínimo	0.046934	0.035394	0.026423
		Máximo	1.3807	0.13752	0.071993
		Media	300000	300000	300000
		Std	0	0	0
f_9	Valor del mínimo encontrado	Media	0.00039943	9.5471e-11	7.5628e-09
		Std	0.0010706	7.9514e-12	1.6563e-08
		Mediana	2.4475e-06	9.8632e-11	1.9958e-09
		Mínimo	5.424e-09	6.3576e-11	5.1454e-10
		Máximo	0.0045016	9.9941e-11	8.9868e-08
	Evaluaciones de función	Media	300000	230870	300000
		Std	0	10396	0
		Porcentaje de éxito	0	100	100
	Evaluaciones de función	Media	3.9435e-05	9.6701e-11	1.0968e-10
		Std	0.00021069	5.1994e-12	9.1622e-11
		Mediana	1.9433e-08	9.8423e-11	9.5994e-11
		Mínimo	9.9995e-11	7.3362e-11	6.0027e-11
		Máximo	0.0011549	9.995e-11	5.9265e-10
f_{10}	Valor del mínimo encontrado	Media	299390	213990	288100
		Std	3324.7	9644.3	5877
		Porcentaje de éxito	3.3333	100	96.667
		Media	3.9435e-05	9.6701e-11	9.7382e-11
		Std	0.00021069	5.1994e-12	2.4277e-12
	Evaluaciones de función	Mediana	1.9433e-08	9.8423e-11	9.8289e-11
		Mínimo	9.9995e-11	7.3362e-11	9.1258e-11
		Máximo	0.0011549	9.995e-11	9.9938e-11
		Media	299390	213990	110130
		Std	3324.7	9644.3	3295.5
f_{11}	Valor del mínimo encontrado	Porcentaje de éxito	3.3333	100	100
		Media	7.8763e-07	9.767e-11	8.904e-11
		Std	2.5049e-06	2.6109e-12	1.6676e-11
		Mediana	1.6092e-09	9.8682e-11	9.5327e-11
		Mínimo	9.2468e-11	8.7358e-11	3.0734e-11
	Evaluaciones de función	Máximo	1.2165e-05	9.9886e-11	9.9887e-11
		Media	297740	203160	281000
		Std	8971.2	8468.6	7557.7
		Porcentaje de éxito	13.333	100	100
		Media	7.8763e-07	9.767e-11	105340
f_{12}	Valor del mínimo encontrado	Std	2.5049e-06	2.6109e-12	4.1261e-12
		Mediana	1.6092e-09	9.8682e-11	9.783e-11
		Mínimo	9.2468e-11	8.7358e-11	8.4636e-11
		Máximo	1.2165e-05	9.9886e-11	9.9944e-11
	Evaluaciones de función	Media	122220	164840	171450
		Std	10523	11253	13433
		Porcentaje de éxito	100	100	100
		Media	9.6895e-11	9.6877e-11	5.1966e-11
		Std	3.7721e-12	4.4269e-12	2.6175e-11

Tabla 6.16: Resultados obtenidos en las funciones f_7 , f_8 , f_9 , f_{10} , f_{11} y f_{12} en dimensión 100

Función	Estadístico reportado	Algoritmo				
		PSO Canónico	H. D-LPSO	PSO-ITC	PSO-MI	
f_{13}	Valor del mínimo encontrado	Media	3.3138e-08	9.3674e-11	8.9994e-11	9.7295e-11
		Std	1.6461e-07	1.2e-11	1.344e-11	3.282e-12
		Mediana	9.9367e-11	9.7071e-11	9.5898e-11	9.8285e-11
		Mínimo	6.8762e-11	3.7681e-11	4.5452e-11	8.6662e-11
		Máximo	9.0184e-07	9.9887e-11	9.9987e-11	9.9817e-11
	Evaluaciones de función	Media	278070	174720	196390	85561
		Std	21213	8343.7	5827.2	2763
	Porcentaje de éxito		66.667	100	100	100
	Valor del mínimo encontrado	Media	5.1403e-08	9.7534e-11	1.174e-10	9.7458e-11
		Std	2.5956e-07	4.624e-12	3.7194e-11	2.6417e-12
		Mediana	6.539e-10	9.8892e-11	9.9513e-11	9.8372e-11
		Mínimo	9.2631e-11	7.5704e-11	7.9952e-11	8.737e-11
		Máximo	1.425e-06	9.9991e-11	2.1726e-10	9.9999e-11
	Evaluaciones de función	Media	292530	204260	295780	107180
		Std	14322	9230.1	5291.4	4210.4
		Porcentaje de éxito	36.667	100	66.667	100
f_{15}	Valor del mínimo encontrado	Media	1115.3	8094	12164	410.63
		Std	391.45	2548.7	12425	124.62
		Mediana	1165.1	7778.3	6903.5	381.56
		Mínimo	435.58	3731.4	1757.7	228.44
		Máximo	1748.5	13503	43046	640.01
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
	Valor del mínimo encontrado	Media	113.42	76.941	8.9206e-11	4.1805
		Std	27.236	22.986	1.3777e-11	2.3698
		Mediana	111.66	75.759	9.4388e-11	3.6793
		Mínimo	69.399	35.295	4.999e-11	0.72291
		Máximo	197.87	123.46	9.9596e-11	10.488
	Evaluaciones de función	Media	300000	300000	191450	300000
		Std	0	0	30562	0
		Porcentaje de éxito	0	0	100	0
f_{17}	Valor del mínimo encontrado	Media	5.5799	0.96341	1.4476	1.4732
		Std	2.0131	0.16714	0.18737	0.35518
		Mediana	5.3999	0.89987	1.3999	1.3999
		Mínimo	2.5999	0.79987	1.0999	0.99987
		Máximo	12.1	1.4999	1.8088	2.2999
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0

Tabla 6.17: Resultados obtenidos en las funciones f_{13} , f_{14} , f_{15} , f_{16} y f_{17} en dimensión 100

6.4 Comparación de algoritmos

En la presente sección hacemos una comparación entre los algoritmos utilizados para resolver las funciones de prueba, con la finalidad de evaluar el desempeño del algoritmo propuesto. Esta comparación se presenta con base en los resultados mostrados en las tablas de la sección 6.3. En las figuras de la presente sección mostramos tres tipos de comparaciones. En el primer caso comparamos las evaluaciones de función necesarias para alcanzar el óptimo, cuando la mayoría de los algoritmos alcanzó el mínimo con la tolerancia permitida (figuras 6.2 y 6.3). En este caso, se considera que un algoritmo es mejor que otro cuando necesita un menor número de evaluaciones de función para alcanzar el óptimo. En el segundo caso, comparamos de acuerdo con el porcentaje de éxito de cada algoritmo. El porcentaje de éxito es obtenido dividiendo el número de veces que un algoritmo alcanzó el óptimo global entre el número total de corridas (en nuestro caso 30) y multiplicando este cociente por 100. Esta comparación se muestra en la figura 6.4. En este caso, un algoritmo es mejor que otro si tiene un mayor porcentaje de éxito. El último tipo de comparación (figura 6.5) que hacemos es con base en la media del valor de función de aptitud “alcanzada” por cada algoritmo. Esta comparación la utilizamos cuando ningún algoritmo (o sólo uno) alcanzó el valor conocido del mínimo global. En este caso, se reporta el promedio del mínimo valor encontrado por cada algoritmo en las 30 corridas. Un algoritmo se considera mejor que otro cuando encuentra un valor más pequeño de la función de aptitud.

Como se puede ver en las figuras 6.2 y 6.3, generalmente PSO-MI necesita menos evaluaciones de función para alcanzar el óptimo. Este comportamiento se puede observar con mayor notoriedad cuando las funciones aumentan de dimensión. Consideraremos que esta característica es de vital importancia para el algoritmo, ya que la dimensionalidad es uno de los problemas más importantes que tienen los algoritmos de optimización. Debemos hacer notar que la mayoría de las funciones en este caso son convexas, por lo que se puede considerar que el algoritmo muestra un buen desempeño en este tipo de funciones, quizás por los parámetros que asignamos al algoritmo, especialmente el historial de las partículas. En el caso de las funciones Penalized 1, Penalized 2 y Ackley (figuras 6.2b, 6.2c y 6.2d; respectivamente), pese a que son funciones multimodales, se obtienen buenos resultados. Ahora bien, en el caso de las funciones comparadas en la figura 6.4, se puede observar, por lo general, un mejor desempeño del algoritmo PSO-ITC. Aunque es importante mencionar que en las funciones Penalized 1, Penalized 2 y Ackley (figuras 6.4c, 6.4d y 6.4e; respectivamente), pese a que PSO-ITC tiene, ligeramente,

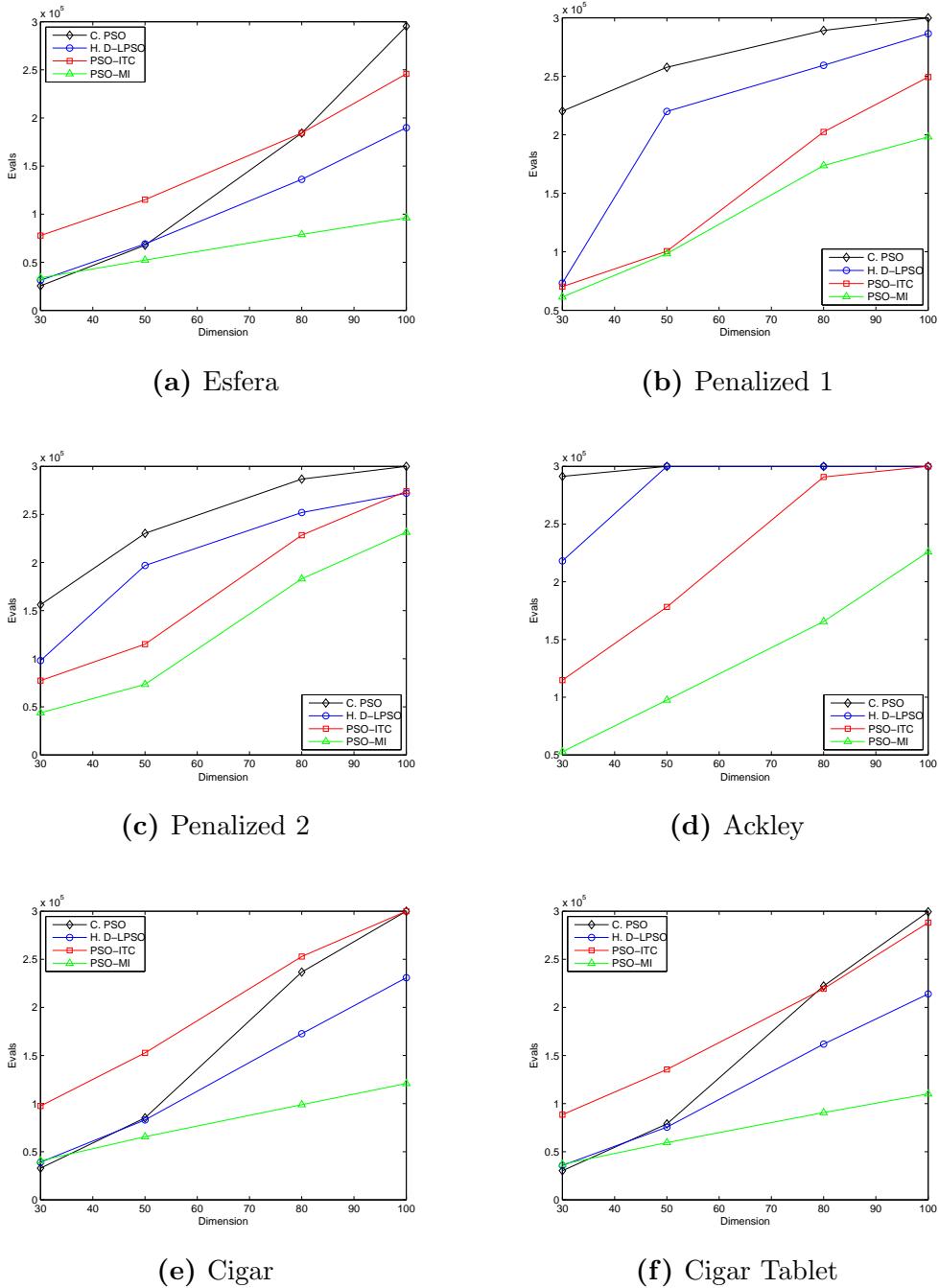


Figura 6.2: Comparación del número de evaluaciones de función requeridas por cada algoritmo para alcanzar el óptimo (parte 1)

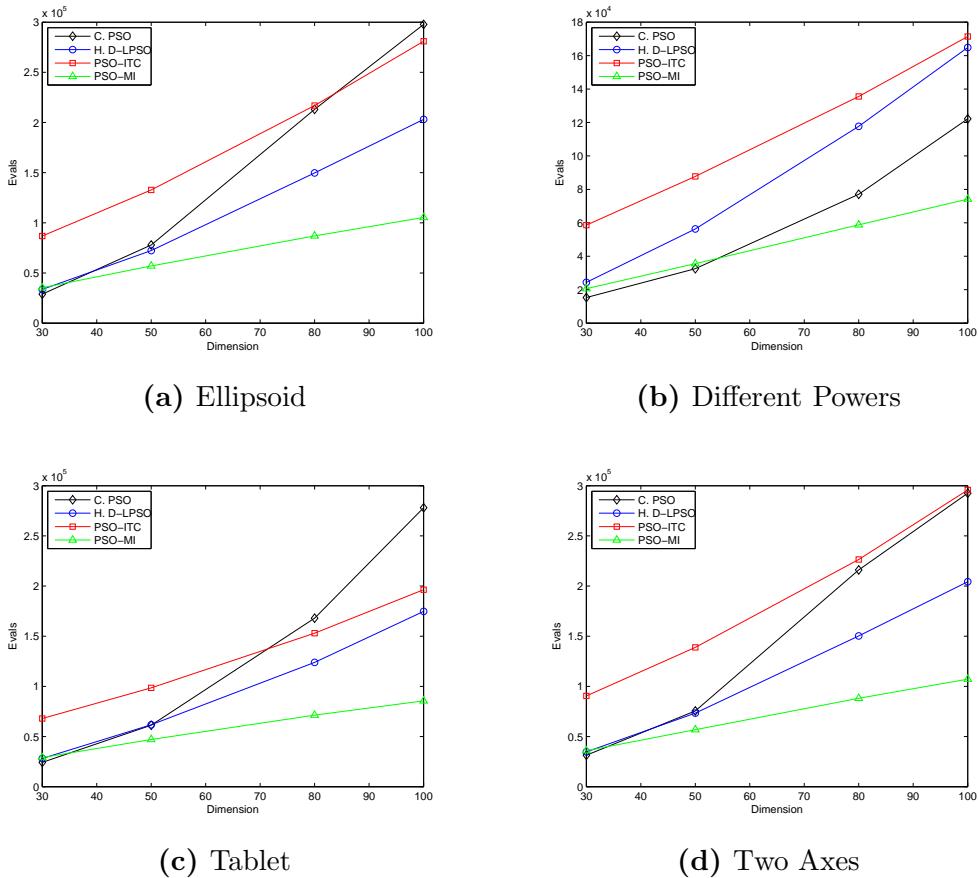


Figura 6.3: Comparación del número de evaluaciones de función requeridas por cada algoritmo para alcanzar el óptimo (parte 2)

un mayor porcentaje de éxito (ventaja que se pierde en dimensión 100), PSO-MI requiere un menor número de evaluaciones de función para alcanzar el óptimo (figuras 6.2b, 6.2c y 6.2d). Por esta razón consideramos que nuestra propuesta presenta un muy buen desempeño en estas funciones. Por último, en el caso de las funciones de la figura 6.5, se puede observar que ningún algoritmo (salvo para la función Zakharov en dimensión 30) pudo obtener el óptimo global de las funciones, obteniendo resultados muy similares en todos los casos, aunque de nuevo hay que hacer notar que en dimensión 100 en las funciones Rosenbrock y Zakharov (figuras 6.5a y 6.5c; respectivamente) se obtiene un valor más pequeño.

En la figura 6.6 se muestra la media de la población y el valor de aptitud del mejor individuo encontrado durante las primera 2000 generaciones de la función Esfera en dimensión 100, utilizando los parámetros mencionados

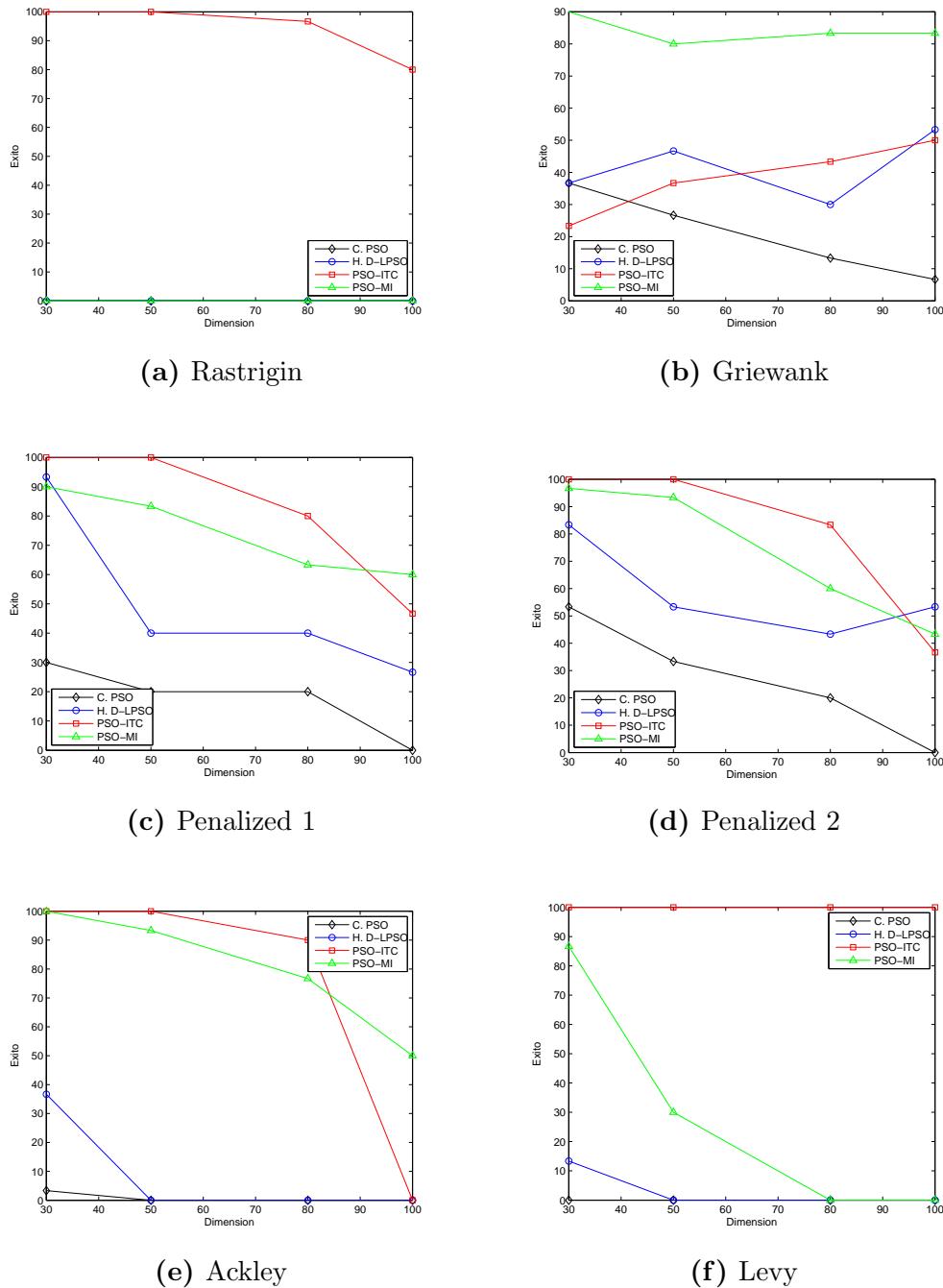
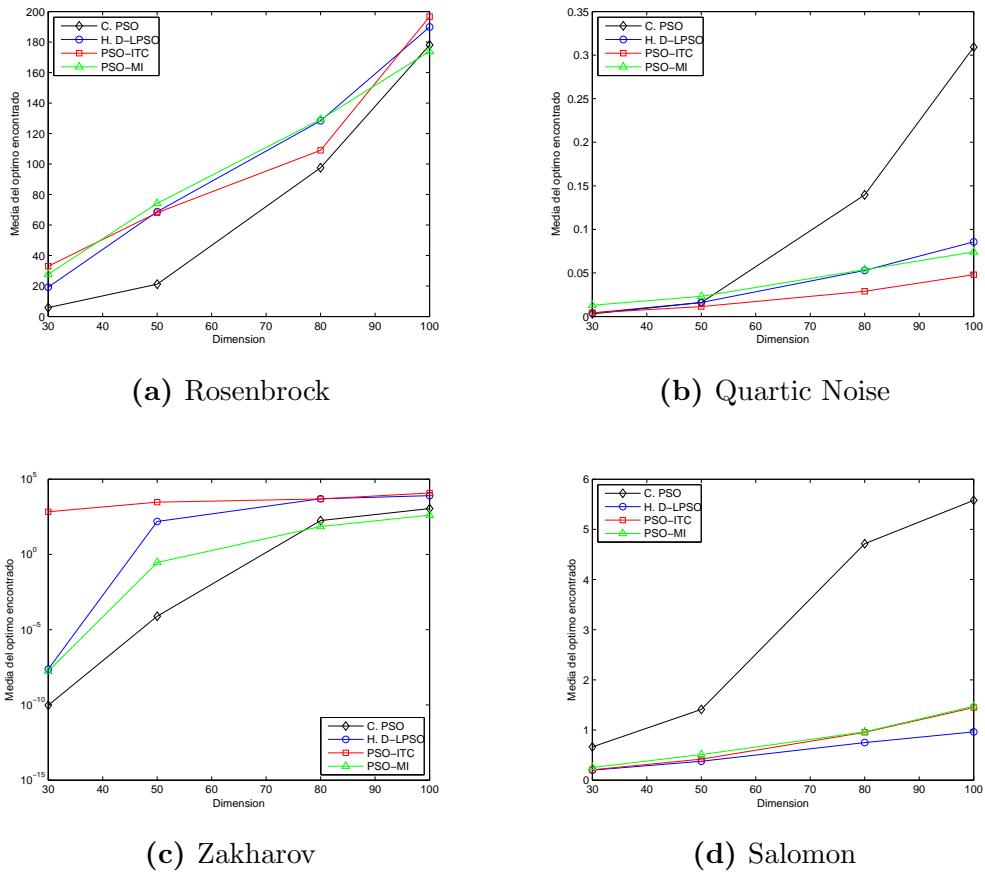


Figura 6.4: Comparación del porcentaje de éxito de cada algoritmo



(a) Rosenbrock

(b) Quartic Noise

(c) Zakharov

(d) Salomon

Figura 6.5: Comparación de la media de la función de aptitud alcanzada por cada algoritmo

en la sección 6.3. Como se puede observar en la figura, cuando se utiliza PSO-MI el descenso de las partículas es más rápido, ya que la media de la población y el mejor individuo tienen valores muy cercanos, lo que implica que las partículas se mueven en conjunto hacia el óptimo. En el caso de PSO-ITC, se puede ver que, pese a que el mejor valor de aptitud encontrado desciende a una buena velocidad, la media de la población se encuentra en un valor “alto”, lo que implica que las partículas se encuentran en una etapa de exploración, cuando deberían tener prioridad a la etapa de explotación, por el tipo de función que se está resolviendo. En el caso de H. D-LPSO, el mejor individuo mejora constantemente con el paso de las generaciones, pero la media de la población mejora cada 400 generaciones, cuando la conectividad de los árboles aumenta, lo que implica que existe una mejora en la población cuando la topología cambia, pero después se produce un estancamiento de las partículas.

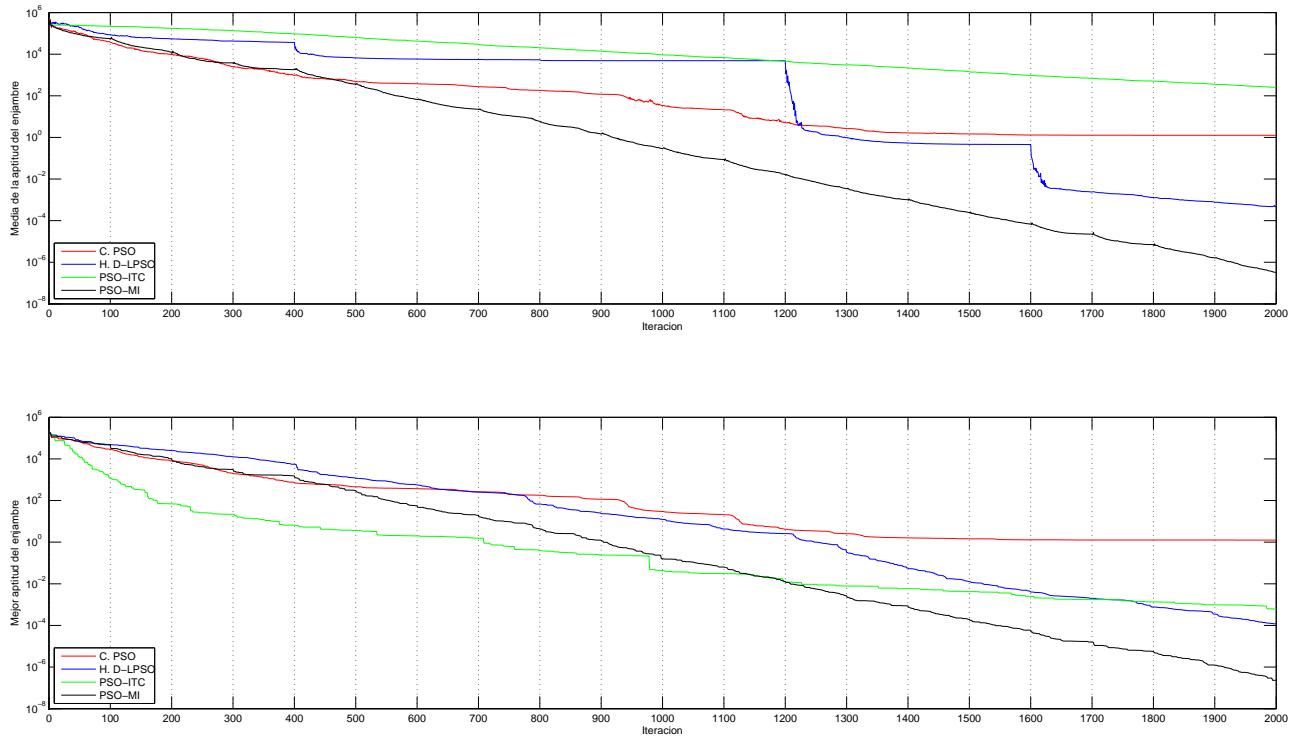


Figura 6.6: Evolución de la media de la población y del mejor individuo encontrado para los algoritmos comparados durante las primeras 2000 generaciones para la función Esfera en dimensión 100

6.5 Impacto del uso de la información mutua

En el presente experimento se buscó demostrar la conveniencia de utilizar la información mutua como una medida de dependencia entre las partículas para construir las topologías. En este, se compararon los resultados de PSO con una topología de anillo (estática); PSO con una topología de anillos aleatorios y diferente para cada dimensión (emulando la construcción de las topologías con PSO-MI, asignada al principio del algoritmo y manteniéndose estática durante el desarrollo de este); y un anillo MIMIC, utilizando PSO-MI actualizando las topologías con el algoritmo 12 (MIMIC), pero uniendo los extremos de las cadenas formados para crear un anillo en cada dimensión. Los parámetros utilizados fueron $\omega = 0.578766$, $c_1 = c_2 = 1.49618$, $S = 30$; y para el caso de PSO-MI se utilizaron $f = 50$, $h = 50$ y topología inicial todos desconectados. Se probaron las funciones Esfera en dimensión 30, 50, 80 y 100 (tabla 6.18) y Griewank en dimensión 30 (tabla 6.19). El algoritmo se detuvo cuando se obtuvo el óptimo global con una tolerancia $\epsilon = 10^{-10}$ o se alcanzaron las 300000 evaluaciones de función. Para cada caso se reportan los resultados de 30 corridas.

Como se puede observar en la tabla 6.19, para el caso de la función Griewank se obtiene un mayor porcentaje de éxito cuando se utilizan los anillos construidos con MIMIC, mientras que, cuando se utilizan los anillos aleatorios no se obtienen muy buenos resultados. En el caso de la función Esfera, los resultados entre el anillo tradicional y el anillo MIMIC son muy similares, aunque cuando la dimensionalidad aumenta el anillo MIMIC empieza a converger un poco más rápido. Por otro lado, cuando se utilizan anillos aleatorios por dimensión, los resultados son malos, aún más en dimensiones grandes, en donde ya no converge. De esta forma podemos observar que el uso de la información mutua es benéfico para las partículas, aumentando la velocidad de convergencia de estas, incluso en funciones multimodales como la Griewank.

Otros experimentos que se realizaron en donde se muestra la influencia de la información mutua de las topologías en el comportamiento de las partículas (y viceversa), se muestran en el anexo D.

Dimensión	Estadístico reportado	Topología		
		Anillo	Anillo aleatorio	Anillo MIMIC
30	Valor del mínimo encontrado	Media	9.3293e-11	9.3189e-11
		Std	6.1785e-12	7.0987e-12
		Mediana	9.4162e-11	9.5946e-11
		Mínimo	7.1764e-11	7.198e-11
		Máximo	9.9652e-11	9.9723e-11
	Evaluaciones de función	Media	24717	33144
		Std	520.47	6912.1
	Porcentaje de éxito		100	100
	Valor del mínimo encontrado	Media	9.4321e-11	358.72
		Std	4.5444e-12	1964.7
		Mediana	9.5383e-11	9.8515e-11
		Mínimo	8.2301e-11	7.8372e-11
		Máximo	9.9783e-11	10761
50	Valor del mínimo encontrado	Media	39925	163820
		Std	742.2	102730
		Porcentaje de éxito	100	66.667
		Media	33144	38994
		Std	100	526.17
	Valor del mínimo encontrado	Media	9.7149e-11	3715.1
		Std	2.8859e-12	6673.3
		Mediana	9.7837e-11	340.36
		Mínimo	8.6452e-11	9.9958e-11
		Máximo	9.9847e-11	25657
80	Valor del mínimo encontrado	Media	62963	297190
		Std	1611.4	15402
		Porcentaje de éxito	100	3.3333
		Media	9.8225e-11	9.4819e-11
		Std	2.8482e-12	5.6434e-12
	Valor del mínimo encontrado	Mediana	9.8906e-11	568.57
		Mínimo	8.4443e-11	0.094743
		Máximo	9.9982e-11	37873
		Evaluaciones de función	Media	58971
		Std	1215	100
100	Porcentaje de éxito		100	100
	Valor del mínimo encontrado	Media	80540	300000
		Std	2268	0
		Porcentaje de éxito	100	0
		Media	71762	1310.9
	Evaluaciones de función	Std	100	100

Tabla 6.18: Resultados obtenidos con topologías de anillo, anillo aleatorio y anillo MIMIC en la función Esfera

Dimensión	Estadístico reportado	Topología		
		Anillo	Anillo aleatorio	Anillo MIMIC
30	Valor del mínimo encontrado	Media	0.0027919	0.038482
		Std	0.0053629	0.19011
		Mediana	9.855e-11	0.00032305
		Mínimo	8.2954e-11	8.1918e-11
		Máximo	0.017241	1.0444
	Evaluaciones de función	Media	96847	197920
		Std	116840	122380
	Porcentaje de éxito		76.667	46.667
				96.667

Tabla 6.19: Resultados obtenidos con topologías de anillo, anillo aleatorio y anillo MIMIC en la función Griewank

Capítulo 7

Conclusiones y trabajo a futuro

Uno de los principales retos, y a su vez una gran área de oportunidad de optimización por enjambre de partículas, es idear cómo construir las topologías de las partículas de tal forma que estas provoquen un mejor desempeño del algoritmo. Diversos estudios han demostrado que las topologías dinámicas, con una adecuada selección de parámetros, permiten a PSO obtener mejores resultados en la búsqueda del óptimo de una función. Debido a esto, consideramos que proponer una nueva forma de conectar a las partículas es una aportación pertinente a la investigación en este tópico. Gran parte de los trabajos realizados en PSO están orientados a proporcionar a las partículas “inteligencia” en su búsqueda del óptimo, pero no en proporcionarles “inteligencia” para saber con qué otras partículas comunicarse. Los modelos gráficos son una herramienta que sirven para extraer y aprender información útil de un conjunto de datos, y que a su vez indican dependencias entre un conjunto de variables aleatorias. Una de las medidas ampliamente utilizada para obtener estas dependencias es la información mutua. Entonces, cuando se construyen las topologías utilizando modelos gráficos que maximizan la información mutua, se están conectando partículas con alta dependencia entre ellas. Además, con estas topologías se le esta dotando a las partículas de un aprendizaje de la función a optimizar.

Uno de los principales problemas a los que se enfrentan en general los algoritmos de optimización es la dimensionalidad. PSO no esta exento de este problema, y es importante buscar los parámetros adecuados (incluida la topología) de tal forma que el algoritmo pueda afrontar problemas de alta dimensionalidad de una mejor forma. De la presente tesis se puede concluir que PSO-MI tiene un buen desempeño, pero su velocidad de convergencia es más notable en funciones de alta dimensión, en comparación con los demás algoritmos probados en este trabajo de investigación. Es por esto que consideramos a nuestra propuesta como una aportación interesante, ya que algunas ideas aquí expuestas pueden ayudar a proponer mejores formas de enfrentarse a la dimensionalidad de las funciones. También consideramos interesante el

hecho de construir una topología distinta para cada dimensión, ya que lo común en las implementaciones de PSO es tener la misma topología para todas las dimensiones.

Es importante mencionar que aunque se obtuvieron resultados muy favorables en varias de las funciones de prueba, en algunas funciones multimodales los resultados no fueron del todo buenos (ni malos, en comparación con los demás algoritmos). Entonces, es apropiado en un futuro realizar mejoras al algoritmo de tal forma que se puedan resolver una mayor cantidad de funciones multimodales.

Algunos experimentos realizados parecen mostrar que existe relación entre la información mutua de los vecindarios y la convergencia del algoritmo, aunque sin duda alguna habría que realizar más pruebas sobre ello, e incluso, realizar demostraciones formales para mostrar esto.

Aunque la construcción de las topologías utilizando modelos de cadena y árbol que maximizan la información mutua mostró una influencia benéfica en el algoritmo, se debe considerar construir los vecindarios utilizando otros modelos gráficos y otras medidas de dependencia entre las variables aleatorias. También, se pueden probar otros parámetros de PSO-MI para encontrar que valores de estos producen una mejora del algoritmo, dependiendo de la función que se desea optimizar.

De igual forma, se debe considerar que el construir una topología para cada dimensión es muy costoso computacionalmente hablando, por lo que se deben buscar formas de construir topologías en común para un cierto número de variables, de preferencia aquellas que tengan dependencias entre sí, o inclusive, una topología para todas las dimensiones, como normalmente trabaja PSO.

En general, podemos concluir que el utilizar topologías que maximizan la información mutua de las partículas resultó benéfico en la velocidad de convergencia de las partículas, principalmente en funciones unimodales de alta dimensión.

Bibliografía

- [Abi01] Mohammad Ali Abido. Particle swarm optimization for multimachine power system stabilizer design. *Power Engineering Society Summer Meeting*, 3:1346–1351, Julio 2001.
(Referenciado en la página 36.)
- [AMLC07] Wilfredo Alfonso, Mario A. Muñoz, Jesús A. López, y Eduardo Caicedo. Optimización de funciones inspirada en el comportamiento de búsqueda de néctar en abejas. *Memorias del Congreso Internacional de Inteligencia Computacional (CIIC2007)*, 2007.
(Referenciado en la página 19.)
- [Bar13] David Barber. *Bayesian Reasoning and Machine Learning*. Independiente, séptima edición, Enero 2013.
(Referenciado en las páginas 59, 60, y 61.)
- [BB96] Gilles Brassard y Paul Bratley. *Fundamentals of Algorithmsics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.
(Referenciado en la página 77.)
- [BE06] Frans Van Den Bergh y Andries Petrus Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971, 2006.
(Referenciado en las páginas 35, 37, 81, y 89.)
- [Bis89] J. M. Bishop. Stochastic searching networks. En *Proceedings 1st IEEE Conf. Artificial Neural Networks*, 1989.
(Referenciado en la página 18.)
- [BJV96] Jeremy De Bonet, Charles Lee Isbell Jr., y Paul Viola. Mimic: Finding optima by estimating probability densities. En Michael Mozer, Michael Jordan, y Thomas Petsche, editores, *Advances in Neural Information Processing Systems*, páginas 424–430.

- MIT Press, 1996.
 (Referenciado en las páginas 64 y 74.)
- [BM09] Guangqing Bao y K. F. Mao. Particle swarm optimization algorithm with asymmetric time varying acceleration coefficients. En *ROBIO*, páginas 2134–2139. IEEE, 2009.
 (Referenciado en la página 33.)
- [Box57] George E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Society, 6(2)*:81–101, Junio 1957.
 (Referenciado en la página 18.)
- [Bre62] H. J. Bremermann. Optimization through evolution and recombination. En M. C. Yovits, G.T. Jacobi, y G.D. Golstine, editores, *In Proceedings of the Conference on Self-Organizing Systems*, páginas 93–106, Washington, D.C., 1962. Spartan Books.
 (Referenciado en la página 18.)
- [BW89] Gerardo Beni y Jing Wang. Swarm intelligence in cellular robotic systems. *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, 102:703–712, 1989.
 (Referenciado en la página 26.)
- [CCC09] Zhihua Cui, Yongfang Chu, y Xingjuan Cai. Nearest neighbor interaction pso based on small-world model. En Emilio Corchado y Hujun Yin, editores, *IDEAL*, volúmen 5788 de *Lecture Notes in Computer Science*, páginas 633–640. Springer, 2009.
 (Referenciado en la página 2.)
- [CK02] Maurice Clerc y James Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.
 (Referenciado en la página 36.)
- [CL68] C. K. Chow y C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, Mayo 1968.
 (Referenciado en las páginas 65 y 74.)
- [CSM16] Carlos Coello, Carlos Segura, y Gara Miranda. History and philosophy of the evolutionary computation. En Plamen Parvanov Angelov, editor, *Handbook on Computational Intelligence*,

- volúmen 2 de *Series on Computational Intelligence*. World Scientific Publishing Company, 2016.
(Referenciado en la página 29.)
- [DC09] Chen DeBao y Zhao ChunXia. Particle swarm optimization with adaptive population size and its application. *Applied Soft Computing*, 9(1):39–48, Enero 2009.
(Referenciado en la página 37.)
- [Dom12] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, Octubre 2012.
(Referenciado en la página 73.)
- [Dor99] Marco Dorigo. Ant colony optimization: A new meta-heuristic. En *Proceedings of the Congress on Evolutionary Computation*, páginas 1470–1477. IEEE Press, 1999.
(Referenciado en la página 18.)
- [EK95] Russell Eberhart y James Kennedy. A new optimizer using particle swarm theory. En *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995.*, páginas 39–43. IEEE, Octubre 1995.
(Referenciado en las páginas 29 y 30.)
- [ES00] Russell Eberhart y Yuhui Shi. Comparing inertia weights and constriction factors in particle swarm optimization. En *Proceedings of the Congress on Evolutionary Computation, 2000*, volúmen 1, páginas 84–88, 2000.
(Referenciado en la página 35.)
- [ES01] Russell Eberhart y Yuhui Shi. Tracking and optimizing dynamic systems with particle swarms. En *Proceedings of the Congress on Evolutionary Computation, 2001*, volúmen 1, páginas 94–100, 2001.
(Referenciado en la página 35.)
- [ESD96] Russell Eberhart, Patrick K. Simpson, y Roy W. Dobbins. *Computational Intelligence PC Tools*. Academic Press Professional, Inc., San Diego, CA, USA, 1996.
(Referenciado en la página 30.)
- [FDN59] R.M. Friedberg, B. Dunham, y J. H. North. A learning machine: Part ii. *IBM Journal of Research and Development*, 3:282–287,

- Julio 1959.
(Referenciado en la página 17.)
- [Fog62] Lawrence J. Fogel. Autonomous automata. *Industrial Research Magazine*, 4(2):14–19, Febrero 1962.
(Referenciado en la página 18.)
- [Fog64] Lawrence J. Fogel. *On the Organization of Intellect*. Tesis de doctorado, University of California, Los Angeles, 1964.
(Referenciado en la página 18.)
- [Fog98] David Fogel. *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, primera edición, 1998.
(Referenciado en las páginas 18 y 29.)
- [FPP86] J. Doyne Farmer, Norman H. Packard, y Alan S. Perelson. The immune system, adaptation, and machine learning. *Physica 22D*, páginas 187–204, 1986.
(Referenciado en la página 18.)
- [Fri58] R.M. Friedberg. A learning machine: Part i. *IBM Journal of Research and Development*, 2:2–13, Enero 1958.
(Referenciado en la página 17.)
- [FS01] H. Fan y Y. Shi. Study on vmax of particle swarm optimization. *Proceedings Workshop on Particle Swarm Optimization*, Abril 2001.
(Referenciado en la página 36.)
- [GBR06] Jörn Grahl, Peter Bosman, y Franz Rothlauf. The correlation-triggered adaptive variance scaling idea. En *In proceedings of the 8th Conference on Genetic and Evolutionary Computation*, páginas 397–404. ACM Press, 2006.
(Referenciado en la página 79.)
- [GKL01] Zong Woo Geem, Joong-Hoon Kim, y G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.
(Referenciado en la página 19.)
- [GZDA11] Pradipta Ghosh, Hamim Zafar, Swagatam Das, y Ajith Abraham. Hierarchical dynamic neighborhood based particle swarm optimization for global optimization. En *IEEE Congress on*

- Evolutionary Computation*, páginas 757–764. IEEE, 2011.
 (Referenciado en las páginas 3, 41, y 89.)
- [Ham08] S. A. Hamdan. Hybrid particle swarm optimiser using multi-neighborhood topologies. *INFOCOMP Journal of Computer Science*, 7(1):36–44, 2008.
 (Referenciado en la página 2.)
- [HG90] Frank H. Heppner y Ulf Grenander. *A stochastic nonlinear model for coordinated bird flocks*, páginas 233–238. AAAS Publications, 1990.
 (Referenciado en las páginas 1 y 30.)
- [Hol62] John H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297–314, 1962.
 (Referenciado en la página 18.)
- [Hol75] John H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.
 (Referenciado en la página 18.)
- [HR77] John H. Holland y Judith S. Reitman. Cognitive systems based on adaptive algorithms. *ACM SIGART Bulletin*, (63), Junio 1977.
 (Referenciado en la página 18.)
- [HSLT09] Sheng-Ta Hsieh, Tsung-Ying Sun, Chan-Cheng Liu, y Shang-Jeng Tsai. Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems, Man and Cybernetics. Part B: Cybernetics*, 39(2):444–456, Abril 2009.
 (Referenciado en la página 37.)
- [Jon75] Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Tesis de doctorado, Ann Arbor, MI, USA, 1975.
 (Referenciado en la página 79.)
- [JY13] Momin Jamil y Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *IJMNO*, 4(2):150–194, 2013.
 (Referenciado en la página 79.)

- [Kar05] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Türkiye, Octubre 2005.
(Referenciado en la página 19.)
- [KE95] James Kennedy y Russell Eberhart. Particle swarm optimization. En *Conference on Neural Networks, 1995. Proceedings., IEEE International*, volúmen 4, páginas 1942–1948. IEEE, Noviembre 1995.
(Referenciado en las páginas 1, 18, 29, 30, y 89.)
- [Ken98] James Kennedy. The behavior of particles. En *Proceedings of the 7th International Conference on Evolutionary Programming*, EP '98, páginas 581–589, London, UK, 1998. Springer-Verlag.
(Referenciado en la página 36.)
- [Ken99] James Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. En *Congress on Evolutionary Computation*, volúmen 3, páginas 1931–1938, 1999.
(Referenciado en las páginas 2 y 38.)
- [Ken00] James Kennedy. Stereotyping: improving particle swarm performance with cluster analysis. En *Proceedings of the Congress on Evolutionary Computation, 2000*, volúmen 2, páginas 1507–1512, 2000.
(Referenciado en la página 2.)
- [KM02] James Kennedy y Rui Mendes. Population structure and particle swarm performance. En *In Proceedings of the Congress on Evolutionary Computation (CEC 2002)*, páginas 1671–1676. IEEE Press, 2002.
(Referenciado en las páginas 2 y 38.)
- [Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
(Referenciado en la página 18.)
- [Kru56] Joseph Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American*

- Mathematical Society*, 7(1):48–50, Febrero 1956.
 (Referenciado en la página 62.)
- [LEG08] James Lane, Andries Engelbrecht, y James Gain. Particle swarm optimization with spatially meaningful neighbours. En *In Proceedings of the IEEE Swarm Intelligence Symposium. 2008*, páginas 1–8, Piscataway, NJ, 2008. Press.
 (Referenciado en la página 2.)
- [LI14] Wei Hong Lim y Nor Ashidi Mat Isa. Particle swarm optimization with increasing topology connectivity. *Engineering Applications of Artificial Intelligence*, 27:80–102, 2014.
 (Referenciado en las páginas 3, 43, y 89.)
- [LM85] A. V. Levy y Ada Montalvo. The tunneling algorithm for the global minimization of functions. *SIAM Journal on Scientific and Statistical Computing*, 6(1):15–29, 1985.
 (Referenciado en la página 79.)
- [LS05] Jing Liang y Ponnuthurai Suganthan. Dynamic multi-swarm particle swarm optimizer with local search. En *Congress on Evolutionary Computation*, páginas 522–528. IEEE, 2005.
 (Referenciado en la página 2.)
- [LT] P. Lucic y D. Teodorovic. Transportation modeling: an artificial life approach. En *14th IEEE International Conference on Tools with Artificial Intelligence. Proceedings*.
 (Referenciado en la página 19.)
- [LY06] Wen-Fung Leong y Gary Yen. Dynamic population size in PSO-based multiobjective optimization. En *IEEE Congress on Evolutionary Computation*, páginas 1718–1725, 2006.
 (Referenciado en la página 37.)
- [MLC09] Mario A. Muñoz, Jesús A. López, y Eduardo Caicedo. An artificial beehive algorithm for continuous optimization. *International Journal of Intelligent System*, 24(11):1080–1093, Noviembre 2009.
 (Referenciado en la página 19.)
- [MMWP05] Arvind Mohais, Rui Mendes, Christopher Ward, y Christian Posthoff. Neighborhood re-structuring in particle swarm optimization. En *Proceedings of the 18th Australian Joint Confer-*

- ence on Artificial Intelligence*, páginas 776–785. Springer, 2005.
 (Referenciado en la página 2.)
- [MN04] Rui Mendes y José Neves. What makes a successful society? experiments with population topologies in particle swarms. En Ana Bazzan y Sofiane Labidi, editores, *SBIA*, volúmen 3171 de *Lecture Notes in Computer Science*, páginas 346–355. Springer, 2004.
 (Referenciado en las páginas 2 y 38.)
- [MWP04] Arvind Mohais, Christopher Ward, y Christian Posthoff. Randomized directed neighborhoods with edge migration in particle swarm optimization. En *Proceedings of the IEEE Congress on Evolutionary Computation, 2004*, páginas 548–555, Portland, Oregon, Junio 2004. IEEE Press.
 (Referenciado en la página 2.)
- [NW99] Jorge Nocedal y Stephen J. Wright. *Numerical Optimization*. Springer, 1999.
 (Referenciado en la página 8.)
- [OSBD09] Marco Antonio Montes De Oca, Thomas Sttzle, Mauro Birattari, y Marco Dorigo. Frankenstein’s pso: A composite particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 13(5):1120–1132, 2009.
 (Referenciado en la página 2.)
- [Pas02] Kevin M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE*, 22(3):52–67, Junio 2002.
 (Referenciado en la página 18.)
- [PGK⁺06] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, y M. Zaidi. The bees algorithm, a novel tool for complex optimisation problems. En *Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2006)*, páginas 454–459. Elsevier, 2006.
 (Referenciado en la página 19.)
- [Rec73] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15. Frommann-Holzboog, 1973.
 (Referenciado en la página 18.)

- [Rec94] Ingo Rechenberg. *Evolutionsstrategie '94*. Frommann-Holzboog, Septiembre 1994.
 (Referenciado en la página 18.)
- [Sch75] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. Technische Universität Berlin, 1975.
 (Referenciado en la página 18.)
- [Sch93] Hans-Paul Schwefel. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., New York, NY, USA, 1993.
 (Referenciado en la página 18.)
- [SE98] Yuhui Shi y Russell Eberhart. A modified particle swarm optimizer. En *The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence.*, páginas 69–73. IEEE, Mayo 1998.
 (Referenciado en la página 33.)
- [Sha48] Claude Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.
 (Referenciado en la página 53.)
- [SP97] Rainer Storn y Kenneth Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Diciembre 1997.
 (Referenciado en la página 18.)
- [Spi65] Michael Spivak. *Calculus on manifolds, a modern approach to classical theorems of advanced calculus*. Addison-Wesley Publishing Company, Octubre 1965.
 (Referenciado en la página 6.)
- [Sti99] David Stirzaker. *Probability and Random Variables - A Beginner's Guide*. Cambridge University Press, primera edición, 1999.
 (Referenciado en la página 52.)
- [Sug99] Ponnuthurai Suganthan. Particle swarm optimiser with neighbourhood operator. En *Proceedings of the Congress on Evolutionary Computation, 1999. CEC 99.*, volúmen 3, 1999.
 (Referenciado en las páginas 2 y 41.)

- [Tal09] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.
(Referenciado en las páginas 14 y 29.)
- [Wei08] Thomas Weise. *Global Optimization Algorithms - Theory and Application*. Independiente, segunda edición, Agosto 2008.
(Referenciado en las páginas 7, 9, 10, 11, 12, 13, 14, 20, 21, 23, y 24.)
- [ZLJ04] Weicai Zhong, Jing Liu, y Licheng Jiao. A novel genetic algorithm based on multi-agent systems. En *Intelligent Information Processing and Web Mining*, volúmen 25 de *Advances in Soft Computing*, páginas 169–178. Springer Berlin Heidelberg, 2004.
(Referenciado en la página 79.)
- [ZMZQ03] Yong Ling Zheng, Long Hua Ma, Li Yan Zhang, y Ji Xin Qian. On the convergence analysis and parameter selection in particle swarm optimization. En *International Conference on Machine Learning and Cybernetics*, volúmen 3, páginas 1802–1807, 2003.
(Referenciado en la página 35.)
- [ZXZC07] Zhi Hui Zhan, Jing Xiao, Jun Zhang, y Neng Wei Chen. Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis. En *IEEE Congress on Evolutionary Computation*, páginas 3276–3282. IEEE, 2007.
(Referenciado en la página 33.)

Anexo A

Resultados obtenidos actualizando las topologías cada 50 generaciones con Chow-Liu

En el presente anexo se muestran los resultados obtenidos en las funciones de la tabla 6.1 en dimensiones 30, 50, 80 y 100; utilizando PSO-MI con los parámetros mencionados en la sección 6.3, pero con $f = 50$, $h = 50$. De igual forma, se reportan los mismos estadísticos de la sección 6.3. En la tabla A.1 se muestran los resultados obtenidos de las funciones f_1 a f_6 , en la tabla A.2 se muestran los resultados obtenidos de las funciones f_7 a f_{12} y en la tabla A.3 se muestran los resultados obtenidos de las funciones f_{13} a f_{17} .

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_1	Valor del mínimo encontrado	Media	9.1489e-11	9.208e-11	9.6323e-11	9.6419e-11
		Std	8.5977e-12	5.2487e-12	3.7783e-12	4.4292e-12
		Mediana	9.5574e-11	9.2361e-11	9.7868e-11	9.7782e-11
		Mínimo	7.0021e-11	8.2623e-11	8.424e-11	7.7031e-11
		Máximo	9.9627e-11	9.9983e-11	9.9692e-11	9.999e-11
	Evaluaciones de función	Media	25925	40387	61703	75748
		Std	547.99	978.18	1218.6	1524.7
f_2	Valor del mínimo encontrado	Porcentaje de éxito	100	100	100	100
		Media	7.682	36.888	139.82	158.74
		Std	12.455	34.289	107.37	58.542
		Mediana	4.2874	23.795	114.01	164.12
		Mínimo	0.06516	0.00022925	15.385	75.131
	Evaluaciones de función	Máximo	67.433	98.952	654.17	288.17
		Media	300000	300000	300000	300000
f_3	Valor del mínimo encontrado	Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
		Media	20.472	70.31	205.55	262.57
		Std	12.409	32.228	78.53	95.09
		Mediana	18.977	60.046	213.73	263.75
	Evaluaciones de función	Mínimo	3.1651	22.884	57.708	84.571
		Máximo	45.41	132.67	328.78	416.67
f_4	Valor del mínimo encontrado	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
		Media	9.1123e-11	0.00049307	0.0055286	0.0035842
		Std	8.3549e-12	0.0018764	0.017804	0.013361
	Evaluaciones de función	Mediana	9.3923e-11	9.7746e-11	9.83e-11	9.8257e-11
		Mínimo	6.8827e-11	7.7557e-11	8.5091e-11	9.0833e-11
f_5	Valor del mínimo encontrado	Máximo	9.9879e-11	0.007396	0.096854	0.065795
		Media	38189	59107	126020	97494
		Std	34675	65515	106850	68676
		Porcentaje de éxito	100	93.333	73.333	90
		Media	0.0069113	0.051912	0.057324	0.045698
	Evaluaciones de función	Std	0.026302	0.11817	0.13855	0.085032
		Mediana	9.5953e-11	9.7162e-11	9.8314e-11	9.9556e-11
f_6	Valor del mínimo encontrado	Mínimo	7.7582e-11	7.0983e-11	8.192e-11	9.0157e-11
		Máximo	0.10367	0.49948	0.66894	0.34325
		Media	45399	113040	152070	194180
		Std	69224	114690	106540	100910
		Porcentaje de éxito	93.333	73.333	66.667	53.333
	Evaluaciones de función	Media	9.345e-11	0.00036625	0.059312	0.074556
		Std	5.0143e-12	0.002006	0.17882	0.29819
		Mediana	9.5123e-11	9.8167e-11	9.9034e-11	9.9087e-11
		Mínimo	8.1597e-11	8.23e-11	8.2256e-11	9.1333e-11
		Máximo	9.9268e-11	0.010987	0.89746	1.5975
	Porcentaje de éxito	Media	27390	53555	159290	159230
		Std	850.06	46594	109000	87563
	Porcentaje de éxito	100	96.667	63.333	73.333	

Tabla A.1: Resultados obtenidos en las funciones f_1 , f_2 , f_3 , f_4 , f_5 y f_6 actualizando las topologías cada 50 generaciones con Chow-Liu

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_7	Valor del mínimo encontrado	Media	9.639e-11	0.10586	0.079124	0.24128
		Std	3.4188e-12	0.32714	0.3025	0.4944
		Mediana	9.7251e-11	9.8246e-11	9.8988e-11	9.926e-11
		Mínimo	8.8172e-11	9.2133e-11	9.1835e-11	9.4745e-11
		Máximo	9.9999e-11	1.2697	1.2967	1.4235
	Evaluaciones de función	Media	41399	87430	110790	155600
		Std	905.17	72079	51473	73472
	Porcentaje de éxito		100	90	93.333	80
	Valor del mínimo encontrado	Media	0.018571	0.035128	0.063837	0.07973
		Std	0.0073759	0.011178	0.013956	0.022218
		Mediana	0.015794	0.032244	0.06559	0.07944
		Mínimo	0.0095464	0.016239	0.039643	0.046174
		Máximo	0.040753	0.066465	0.08813	0.16244
f_8	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
	Valor del mínimo encontrado	Media	9.2124e-11	9.4734e-11	9.7081e-11	9.6461e-11
		Std	8.0681e-12	6.0241e-12	2.5879e-12	2.9808e-12
		Mediana	9.2977e-11	9.7076e-11	9.8336e-11	9.7036e-11
		Mínimo	6.5005e-11	7.4659e-11	9.1293e-11	8.8654e-11
		Máximo	9.9996e-11	9.9961e-11	9.9779e-11	9.9856e-11
	Evaluaciones de función	Media	32393	50411	76659	94307
		Std	837.14	1074.1	1672.6	1850.4
	Porcentaje de éxito		100	100	100	100
f_9	Valor del mínimo encontrado	Media	9.1197e-11	9.358e-11	9.6087e-11	9.6533e-11
		Std	6.8390e-12	6.1214e-12	3.7099e-12	4.1756e-12
		Mediana	9.2215e-11	9.6206e-11	9.7784e-11	9.8096e-11
		Mínimo	7.4718e-11	7.925e-11	8.7227e-11	8.2075e-11
		Máximo	9.9884e-11	9.996e-11	9.9851e-11	9.9871e-11
	Evaluaciones de función	Media	29622	46529	70035	86506
		Std	784.11	956.16	1302.8	2307.1
	Porcentaje de éxito		100	100	100	100
f_{10}	Valor del mínimo encontrado	Media	9.2062e-11	9.4632e-11	9.6975e-11	9.5869e-11
		Std	9.6586e-12	5.344e-12	3.3712e-12	4.2052e-12
		Mediana	9.5416e-11	9.6489e-11	9.8415e-11	9.6923e-11
		Mínimo	4.8147e-11	7.8497e-11	8.8013e-11	8.0405e-11
		Máximo	9.8676e-11	9.9996e-11	9.9997e-11	9.9826e-11
	Evaluaciones de función	Media	28149	44135	66777	82806
		Std	788.04	785.79	1295.1	1549.2
	Porcentaje de éxito		100	100	100	100
f_{11}	Valor del mínimo encontrado	Media	8.2286e-11	8.9893e-11	9.6128e-11	9.3854e-11
		Std	1.5383e-11	9.3953e-12	2.9763e-12	5.1537e-12
		Mediana	8.5244e-11	9.2242e-11	9.6726e-11	9.523e-11
		Mínimo	4.4947e-11	6.1195e-11	8.5805e-11	7.9357e-11
		Máximo	9.9435e-11	9.9107e-11	9.9634e-11	9.9793e-11
	Evaluaciones de función	Media	16640	28536	47766	60476
		Std	875.68	1331	1890.4	2236.6
	Porcentaje de éxito		100	100	100	100
f_{12}	Valor del mínimo encontrado	Media	8.2286e-11	8.9893e-11	9.6128e-11	9.3854e-11
		Std	1.5383e-11	9.3953e-12	2.9763e-12	5.1537e-12
		Mediana	8.5244e-11	9.2242e-11	9.6726e-11	9.523e-11
		Mínimo	4.4947e-11	6.1195e-11	8.5805e-11	7.9357e-11
		Máximo	9.9435e-11	9.9107e-11	9.9634e-11	9.9793e-11
	Evaluaciones de función	Media	16640	28536	47766	60476
		Std	875.68	1331	1890.4	2236.6
	Porcentaje de éxito		100	100	100	100

Tabla A.2: Resultados obtenidos en las funciones f_7 , f_8 , f_9 , f_{10} , f_{11} y f_{12} actualizando las topologías cada 50 generaciones con Chow-Liu

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_{13}	Valor del mínimo encontrado	Media	9.4153e-11	9.3938e-11	9.5912e-11	9.6929e-11
		Std	4.0822e-12	5.8838e-12	3.7618e-12	3.2861e-12
		Mediana	9.4389e-11	9.5899e-11	9.664e-11	9.7947e-11
		Mínimo	8.4988e-11	7.8934e-11	8.2673e-11	8.4064e-11
		Máximo	9.9757e-11	9.9926e-11	9.9814e-11	9.9989e-11
	Evaluaciones de función	Media	23111	35946	54796	67527
		Std	487.28	743.76	1072.3	1499.2
		Porcentaje de éxito	100	100	100	100
f_{14}	Valor del mínimo encontrado	Media	9.0629e-11	9.3923e-11	9.5096e-11	9.6065e-11
		Std	6.9521e-12	5.4662e-12	4.9543e-12	4.6046e-12
		Mediana	9.1965e-11	9.5796e-11	9.6876e-11	9.8094e-11
		Mínimo	7.236e-11	7.6957e-11	7.8549e-11	8.1229e-11
		Máximo	9.9581e-11	9.9797e-11	9.9995e-11	9.9901e-11
	Evaluaciones de función	Media	28308	44665	67785	83499
		Std	717.61	1040.1	1293.3	1592.2
		Porcentaje de éxito	100	100	100	100
f_{15}	Valor del mínimo encontrado	Media	9.7043e-11	0.014343	61.162	461.75
		Std	2.4453e-12	0.027138	36.367	151.11
		Mediana	9.7757e-11	0.0035077	53.55	443.24
		Mínimo	8.9947e-11	0.0002462	25.579	234.9
		Máximo	9.9799e-11	0.12612	202.66	872.38
	Evaluaciones de función	Media	214040	300000	300000	300000
		Std	16754	0	0	0
		Porcentaje de éxito	100	0	0	0
f_{16}	Valor del mínimo encontrado	Media	0.066545	0.78749	1.7018	3.7299
		Std	0.15634	0.93126	1.3401	2.4116
		Mediana	9.5287e-11	0.45432	0.99818	3.3509
		Mínimo	7.2072e-11	9.1317e-11	9.1271e-11	9.9465e-11
		Máximo	0.45432	3.9098	4.7477	11.035
	Evaluaciones de función	Media	79523	239040	283940	292370
		Std	112130	112390	61130	41797
		Porcentaje de éxito	80	23.333	6.6667	3.3333
f_{17}	Valor del mínimo encontrado	Media	0.21987	0.3699	0.73654	1.0468
		Std	0.040684	0.065107	0.22358	0.31143
		Mediana	0.19987	0.39987	0.69987	0.99987
		Mínimo	0.19987	0.29987	0.39987	0.59991
		Máximo	0.29987	0.59987	1.3999	2.1999
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0

Tabla A.3: Resultados obtenidos en las funciones f_{13} , f_{14} , f_{15} , f_{16} y f_{17} actualizando las topologías cada 50 generaciones con Chow-Liu

Anexo B

Resultados obtenidos construyendo las topologías con MIMIC

En el presente anexo se muestran los resultados obtenidos en las funciones de la tabla 6.1 en dimensiones 30, 50, 80 y 100; utilizando PSO-MI con los parámetros mencionados en la sección 6.3, pero construyendo las topologías con MIMIC. Se reportan los mismos estadísticos de la sección 6.3. En la tabla B.1 se muestran los resultados obtenidos de las funciones f_1 a f_6 , en la tabla B.2 se muestran los resultados obtenidos de las funciones f_7 a f_{12} y en la tabla B.3 se muestran los resultados obtenidos de las funciones f_{13} a f_{17} ; en este caso utilizando $h = 50$ y $f = 50$. Los resultados obtenidos con $h = 100$ y $f = 100$ se reportan en las tablas B.4, B.5 y B.6; para las funciones f_1 a f_6 , f_7 a f_{12} y f_{13} a f_{17} , respectivamente.

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_1	Valor del mínimo encontrado	Media	9.0047e-11	9.3954e-11	9.4819e-11	9.6423e-11
		Std	9.4949e-12	5.866e-12	5.6434e-12	3.3954e-12
		Mediana	9.1019e-11	9.6209e-11	9.6506e-11	9.7965e-11
		Mínimo	5.3589e-11	7.9587e-11	7.3882e-11	8.3902e-11
		Máximo	9.9997e-11	9.9951e-11	9.9801e-11	9.9658e-11
	Evaluaciones de función	Media	24788	38994	58971	71762
		Std	591.35	526.17	1215	1310.9
		Porcentaje de éxito	100	100	100	100
	Valor del mínimo encontrado	Media	7.2697	58.482	114.38	159.64
		Std	5.706	34.869	35.735	51.319
		Mediana	5.0348	70.538	109.99	141.68
		Mínimo	0.020226	6.4666	56.238	89.505
		Máximo	15.344	174.54	180.25	269.77
f_2	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
		Media	300000	300000	300000	300000
		Std	0	0	0	0
	Valor del mínimo encontrado	Media	13.778	73.346	192.09	330.02
		Std	10.087	41.171	73.376	120.04
		Mediana	10.055	65.275	199.47	346.28
		Mínimo	3.9798	12.934	50.743	92.531
		Máximo	46.188	153.83	368.05	519.77
f_3	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
		Media	300000	300000	300000	300000
		Std	0	0	0	0
	Valor del mínimo encontrado	Media	0.00024653	0.00065592	0.00024653	0.0011493
		Std	0.0013503	0.0035926	0.0013503	0.0037554
		Mediana	9.5082e-11	9.603e-11	9.659e-11	9.7804e-11
		Mínimo	3.8665e-11	7.8057e-11	7.3749e-11	8.8252e-11
		Máximo	0.007396	0.019678	0.007396	0.017226
f_4	Evaluaciones de función	Media	47502	48829	66613	94210
		Std	54652	47546	44096	69783
		Porcentaje de éxito	96.667	96.667	96.667	90
		Media	9.0656e-11	0.0020734	0.010367	0.021776
		Std	8.3078e-12	0.011356	0.020247	0.04845
	Valor del mínimo encontrado	Mediana	9.2036e-11	9.5157e-11	9.8086e-11	9.8762e-11
		Mínimo	5.9371e-11	7.4129e-11	8.3847e-11	8.0725e-11
		Máximo	9.9117e-11	0.062201	0.077744	0.21791
		Media	25480	49929	120850	145550
		Std	1345.9	47282	100560	94900
f_5	Porcentaje de éxito	Media	100	96.667	76.667	73.333
		Std	0	0	0	0
		Mediana	0	0	0	0
		Mínimo	0	0	0	0
		Máximo	0	0	0	0
	Valor del mínimo encontrado	Media	9.2197e-11	0.11992	0.087227	0.3236
		Std	8.555e-12	0.6568	0.45764	1.4915
		Mediana	9.5227e-11	9.3373e-11	9.8119e-11	9.8009e-11
		Mínimo	6.003e-11	7.3447e-11	9.0321e-11	8.7826e-11
		Máximo	9.9475e-11	3.5975	2.5085	8.0665
f_6	Evaluaciones de función	Media	26144	50498	91494	123760
		Std	527.01	47146	70776	80314
		Porcentaje de éxito	100	96.667	90	83.333
		Media	9.0656e-11	0.0020734	0.010367	0.021776
		Std	8.3078e-12	0.011356	0.020247	0.04845

Tabla B.1: Resultados obtenidos en las funciones f_1 , f_2 , f_3 , f_4 , f_5 y f_6 actualizando las topologías cada 50 generaciones con MIMIC

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_7	Valor del mínimo encontrado	Media	9.5599e-11	9.6394e-11	9.7975e-11	9.8104e-11
		Std	4.8651e-12	3.1136e-12	2.176e-12	1.5844e-12
		Mediana	9.7375e-11	9.7708e-11	9.858e-11	9.8262e-11
		Mínimo	8.1525e-11	8.8943e-11	8.9689e-11	9.3096e-11
		Máximo	9.985e-11	9.9939e-11	9.9711e-11	9.9917e-11
	Evaluaciones de función	Media	39662	61580	91975	111790
		Std	576.68	1314.1	1104	1200.7
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	0.020447	0.035491	0.061069	0.073545
		Std	0.0064794	0.010213	0.011247	0.012401
		Mediana	0.020105	0.034778	0.060496	0.074937
		Mínimo	0.0082596	0.016399	0.042615	0.042053
		Máximo	0.031014	0.057428	0.08806	0.099313
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
	Porcentaje de éxito		0	0	0	0
f_9	Valor del mínimo encontrado	Media	9.0011e-11	9.3015e-11	9.5206e-11	9.5456e-11
		Std	7.9334e-12	6.3194e-12	4.9814e-12	4.8239e-12
		Mediana	9.0482e-11	9.4948e-11	9.6917e-11	9.7603e-11
		Mínimo	6.7205e-11	7.644e-11	7.8778e-11	8.1048e-11
		Máximo	9.9964e-11	9.9988e-11	9.9552e-11	9.9656e-11
	Evaluaciones de función	Media	30742	48454	73865	89573
		Std	627.01	784.16	1188.1	1214
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	9.1131e-11	9.3689e-11	9.5418e-11	9.5861e-11
		Std	6.9818e-12	6.4222e-12	4.1509e-12	4.6855e-12
		Mediana	9.1697e-11	9.5224e-11	9.6583e-11	9.7959e-11
		Mínimo	7.6501e-11	7.5105e-11	8.3398e-11	8.0098e-11
		Máximo	9.9879e-11	9.9929e-11	9.9894e-11	9.9915e-11
	Evaluaciones de función	Media	28498	44855	66999	82050
		Std	523.78	571.66	1001.1	1435.8
	Porcentaje de éxito		100	100	100	100
f_{10}	Valor del mínimo encontrado	Media	9.2061e-11	9.2836e-11	9.4424e-11	9.6255e-11
		Std	8.4348e-12	6.5084e-12	6.2862e-12	4.0054e-12
		Mediana	9.4661e-11	9.3494e-11	9.7327e-11	9.8255e-11
		Mínimo	7.0894e-11	7.1495e-11	7.5632e-11	8.5767e-11
		Máximo	9.9789e-11	9.9728e-11	9.9639e-11	9.9901e-11
	Evaluaciones de función	Media	26993	42194	64026	78329
		Std	673.11	712.14	862.72	1299.5
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	8.5271e-11	9.1115e-11	9.179e-11	9.3017e-11
		Std	1.2299e-11	6.4511e-12	7.1051e-12	6.1106e-12
		Mediana	8.6902e-11	9.2683e-11	9.3731e-11	9.5145e-11
		Mínimo	5.3285e-11	7.5951e-11	6.7044e-11	7.1101e-11
		Máximo	9.9655e-11	9.9998e-11	9.9884e-11	9.9773e-11
	Evaluaciones de función	Media	15843	26810	44766	57958
		Std	651.45	834.49	1674.8	2217.2
	Porcentaje de éxito		100	100	100	100
f_{12}	Valor del mínimo encontrado	Media	8.5271e-11	9.1115e-11	9.179e-11	9.3017e-11
		Std	1.2299e-11	6.4511e-12	7.1051e-12	6.1106e-12
		Mediana	8.6902e-11	9.2683e-11	9.3731e-11	9.5145e-11
		Mínimo	5.3285e-11	7.5951e-11	6.7044e-11	7.1101e-11
		Máximo	9.9655e-11	9.9998e-11	9.9884e-11	9.9773e-11
	Evaluaciones de función	Media	15843	26810	44766	57958
		Std	651.45	834.49	1674.8	2217.2
	Porcentaje de éxito		100	100	100	100

Tabla B.2: Resultados obtenidos en las funciones f_7 , f_8 , f_9 , f_{10} , f_{11} y f_{12} actualizando las topologías cada 50 generaciones con MIMIC

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_{13}	Valor del mínimo encontrado	Media	9.0871e-11	9.3506e-11	9.5495e-11	9.5436e-11
		Std	9.3005e-12	7.4428e-12	3.6511e-12	5.7557e-12
		Mediana	9.3723e-11	9.7243e-11	9.5716e-11	9.7558e-11
		Mínimo	6.7385e-11	7.2387e-11	8.7916e-11	7.5169e-11
		Máximo	9.9898e-11	9.9882e-11	9.9973e-11	9.9895e-11
	Evaluaciones de función	Media	22437	34928	52617	63751
		Std	605.73	798.17	1156.4	858.04
		Porcentaje de éxito	100	100	100	100
f_{14}	Valor del mínimo encontrado	Media	9.3628e-11	9.3988e-11	9.6368e-11	9.6998e-11
		Std	5.2743e-12	7.0205e-12	4.072e-12	2.5241e-12
		Mediana	9.5188e-11	9.6371e-11	9.7377e-11	9.8013e-11
		Mínimo	7.769e-11	6.5601e-11	7.9612e-11	9.0518e-11
		Máximo	9.9984e-11	9.9819e-11	9.9985e-11	9.999e-11
	Evaluaciones de función	Media	27090	42849	65211	79320
		Std	442.03	724.05	1027.4	1434.4
		Porcentaje de éxito	100	100	100	100
f_{15}	Valor del mínimo encontrado	Media	9.5068e-11	0.00091791	40.87	308.54
		Std	5.8517e-12	0.0023793	25.338	69.632
		Mediana	9.7451e-11	0.00024283	32.995	299.48
		Mínimo	7.5441e-11	9.5414e-06	12.988	148.06
		Máximo	9.9812e-11	0.012278	142.14	444.04
	Evaluaciones de función	Media	188920	300000	300000	300000
		Std	9656.6	0	0	0
		Porcentaje de éxito	100	0	0	0
f_{16}	Valor del mínimo encontrado	Media	0.030288	0.066545	0.62804	0.99839
		Std	0.11527	0.16507	0.64622	0.8773
		Mediana	9.5539e-11	9.7024e-11	0.45432	0.67814
		Mínimo	3.6914e-11	7.5833e-11	8.8887e-11	8.8594e-11
		Máximo	0.45432	0.54385	2.2716	3.9098
	Evaluaciones de función	Media	42040	81088	259290	292350
		Std	70124	99576	92595	41895
		Porcentaje de éxito	93.333	83.333	16.667	3.3333
f_{17}	Valor del mínimo encontrado	Media	0.19345	0.30654	0.50321	0.67654
		Std	0.03587	0.052083	0.1129	0.16333
		Mediana	0.19987	0.29987	0.49987	0.59987
		Mínimo	0.099873	0.19987	0.39987	0.49987
		Máximo	0.29987	0.39987	0.89987	1.1999
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0

Tabla B.3: Resultados obtenidos en las funciones f_{13} , f_{14} , f_{15} , f_{16} y f_{17} actualizando las topologías cada 50 generaciones con MIMIC

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_1	Valor del mínimo encontrado	Media	9.2612e-11	9.5049e-11	9.6345e-11	9.567e-11
		Std	7.864e-12	4.6664e-12	3.0913e-12	4.756e-12
		Mediana	9.6099e-11	9.685e-11	9.7007e-11	9.7907e-11
		Mínimo	7.0686e-11	8.1631e-11	8.8715e-11	8.3124e-11
		Máximo	9.9879e-11	9.999e-11	9.9778e-11	9.996e-11
	Evaluaciones de función	Media	29106	47200	73098	89915
		Std	1057.3	1426.5	1625.7	2234.3
		Porcentaje de éxito	100	100	100	100
	Valor del mínimo encontrado	Media	13.052	66.9	130.72	179.84
		Std	18.637	38.849	49.168	68.859
		Mediana	8.1801	80.42	114.23	168.56
		Mínimo	0.00031435	0.38279	12.997	84.028
		Máximo	78.715	143.98	226.8	352.18
f_2	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
	Valor del mínimo encontrado	Media	48.472	155.2	347.45	491.97
		Std	18.178	32.373	46.107	82.041
		Mediana	46.602	153.97	351.53	496.6
		Mínimo	6.9753	93.456	268.75	331.45
		Máximo	85.032	219.51	449.82	639.47
	Porcentaje de éxito	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0
f_3	Valor del mínimo encontrado	Media	0.00025411	0.00024653	0.00073911	0.00065707
		Std	0.0013918	0.0013503	0.0028312	0.0025827
		Mediana	9.2094e-11	9.6403e-11	9.8354e-11	9.6978e-11
		Mínimo	6.4831e-11	8.5547e-11	8.1805e-11	7.3847e-11
		Máximo	0.0076233	0.007396	0.012316	0.012316
	Evaluaciones de función	Media	50884	59931	88242	104410
		Std	53416	45914	57613	53325
		Porcentaje de éxito	96.667	96.667	93.333	93.333
	Porcentaje de éxito	Media	0.013822	0.010367	0.016846	0.044599
		Std	0.075708	0.046442	0.034895	0.069196
		Mediana	9.3527e-11	9.6141e-11	9.9066e-11	9.9475e-11
		Mínimo	6.2336e-11	8.0839e-11	7.6927e-11	8.9728e-11
		Máximo	0.41467	0.2488	0.1555	0.21793
f_5	Valor del mínimo encontrado	Media	39894	68252	145780	183960
		Std	49155	63045	94825	90383
		Porcentaje de éxito	96.667	93.333	73.333	63.333
		Media	9.1999e-11	0.00073249	0.0018312	0.0021975
		Std	9.2052e-12	0.0027876	0.0041648	0.0083626
	Evaluaciones de función	Mediana	9.5212e-11	9.7416e-11	9.7874e-11	9.8295e-11
		Mínimo	6.0058e-11	7.8333e-11	8.6438e-11	7.5297e-11
f_6	Porcentaje de éxito	Media	31404	66856	118540	125500
		Std	1250.7	63401	82576	59400
		Porcentaje de éxito	100	93.333	83.333	90

Tabla B.4: Resultados obtenidos en las funciones f_1 , f_2 , f_3 , f_4 , f_5 y f_6 actualizando las topologías cada 100 generaciones con MIMIC

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_7	Valor del mínimo encontrado	Media	9.5533e-11	9.7652e-11	9.7464e-11	9.8332e-11
		Std	3.2762e-12	2.191e-12	3.2374e-12	1.8832e-12
		Mediana	9.6034e-11	9.8153e-11	9.8692e-11	9.8724e-11
		Mínimo	8.6772e-11	9.154e-11	8.8272e-11	9.0176e-11
		Máximo	9.974e-11	9.9953e-11	9.9999e-11	9.9807e-11
	Evaluaciones de función	Media	46508	75121	114660	140010
		Std	1339.7	1642.9	2366.7	2463.6
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	0.012858	0.020208	0.03364	0.046709
		Std	0.0037376	0.0052452	0.0063556	0.0078583
		Mediana	0.012567	0.01932	0.033245	0.047323
		Mínimo	0.0073195	0.012506	0.023157	0.032952
		Máximo	0.021734	0.03437	0.044251	0.059256
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
	Porcentaje de éxito		0	0	0	0
f_9	Valor del mínimo encontrado	Media	9.2078e-11	9.3986e-11	9.6029e-11	9.5497e-11
		Std	6.9694e-12	6.6521e-12	4.7414e-12	3.8628e-12
		Mediana	9.3207e-11	9.6293e-11	9.8264e-11	9.6411e-11
		Mínimo	7.1299e-11	6.9866e-11	8.0046e-11	8.1693e-11
		Máximo	9.9794e-11	9.9713e-11	9.9973e-11	9.9941e-11
	Evaluaciones de función	Media	35574	58536	91727	113220
		Std	880.63	1665.8	1832.2	2437.7
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	9.0395e-11	9.6135e-11	9.5224e-11	9.7067e-11
		Std	7.6158e-12	3.0541e-12	4.7822e-12	2.602e-12
		Mediana	9.1381e-11	9.631e-11	9.6821e-11	9.7669e-11
		Mínimo	6.7217e-11	8.9946e-11	7.585e-11	8.9283e-11
		Máximo	9.9806e-11	9.9975e-11	9.9327e-11	9.9977e-11
	Evaluaciones de función	Media	33027	53576	83817	103420
		Std	1144.8	1307.2	1705.2	2238
	Porcentaje de éxito		100	100	100	100
f_{11}	Valor del mínimo encontrado	Media	9.2103e-11	9.4989e-11	9.3929e-11	9.7397e-11
		Std	6.8965e-12	6.0887e-12	6.4793e-12	2.527e-12
		Mediana	9.404e-11	9.7398e-11	9.6634e-11	9.8093e-11
		Mínimo	7.2591e-11	7.188e-11	7.472e-11	8.9655e-11
		Máximo	9.9898e-11	9.9647e-11	9.9917e-11	9.9898e-11
	Evaluaciones de función	Media	31430	50796	79418	97924
		Std	1002.2	1521.3	1416.2	2268.4
	Porcentaje de éxito		100	100	100	100
	Valor del mínimo encontrado	Media	8.9122e-11	9.2147e-11	9.4109e-11	9.578e-11
		Std	1.0397e-11	8.1916e-12	7.1297e-12	4.392e-12
		Mediana	9.1039e-11	9.4797e-11	9.6726e-11	9.7212e-11
		Mínimo	5.8279e-11	6.3947e-11	7.4199e-11	8.1703e-11
		Máximo	9.9985e-11	9.9922e-11	9.9745e-11	9.9832e-11
	Evaluaciones de función	Media	18751	31802	53120	67962
		Std	853.32	1039.7	1862.7	2291.9
	Porcentaje de éxito		100	100	100	100

Tabla B.5: Resultados obtenidos en las funciones f_7 , f_8 , f_9 , f_{10} , f_{11} y f_{12} actualizando las topologías cada 100 generaciones con MIMIC

Función	Estadístico reportado	Dimensión				
		30	50	80	100	
f_{13}	Valor del mínimo encontrado	Media	9.3874e-11	9.5139e-11	9.3392e-11	9.7347e-11
		Std	5.6499e-12	3.7932e-12	7.0325e-12	3.125e-12
		Mediana	9.6757e-11	9.6304e-11	9.5659e-11	9.8598e-11
		Mínimo	8.1488e-11	8.1057e-11	6.7642e-11	8.8105e-11
		Máximo	9.9576e-11	9.9435e-11	9.9985e-11	9.9988e-11
	Evaluaciones de función	Media	26594	42212	65737	79555
		Std	929.37	1342.8	1580.1	1496.4
		Porcentaje de éxito	100	100	100	100
f_{14}	Valor del mínimo encontrado	Media	9.1449e-11	9.3698e-11	9.7525e-11	9.6044e-11
		Std	7.7833e-12	7.4059e-12	3.3929e-12	5.5721e-12
		Mediana	9.3932e-11	9.5483e-11	9.853e-11	9.8041e-11
		Mínimo	6.9859e-11	7.0827e-11	8.3551e-11	7.4689e-11
		Máximo	9.9854e-11	9.9908e-11	9.9881e-11	9.9838e-11
	Evaluaciones de función	Media	31087	51160	79764	98195
		Std	793.17	1573	1775.2	2207.5
		Porcentaje de éxito	100	100	100	100
f_{15}	Valor del mínimo encontrado	Media	9.5191e-11	0.013234	41.128	312.82
		Std	5.0335e-12	0.016145	32.584	90.019
		Mediana	9.6817e-11	0.0089412	33.353	305.32
		Mínimo	8.0287e-11	0.00049715	11.46	144.28
		Máximo	9.9951e-11	0.082534	199.71	496.71
	Evaluaciones de función	Media	246380	300000	300000	300000
		Std	17876	0	0	0
		Porcentaje de éxito	100	0	0	0
f_{16}	Valor del mínimo encontrado	Media	0.033273	0.096833	0.64101	1.3987
		Std	0.11561	0.18319	0.8495	1.3182
		Mediana	9.4433e-11	9.758e-11	0.45432	0.90865
		Mínimo	7.1797e-11	7.8496e-11	9.1781e-11	9.5121e-11
		Máximo	0.45432	0.45432	2.8882	4.4537
	Evaluaciones de función	Media	55565	113810	239280	285940
		Std	82875	114210	102420	53521
		Porcentaje de éxito	90	73.333	26.667	6.6667
f_{17}	Valor del mínimo encontrado	Media	0.19989	0.30654	0.53654	0.65988
		Std	0.037104	0.052083	0.08899	0.11626
		Mediana	0.19987	0.29987	0.49987	0.59987
		Mínimo	0.099924	0.19987	0.39987	0.49987
		Máximo	0.29987	0.39987	0.69987	0.99987
	Evaluaciones de función	Media	300000	300000	300000	300000
		Std	0	0	0	0
		Porcentaje de éxito	0	0	0	0

Tabla B.6: Resultados obtenidos en las funciones f_{13} , f_{14} , f_{15} , f_{16} y f_{17} actualizando las topologías cada 100 generaciones con MIMIC

Anexo C

Comparación de los resultados obtenidos con distintas frecuencia de actualización y formas de construir las topologías

En el presente anexo hacemos una comparación de los resultados obtenidos por PSO-MI en las funciones de la tablas 6.1, utilizando distintas frecuencias de actualización, historial de las partículas y actualizando con Chow-Liu y MIMIC. Los cuatro casos evaluados son: $h = 100$, $f = 100$ y actualización de las topologías con Chow-Liu (Chow-Liu 100); $h = 50$, $f = 50$ y actualización de las topologías con Chow-Liu (Chow-Liu 50); $h = 100$, $f = 100$ y actualización de las topologías con MIMIC (MIMIC 100) y $h = 50$, $f = 50$ y actualización de las topologías con MIMIC (MIMIC 50). Esta comparación se presenta con base en los resultados mostrados en las tablas de los anexos A y B. Las figuras del presente anexo se separan de la misma forma que en la sección 6.4, mostrándose en las figuras C.1 y C.2 las funciones comparadas por el número de evaluaciones de función necesarias para alcanzar el óptimo, en la figura C.3 las funciones comparadas por el valor de aptitud alcanzado y en la figura C.4 las funciones comparadas por su porcentaje de éxito. Como se puede observar en las figuras, en las funciones en donde se comparan las evaluaciones de función necesarias para alcanzar el óptimo, se obtiene una convergencia más rápida cuando se actualizan las topologías cada 50 generaciones, siendo un poco mejor al actualizarlas con MIMIC. En las funciones en donde se compara por el valor de aptitud alcanzado, por lo general, se puede observar resultados más “cercanos” al óptimo cuando las topologías se actualizan cada 50 generaciones con MIMIC. Por último,

en las funciones en donde se compara el porcentaje de éxito de cada combinación algoritmo-frecuencia de actualización, se puede observar un mejor desempeño cuando las topologías se actualizan con MIMIC, obteniéndose porcentajes de éxito ligeramente mayores cuando las topologías se actualizan cada 50 generaciones. Otro detalle importante a observar es que de las cuatro combinaciones probadas, la peor fue, por lo general, cuando las topologías se actualizan cada 100 generaciones con Chow-Liu, que fue la combinación que se comparó con los demás algoritmos del estado del arte. Es importante hacer notar esta situación, ya que, pese a esto, en muchas de las funciones probadas se obtuvieron mejores resultados.

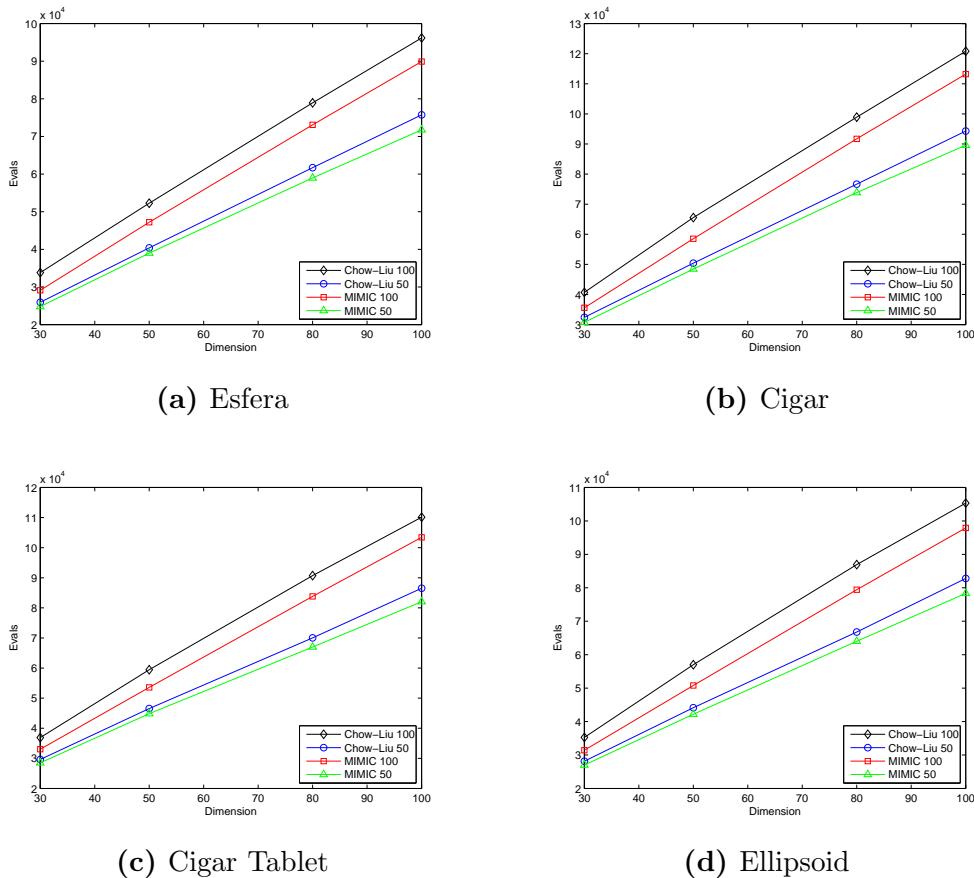


Figura C.1: Comparación del número de evaluaciones de función requeridas por cada combinación algoritmo-frecuencia de actualización para alcanzar el óptimo (parte 1)

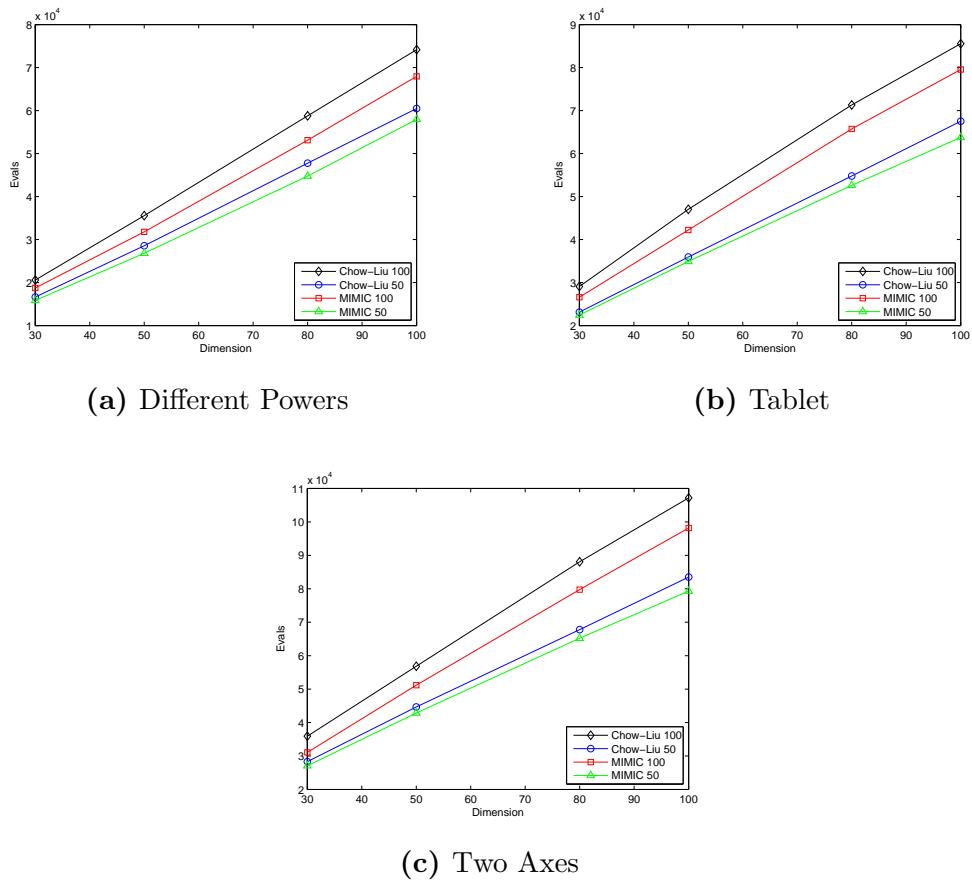


Figura C.2: Comparación del número de evaluaciones de función requeridas por cada combinación algoritmo-frecuencia de actualización para alcanzar el óptimo (parte 2)

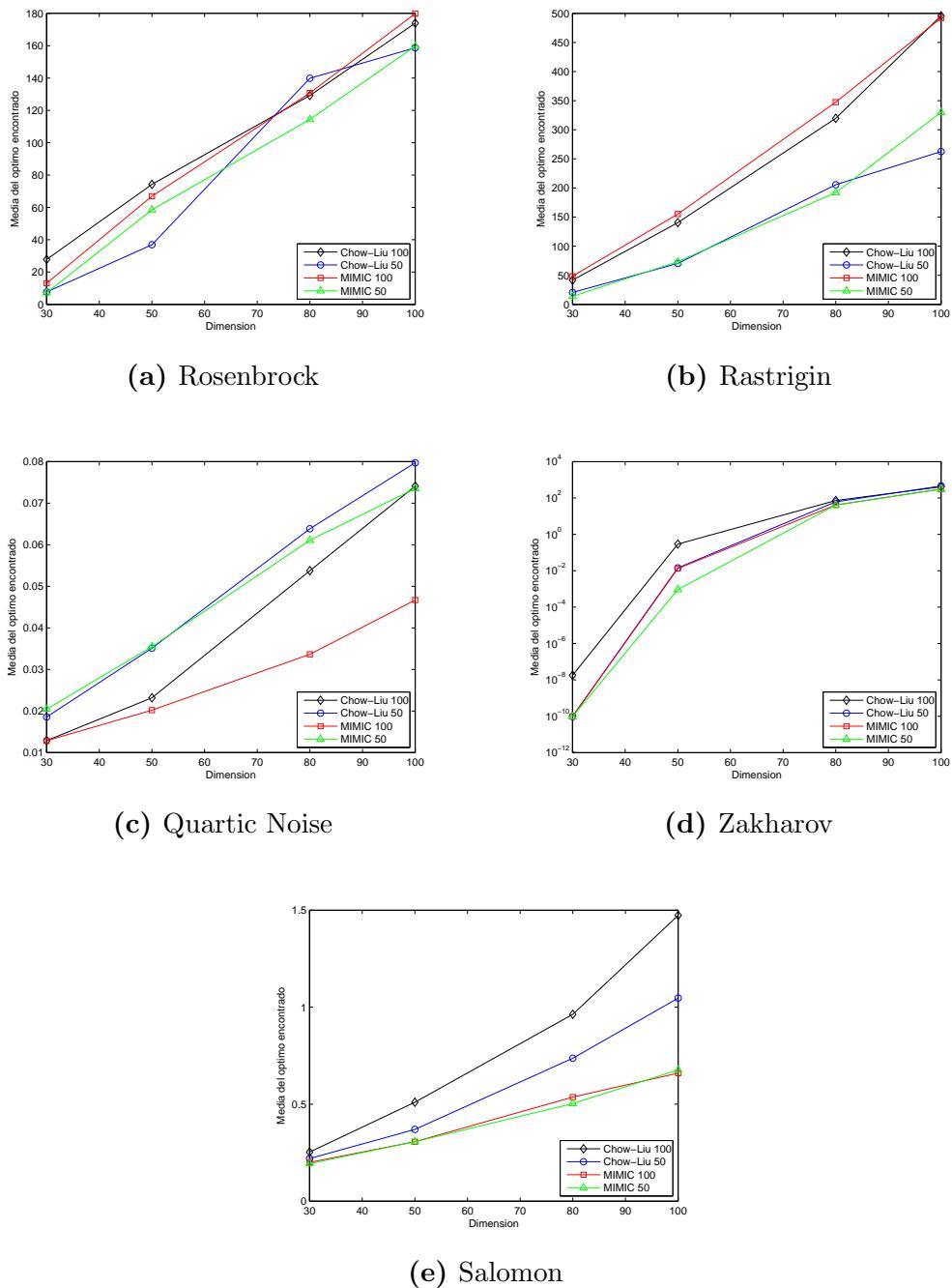


Figura C.3: Comparación de la media de la función de aptitud alcanzada por cada combinación algoritmo-frecuencia de actualización

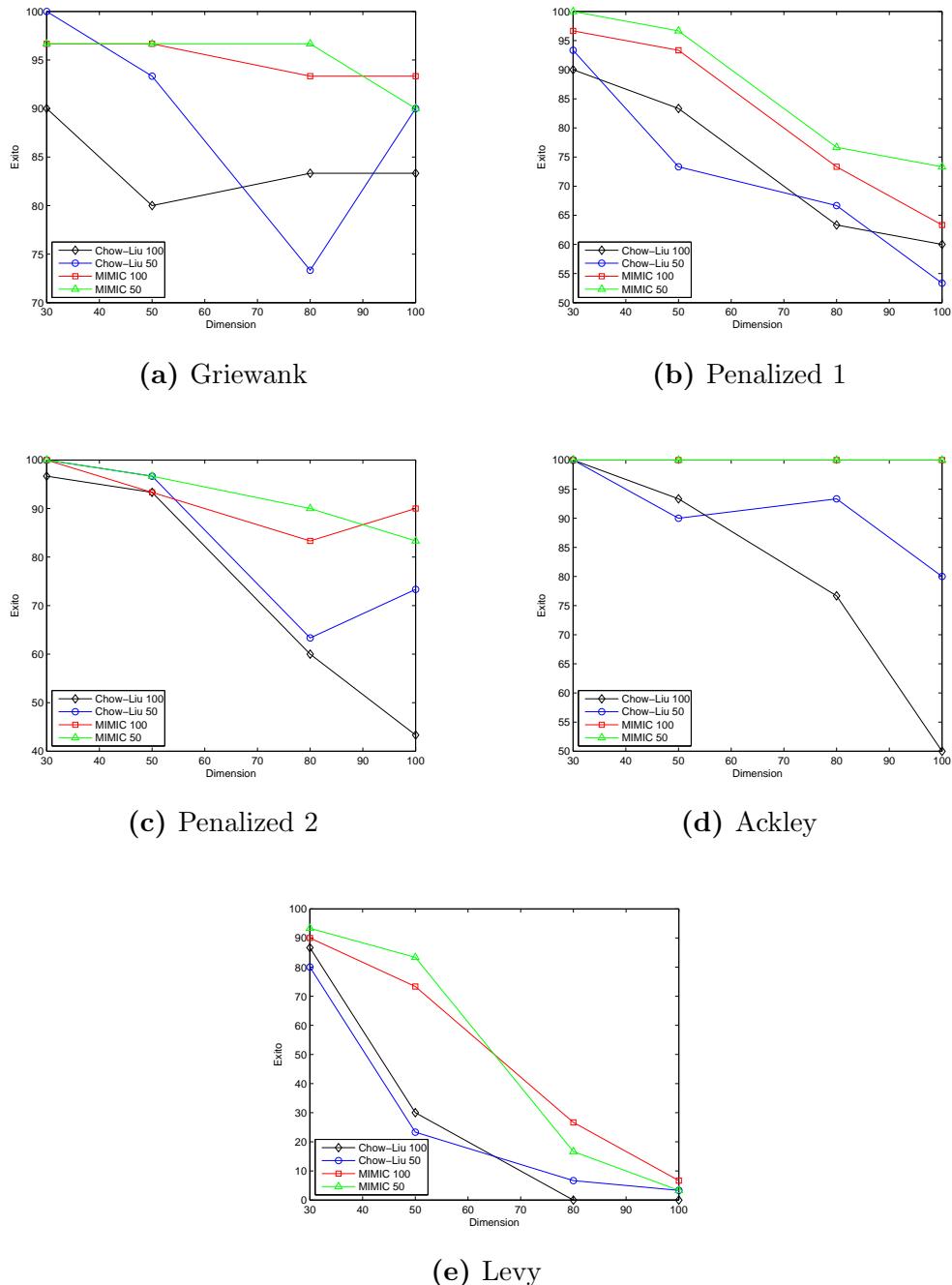


Figura C.4: Comparación del porcentaje de éxito de cada combinación algoritmo-frecuencia de actualización

Anexo D

Otros experimentos

En este anexo se presentan algunos experimentos realizados, en los que intentamos observar la evolución de las topologías y como el comportamiento de las partículas influye en la construcción de estas. También, se muestran experimentos en los que se pretende observar la influencia de la información mutua en la convergencia del algoritmo. Para realizar las gráficas de los árboles construidos se utilizó Bayes Net Toolbox for Matlab, el cual se puede descargar en <http://code.google.com/p/bnt>.

D.1 Evolución de los árboles construidos y movimiento de las partículas

El objetivo de este experimento es observar como se comportan las topologías de PSO al construirlas con Chow-Liu (algoritmo 13). También se busca ver como influyen las posiciones que han visitado las partículas en la construcción del modelo gráfico, así como la influencia del modelo gráfico correspondiente en el comportamiento de las partículas. Para este experimento se resolvió la función esfera en dimensión 3 con PSO-MI, con parámetros $\omega = 0.578766$, $c_1 = c_2 = 1.49618$, $S = 10$, $f = 20$, topología inicial todos desconectados y construyendo los vecindarios con Chow-Liu. El algoritmo se detuvo cuando se obtuvo el valor del óptimo con una tolerancia de $\epsilon = 10^{-10}$. En las figuras D.1 y D.2 se muestra esta evolución para la dimensión 1 de la función, las figuras D.3 y D.4 para la dimensión 2 y las figuras D.5 y D.6 para la dimensión 3 del problema. En las figuras se muestra el movimiento de las partículas por cada 20 generaciones (excepto cuando converge el algoritmo, en cuyo caso son 15 iteraciones), y posteriormente el árbol que se construyó con estos datos. También, después de cada árbol se muestra como se movieron las partículas. De igual forma, en las figuras en donde se muestran los árboles se agrega el conjunto de partículas conectadas entre si y cual fue la información mutua en cada caso.

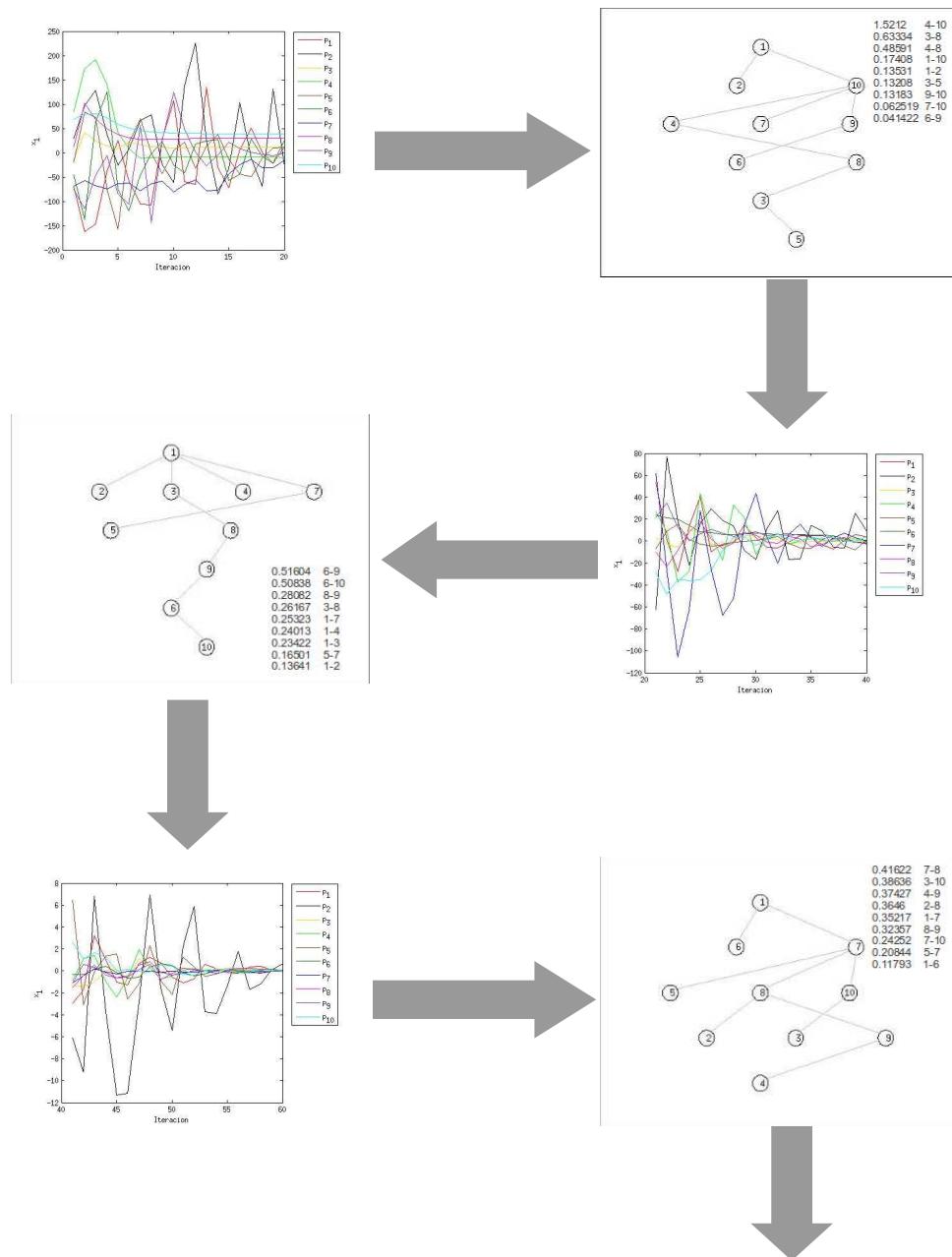


Figura D.1: Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 1, parte 1

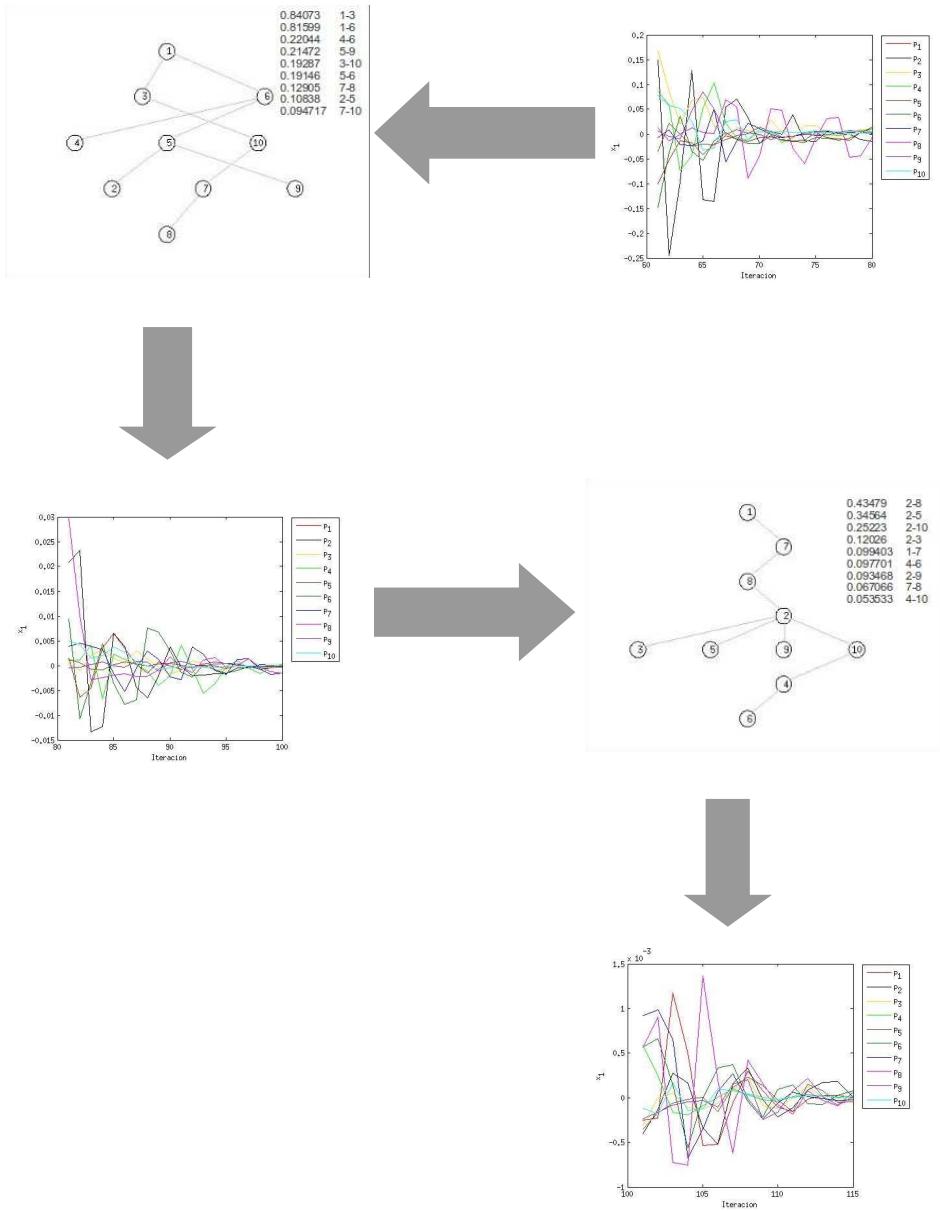


Figura D.2: Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 1, parte 2

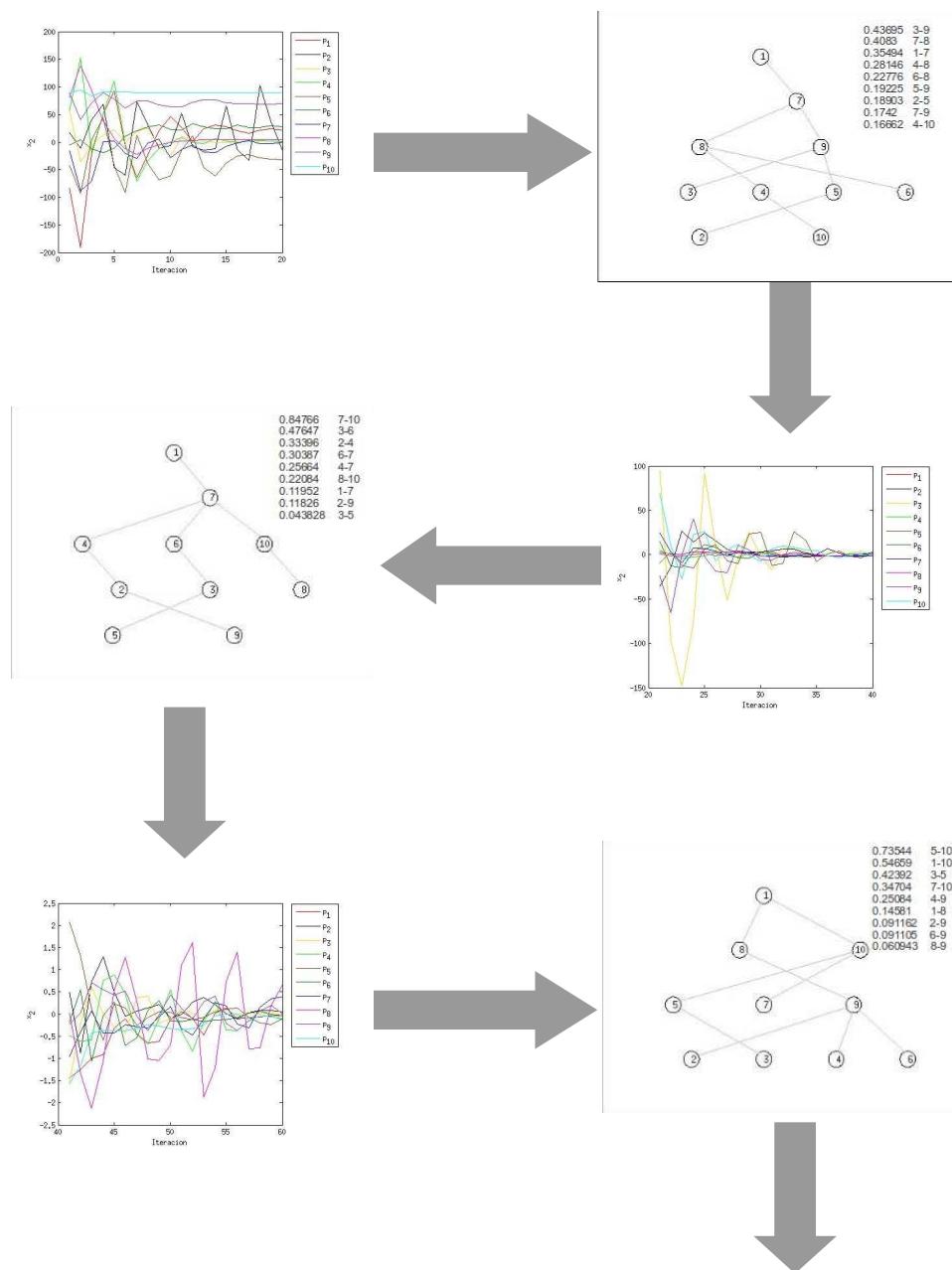


Figura D.3: Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 2, parte 1

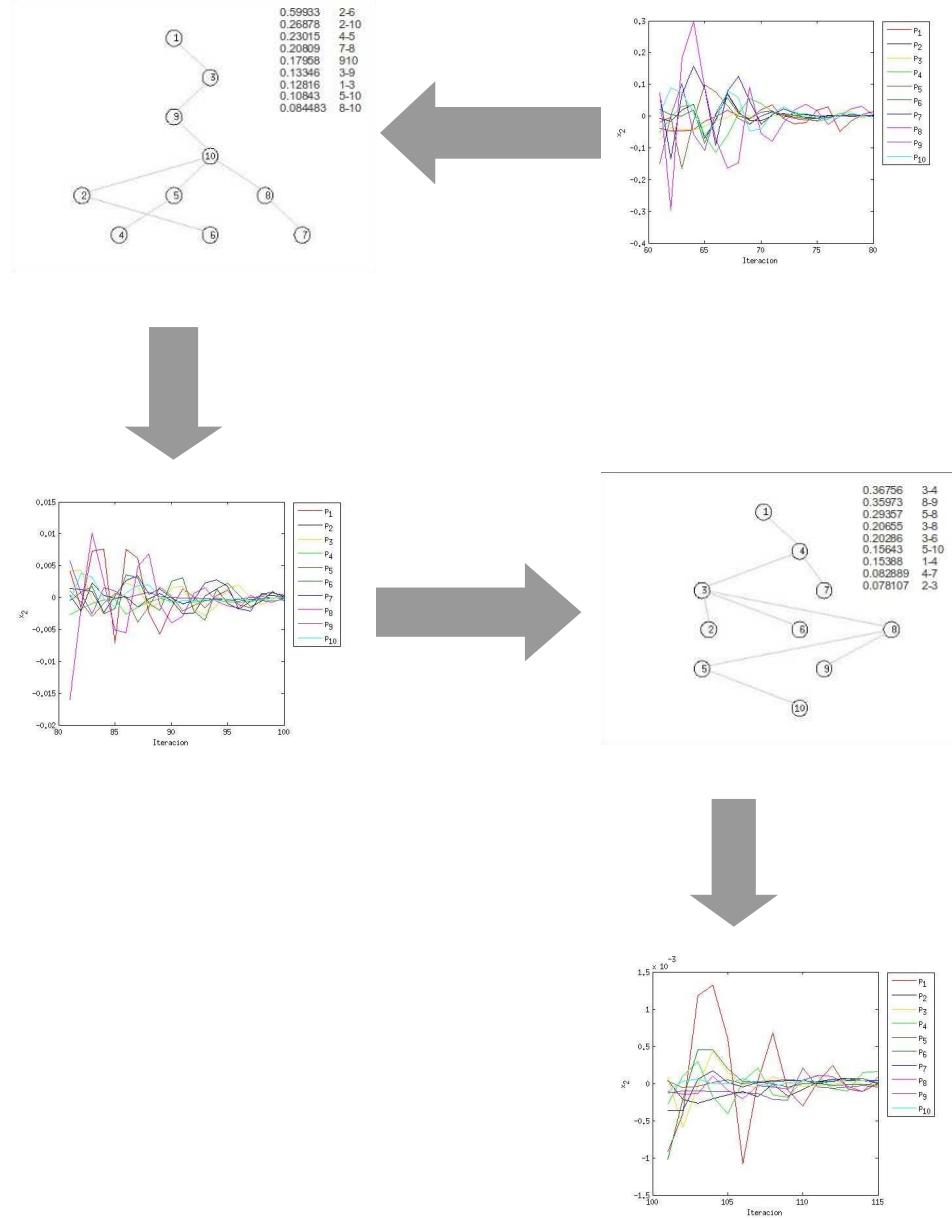


Figura D.4: Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 2, parte 2

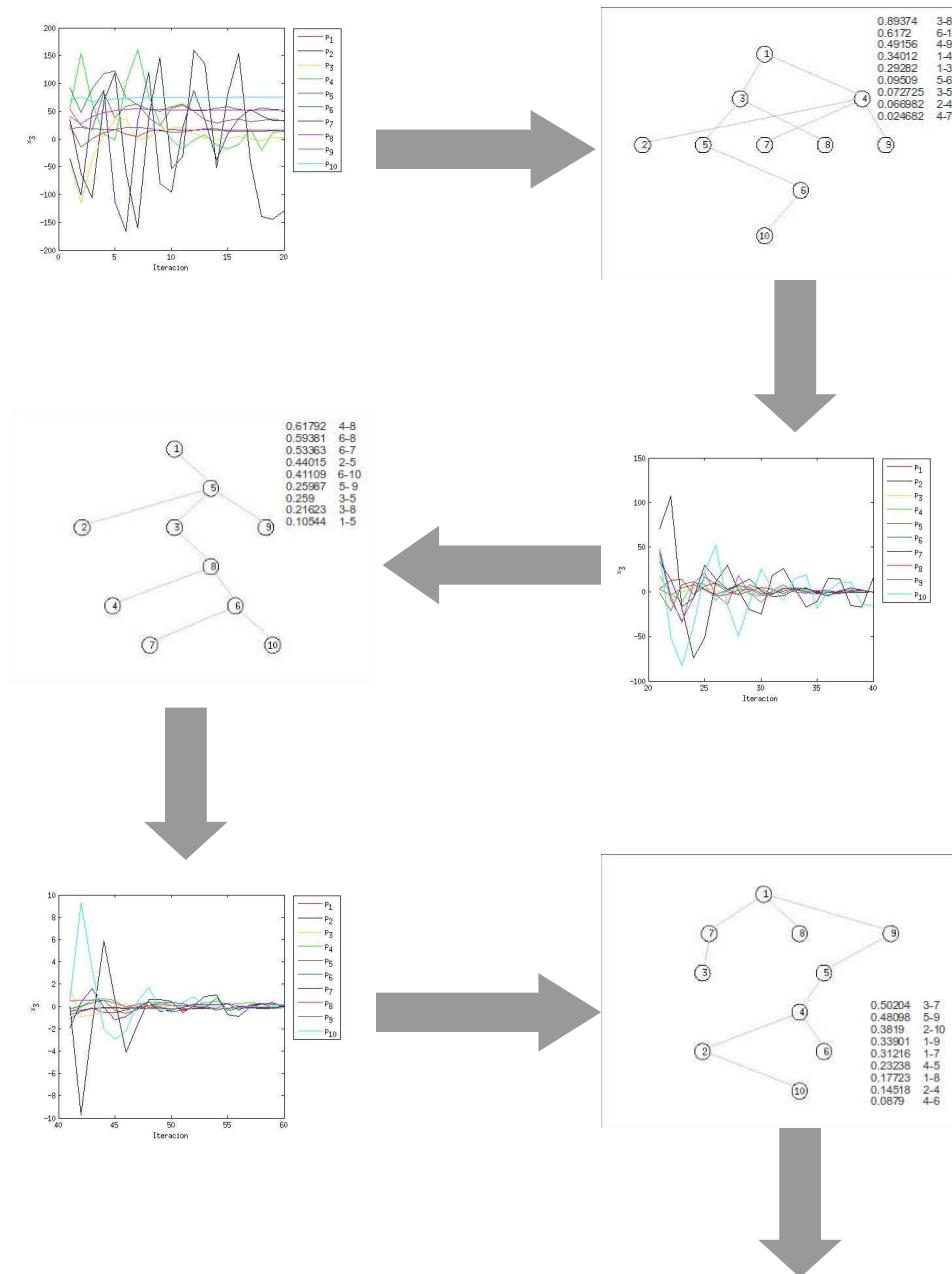


Figura D.5: Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 3, parte 1

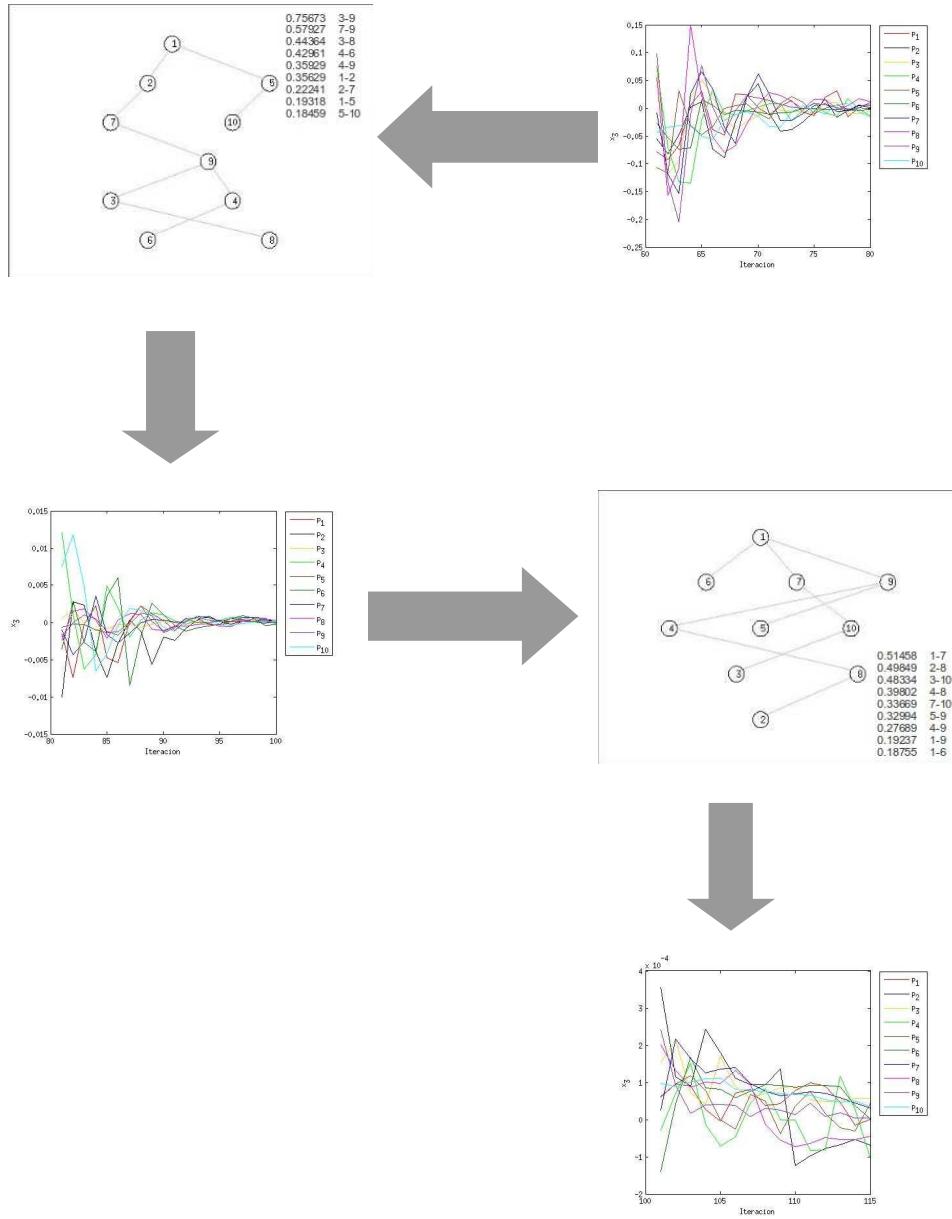


Figura D.6: Evolución de las topologías y movimiento de las partículas para la función esfera en dimensión 3. Variable 3, parte 2

D.2 Evolución de los árboles construidos y movimiento de las partículas por pares conectados

Este experimento es similar al anterior, sólo que ahora en lugar de ver las partículas juntas, decidimos separarlos por pares conectados, con la finalidad de observar si presentaban algún comportamiento similar que provocará que estas se conectarán en el modelo gráfico. Es decir, si en el modelo gráfico las partículas 5 y 15 están conectadas, entonces en una subimagen se muestra el movimiento de estas dos partículas por separado. En la figura D.7 se muestra el movimiento de las partículas en las primeras cien generaciones. Posteriormente, en la figura D.8 se muestra el primer árbol construido, proveniente del comportamiento de las partículas mostrado en la figura D.7. Después, en la figura D.9 se muestra el movimiento de las partículas de la generación 100 a la 200. Este comportamiento de las partículas se ve afectado por la topología mostrada en la figura D.8, y a su vez genera el árbol mostrado en la figura D.16. De este modo, el comportamiento mostrado en la figura D.11 corresponde al árbol de la figura D.11 y así sucesivamente hasta las figuras D.27 y D.28. Para este experimento se resolvió la función esfera en dimensión 30 con PSO-MI, con parámetros $\omega = 0.578766$, $c_1 = c_2 = 1.49618$, $S = 30$, $f = 100$, topología inicial todos desconectados y construyendo los vecindarios con Chow-Liu. El algoritmo se detuvo cuando se obtuvo el valor del óptimo con una tolerancia de $\epsilon = 10^{-10}$. Las figuras mostradas corresponden solamente a la dimensión 1 del problema a resolver. Como se puede observar en las figuras, en mucha ocasiones si se conectan partículas que tuvieron un movimiento parecido.

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

148

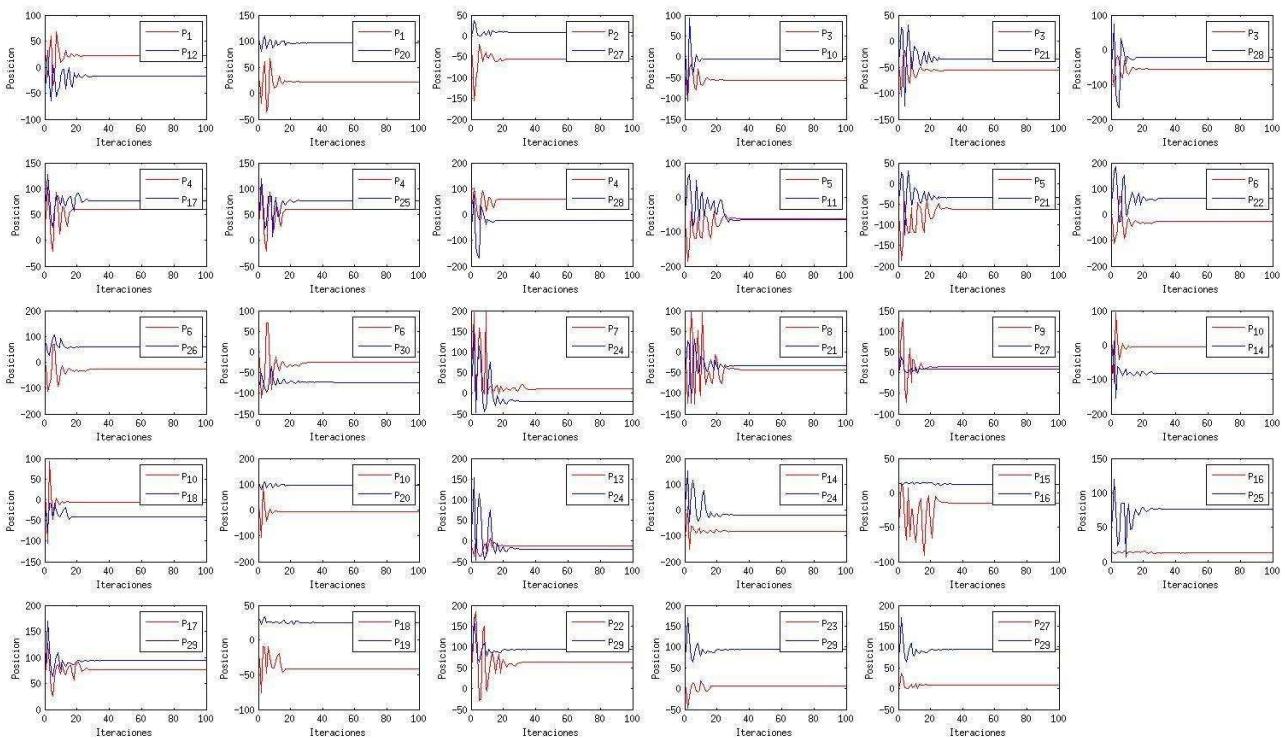


Figura D.7: Movimiento de las partículas antes de la construcción del primer árbol, separadas por pares de partículas conectadas

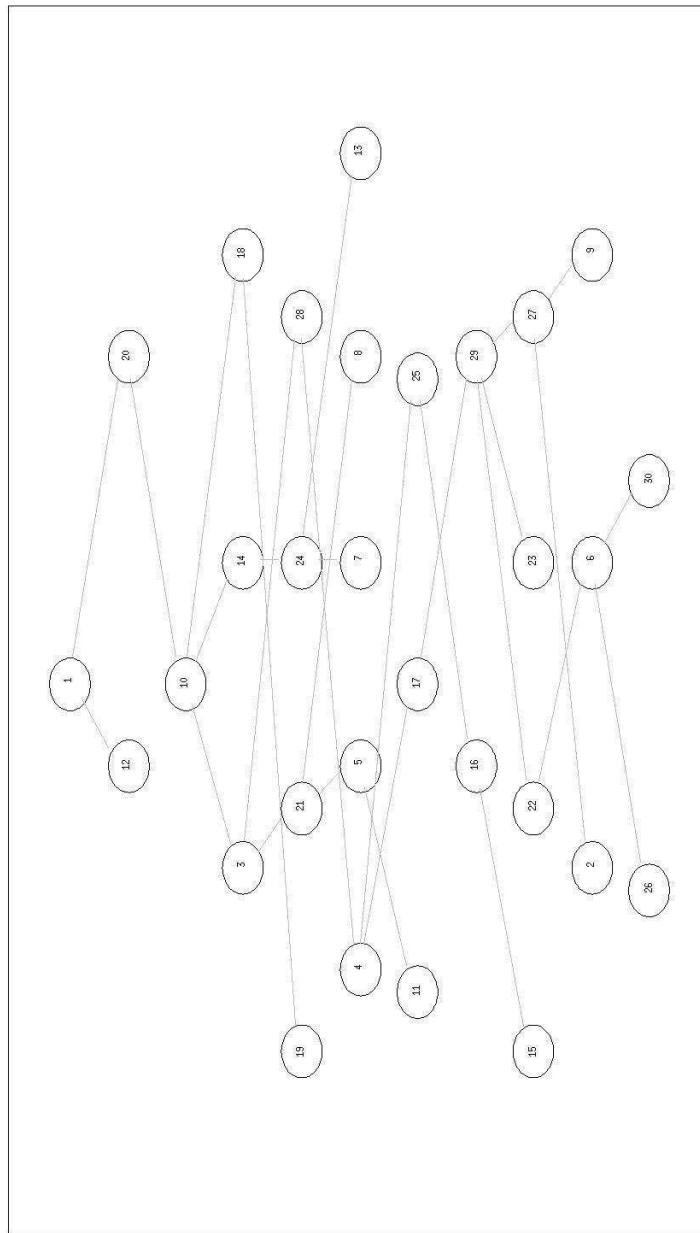


Figura D.8: Primer árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

150

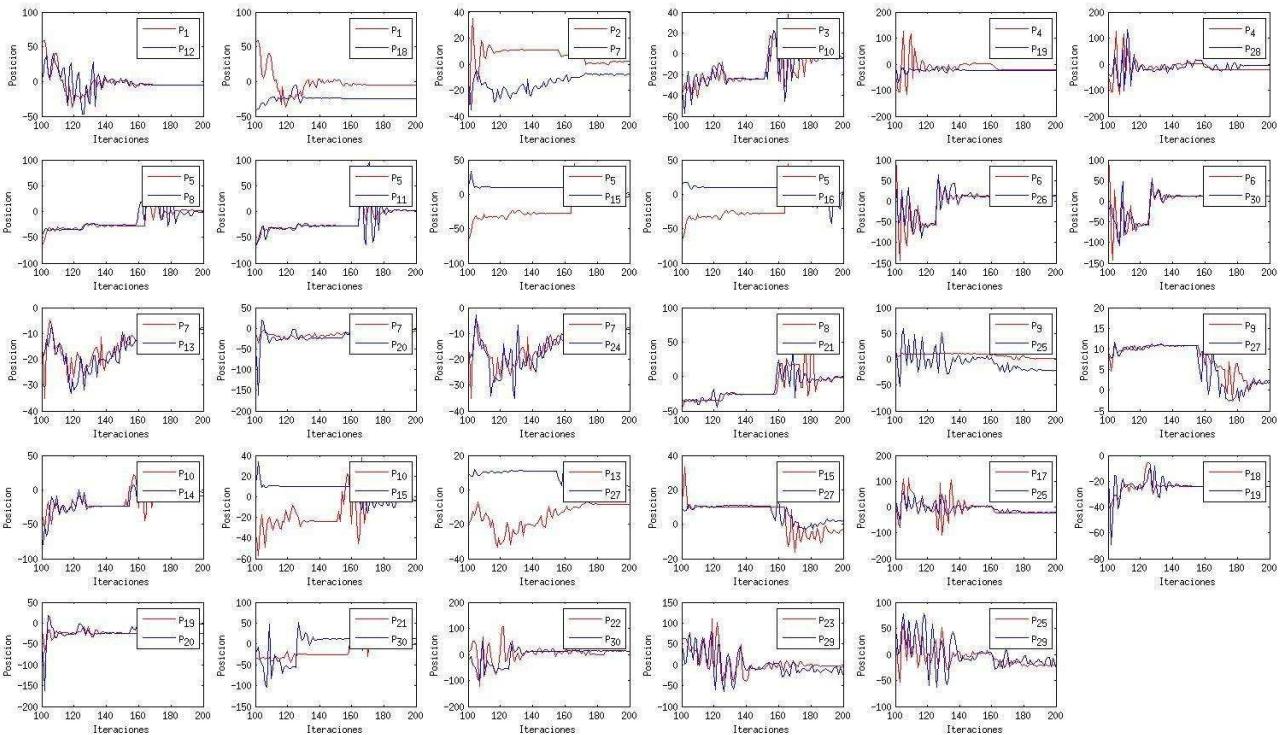


Figura D.9: Movimiento de las partículas antes de la construcción del segundo árbol, separadas por pares de partículas conectadas

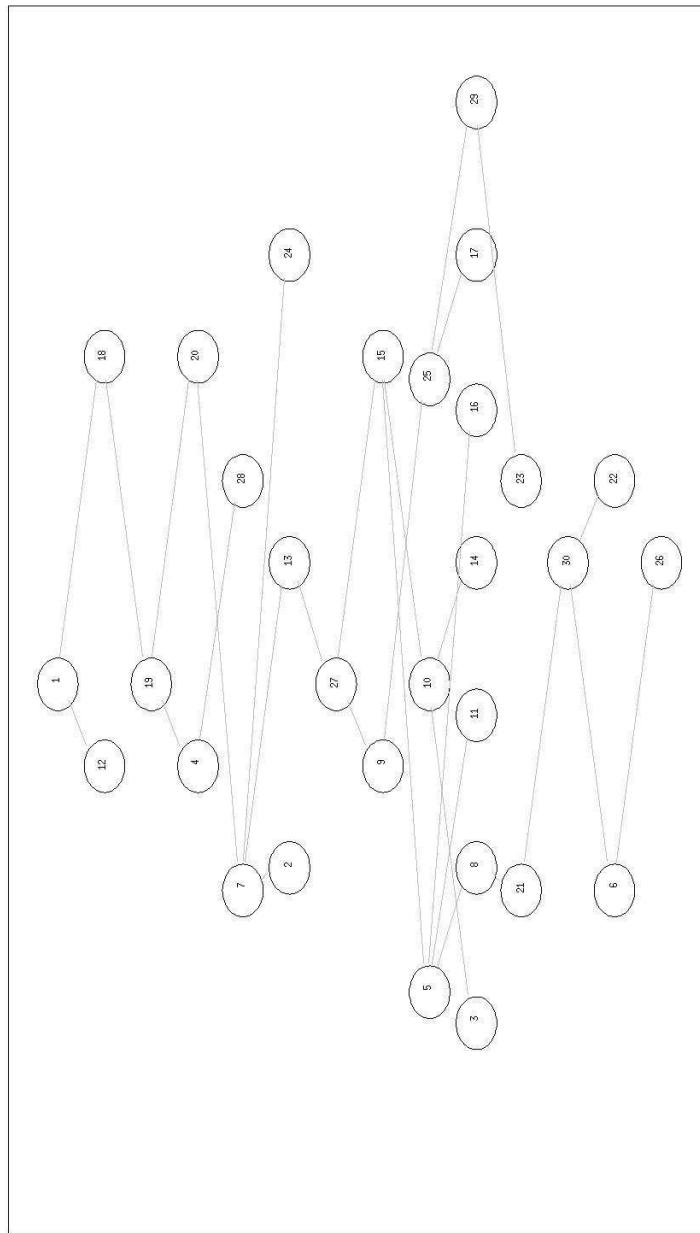


Figura D.10: Segundo árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

152

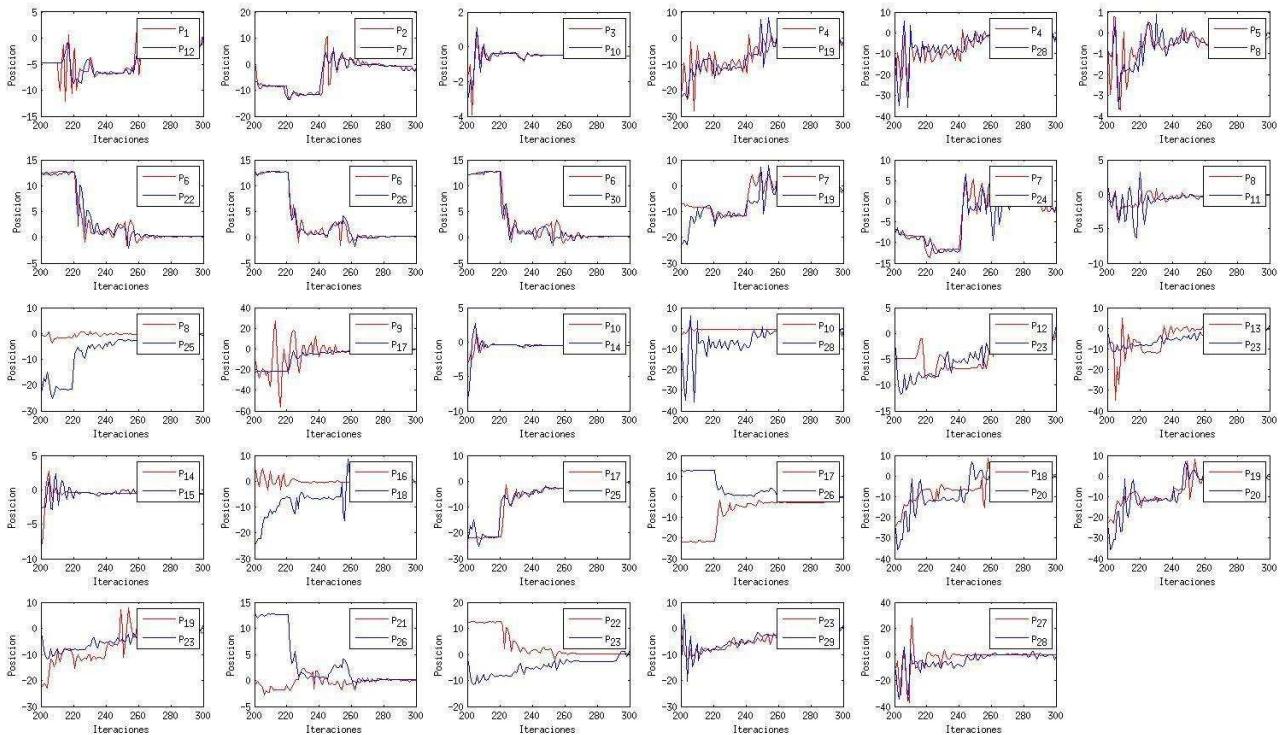


Figura D.11: Movimiento de las partículas antes de la construcción del tercer árbol, separadas por pares de partículas conectadas

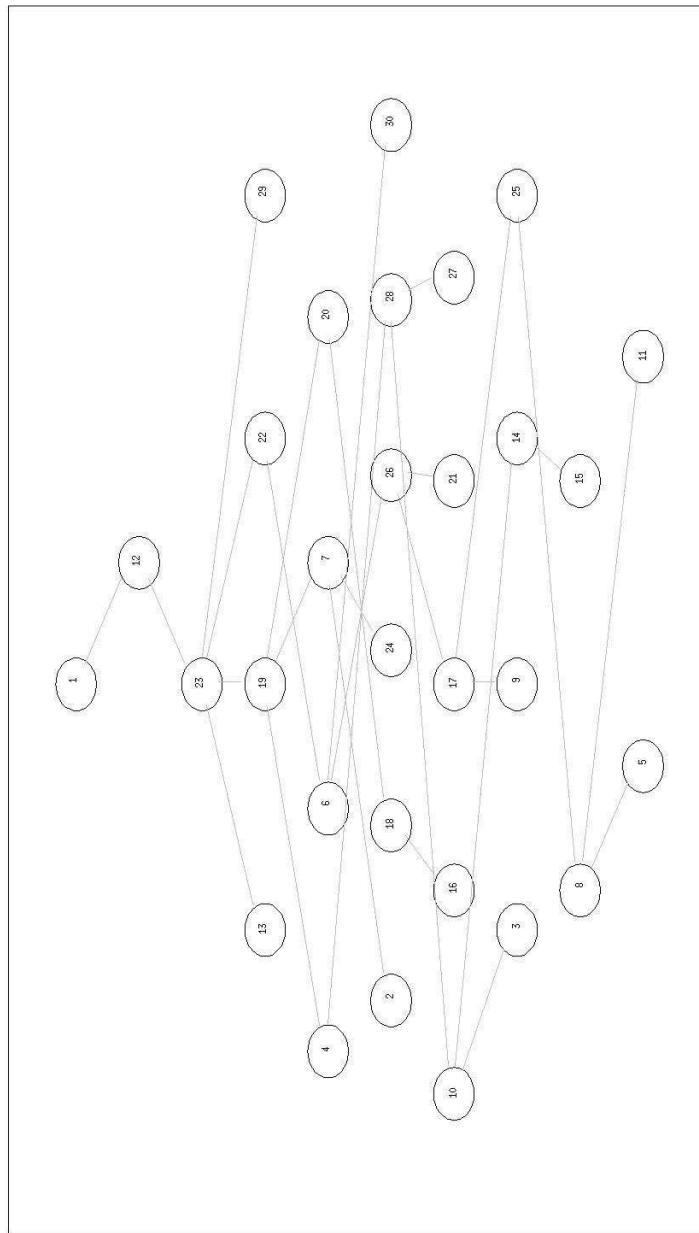


Figura D.12: Tercer árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

154

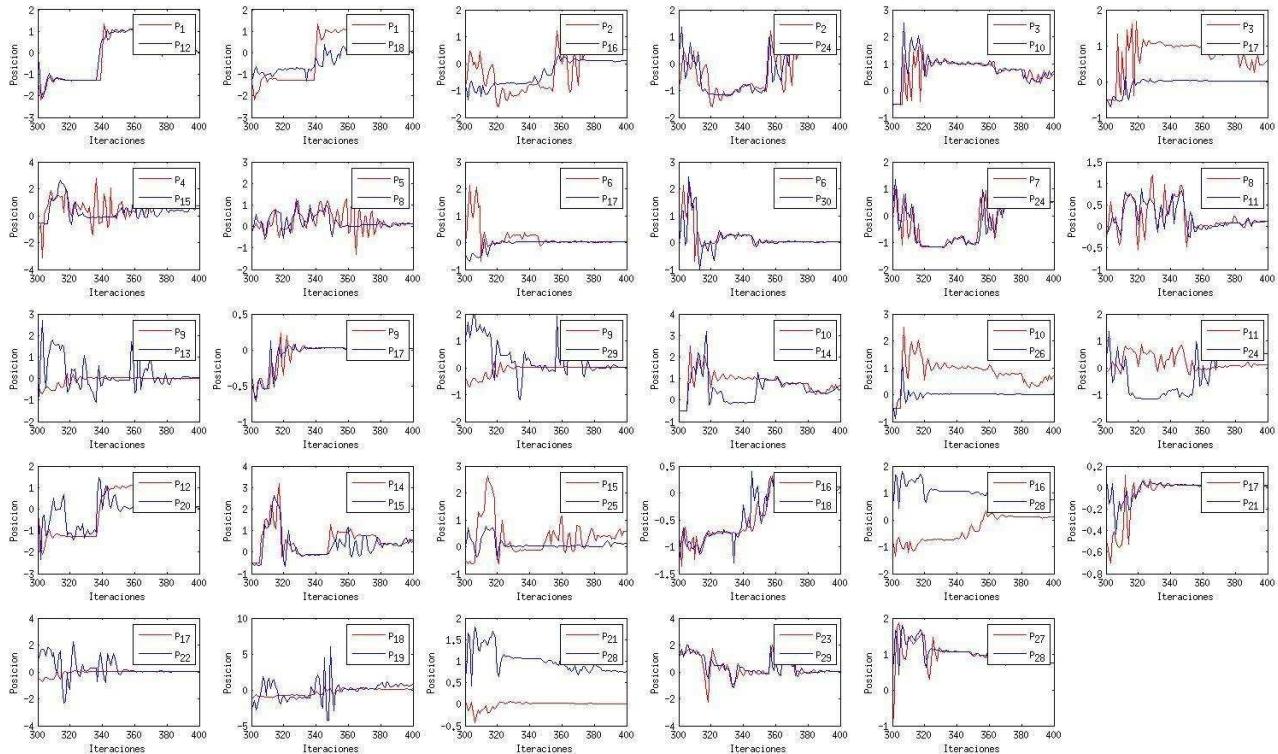


Figura D.13: Movimiento de las partículas antes de la construcción del cuarto árbol, separadas por pares de partículas conectadas

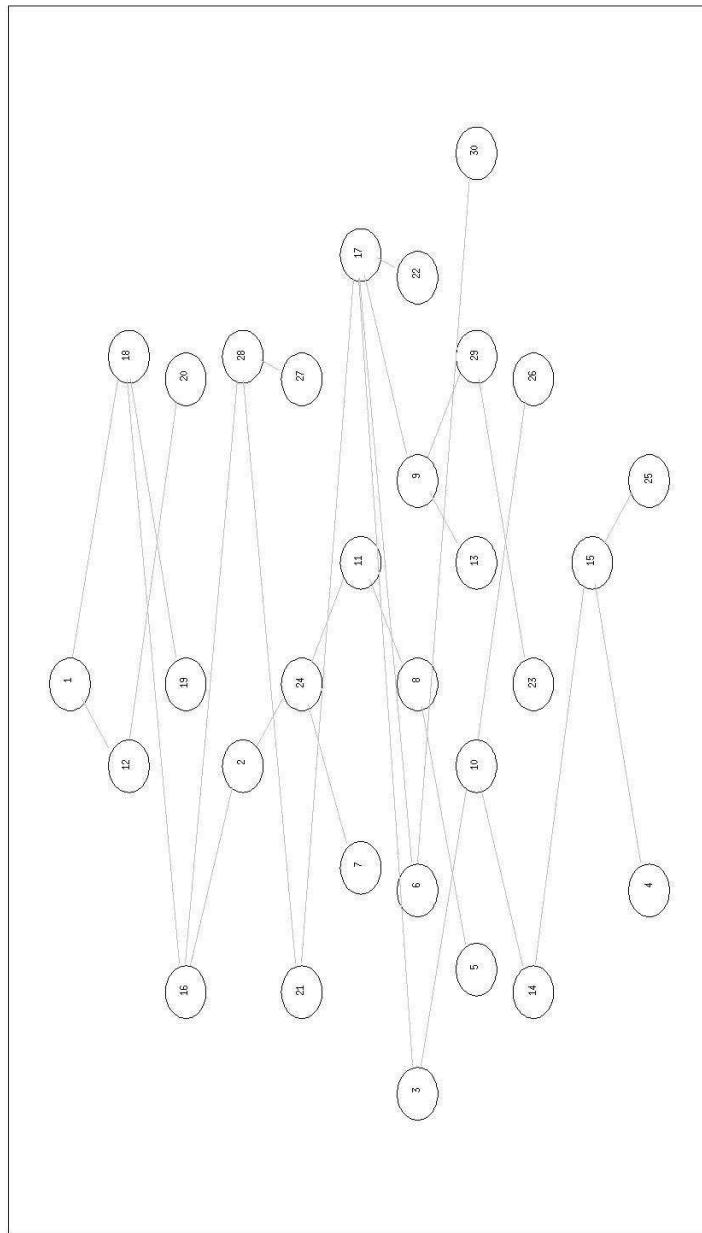


Figura D.14: Cuarto árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

156

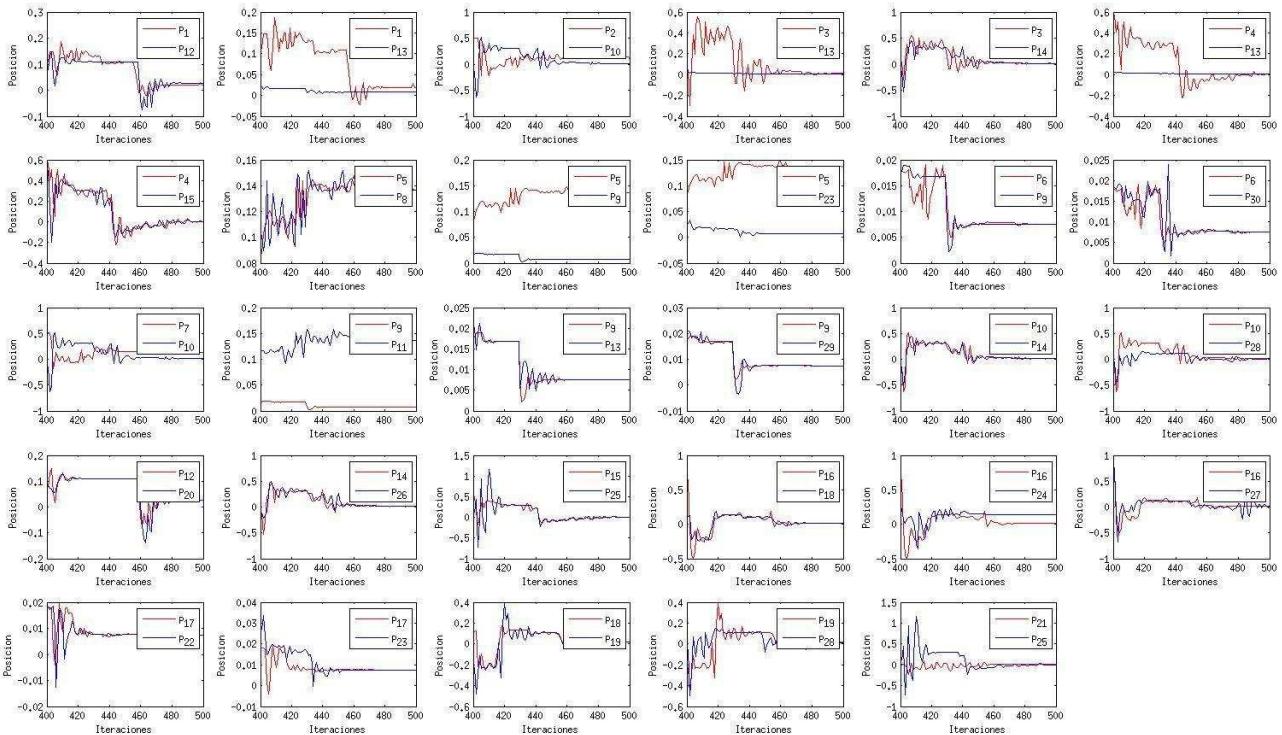


Figura D.15: Movimiento de las partículas antes de la construcción del quinto árbol, separadas por pares de partículas conectadas

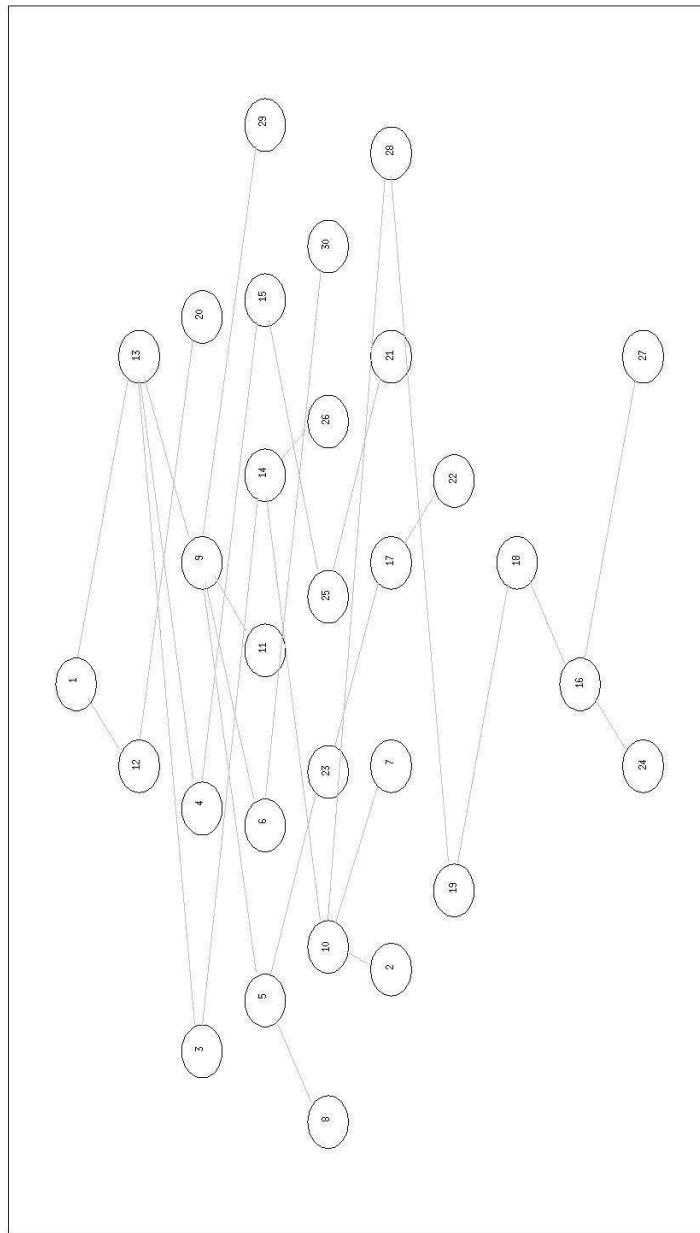


Figura D.16: Quinto árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

158

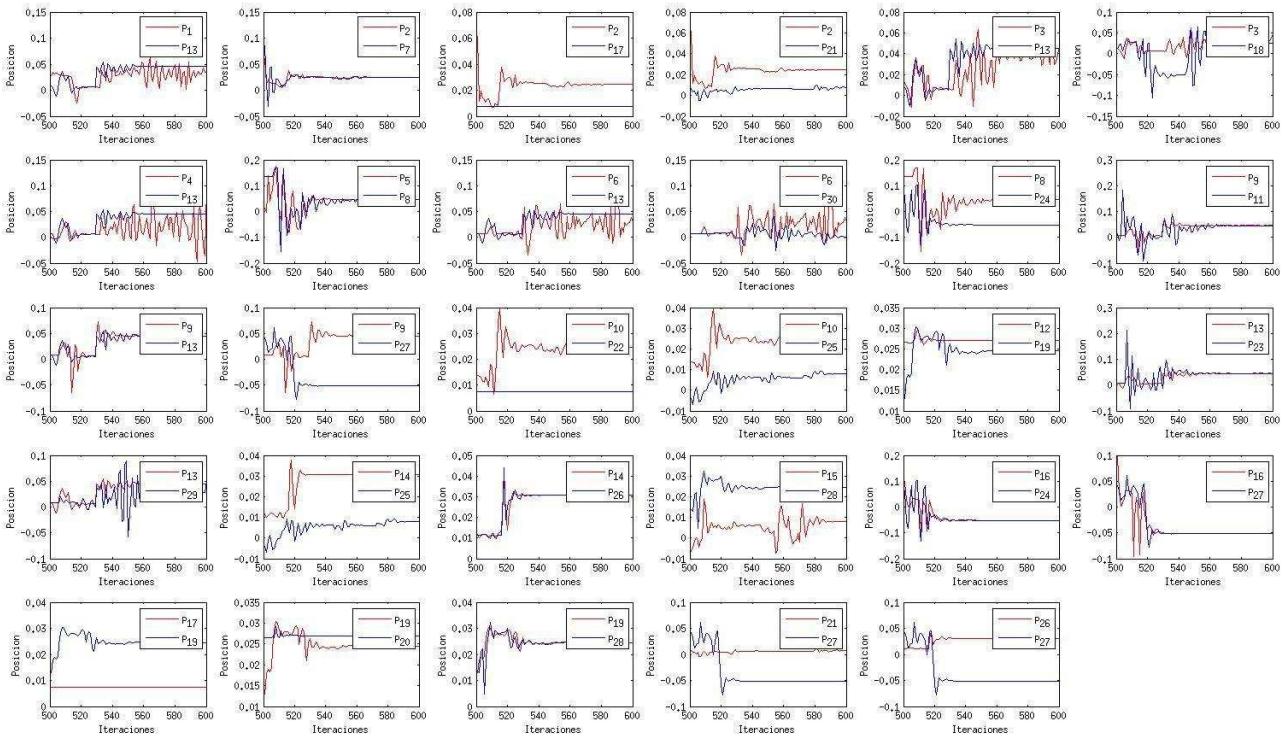


Figura D.17: Movimiento de las partículas antes de la construcción del sexto árbol, separadas por pares de partículas conectadas

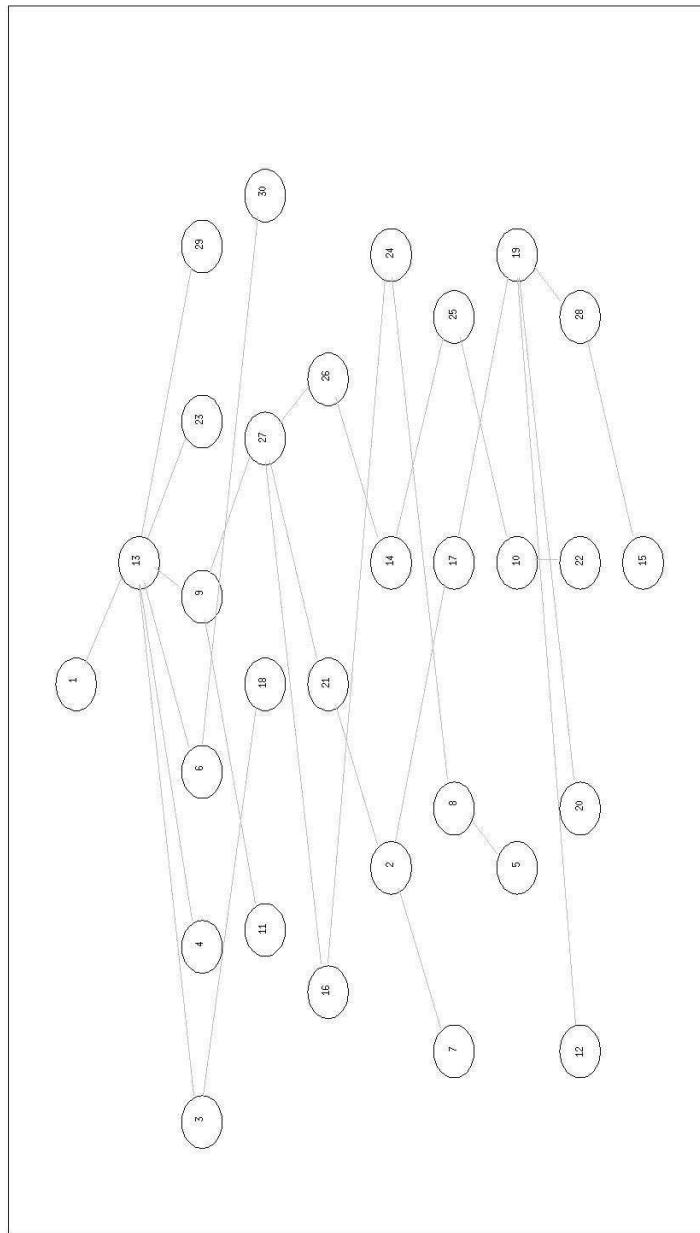


Figura D.18: Sexto árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

160

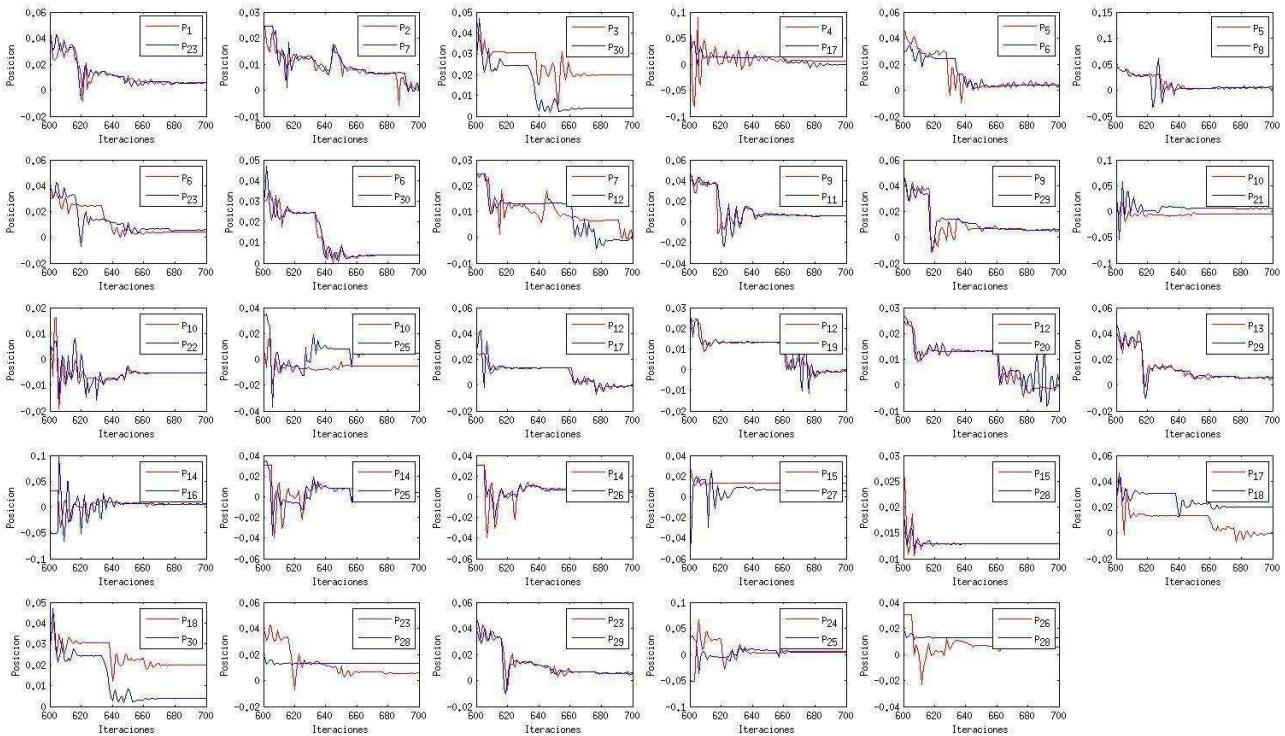


Figura D.19: Movimiento de las partículas antes de la construcción del séptimo árbol, separadas por pares de partículas conectadas

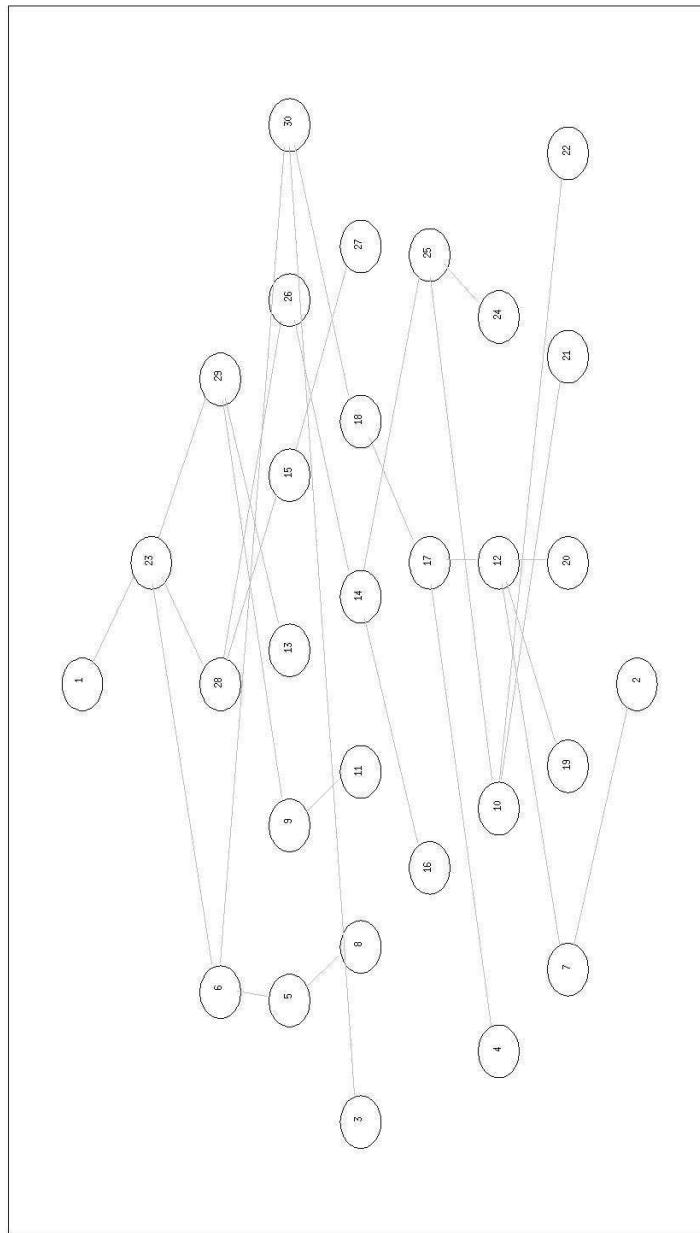


Figura D.20: Séptimo árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

162

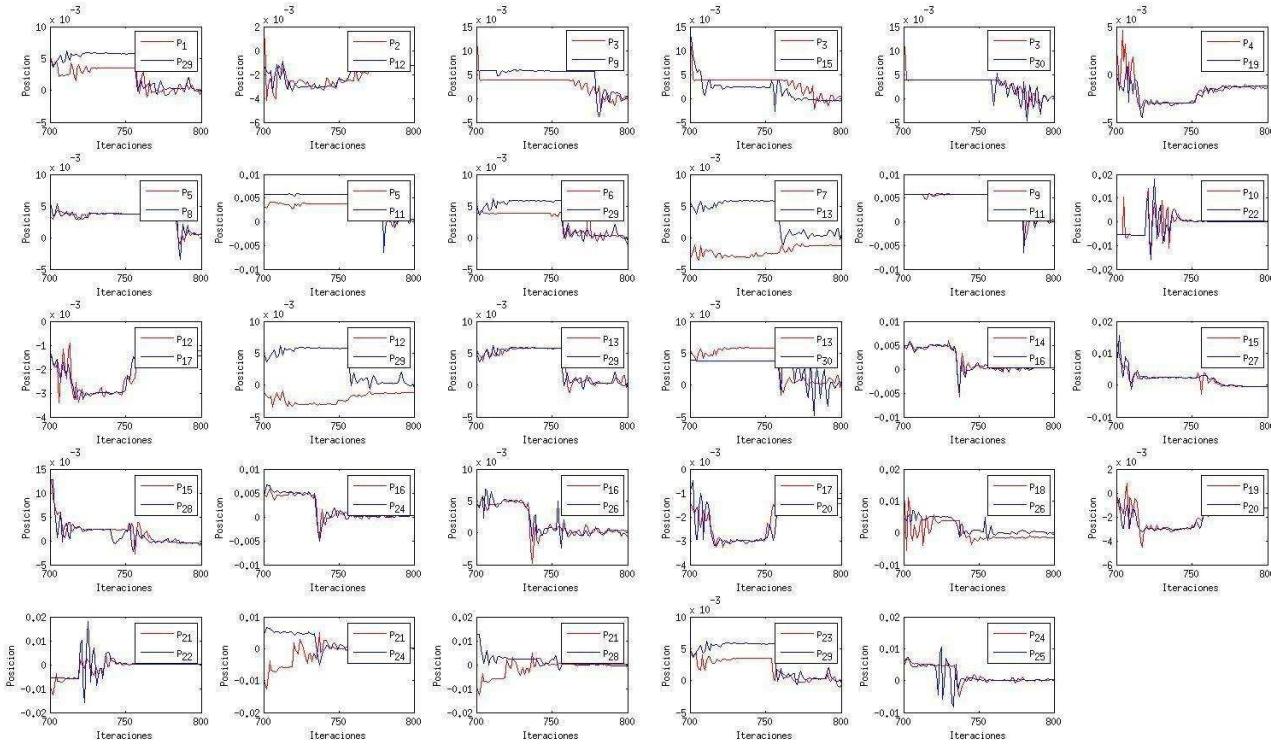


Figura D.21: Movimiento de las partículas antes de la construcción del octavo árbol, separadas por pares de partículas conectadas

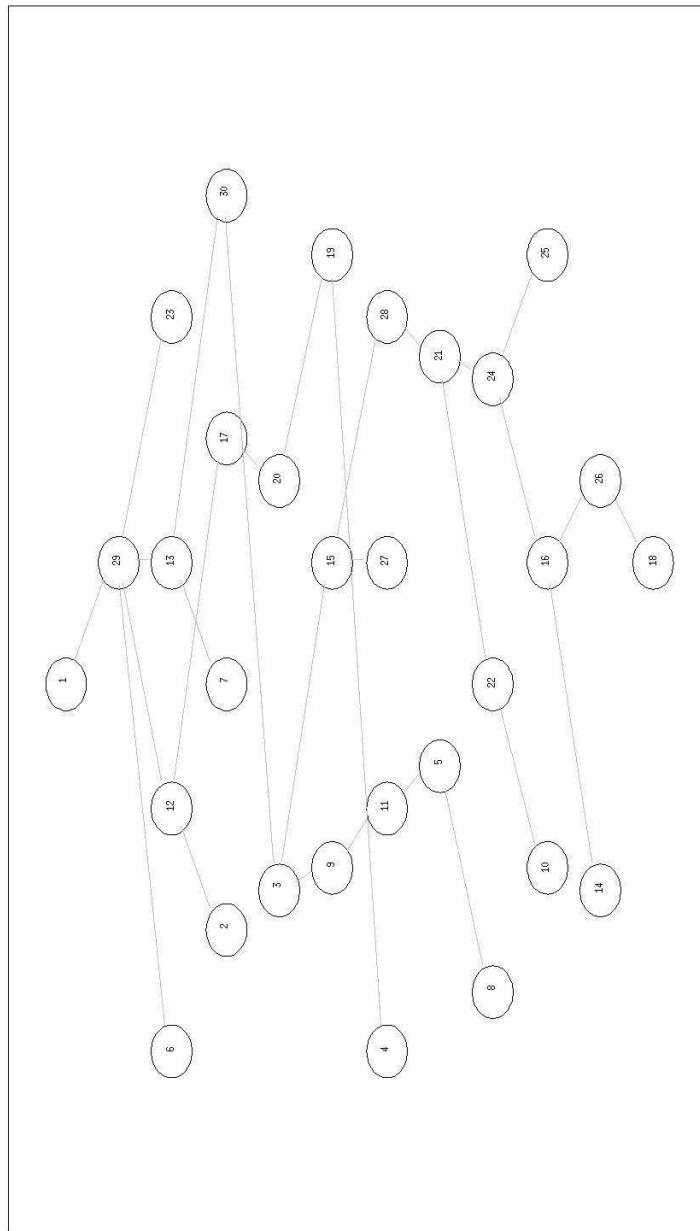


Figura D.22. Octavo árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

164

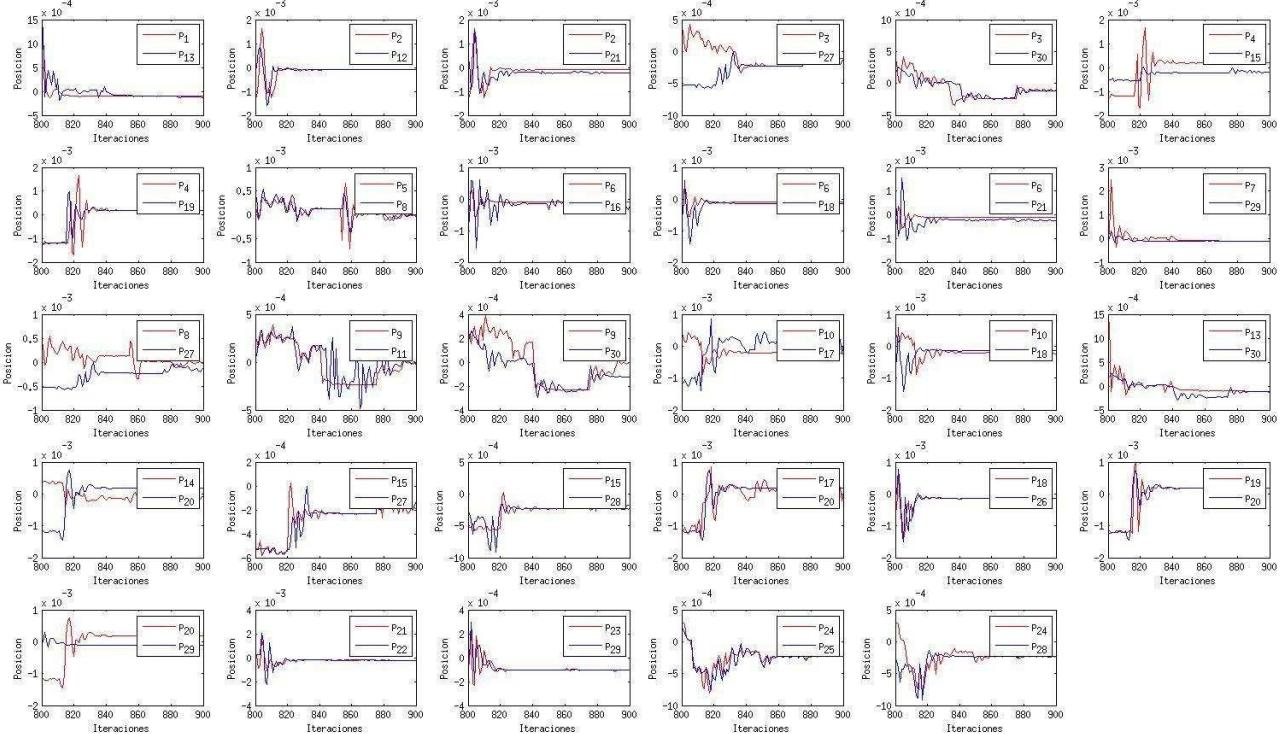


Figura D.23: Movimiento de las partículas antes de la construcción del noveno árbol, separadas por pares de partículas conectadas

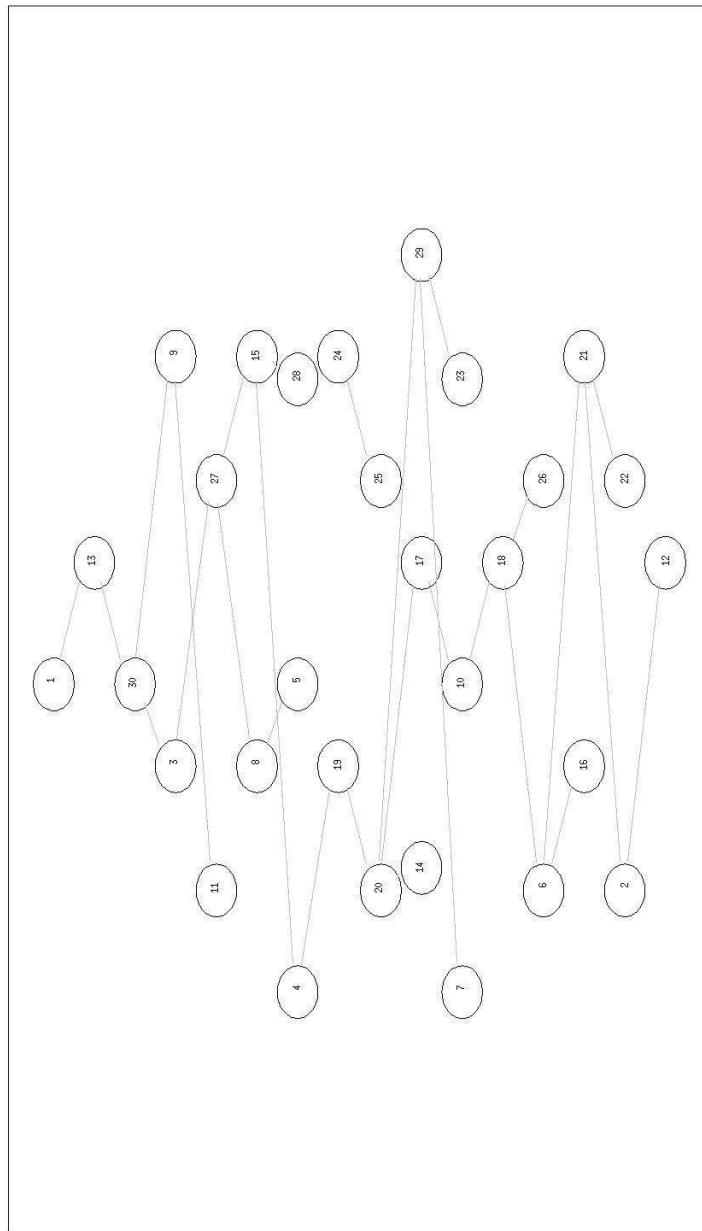


Figura D.24: Noveno árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

166

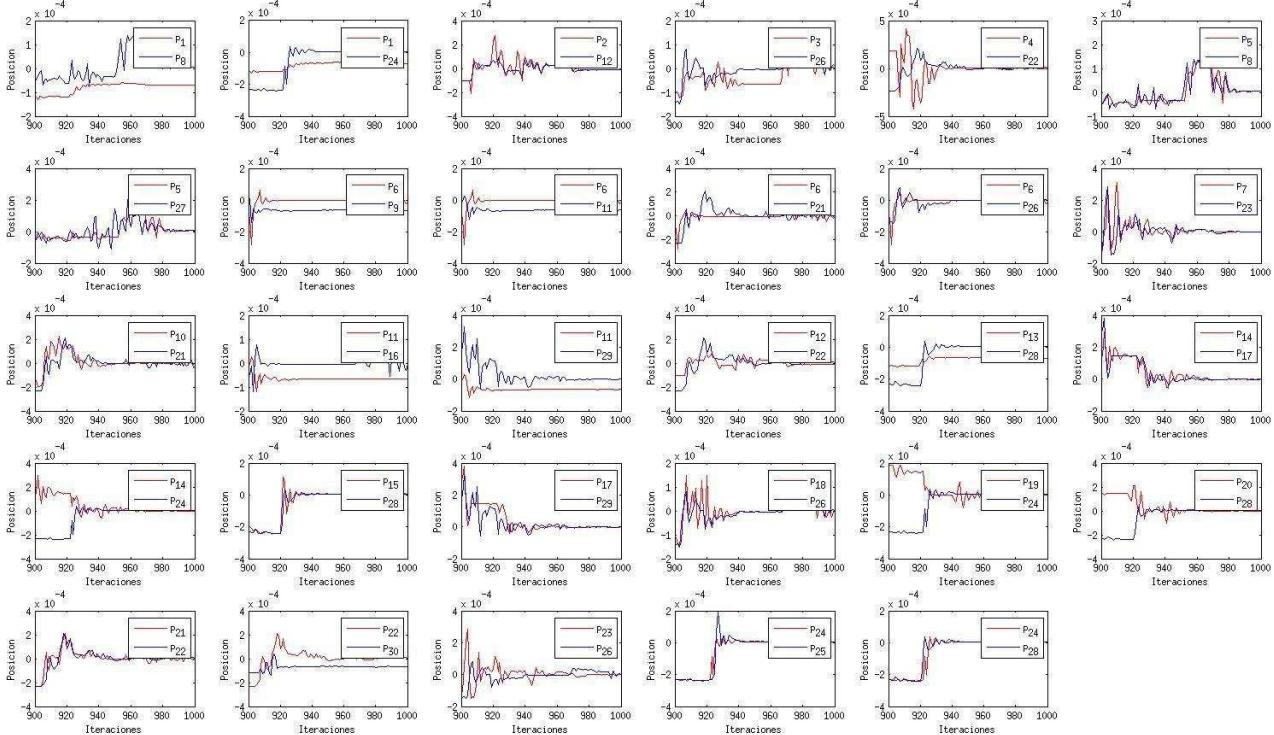


Figura D.25: Movimiento de las partículas antes de la construcción del décimo árbol, separadas por pares de partículas conectadas

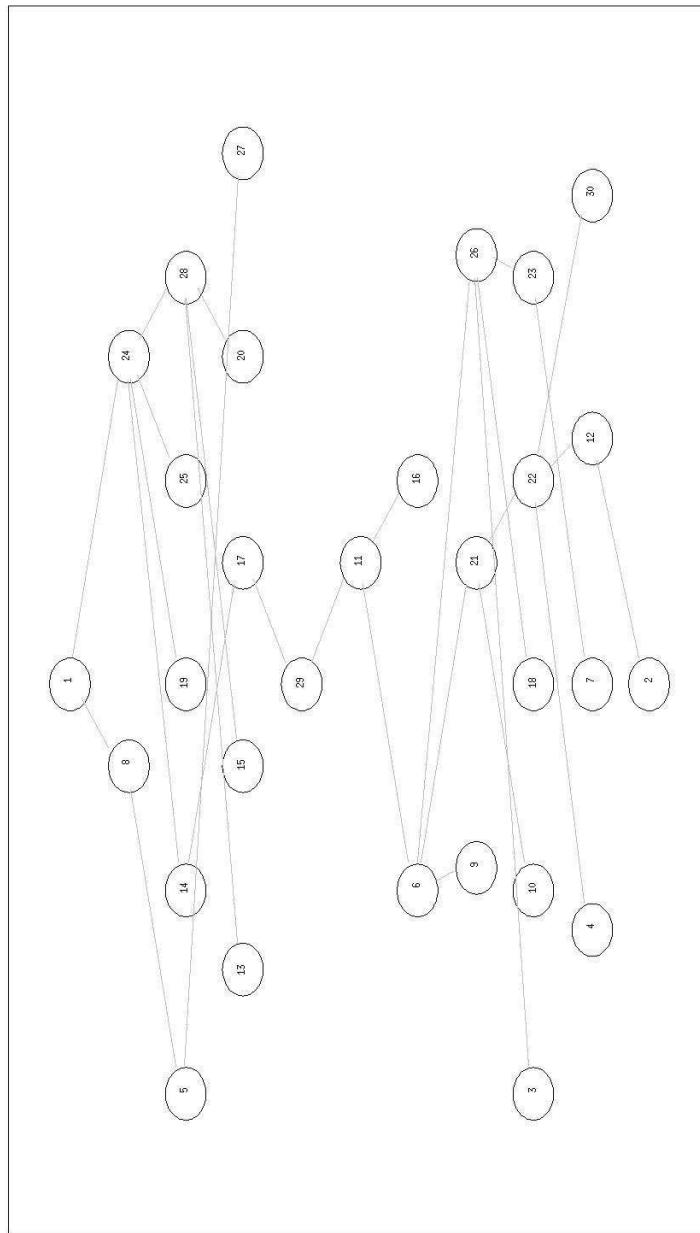


Figura D.26: Décimo árbol de partículas construido

D.2. EVOLUCIÓN DE LOS ÁRBOLES CONSTRUIDOS Y MOVIMIENTO DE LAS PARTÍCULAS POR PARES CONECTADOS

168

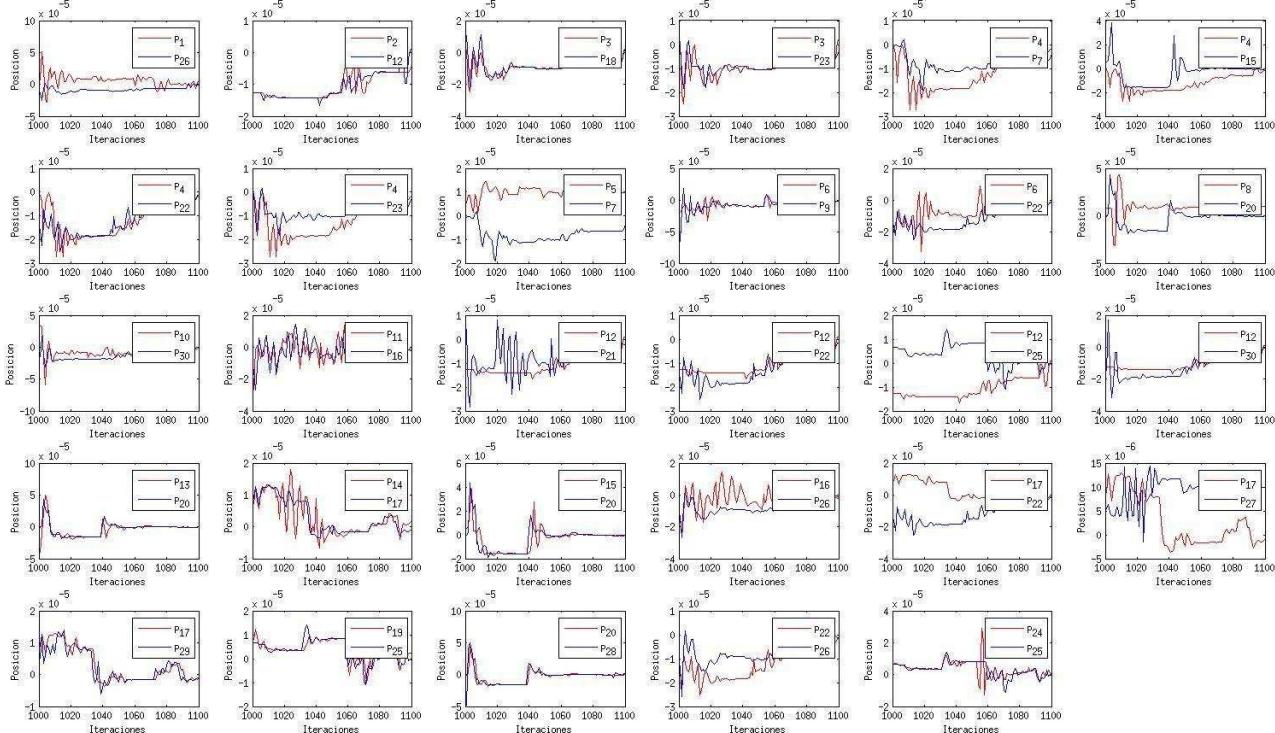


Figura D.27: Movimiento de las partículas antes de la construcción del undécimo árbol, separadas por pares de partículas conectadas

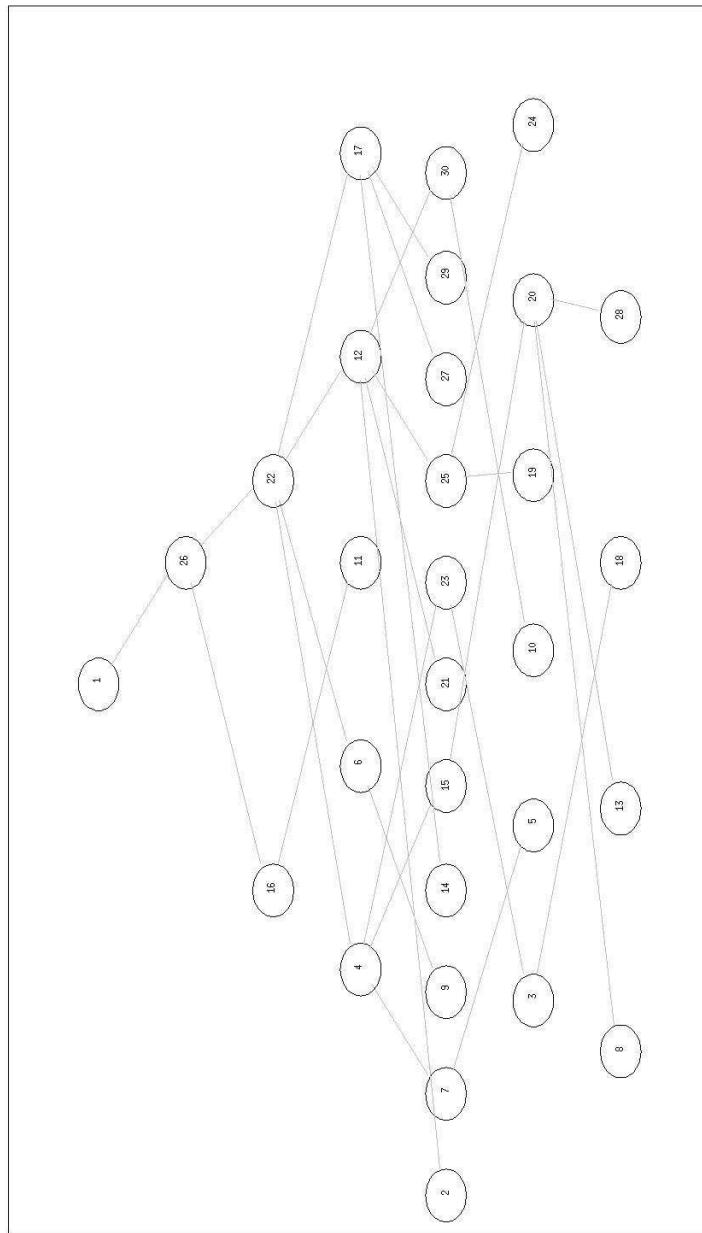


Figura D.28: Undécimo árbol de partículas construido

D.3 Conectividad de los árboles generados

En este experimento se pretende observar cual es la máxima conectividad que puede tener una partícula en las topologías construidas con Chow-Liu. Cada vez que se actualizaban los vecindarios, se buscaba la partícula que tuviera mayor conectividad y este valor es el que se grafica. En la figura D.29 se muestra la máxima conectividad de todas las topologías para cada dimensión durante el desarrollo del algoritmo. Para este experimento se resolvió la función Esfera en dimensión 30 con los mismos parámetros utilizados en el experimento de la sección D.2. Como se puede observar, la topología nunca rebasa una conectividad de 8 vecinos. Pensamos que este comportamiento es bueno, ya que estos valores pueden mantener un equilibrio entre la exploración y la explotación de las partículas. De igual forma, si se quiere limitar la conectividad de los vecindarios, la mejor opción sería construir estos con MIMIC (algoritmo 12), ya que en este caso la conectividad de cada partícula sería 2.

D.4 Información mutua de los vecindarios

El objetivo de este experimento es ver cual es la magnitud de la información mutua de las topologías construidas con Chow-Liu y de la topología todos conectados, así como ver la influencia de esta en la convergencia al óptimo de las partículas. Para este experimento se resolvió la función esfera en dimensión 30 con PSO-MI, con parámetros $c_1 = c_2 = 1.49618$, $S = 30$, $f = 100$, topología inicial todos desconectados y construyendo los vecindarios con Chow-Liu. Para este caso, se probaron dos factores de inercia, $\omega_1 = 0.578766$ y $\omega_2 = 0.7298$ (figuras D.30 y D.31, respectivamente). El algoritmo se detuvo cuando se obtuvo el valor del óptimo con una tolerancia de $\epsilon = 10^{-10}$. En las figuras se muestra la suma de las informaciones mutuas de las aristas de cada árbol para cada dimensión del problema. También se resolvió la función esfera en dimensión 30 con PSO canónico (topología todos conectados), con parámetros $c_1 = c_2 = 1.49618$ y $S = 30$, con factores de inercia $\omega_1 = 0.578766$, $\omega_2 = 0.7298$ y $\omega_3 = 0.8$ (figuras D.32, D.33 y D.34; respectivamente). Debido a que en la topología todos conectados no existe un cambio dinámico, para mostrar resultados similares y poder comparar con los vecindarios construidos con Chow-Liu, cada 100 generaciones se obtuvo la información mutua de esta topología utilizando los 100 datos previos por cada dimensión.

A manera de análisis, como se puede observar en las figuras de esta sección y con base en los resultados mostrados en las tablas de la sección

6.3, al construir las topologías con Chow-Liu utilizando un factor de inercia $\omega = 0.578766$ se obtienen valores de la información mutua de entre 10 y 15 (generalmente). Por el contrario, cuando se utiliza un factor de inercia $\omega = 0.7298$, los valores de la información mutua son menores a 10. De hecho, cuando se utiliza un factor de inercia $\omega = 0.7298$ se obtiene una convergencia más rápida que cuando se utiliza un factor de inercia $\omega = 0.578766$ (ver tabla 6.3). En cuanto a la topología todos conectados, se puede observar que cuando se utiliza un factor de inercia $\omega = 0.7298$ se obtienen valores de la información mutua más altos que cuando se utiliza un factor de inercia $\omega = 0.8$, convergiendo más rápido el primero. De hecho, en el caso de $\omega = 0.7298$, se pueden observar algunos “picos” en la información mutua, que suponemos ayudan a una convergencia más rápida que cuando se construyen los vecindarios con Chow-Liu con un factor de inercia $\omega = 0.578766$. Asumimos que este comportamiento se debe a que como las partículas mantienen siempre la misma estructura, estas “aprenden” de sus vecinos con el paso del algoritmo, por lo que comienzan a tener movimientos parecidos, obteniendo altos valores de la información mutua. Un caso “raro” ocurre cuando se utiliza un factor de inercia $\omega = 0.578766$ con la topología todos conectados, donde se pueden observar valores de la información mutua altos y oscilantes, para después mantener valores muy altos o muy bajos. Esto puede deberse a un estancamiento de las partículas. De hecho, en este caso no se obtuvo el valor del óptimo. De lo descrito anteriormente, pareciera que existe una relación entre la información mutua de los vecindarios y la convergencia del algoritmo, aunque habría que realizar más pruebas para corroborar esto.

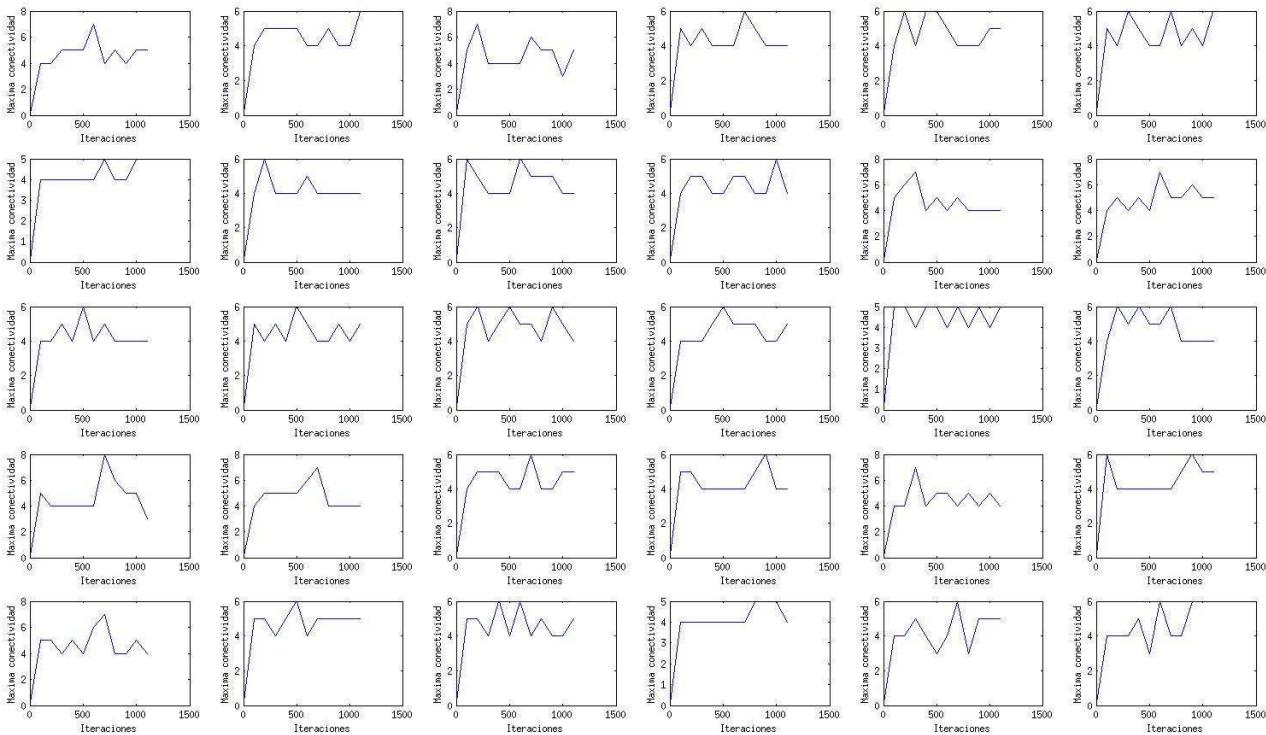


Figura D.29: Máxima conectividad de los vecindarios construidos con Chow-Liu

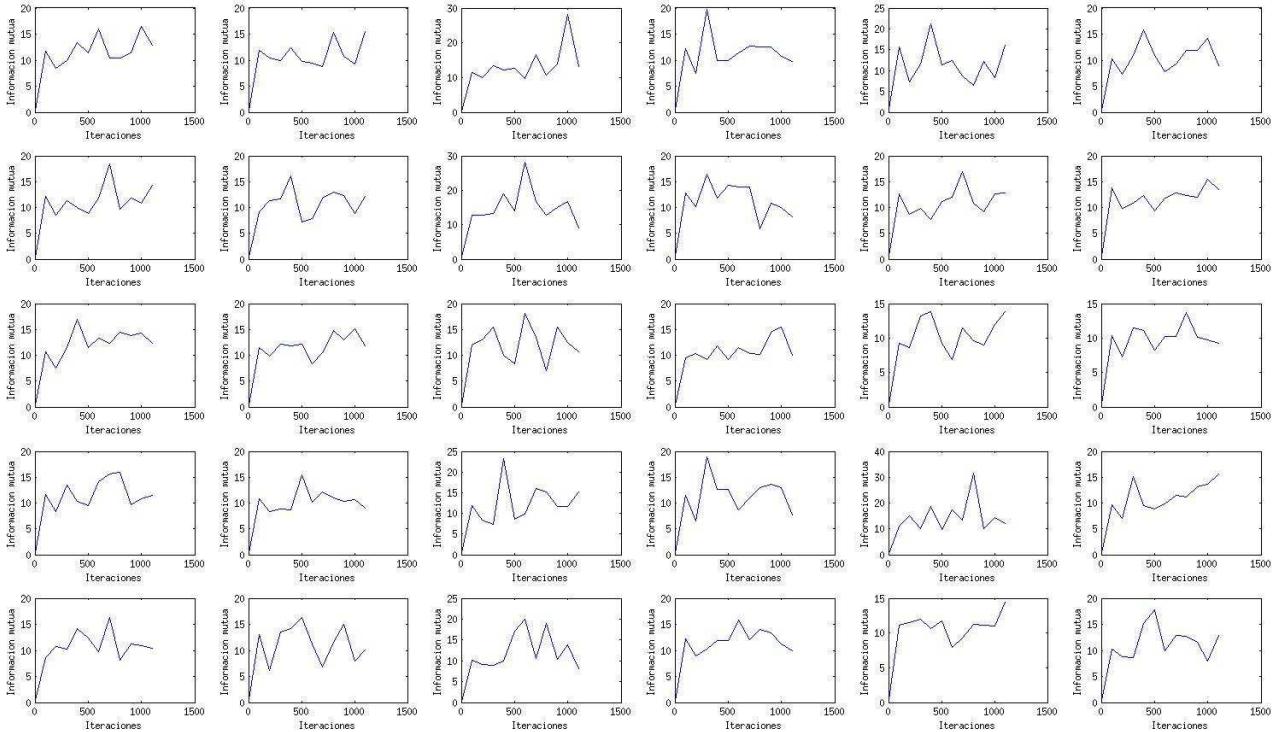


Figura D.30: Información mutua de los vecindarios construidos con Chow-Liu. Factor de inercia $\omega = 0.578766$

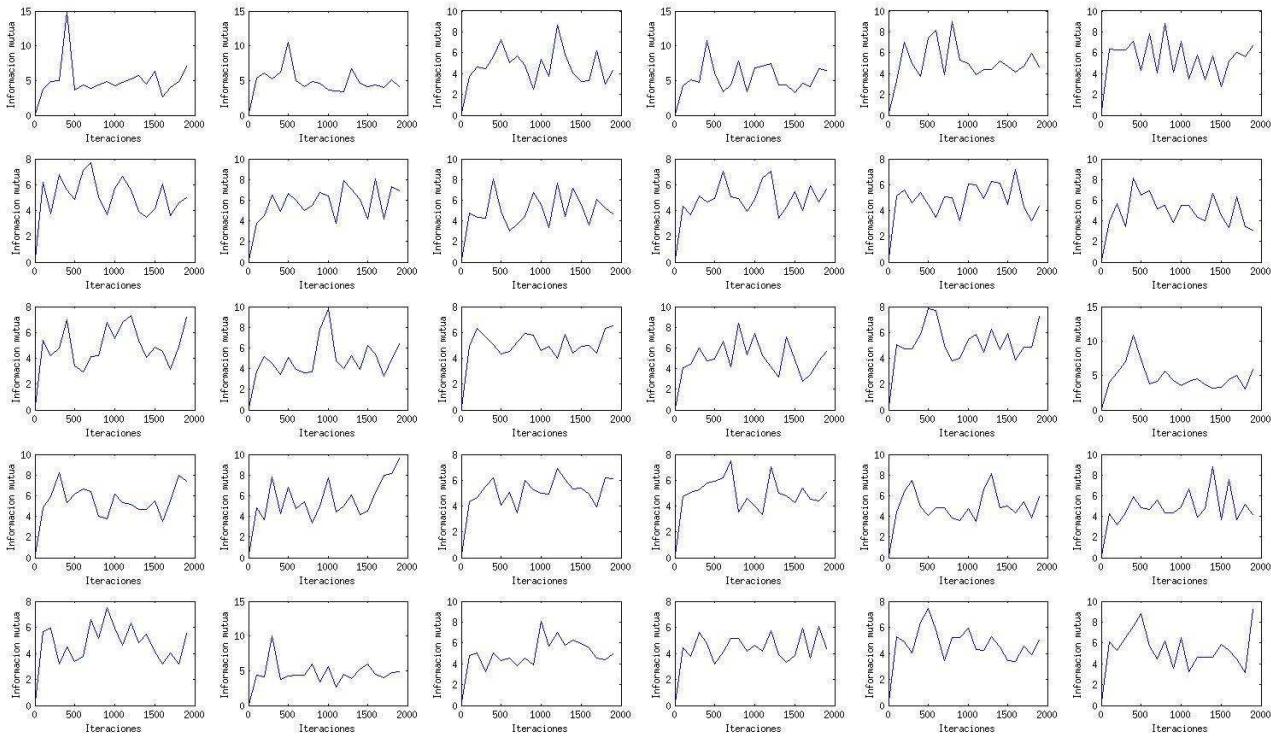


Figura D.31: Información mutua de los vecindarios construidos con Chow-Liu. Factor de inercia $\omega = 0.7298$

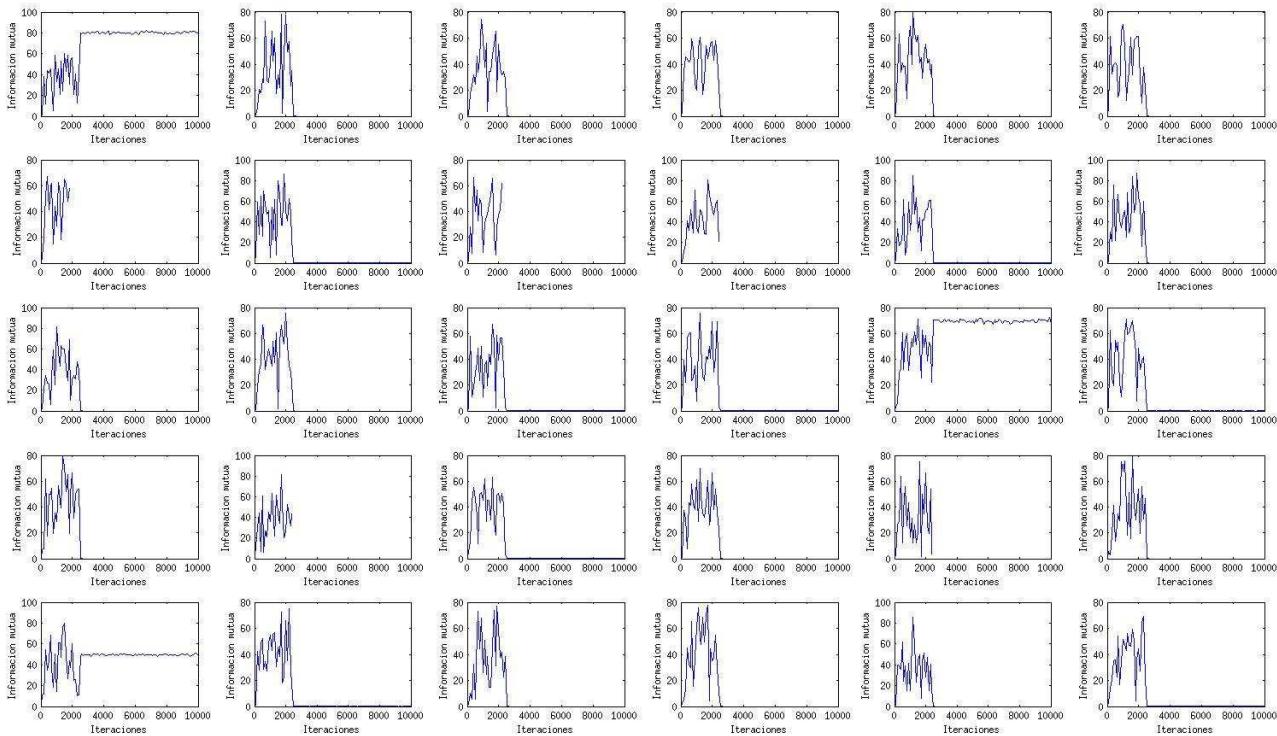


Figura D.32: Información mutua de la topología todos conectados. Factor de inercia $\omega = 0.578766$

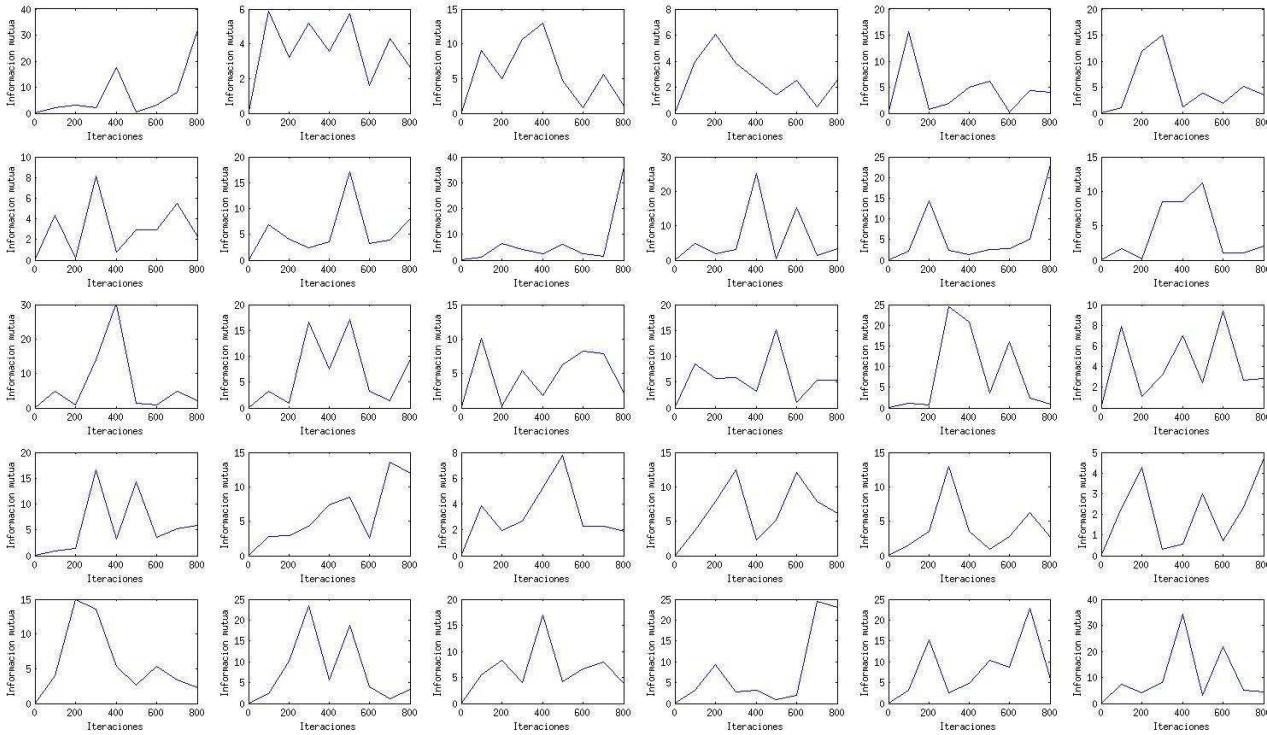


Figura D.33: Información mutua de la topología todos conectados. Factor de inercia $\omega = 0.7298$

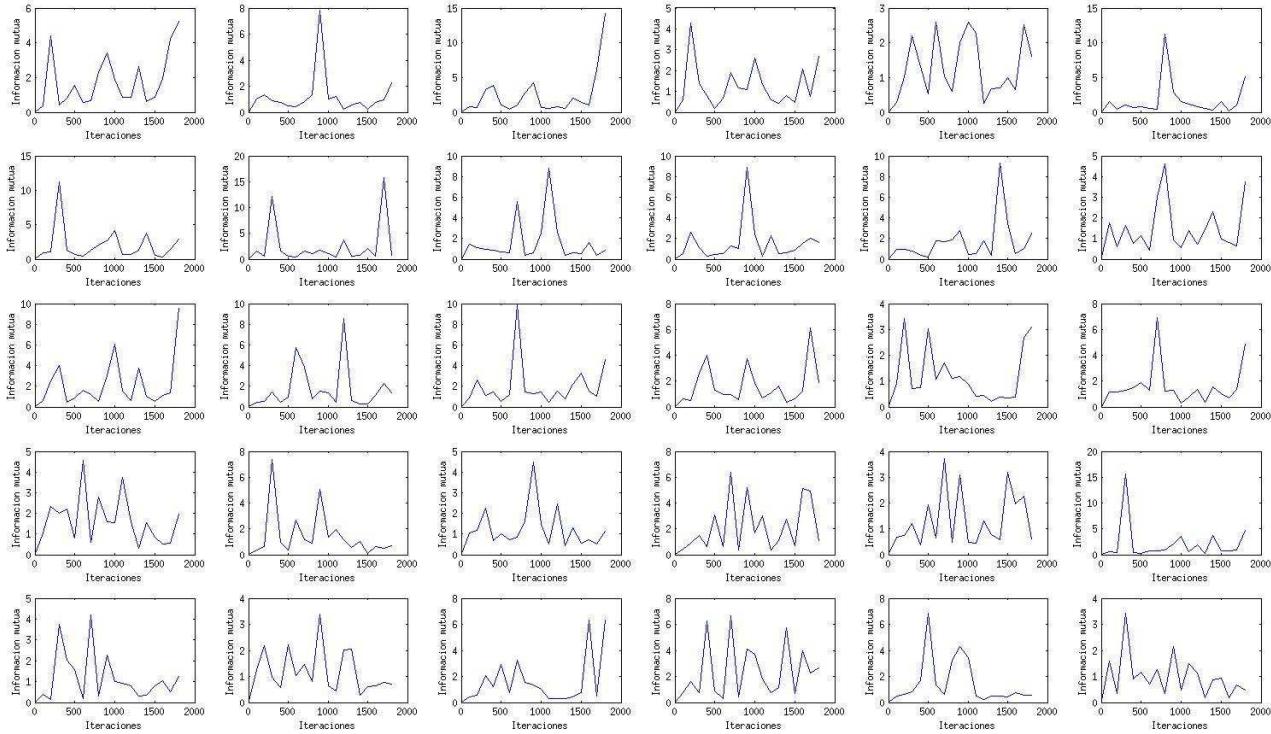
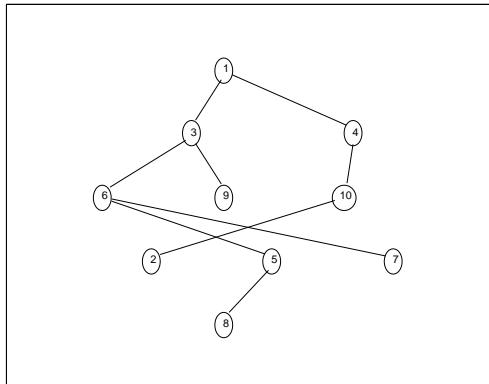


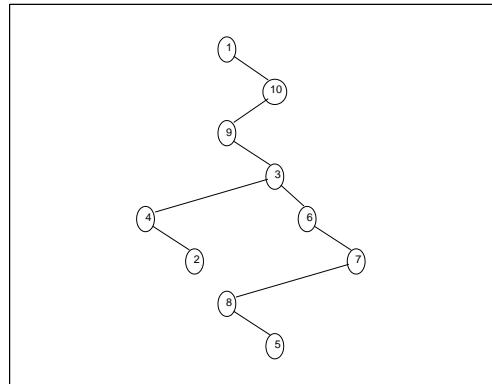
Figura D.34: Información mutua de la topología todos conectados. Factor de inercia $\omega = 0.8$

D.5 Evolución de las distribuciones

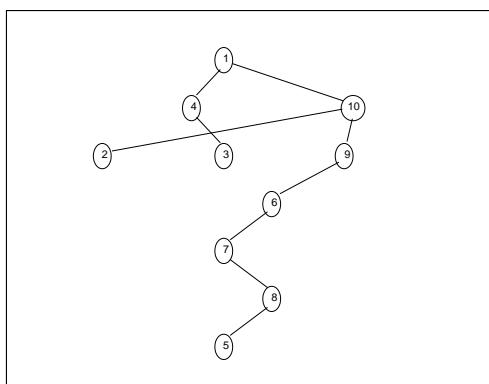
En experimentos anteriores nos hemos centrado en observar la influencia de las partículas en la construcción de los árboles, esto con base en el movimiento de estas. En este experimento, más que ver el movimiento de las partículas quisimos observar la distribución que tenían las partículas antes de la construcción de cada árbol. Para este experimento, se resolvió la función Esfera en dimensión 30 con PSO-MI, con parámetros $\omega = 0.578766$, $c_1 = c_2 = 1.49618$, $S = 10$, $f = 100$, topología inicial todos desconectados y construyendo los vecindarios con Chow-Liu. El algoritmo se corrió durante 1200 generaciones. Sólo se muestran las árboles y las distribuciones para la dimensión 1 de la función. En la figura D.37 se muestra la distribución de las partículas en las primeras 100 iteraciones. Estas distribuciones son las que se utilizaron para la construcción del primer árbol, mostrado en la figura D.35a. Este árbol influye en la distribución de las partículas de la generación 101 a la 200. Estas distribuciones se muestran en la figura D.38, que a su vez son las que se utilizan para construir el segundo árbol, el cual se muestra en la figura D.35b. Este proceso se repite hasta las distribuciones de las partículas de la iteración 1101 a la 1200 (figura D.48), con las cuales se construye el duodécimo árbol, que se muestra en la figura D.36f. Además, en las figuras en donde se muestran las distribuciones se agregan la media y la desviación estándar de las posiciones de las partículas, ya que hay que recordar que estamos asumiendo que las partículas tienen una distribución normal.



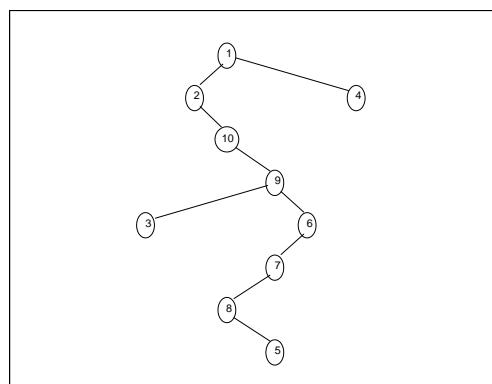
(a) Primer árbol



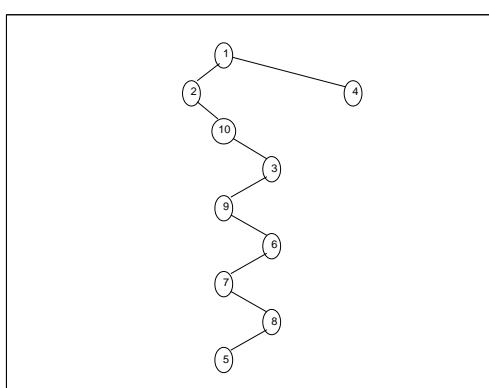
(b) Segundo árbol



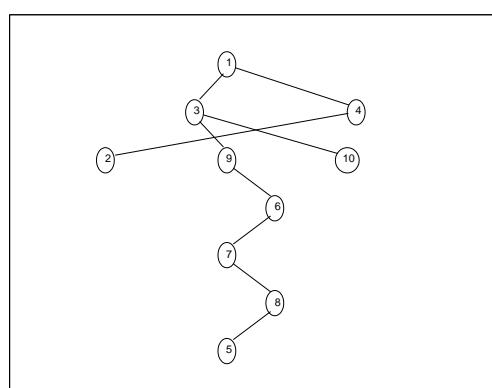
(c) Tercer árbol



(d) Cuarto árbol

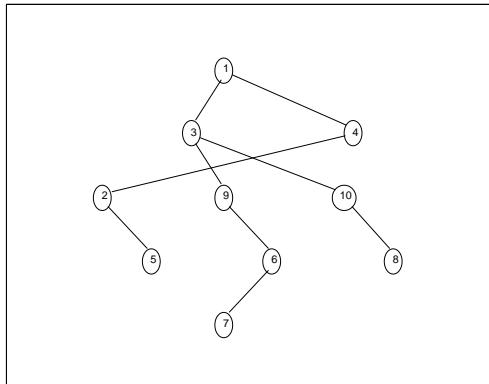


(e) Quinto árbol

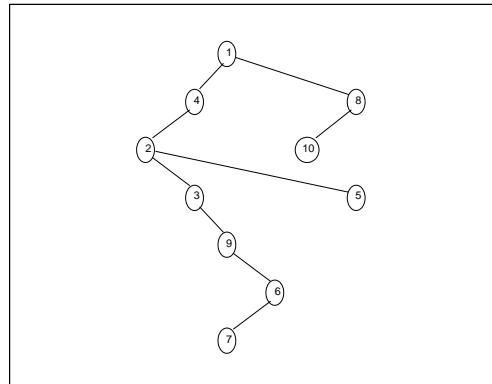


(f) Sexto árbol

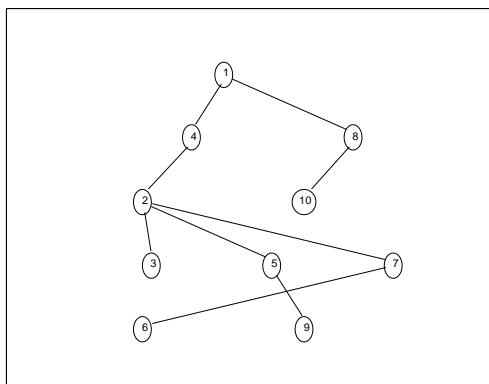
Figura D.35: Evolución de los árboles al resolver la función esfera en dimensión 30 con 10 partículas (parte 1)



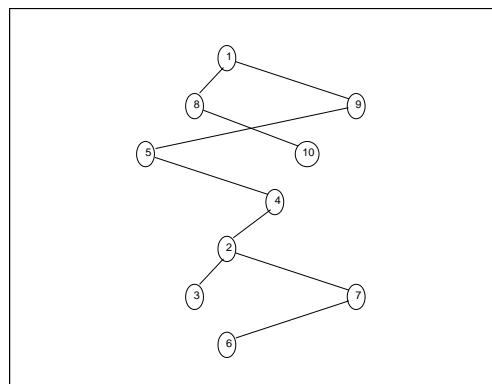
(a) Séptimo árbol



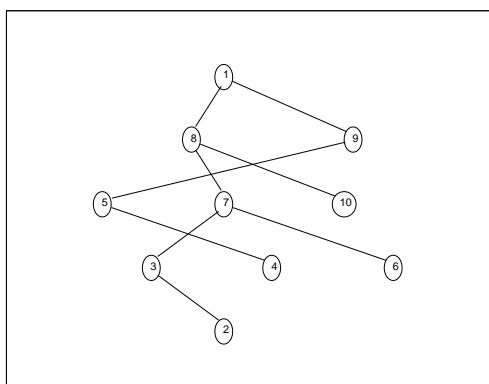
(b) Octavo árbol



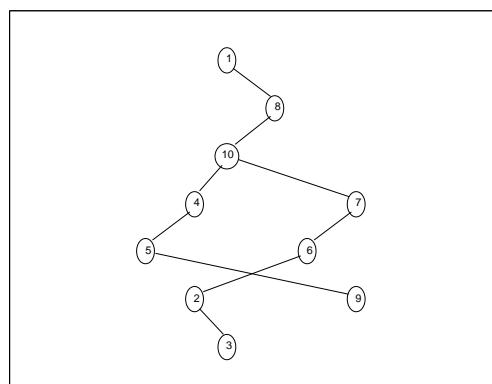
(c) Noveno árbol



(d) Décimo árbol



(e) Undécimo árbol



(f) Duodécimo árbol

Figura D.36: Evolución de los árboles al resolver la función esfera en dimensión 30 con 10 partículas (parte 2)

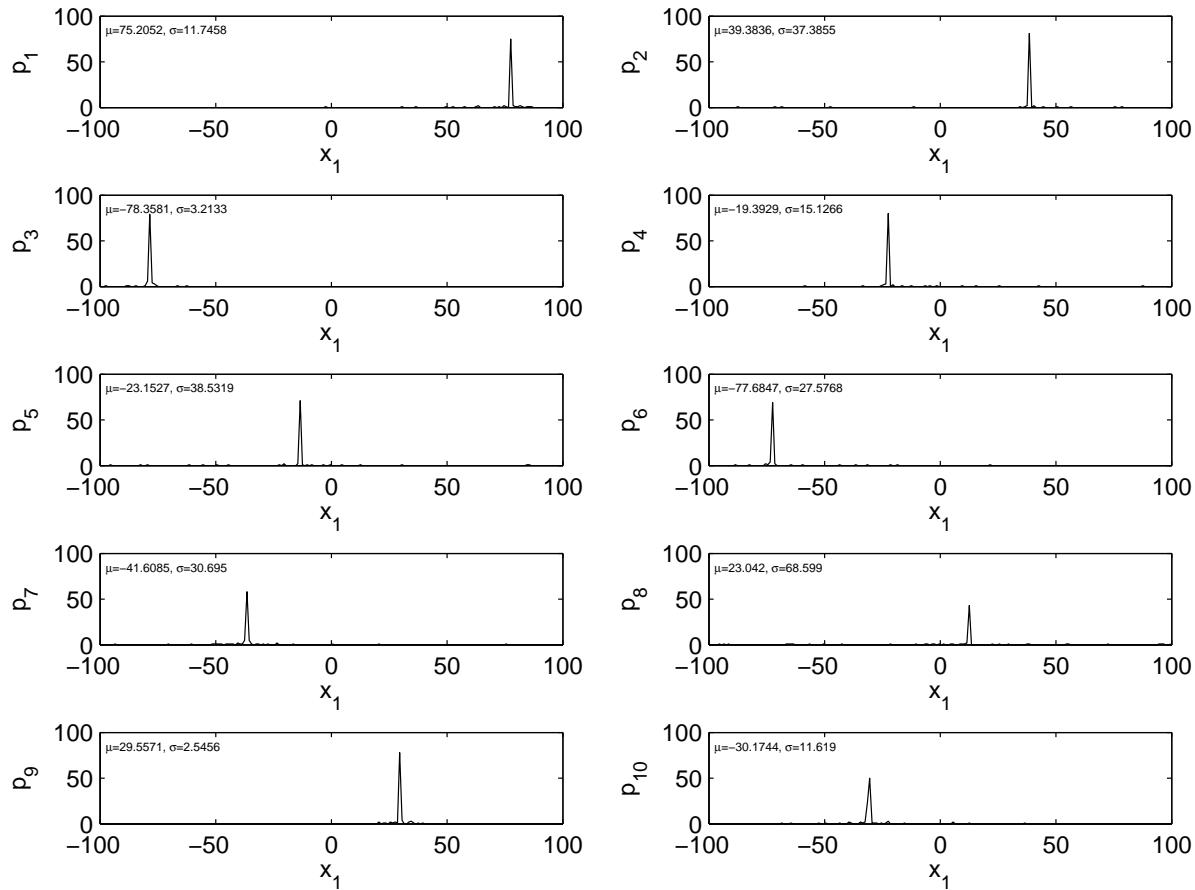


Figura D.37: Distribución de las partículas en el espacio de búsqueda de la generación 1 a la 100

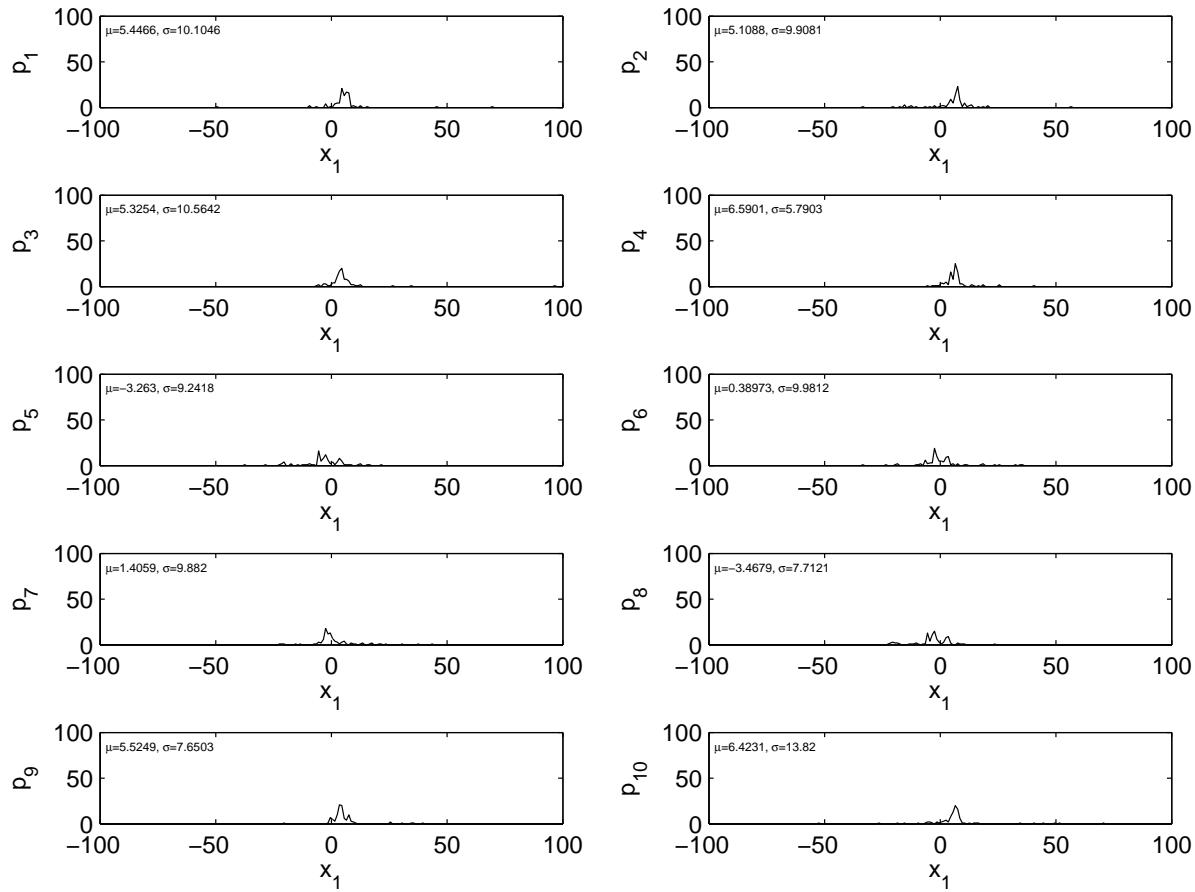


Figura D.38: Distribución de las partículas en el espacio de búsqueda de la generación 101 a la 200

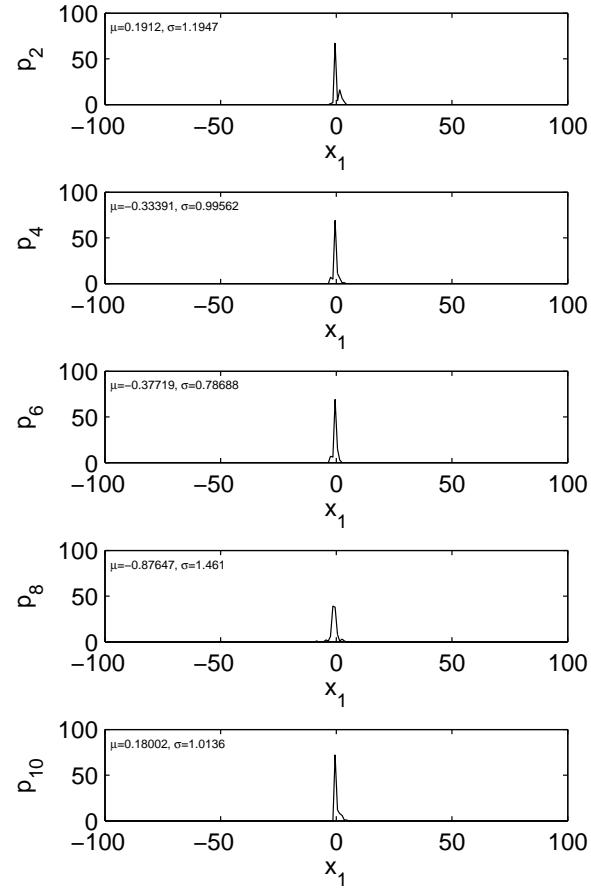


Figura D.39: Distribución de las partículas en el espacio de búsqueda de la generación 201 a la 300

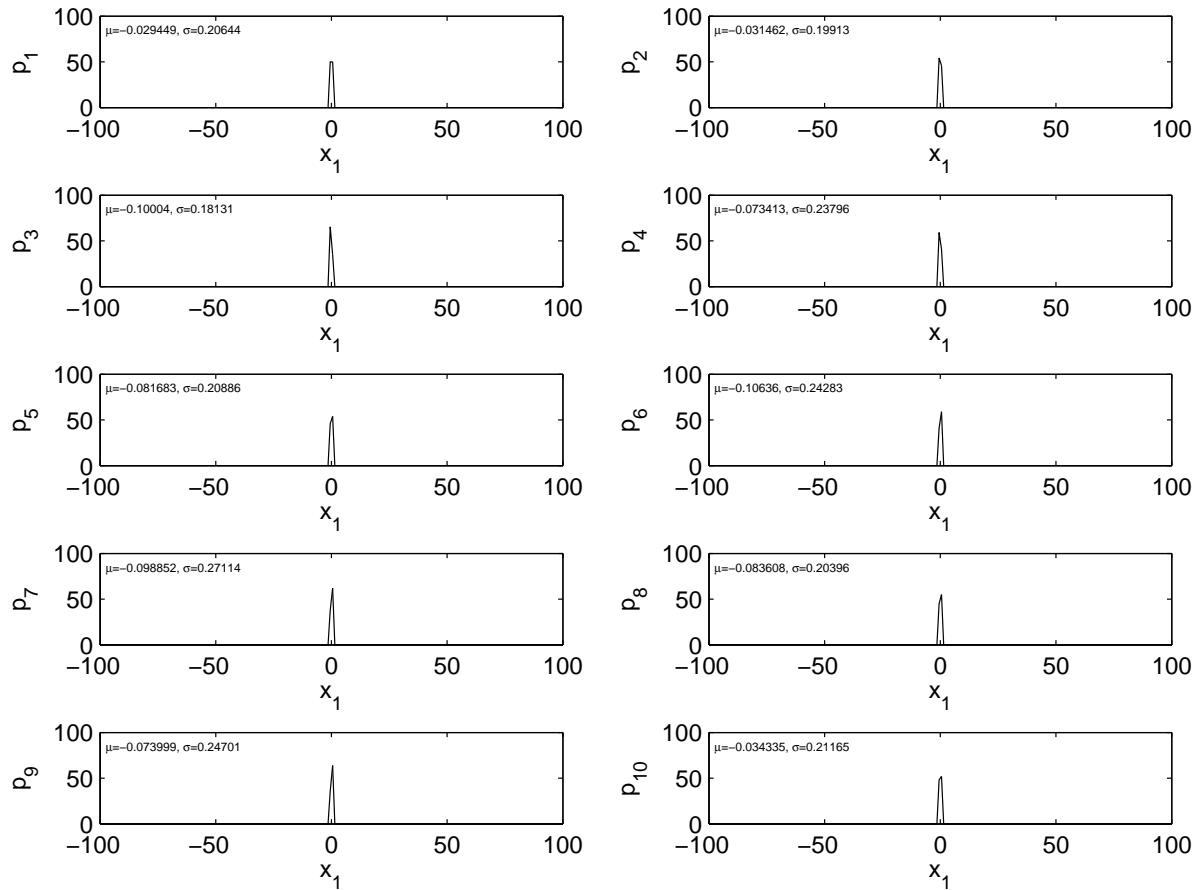


Figura D.40: Distribución de las partículas en el espacio de búsqueda de la generación 301 a la 400

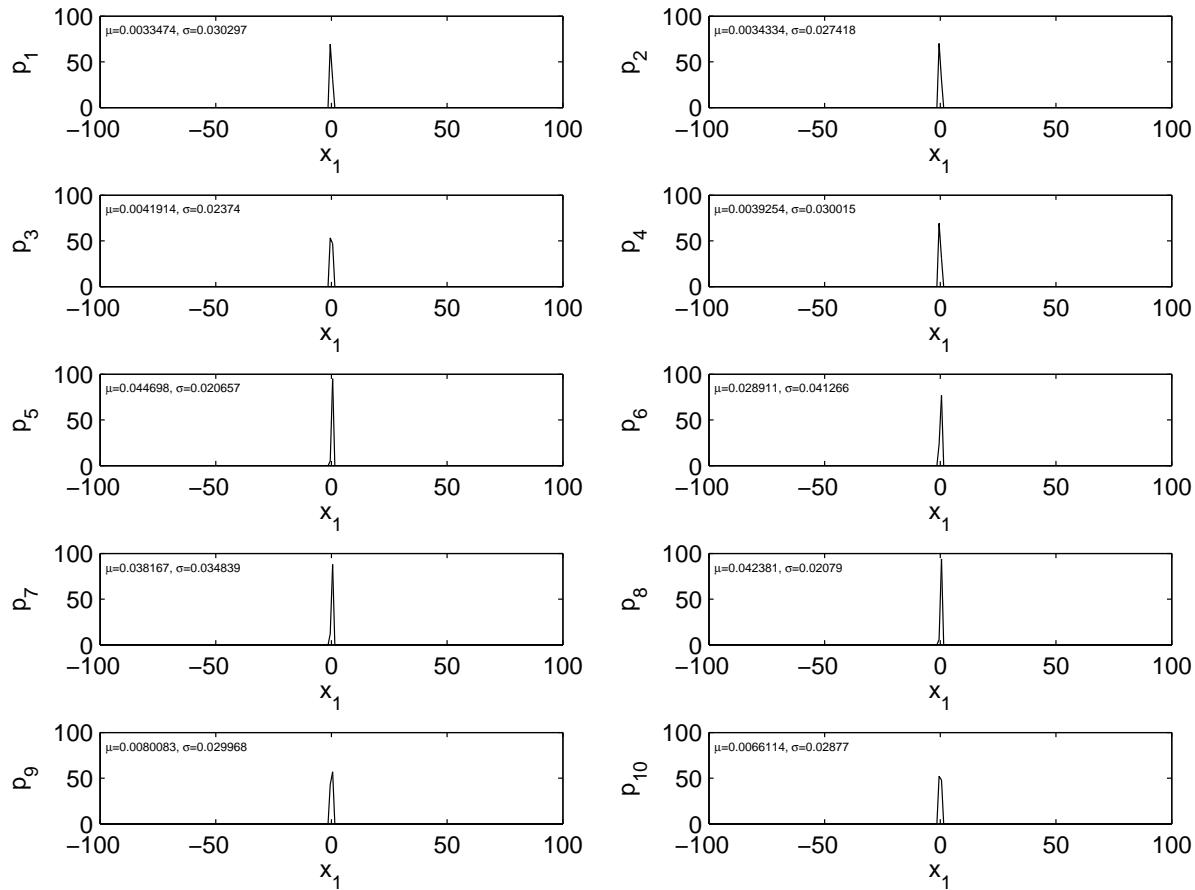


Figura D.41: Distribución de las partículas en el espacio de búsqueda de la generación 401 a la 500

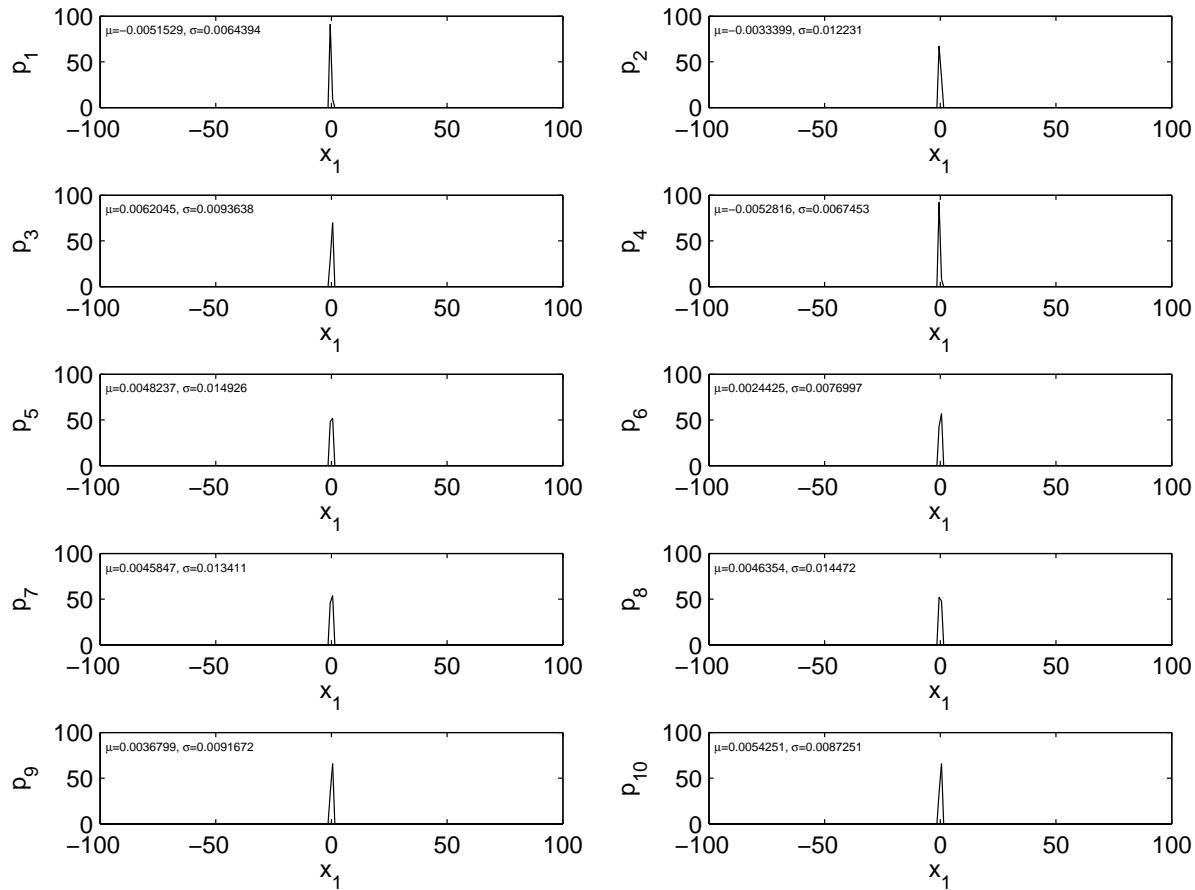


Figura D.42: Distribución de las partículas en el espacio de búsqueda de la generación 501 a la 600

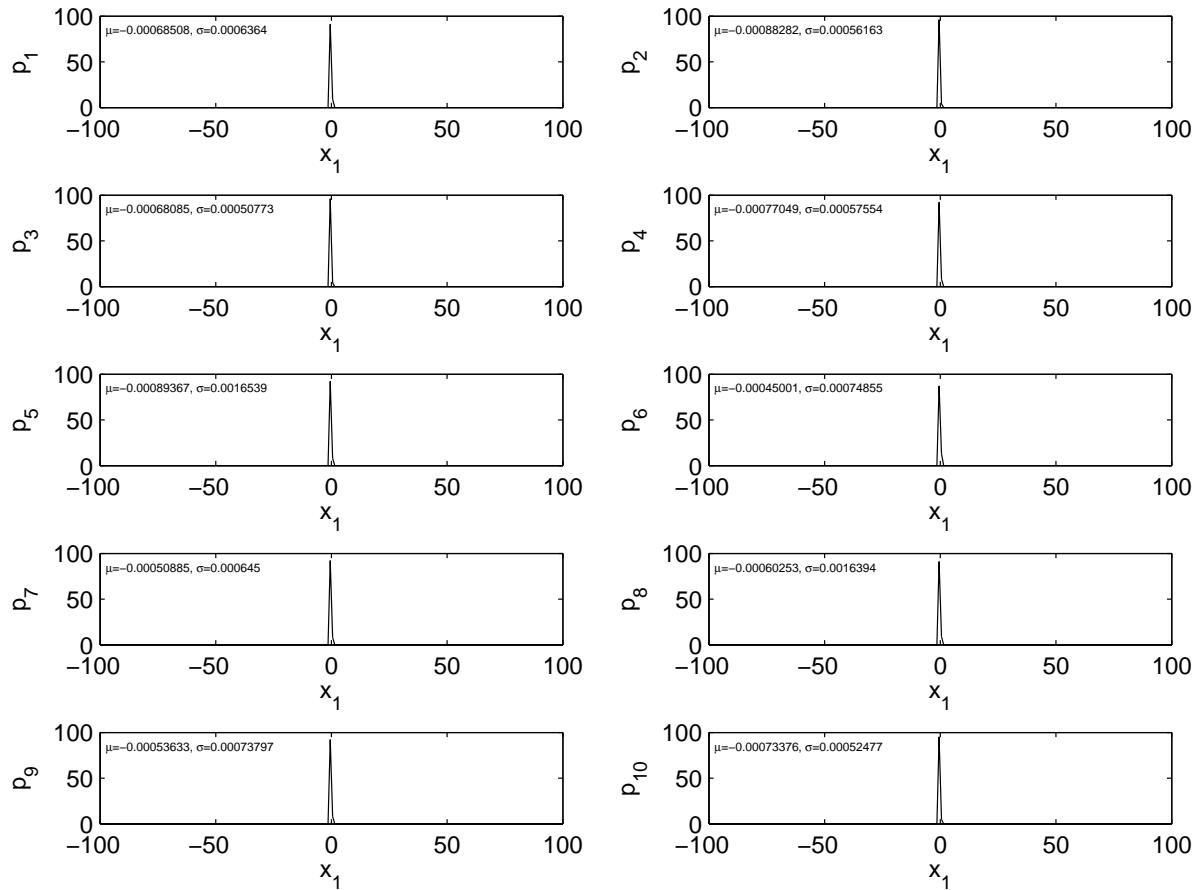


Figura D.43: Distribución de las partículas en el espacio de búsqueda de la generación 601 a la 700

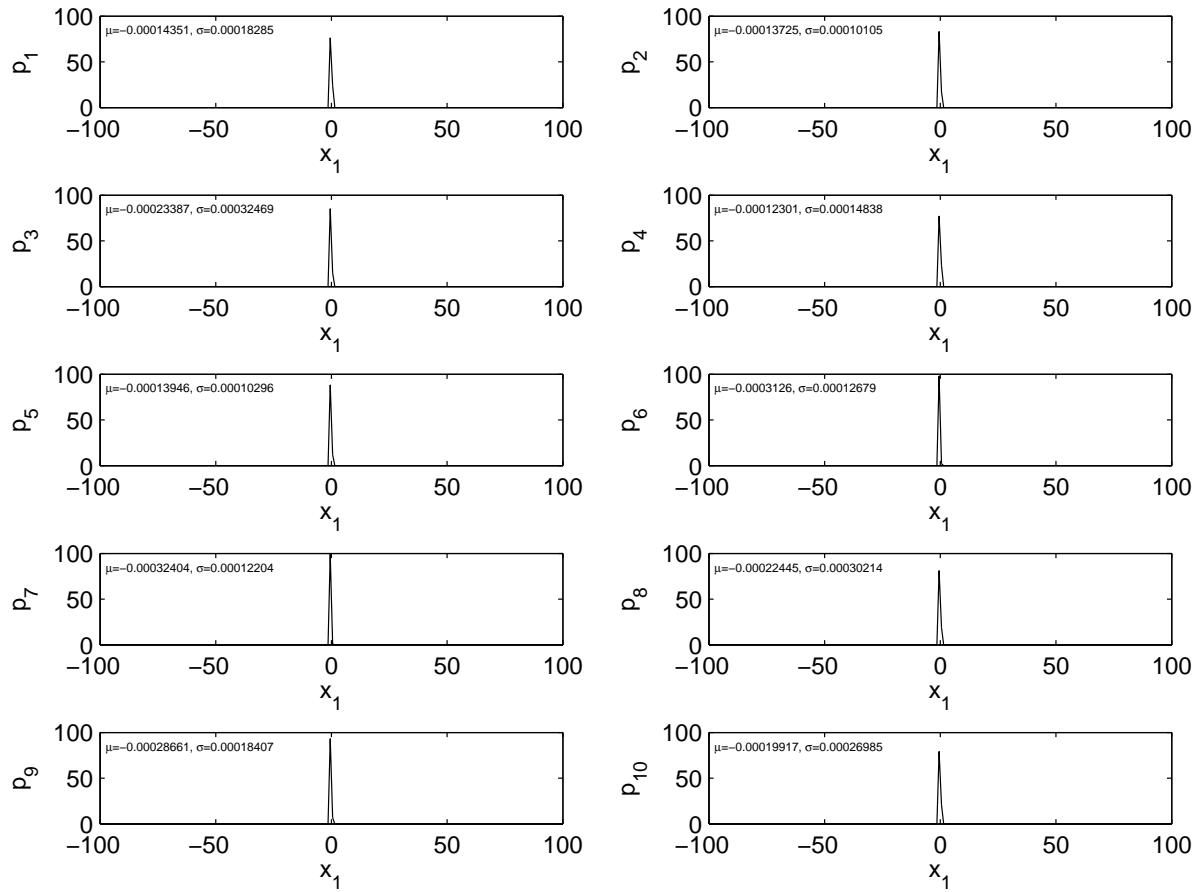


Figura D.44: Distribución de las partículas en el espacio de búsqueda de la generación 701 a la 800

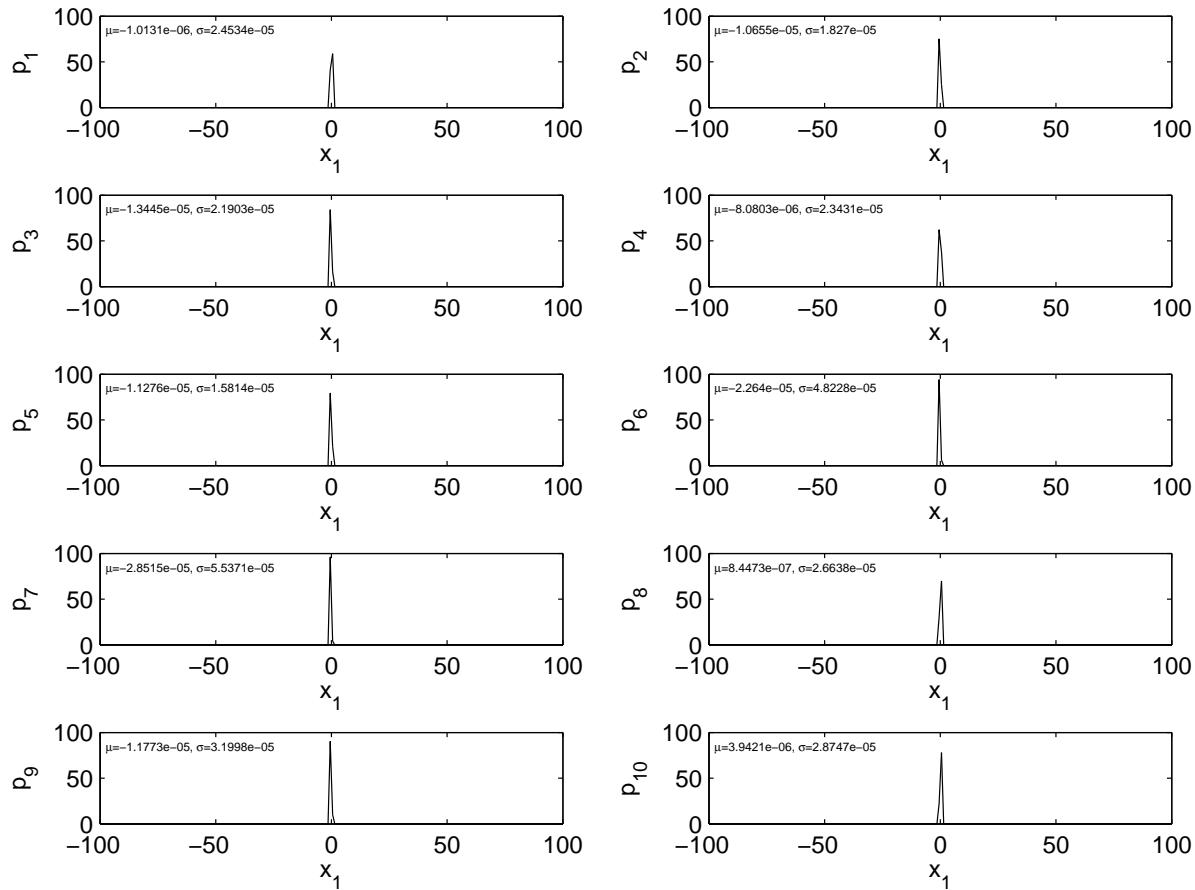


Figura D.45: Distribución de las partículas en el espacio de búsqueda de la generación 801 a la 900

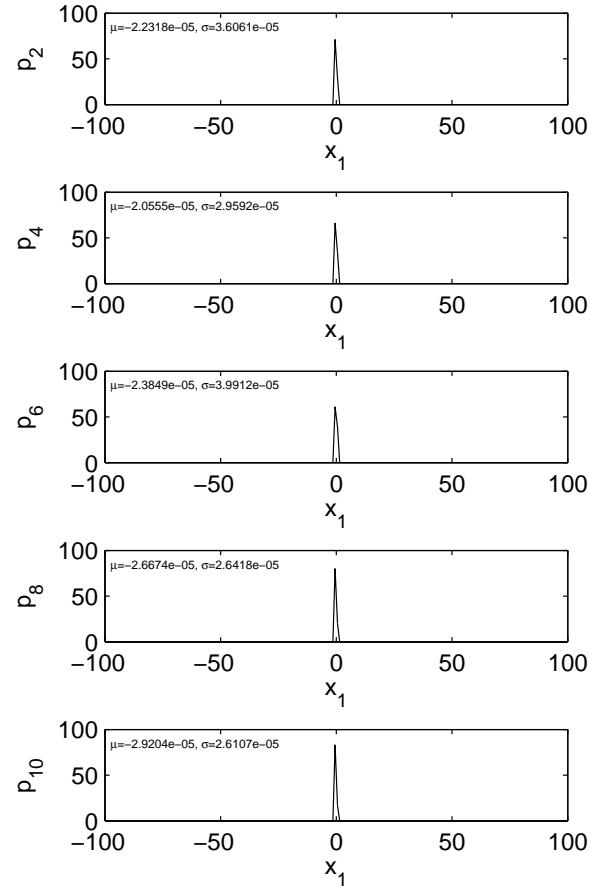


Figura D.46: Distribución de las partículas en el espacio de búsqueda de la generación 901 a la 1000

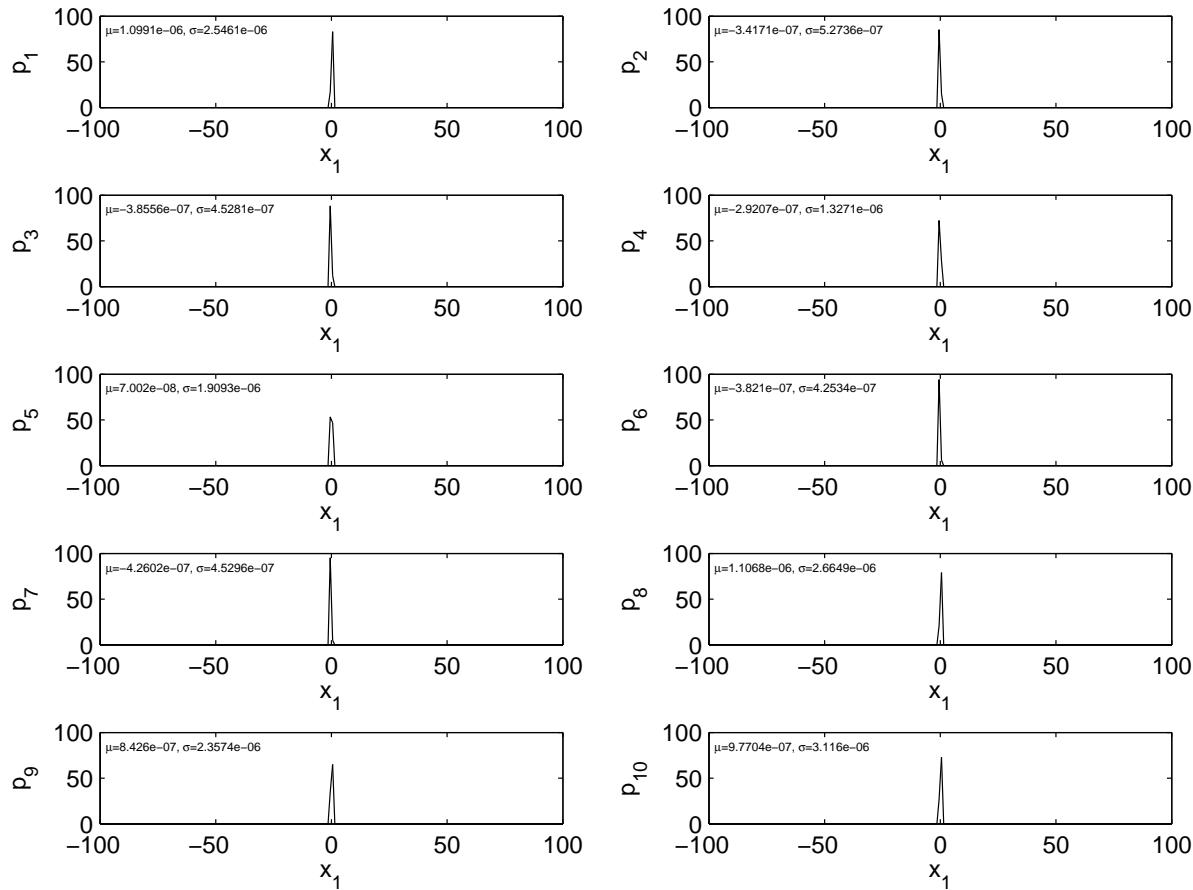


Figura D.47: Distribución de las partículas en el espacio de búsqueda de la generación 1001 a la 1100

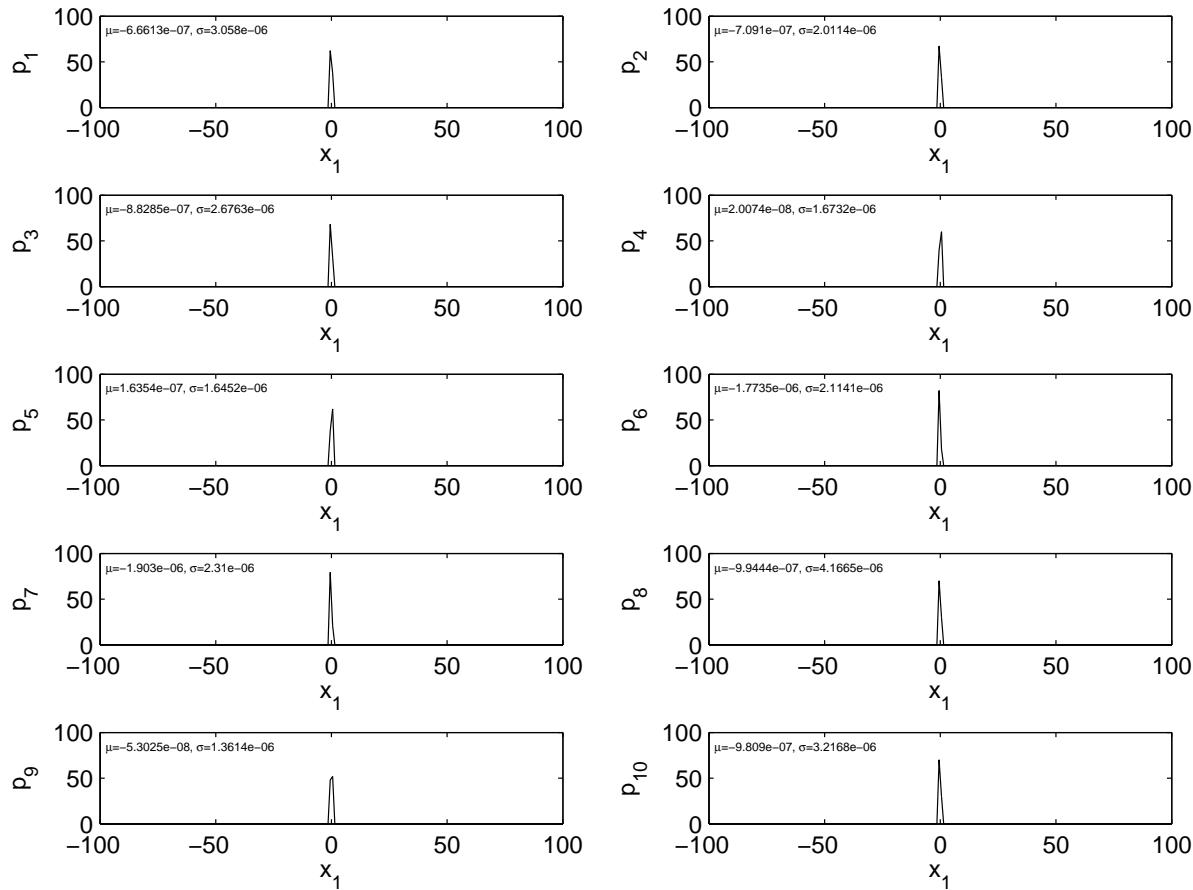


Figura D.48: Distribución de las partículas en el espacio de búsqueda de la generación 1101 a la 1200

D.6 Información mutua utilizando MIMIC

En este experimento se buscó observar la influencia de la información mutua en la convergencia de las partículas, pero ahora utilizando el modelo de cadena MIMIC y comparándolo con una topología de anillo y con anillos creados aleatoriamente, como se describe en la sección 6.5. Para este experimento se resolvió la función Esfera en dimensión 100 utilizando PSO con topología de anillo, PSO-MI construyendo las topologías con MIMIC y anillos aleatorios. Los parámetros utilizados fueron $\omega = 0.578766$, $c_1 = c_2 = 1.49618$, $S = 30$, para todos los casos. En el caso de PSO-MI, se utilizó topología inicial todos desconectados y frecuencias de actualización $f = 50$ (figura D.50) y $f = 100$ (figura D.49). En las figuras se muestra el promedio de las sumas de las informaciones mutuas de las topologías de cada dimensión (esto es, la suma de todas las informaciones mutuas de las aristas de las topologías de las 100 dimensiones, entre las 100 dimensiones), cada vez que se actualiza la topología (50 ó 100 generaciones, según sea el caso). En el caso del anillo, se calcula la información mutua para la misma estructura en las 100 dimensiones, pero con los datos correspondientes de cada dimensión. Como se puede observar en la figura D.49, la información mutua de las topologías construidas con MIMIC es menor que las topologías de anillo, y a su vez la convergencia de las partículas es más rápida. Por otro lado, cuando las topologías de PSO-MI se actualizan cada 50 generaciones, la información mutua entre las topologías de anillo y la cadena MIMIC son muy parecidas, observándose una convergencia al óptimo muy similar, aunque un poco más rápida en PSO-MI, comportamiento que se puede deber a la construcción dinámica de los vecindarios. En el caso de PSO con anillos aleatorios, se puede observar que en un principio la información mutua entre las partículas es 0, debido a la asignación aleatoria de los vecindarios, y que después de algunas iteraciones la información mutua tiende a infinito (aunque en la gráfica se muestra en un valor de 20, sólo por objetivos de visualización), lo cual se debe al estancamiento de las partículas.

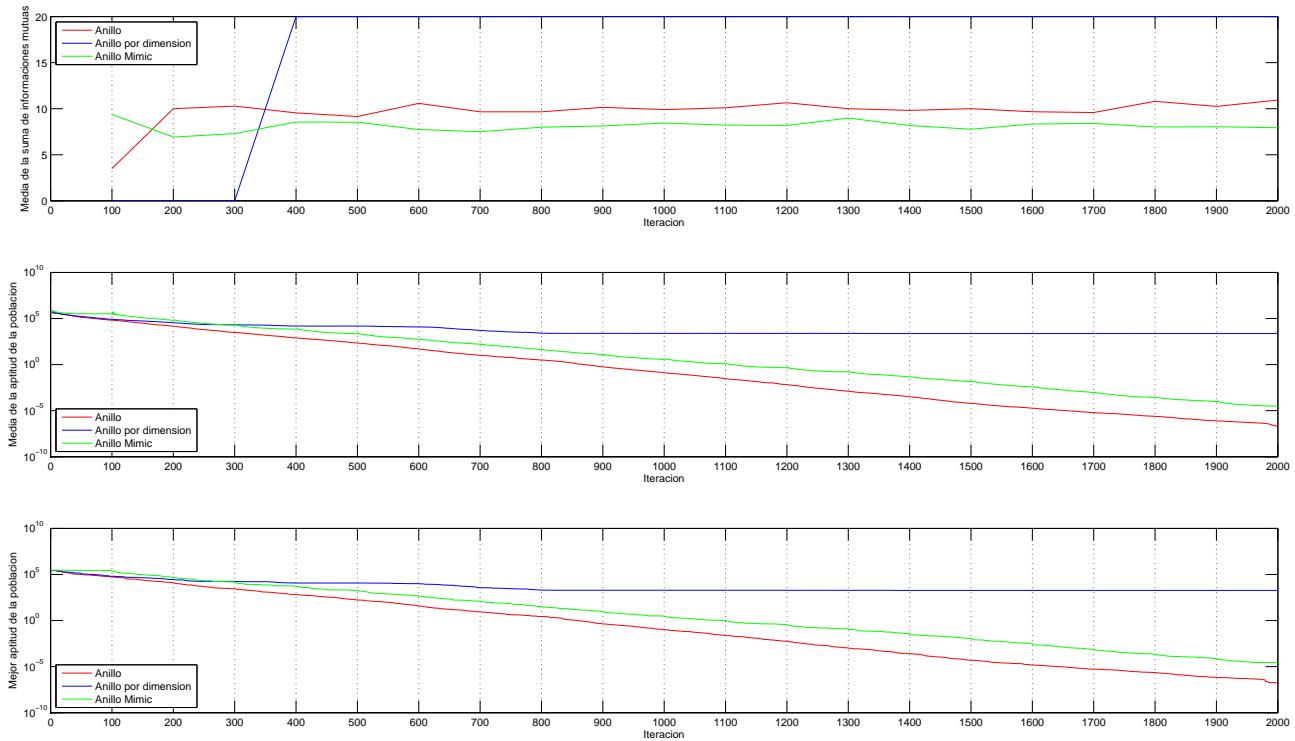


Figura D.49: Información mutua en la topología anillo y cadena MIMIC utilizando una frecuencia de actualización de 100 generaciones

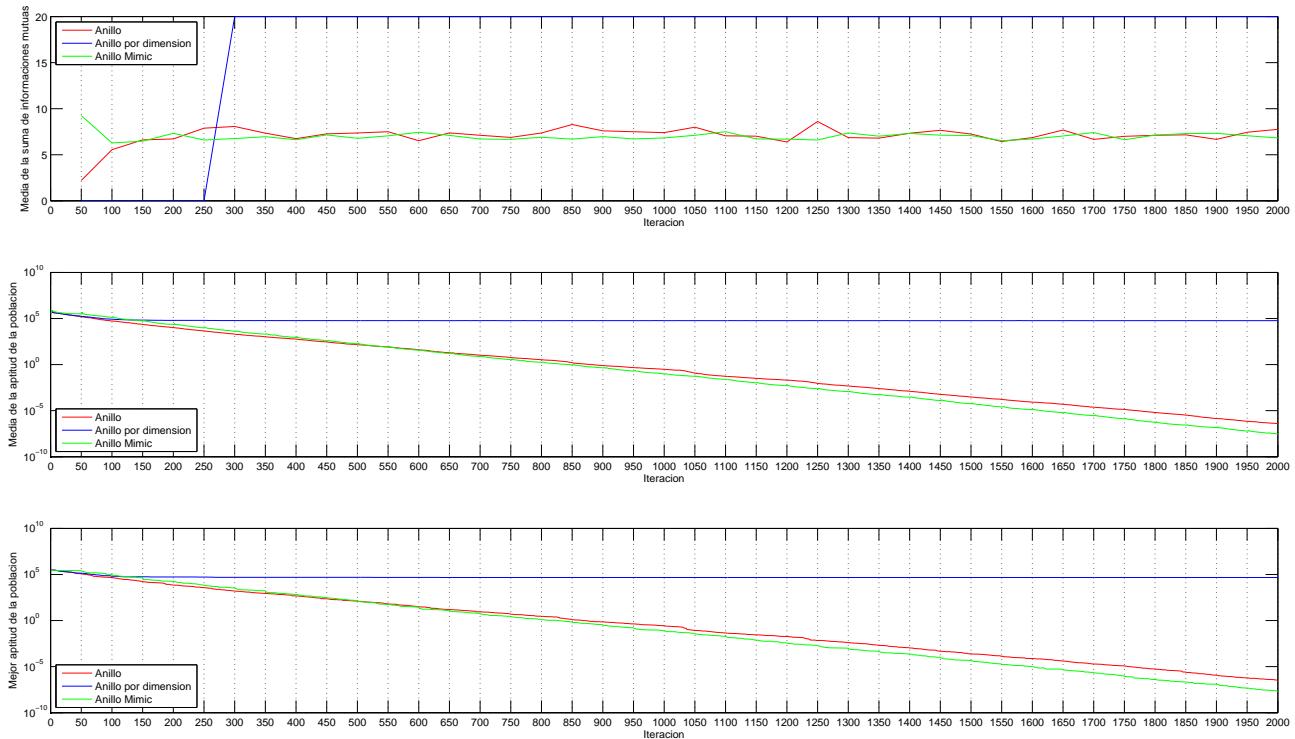


Figura D.50: Información mutua en la topología anillo y cadena MIMIC utilizando una frecuencia de actualización de 50 generaciones