



**INSTITUTO POLITÉCNICO NACIONAL**

Centro de Investigación en Computación



**Modelos alternativos de heurísticas bioinspiradas  
espacialmente estructuradas**

**T E S I S**

Que para obtener el grado de

**Doctor en Ciencias de la Computación**

Presenta

**M. en C. Javier Arellano Verdejo**

Directores de tesis

**Dr. Adolfo Guzmán Arenas  
Dr. Salvador Godoy Calderón**

México D.F.

Agosto, 2014



# INSTITUTO POLITÉCNICO NACIONAL SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

## ACTA DE REVISIÓN DE TESIS

En la Ciudad de     México, D.F.     siendo las     16:00     horas del día     4     del mes de     Junio     de     2014     se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:  
**Centro de Investigación en Computación**  
para examinar la tesis titulada:

**“Modelos alternativos de heurísticas bioinspiradas espacialmente estructuradas”**

Presentada por el alumno:

<b>Arellano</b>	<b>Verdejo</b>	<b>Javier</b>						
Apellido paterno	Apellido materno	Nombre(s)						
		Con registro:						
		B	1	0	2	2	4	8

aspirante de: **DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.


### LA COMISIÓN REVISORA

Directores de tesis

  
\_\_\_\_\_  
Dr. Salvador Godoy Calderón

  
\_\_\_\_\_  
Dr. Adolfo Guzmán Arenas


  
\_\_\_\_\_  
Dr. Marco Antonio Moreno Armendáriz

  
\_\_\_\_\_  
Dr. René Luna García

  
\_\_\_\_\_  
Dr. Ángel Fernando Kuri Morales

  
\_\_\_\_\_  
Dr. Ricardo Barrón Fernández

PRESIDENTE DEL COLEGIO DE PROFESORES

  
\_\_\_\_\_  
Dr. Luis Alfonso Villa Vargas



INSTITUTO POLITÉCNICO NACIONAL  
CENTRO DE INVESTIGACIÓN  
EN COMPUTACIÓN  
DIRECCIÓN



**INSTITUTO POLITÉCNICO NACIONAL**  
**SECRETARÍA DE INVESTIGACIÓN Y POSGRADO**

**CARTA DE CESIÓN DE DERECHOS**

En la Ciudad de México, D.F. el día **10** del mes de **Junio** del año **2014**, el que suscribe **Javier Arellano Verdejo** alumno del Programa de **Doctorado en Ciencias de la Computación**, con número de registro **B102248**, adscrito al **Laboratorio de Inteligencia Artificial**, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del **Dr. Adolfo Guzmán Arenas y Dr. Salvador Godoy Calderón**, y cede los derechos del trabajo titulado **Modelos alternativos de heurísticas bioinspiradas espacialmente estructuradas**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, graficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección **[javier\\_arellano\\_verdejo@hotmail.com](mailto:javier_arellano_verdejo@hotmail.com)**. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

**Javier Arellano Verdejo**  
Nombre y Firma

# Resumen

En esta tesis, se abordan diferentes variantes de algoritmos bioinspirados espacialmente estructurados los cuales se encuentran adaptados para resolver problemas continuos y para encontrar el número óptimo de grupos en un conjunto de datos; se proponen tres nuevas variantes de EAs, la primera, incorpora un mecanismo de fusión de islas para reducir el número de evaluaciones de la función de aptitud, la segunda propone un algoritmo genético celular híbrido y la tercera un algoritmo de evolución diferencial híbrido.

Para validar las variantes propuestas, se presenta un amplio marco experimental así como un exhaustivo análisis estadístico que muestran la viabilidad de dichas propuestas .



# Abstract

In this thesis, dealt with different optimal variants of spatially structured bioinspired algorithms, which are adapted to solve ongoing problems, and to find the number of groups in a set of data; We propose three new variants of the first EAs, incorporates a mechanism of merging of Islands to reduce the number of fitness function evaluations, the second proposes a new hybrid cellular genetic algorithm and the third a new hybrid differential evolution algorithm.

To validate the proposed variants, presents a broad experimental framework as well as a comprehensive statistical analysis showing the feasibility of these proposals.

*Dedicado a la compañera y mujer de mi vida "Geny"*

*A mi madre y hermanos*

*A la memoria de mi padre*

# Agradecimientos

Agradezco ...

Al Instituto Politécnico Nacional y al Centro de Investigación en Computación por todo el apoyo brindado durante mis estudios.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el gran apoyo económico durante mis estudios de doctorado.

A mis directores de tesis, Dr. Adolfo Guzmán Arenas y Dr. Salvador Godoy Calderón por haber creído en mi y guiarme durante estos años, por compartir sus conocimientos y experiencias. Gracias por la paciencia que tuvieron al enseñarme.

Al Dr. Enrique Alba Torres de la Universidad de Málaga, por haberme adoptado como a uno mas de sus estudiantes durante mi estancia en Málaga España y por haberme enseñado que soy capaz de conseguir cosas muy grandes he importantes en mi vida.

A los integrantes del grupo de investigación “Networking and Emerging Optimization” de la Universidad de Málaga , Sebastian Luna, Francisco Luna, José Manuel Gracia, Javier Matos, Francisco Chicano, Gabriel J. Luque, Jamal Toutouh y Daniel H. Stolfi por su compañía, apoyo y buenos momentos que vivimos durante mi estancia en la Universidad de Málaga.

A los doctores Francisco Hiram Calvo Castro y Marco Antonio Moreno Armendáriz por ayudarme como profesores y amigos en esos momentos difíciles.

Al M. en C. Carlos Guzmán Sánchez (carlitos) por sus sabios consejos y a todos mis compañeros del CIC por esos buenos momentos que vivimos juntos.

A mi madre por su ejemplo de fuerza y amor, gracias mamá y a mis hermanos, por estar siempre ahí mostrándome el camino a seguir y enseñándome la importancia de la familia a todos ustedes muchas gracias.

A mi esposa, una gran mujer, gracias cielo, por tu amor y comprensión, perdón por todas esas horas que no pude estar junto a ti, sin tu apoyo esto no sería hoy una realidad, gracias por mantenerme siempre con los pies en la tierra y por estar a mi lado, la mitad de todo esto es tuyo, gracias por creer en mí y sobre todo gracias por ser quien eres, te amo.

*Finalmente quiero agradecer a un gran amigo que me ha apoyado en todas y cada una de mis ideas, que ha sido capaz de tolerar mis momentos de frustración, coraje y tristeza y que a pesar de todo, siempre se ha mantenido firme y fiel a mi lado, brindándome su amistad y ayuda incondicional en todo momento, compartiendo sus conocimientos y sueños, gracias por todo querido Edgar.*

# Índice

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Objetivo general y objetivos particulares	6
1.2	Resumen de aportaciones	7
1.3	Organización de este texto	8
<b>2</b>	<b>Marco teórico</b>	<b>11</b>
2.1	Principales paradigmas del cómputo evolutivo	12
2.1.1	Algoritmos genéticos	13
2.1.2	Programación genética	14
2.1.3	Estrategias evolutivas	15
2.1.4	Evolución diferencial	16
2.1.5	Micro evolución diferencial	19
2.1.6	Enjambre de partículas	19
2.1.7	Micro PSO con ajuste local	20
2.2	Algoritmos evolutivos paralelos	21
2.2.1	Modelo Maestro-Esclavo	21
2.2.2	Modelo celular	23
2.2.3	Modelo distribuido (islas)	27
2.3	Modelos de búsqueda basados en trayectorias	33
2.3.1	Búsqueda por fuerza bruta	33
2.3.2	Método de Hooke-Jeeves	34
2.3.3	Método de Nelder y Mead	36
2.4	Test estadísticos y pruebas post-hoc	39
2.4.1	Test de Friedman	41
2.4.2	Test de Friedman alineado	42
2.5	Estimación del número de clases dentro de un conjunto de datos	43

2.5.1	Índices de validación de agrupamientos	44
<b>3</b>	<b>Revisión del estado del arte</b>	<b>47</b>
3.1	Algoritmos evolutivos distribuidos	47
3.2	Selección de parámetros en algoritmos distribuidos	51
3.3	Impacto de la migración en los algoritmos evolutivos	54
3.4	Topologías de comunicación en los algoritmos evolutivos	56
3.5	Algoritmos evolutivos celulares	67
<b>4</b>	<b>Modelo de Fusión de poblaciones</b>	<b>60</b>
4.1	Modelo de fusión de poblaciones	61
4.2	Exploración vs Explotación	66
4.3	Experimentos y resultados	72
4.4	Conclusiones parciales	77
4.5	Trabajo futuro	78
<b>5</b>	<b>Modelo Celular Híbrido</b>	<b>79</b>
5.1	cGA Híbrido	79
5.1.1	Representación	81
5.1.2	Función de aptitud	82
5.1.3	Algoritmo cGA/ $\mu$ PSOLA	83
5.2	Experimentación	84
5.2.1	Método de selección de los padres	88
5.2.2	Tipo de vecindario	89
5.2.3	Tamaño del vecindario	91
5.3	Análisis de resultados	93
5.3.1	Medidas estadísticas clásicas	93
5.3.2	Pruebas estadísticas no paramétricas y pruebas <i>post-hoc</i>	95
5.4	Conclusiones parciales	97

5.5	Trabajo futuro	98
<b>6</b>	<b>Modelo diferencial descentralizado</b>	99
6.1	Algoritmo de Evolución Diferencial con Ajuste Local	99
6.1.1	Exploración vs Explotación	105
6.2	Experimentos y resultados	109
6.2.1	Conjuntos de datos	110
6.2.2	Entorno de experimentación	112
6.2.3	Resultados experimentales	113
6.2.4	Análisis estadístico de los resultados experimentales	118
6.3	Conclusiones parciales	120
6.4	Trabajo futuro	121
<b>7</b>	<b>Conclusiones</b>	122
7.1	Algoritmo de fusión de islas	124
7.2	Modelo Celular Hibrido	127
7.3	Modelo Diferencial Descentralizado	134
7.4	Últimas palabras	138
	<b>Referencias</b>	140

# Introducción

# 1

El término "computación evolutiva", ha sido empleado para referirse a la escuela (de pensamiento) que toma la Teoría de la Evolución como inspiración y marco de referencia para diseñar algoritmos heurísticos de búsqueda y optimización. El campo del Cómputo Evolutivo es una aplicación del modelo biológico y la teoría de la evolución de Darwin a la ciencia computacional, y es usada para encontrar soluciones a todo tipo de problemas de búsqueda y optimización.

La característica que distingue a un algoritmo evolutivo de búsqueda de uno tradicional, es que el algoritmo evolutivo opera con poblaciones de posibles soluciones y, a través de la adaptación de sucesivas generaciones (evolución artificial), ejecuta una búsqueda dirigida en forma eficiente.

La técnica más popular en el área de computación evolutiva han sido los Algoritmos Genéticos. En el algoritmo genético clásico (por sus siglas en ingles GA "Genetic Algorithm") propuesto por John Henry Holland [9], cada individuo (posible solución) es representado por una cadena de bits de longitud fija. Se asume que cada posición en la cadena representa una característica específica del individuo y el valor almacenado en dicha posición, representa la forma en como se manifiesta dicha característica en la solución.

No solo los algoritmos genéticos han surgido como una herramienta de búsqueda inspirada en modelos naturales y basada en poblaciones. Las estrategias evolutivas,



programación genética, evolución diferencial, y optimización por cúmulos de partículas (PSO) son sólo otras aproximaciones, que también involucran aspectos comunes del proceso evolutivo como la reproducción, mutación, selección y competencia, aspectos todos ellos esenciales para la evolución [1].

Sin embargo, durante las últimas décadas, la complejidad de los problemas que se intentan resolver mediante el empleo de las herramientas del cómputo evolutivo ha aumentado [7], haciendo que el tiempo de ejecución de éstos incremente de forma exponencial debido a que clásicamente se ha realizado su ejecución sobre un solo núcleo (de forma secuencial).

La naturaleza poblacional de los algoritmos evolutivos aunada con el actual desarrollo acelerado de las tecnologías de hardware multi-núcleo (CPU y GPU) ha convertido a los algoritmos evolutivos en fuertes candidatos para su ejecución en paralelo y así disminuir los tiempos de cálculo no obstante, la implementación de los diversos modelos evolutivos paralelos no solo trae como consecuencia una disminución del tiempo como muchos investigadores desean, sino que también se ha demostrado en estudios recientes[6] que la ejecución de ciertos modelos paralelos afecta el comportamiento del algoritmo con respecto a la forma en como se plantea el proceso de exploración y explotación de este.

Existen reportados en la literatura [5], tres modelos paralelos por excelencia, sin embargo, para conseguir tal paralelización de dichos modelos, se requiere realizar una reestructuración de sus poblaciones dando origen a los siguientes modelos:

- Modelo maestro-esclavo
- Modelo distribuido (modelo de islas)
- Modelo celular

Los algoritmos evolutivos programados bajo el esquema de maestro-esclavo, trabajan con una sola población de individuos o población centralizada [5], la cual es gestionada por un nodo maestro. Este modelo, consiste principalmente en distribuir la evaluación de la función de aptitud (y en algunas ocasiones los operadores de cruce y mutación), entre un conjunto de nodos denominados esclavos, mientras que el resto de las operaciones son ejecutadas por el nodo maestro. Este paradigma es bastante fácil de implementar sin embargo, la exploración y explotación del espacio de búsqueda es conceptualmente idéntica a la realizada por un algoritmo genético serial. Este modelo es más eficiente que la versión serial sólo cuando la evaluación de la función objetivo es costosa.

Para el caso de los algoritmos basados en el modelo distribuido o de islas, las características más importantes, son el uso de múltiples poblaciones y la migración (intercambio) de individuos entre ellas. Dado que cada una de las poblaciones evoluciona independientemente, el radio de migración es decir, la distancia máxima (medida en generaciones) que tiene que viajar un individuo para llegar de una población en un extremo a otra en el extremo contrario, será muy importante para obtener resultados satisfactorios. La principal motivación del uso de estos algoritmos, es obtener ganancias en velocidad de ejecución a través del uso de islas débilmente acopladas. En la mayoría de los trabajos con algoritmos genéticos distribuidos [3], se utilizan islas cuya evolución básica es secuencial, esto supone que cada isla ejecuta un número dado de generaciones antes de llegar a la fase de comunicación de individuos con su(s) vecino(s).

En el caso de los algoritmos celulares, la idea esencial del modelo, es proveer a la población de una estructura que puede definirse como un grafo conectado en el que cada vértice es un individuo que se comunica con sus vecinos más cercanos. En concreto, los individuos se encuentran dispuestos conceptualmente en una rejilla, y sólo se permite realizar la recombinación entre individuos topológicamente conectados de acuerdo a la determinación previa de un vecindario, lo que lleva a un nuevo concepto conocido como:

“*aislamiento por distancia*” lo que ayuda a simular la evolución del sistema desde el punto de vista del individuo y no de la población [1].

Como se observará a lo largo del estudio del estado del arte (capítulo 3), en la mayoría de los casos, se ha establecido claramente que las líneas de investigación, con respecto a la forma en que se lleva a cabo el intercambio de información entre diferentes poblaciones o individuos que conforman a los algoritmos evolutivos con poblaciones estructuradas, circula alrededor de los siguientes tópicos:

- Frecuencia de intercambio de información
- Políticas para la selección y remplazo de los individuos migrados
- Tipo de topología de interconexión
- Cantidad de individuos a ser migrados

Después de haber hecho un exhaustivo estudio del estado del arte, se llegó a la conclusión de que la cantidad de investigación destinada a la exploración de nuevos mecanismos para la estructuración de la población de los algoritmos evolutivos así como la combinación de técnicas de búsqueda que funcionen bajo dicha estructuración, no ha mostrado avances significativos en los últimos años, por lo que en este trabajo, se presenta un nuevo conjunto de modelos alternativos de heurísticas bioinspiradas espacialmente estructuradas es decir, modelos que reestructuran a la población de individuos de forma tal, que ayude a conducir el balance entre el proceso de exploración y explotación del algoritmo. Para ello, se ha estudiado y experimentado intensivamente sobre tres temas muy concretos:

- Restructuración de la población de los algoritmos distribuidos, para modificar el impacto general de los procesos de exploración y explotación dentro de los algoritmos y conseguir así un mejor balance de estos dos aspectos.
- Implementación de un conjunto de micro algoritmos evolutivos (aquellos con poblaciones reducidas de entre 3 y 5 individuos) con ajuste local, los cuales han sido diseñados específicamente para la explotación del espacio de búsqueda (ver capítulos 5 y 6)
- Hibridación de diversas meta-heurísticas, es decir, se ha partido de un conjunto de algoritmos evolutivos (en función del problema) y se ha realizado una combinación de estos para emplear a cada uno de ellos en situaciones bien definidas (ver capítulos 5 y 6) dentro del proceso de búsqueda, lo cual ha conducido a una simbiosis que ha desembocado en resultados alentadores (ver capítulo 7).

Partiendo de la base de que el presente trabajo es de tipo experimental y no teórico, se ha recurrido a un conjunto de técnicas estadísticas para el análisis de los resultados obtenidos y así poder generar conclusiones que sean estadísticamente consistentes. El diseño de experimentos es una tarea de vital importancia en el desarrollo de nuevas propuestas. La validación de nuevos algoritmos, frecuentemente requiere la definición de un marco experimental exhaustivo, incluyendo un amplio abanico de problemas y algoritmos del estado del arte. La parte crítica de estas comparaciones, recae en la validación estadística de los resultados, contrastando las diferencias encontradas entre métodos. En este sentido, es importante contar con una metodología estadística robusta que avale las conclusiones obtenidas. Dentro del conjunto de técnicas disponibles, destacan los tests estadísticos no paramétricos (ver capítulo 2) debido a su flexibilidad y a las pocas restricciones de uso que presentan (en contraste a su contrapartida paramétrica, la cual sufre a menudo problemas derivados de la imposibilidad de cumplir las propiedades de independencia y normalidad necesarias para su uso [8]).

## 1.1 Objetivo general y objetivos particulares

El *objetivo general* de la presente investigación, es proponer un conjunto alternativo de modelos de optimización espacialmente estructurados, en donde la población se encuentre estructurada a través de algún tipo de topología que permita un intercambio ordenado de individuos y supere a los algoritmos clásicos en al menos uno de los siguientes aspectos:

- Velocidad de convergencia (obtengan mas rápido el valor óptimo)
- Calidad de la respuesta (mejorar la respuesta obtenida con respecto al estado del arte)
- Costo computacional (disminuir el número de evaluaciones de la función de aptitud)

Los *objetivos particulares* son:

- Estudiar el comportamiento de la relación exploración/explotación durante procesos de optimización heurística para los modelos estructurados.
- A partir de dicho estudio, proponer cambios a los modelos clásicos que permitan ejercer una mayor regulación sobre el balance entre exploración y explotación
- Establecer una plataforma de pruebas y experimentos para estudiar el comportamiento de la relación exploración/explotación en diversos tipos de problemas de optimización con heurísticas espacialmente estructuradas.
- Experimentar los modelos propuestos en la plataforma previamente establecida para validar y estudiar los resultados obtenidos.
- Publicar, en diversos congresos y revistas nacionales e internacionales, los modelos y resultados obtenidos.

## 1.2 Resumen de aportaciones

En este trabajo de tesis se aportan resultados a la literatura de las heurísticas bioinspiradas espacialmente estructuradas. Para ello la contribución científica que se pretende realizar con nuestro estudio comprende básicamente los siguientes puntos:

- Análisis del comportamiento de los procesos de exploración y explotación en los algoritmos genéticos celulares y de islas.
- Caracterización de forma experimental de los modelos celulares y distribuidos empleando herramientas matemáticas basadas en pruebas de hipótesis, que permitan mostrar estadísticamente, las ventajas de los modelos estructurados con respecto a los modelos existentes en la literatura.
- Diseño de un nuevo modelo evolutivo híbrido que combina un algoritmo genético celular y un micro algoritmo de optimización por cúmulo de partículas, este, es empleado para determinar el número de clases presentes en un conjunto de datos y que como muestran los resultados, es más rápido y eficiente que los algoritmos existentes para dicha tarea.
- Diseño de un nuevo modelo evolutivo híbrido, que combina un algoritmo de evolución diferencial y un micro algoritmo de evolución diferencial cuya sinergia, aporta un nuevo mecanismo capaz de modificar su comportamiento general, balanceando el proceso de exploración y explotación de forma efectiva.
- Diseño de un nuevo modelo distribuido, que soluciona el problema sobre la selección de la topología de comunicación entre poblaciones, así como la tasa y frecuencia de migración de individuos entre estas, implementando un nuevo mecanismo jamás antes visto de fusión de poblaciones, donde una de sus principales ventajas, es la reducción

sistemática, del número de evaluaciones de la función de aptitud con respecto a los modelos distribuidos propuestos en la literatura.

- Implementación de una nueva metodología que divide el proceso de búsqueda en jerarquías, hibridando las heurísticas existentes y como se muestra en los resultados, se mejora la velocidad y calidad de las soluciones encontradas.
- Propuesta, diseño y estudio, del primer modelo ortodoxo de un algoritmo celular híbrido para estimar el número de clases presentes en un conjunto de datos arbitrario.

## 1.3 Organización de este texto

El presente trabajo se encuentra estructurado de la siguiente manera:

En el **capítulo 2**, se presenta el marco teórico y se explican los principales paradigmas del cómputo evolutivo, además de profundizar en la clasificación y selección de los parámetros de operación de los algoritmos genéticos distribuidos, también se realiza un estudio de los diferentes modelos de algoritmos evolutivos con poblaciones estructuradas (descentralizadas) y se concluye con una descripción de las pruebas paramétricas que serán empleadas para el análisis de los resultados experimentales obtenidos por los diferentes modelos propuestos.

En el **capítulo 3**, se realiza una revisión del estado del arte en donde se analizan las propuestas de diferentes autores alrededor de los temas relacionados con la selección de parámetros en los algoritmos evolutivos distribuidos, modelos con poblaciones estructuradas y políticas de selección, remplazo y dispersión de soluciones a través de distintas poblaciones.

En el **capítulo 4**, se presenta el primero de los tres modelos propuestos en este trabajo. El *Modelo de Fusión de Islas*, conserva la estructura de los modelos distribuidos (ver capítulo 2) sin embargo, aporta una serie de características únicas que no están presentes en el modelo distribuido original. El modelo propuesto, responde a la necesidad de la selección de una topología de interconexión entre las poblaciones (islas) dentro de los modelos distribuidos, además, el modelo propuesto analiza el efecto que se obtiene al remplazar la migración de individuos por un nuevo mecanismo de fusión de poblaciones y que como se observará en los resultados y posteriormente en las conclusiones, la fusión de islas supone una alternativa válida para la disminución del número de individuos de la población y como consecuencia del número de evaluaciones de la función de aptitud (es decir la función que determina que tan buena o mala es una solución).

En el **capítulo 5**, se presenta un *Algoritmo Celular Híbrido*, que tiene como objetivo estimar el número de clases contenidas en un conjunto de datos. Esta propuesta, incorpora dos algoritmos evolutivos que permiten dividir los procesos de exploración y explotación, utilizando una población espacialmente estructurada (también conocida como población descentralizada) en forma de malla o rejilla (ver capítulo 2). Como se observará en la sección de conclusiones del capítulo 5, para validar el desempeño y obtener la mejor configuración de los parámetros para el modelo propuesto, se realizaron un total de 19,500 experimentos sobre 10 conjuntos de datos sintéticos con diferentes configuraciones. Para el análisis de los resultados obtenidos, se hicieron pruebas estadísticas basadas en rankings así como pruebas post-hoc que ayudaron a identificar estadísticamente, la configuración del algoritmo con el mejor desempeño.

En el **capítulo 6**, se presenta el tercer y último modelo desarrollado y que a diferencia del capítulo 5 donde el modelo celular simula la evolución del sistema a nivel individuo, el modelo poblacional propuesto, evoluciona a partir del conjunto de creencias de todos los



individuos que conforman a la población, esto elimina el aislamiento por distancia (aislamiento producido por la aparición de vecindarios (ver capítulo 2)). El modelo meta-heurístico propuesto, ha sido enriquecido empleando un método de búsqueda local basado en un micro algoritmo evolutivo y ajuste local. Para realizar la validación del modelo, se realizaron un total de 11,700 experimentos sobre 11 conjuntos de datos sintéticos con diferentes configuraciones en sus clases, variando la compactación y separación de estas así como también su geometría, esto permitió probar, el balance entre la exploración y la explotación del algoritmo presentado. También se realizaron pruebas con dos conjuntos de datos reales (Iris y Wine) los cuales son ampliamente utilizados para la validación de algoritmos de agrupamiento. Para realizar la comparación del rendimiento del algoritmo propuesto, fueron utilizados dos de las meta heurísticas mas utilizadas en el estado del arte para realizar la misma tarea (evolución diferencial y optimización por cúmulo de partículas o PSO). Nuevamente, para validar los resultados se emplearon un conjunto de pruebas estadísticas que ayudaron a determinar de forma cuantitativa al mejor algoritmo de los tres.

Finalmente en el **capítulo 7** se exponen las conclusiones finales derivadas de la investigación realizada durante el desarrollo de los tres modelos propuestos.

Llegados a este punto, se desea aclarar que por la naturaleza de los algoritmos expuestos en esta tesis, se decidió que cada uno de los capítulos donde son presentados los modelos propuestos (capítulos 4, 5 y 6) contengan su propia sección de experimentos, análisis de resultados y conclusiones parciales con la finalidad de no mezclar conceptos y conclusiones al momento de mostrar los resultados.

# Marco Teórico

# 2

El término "*computación evolutiva*", ha sido empleado para referirse a la escuela (de pensamiento) que toma la Teoría de la Evolución como inspiración y marco de referencia para diseñar algoritmos heurísticos de búsqueda y optimización. En 1932 *W. D. Cannon* visualizó la evolución natural como un proceso de aprendizaje y en los años 50 se comenzaron a utilizar los principios de Charles Darwin y Mendel como modelos para la solución de problemas [1], esto representa un esfuerzo conjunto de los investigadores que han tenido diferentes aproximaciones simulando aspectos de la evolución.

Los Algoritmos Genéticos, Estrategias Evolutivas, Programación Genética, Evolución Diferencial, Algoritmos Culturales y Optimización por Cúmulos de Partículas (PSO) son sólo algunas de esas aproximaciones que involucran aspectos comunes del proceso evolutivo como la reproducción, mutación, selección y competencia, aspectos todos ellos esenciales para la evolución [2].

Un programa evolutivo es, desde esta perspectiva, un algoritmo estocástico que mantiene una población de individuos  $P(t) = \{X_1^t, \dots, X_n^t\}$ , donde cada individuo, representa una solución potencial al problema que se desea resolver, y es representado mediante alguna estructura de datos  $S$ . Cada solución  $X_i^t$  es evaluada para obtener una medida de su aptitud. Una nueva población es formada seleccionando a los mejores individuos de la iteración  $t$ , algunos miembros de la nueva población se someterán a transformaciones con operadores de variación (como la cruce y la mutación), para formar finalmente nuevas soluciones [3].

## 2.1 Principales paradigmas del cómputo evolutivo

La Teoría de la Selección Natural de Darwin, propone que las plantas y animales que existen hoy, son el resultado de millones de años de adaptación a las demandas del ambiente [5]. En todo momento, un determinado número de diferentes especies, pueden coexistir y competir por los mismos recursos en el ecosistema, sin embargo, los organismos más capaces (mejor adaptados) para adquirir dichos recursos y además para reproducirse exitosamente, son aquellos cuyos descendientes tendrán una mayor población en el futuro, a diferencia de aquellos menos capaces. Las técnicas de cómputo evolutivo, abstraen estos principios de la evolución en algoritmos que pueden ser usados para buscar soluciones óptimas a problemas.

En un algoritmo de búsqueda convencional, todas las posibles soluciones se encuentran disponibles y la única tarea de dicho algoritmo, es encontrar, de entre ellas, la mejor solución en un tiempo razonable. Para un espacio de búsqueda que contenga solamente un número pequeño de soluciones, todas éstas pueden ser examinadas rápidamente, sin embargo, esta búsqueda exhaustiva puede ser no-eficiente en la medida que el espacio de búsqueda crece [7].

El aspecto clave que distingue a un algoritmo evolutivo de búsqueda de uno tradicional, es que el algoritmo evolutivo se encuentra basado en poblaciones y, a través de la adaptación de sucesivas generaciones, ejecuta una búsqueda dirigida en forma eficiente.

En un algoritmo evolutivo, el esquema de representación del individuo es elegido por el investigador para definir el conjunto de soluciones que forman su espacio de búsqueda [11]; de esa manera, un conjunto de soluciones individuales son creadas para formar una población inicial. La evaluación de los individuos, así como la aplicación sucesiva de los

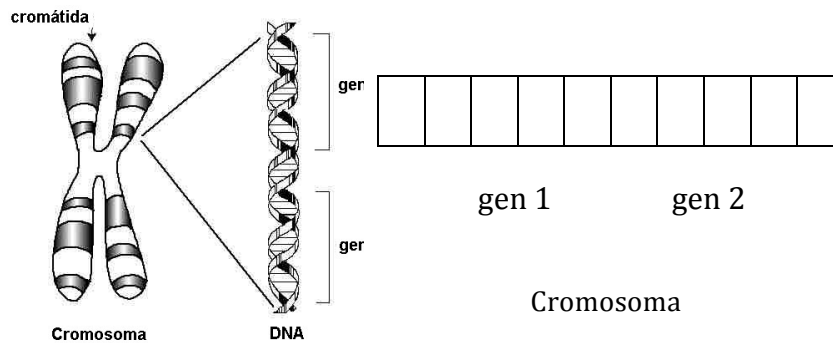
diferentes operadores de variación (cruza, mutación, etc.) son repetidas de forma iterativa hasta encontrar alguna solución que satisfaga el criterio de terminación predefinido.

Cada individuo es evaluado utilizando una función de aptitud (fitness), la cual es específica del problema que se está resolviendo. Con base en los valores de aptitud obtenidos por esta función, un determinado número de individuos es seleccionado para combinar sus características y generar un nuevo individuo (ser progenitores en la terminología biológica); así, nuevos individuos o descendientes son producidos a partir de ellos usando, operadores específicos de reproducción. Los valores de aptitud de estos descendientes son determinados y finalmente los sobrevivientes son seleccionados de la población anterior para formar una nueva población denominada *siguiente generación*.

Existen cuatro paradigmas históricos que sirven como base para los nuevos desarrollos en computación evolutiva: algoritmos genéticos (Holland 1975), programación genética (Koza 1992, 1994), estrategias evolutivas (Recheuberg 1973) y programación evolutiva (Forgel 1966). La diferencia básica entre estos paradigmas consiste en la naturaleza de los esquemas de representación, los operadores de reproducción y los métodos de selección [7].

### **2.1.1 Algoritmos genéticos.**

La técnica más popular en computación evolutiva han sido los algoritmos genéticos (GA por sus siglas en ingles de Genetic Algorithm) [1]. En el GA clásico, cada individuo es representado por una cadena de bits de longitud fija. Se asume que cada posición en la cadena representa una característica específica del individuo y el valor almacenado en dicha posición, representa la forma en que se manifiesta dicha característica en la solución. La analogía con la Teoría de la Evolución radica en que cada individuo (cadena de bits) representa una entidad que es estructuralmente independiente de otra (Figura 2.1) .

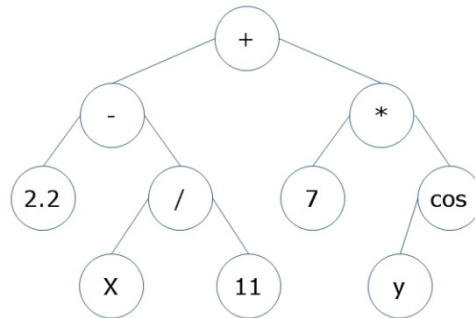


**Figura 2.1.** Analogía entre un cromosoma biológico y una estructura de datos

El principal operador de reproducción usado en esta técnica es la cruce, en la cual dos cadenas de bits representan a los padres y los nuevos individuos serán formados por intercambio entre estos. Otro operador popular utilizado en esta técnica es el de mutación, en la cual, un bit de la cadena puede ser alterado de forma aleatoria para formar así un nuevo individuo.

### 2.1.2 Programación genética.

Una técnica evolutiva que ha incrementado mucho su popularidad en la actualidad es la programación genética. En un programa genético estándar, la representación utilizada para un individuo es un árbol de longitud variable; éste se encuentra formado por un grupo de funciones y variables que codifican al individuo y a la solución del problema. Cada hoja del árbol representa una variable, mientras que cada nodo interno representa una función determinada. (Figura 2.2).



**Figura 2.2.** Representación de datos en Programación Genética

El operador de cruza más utilizado por la programación genética es la cruza de subárboles, en el cual, un subárbol entero es intercambiado entre dos padres. En un programa genético estándar, se asume que todas las funciones devuelven el mismo tipo de dato, este principio permite que cualquier árbol sea considerado estructuralmente igual a otro subárbol, dando como resultado que la operación de cruza genere árboles válidos.

### 2.1.3 Estrategias evolutivas.

La representación de los individuos utilizada por las estrategias evolutivas es un vector de valores reales con longitud fija. Como con las cadenas de bits de los algoritmos genéticos, cada posición en el vector corresponde con una característica del individuo, sin embargo, dichas características son consideradas como un comportamiento más que como una estructura.

El principal operador de reproducción en las estrategias evolutivas es la mutación Gaussiana, en la cual, un valor de una distribución Gaussiana es agregado a cada elemento de un vector individual para crear un nuevo descendiente. Otro operador utilizado es la

recombinación intermedia, en esta, los vectores de dos padres son promediados elemento a elemento para formar un nuevo descendiente.

### 2.1.4 Evolución diferencial.

El algoritmo de Evolución Diferencial (*DE* por sus siglas en inglés Diferencial Evolución), fue desarrollado por Rainer Storn y Kenneth Price [24] para resolver principalmente problemas de optimización en espacios continuos, en este algoritmo, las variables son representadas por medio de arreglos de números reales de dimensión '*D*'. Storn y Price, destacaron diferentes variantes del algoritmo, dichas variantes, difieren en función del número de padres usados para generar el vector de prueba, así como la forma de combinar las partes de los diferentes vectores seleccionados. La variante de *DE* más comúnmente usada se denomina **DE/rand/1/bin**, su nombre deriva del hecho de que el vector base ( $r_0$ ) es seleccionado de forma aleatoria (/rand), solo una diferencia de vectores es sumada es este mismo vector (/1) y finalmente, el número de parámetros heredados sigue una distribución binomial (/bin). En el algoritmo 2.1, se muestra el pseudocódigo de dicha variante.

---

**Algoritmo 2.1:** Pseudocódigo para el algoritmo DE/rand/1/bin

---

```

1  p = GenerateInitialPopulation(Np)
2  do
3      for (i=0; i<NP; i++) do
4          do  $r_0 = \text{floor}(\text{rand}(0,1)*Np)$  while ( $r_0 == i$ )
5          do  $r_1 = \text{floor}(\text{rand}(0,1)*Np)$  while ( $r_1 == r_0$  or  $r_1 == i$ )
6          do  $r_2 = \text{floor}(\text{rand}(0,1)*Np)$  while ( $r_2 == r_1$  or  $r_2 == r_0$  or  $r_2 == i$ )
7           $jrand = \text{floor}(D*\text{rand}(0,1))$ 
8          for (j=0; j<D; j++) do
9              if ( $\text{rand}(0,1) < Cr$  or  $j == jrand$ ) then
10                  $u_{j,i} = x_{j,r_0} + F(x_{j,r_1} - x_{j,r_2})$ 
11             else

```

```

12              $u_{j,i} = x_{j,i}$ 
13         end if
14     end for
15 end for
16 for ( $i=0; i < NP; i++$ ) do
17     If ( $f(u_i) \leq f(x_i)$ ) then
18          $x_i = u_i$ 
19     end if
20 end for
21 while (Convergence Criterion no yet met)

```

---

Como ya se comento, en la *DE*, las variables del problema a optimizar se encuentran codificadas en un vector de números reales  $x_{p,n}^g$ , donde  $g$  representa la generación,  $p$  al individuo de  $g$  y  $n$  el índice de la variable dentro del individuo  $x$  [33]. El dominio de la variable  $x_{p,n}^g$ , se encuentra restringido por los valores  $x_{p,n}^{min}$ , y  $x_{p,n}^{max}$ , es decir:

$$x_{p,n}^g \in \{x_{p,n}^{min}, x_{p,n}^{max}\} \quad (2.1)$$

La inicialización genera una población de vectores aleatorios tal que:

$$x_{p,m}^1 = x_m^{min} + \sigma(x_m^{max} - x_m^{min}) \quad (2.2)$$

Dónde:  $\sigma$  es un número aleatorio uniformemente distribuido en el intervalo  $[0,1]$ ,  $p \in \{1 \dots NP\}$ ,  $NP$  define al número de individuos de la población  $p$  y  $m \in \{1 \dots n\}$  es el índice de la variable del individuo  $x$  [33]. El operador de mutación utilizado en la evolución diferencial clásica se define como:



$$n_p^g = x_c + \lambda(x_a - x_b) \quad (2.3)$$

$n_p^g$  es denominado *vector aleatorio ruidoso* dónde:  $x_a, x_b, x_c$  son individuos elegidos al azar con  $p \neq a \neq b \neq c$  y  $p \in \{1 \dots NP\}$ .  $\lambda \in [0 \dots 1]$  es un parámetro que controla la tasa de mutación [33]. La recombinación o cruza, se efectúa de manera aleatoria comparando los vectores originales con los vectores mutados de la siguiente manera:

$$t_{p,m}^g = \begin{cases} n_{p,m}^g & \text{si } \sigma < \beta \\ x_{p,m}^g & \text{en otro caso} \end{cases} \quad (2.4)$$

Dónde:  $\beta$  es un parámetro que controla la tasa de recombinación.  $t_{p,m}^g$  es denominado vector de prueba (*trial vector*). Finalmente la selección se realiza utilizando la función:

$$x_p^{g+1} = \begin{cases} t_p^g & \text{si } f(t_p^g) \leq f(x_p^g) \\ x_p^g & \text{en otro caso} \end{cases} \quad (2.5)$$

Dónde:  $f$  define a la función de adaptación.

Las condiciones de terminación usualmente más empleadas por la evolución diferencial son [33]:

- Número de generaciones alcanzado
- Tiempo de evolución transcurrido
- Calidad de solución alcanzada

### **2.1.5 Micro Evolución Diferencial ( $\mu DE$ )**

Un micro algoritmo de evolución diferencial ( $\mu DE$ ), consiste en un algoritmo de *DE* pero con población reducida (normalmente entre cuatro y seis individuos). Dada la naturaleza del algoritmo de Evolución Diferencial, el número mínimo de individuos para que un  $\mu DE$  sea funcional, es de cuatro (índices  $i, r_0, r_1$  y  $r_2$ ). Para mantener la diversidad dentro en los  $\mu EA$  (micro algoritmos evolutivos), se incorporan ciertos mecanismos como por ejemplo la re inicialización de la población con cierta frecuencia, esto permite que el algoritmo no se quede atrapado en un óptimo local.

### **2.1.6 Enjambre de partículas**

Un algoritmo de optimización por cumulo de partículas (denominado PSO)[29][56], es una meta-heurística poblacional inspirada en el comportamiento social de las bandadas de aves y bancos de peces, aunque en muchas investigaciones se ha utilizado también el comportamiento de las colonias de hormigas para la solución de problemas de optimización (ACO por sus siglas en inglés Ant Colony Optimization). Este tipo de algoritmos sigue el modelo biológico conocido como Metáfora Social (Kennedy and Eberhart, 1995) la cual establece, que todos los individuos que forman parte de una sociedad, crean sus opiniones a partir de su propio conocimiento y experiencia sin embargo, también consideran en cierta medida, las creencias de la sociedad en general.

En los PSO, las partículas que forman al enjambre (los individuos), tienen una posición y velocidad, y cuentan con dos capacidades de razonamiento esenciales: primero, pueden recordar la mejor posición que han ocupado ellas mismas y segundo, conocen la mejor posición que han ocupado el resto de las partículas en el espacio de búsqueda. Cada uno de las partículas del enjambre, comunican su mejor posición con el resto de los individuos y

basados en dicha información, ajustan su propia posición y velocidad permitiendo de esta manera poder alcanzar el valor óptimo [29].

### 2.1.7 $\mu$ PSO con ajuste local ( $\mu$ PSOLA)

Un  $\mu$ PSO, es un PSO con una población reducida la cual se encuentra típicamente formada de entre 3 a 5 individuos. A diferencia de los algoritmos por cúmulos de partículas clásicos, el  $\mu$ PSO opera empleando de 2 ciclos (ver el pseudocódigo 2.2). El ciclo interno (líneas 6 a la 16) trabajan exactamente como un PSO canónico mientras no se alcance un criterio de convergencia local. El ciclo externo (línea 4) reinicializa la población del algoritmo con cierta frecuencia (línea 18). Para formar la nueva población, el  $\mu$ PSO conserva la mejor solución encontrada hasta el momento (elitismo), mientras que el resto de los individuos son generados aleatoriamente lo cual promueve el proceso de exploración.

---

**Algoritmo 2.2:** Pseudocódigo del  $\mu$ PSOLA

---

```
1  p = GenerarPoblaciónInicial()
2  MejorPosicionParticula = ObtenerMejorPosicionParticula(p)
3  MejorParticulaGlobal = ObtenerMejorParticulaGlobal(p)
4  while not CondiciónDeParada do
5      while not ConvergenciaLocal do
6          for each partícula in p do
7              v = CalcularVelocidad(partícula)
8              x = CalcularPosición(partícula, v)
9              x = AjusteLocal(x)
10             if fitness(x)  $\geq$  fitness(MejorPosicionParticula) then
11                 MejorPosicionParticula = x
12             if fitness(x)  $\geq$  fitness(MejorParticulaGlobal) then
13                 MejorParticulaGlobal = x
14             end if
15         end if
```

```
16         end for
17     end while
18      $p = \text{ReiniciarPoblación}(p, \text{MejorParticulaGlobal})$ 
19 end while
```

---

## 2.2 Algoritmos evolutivos paralelos

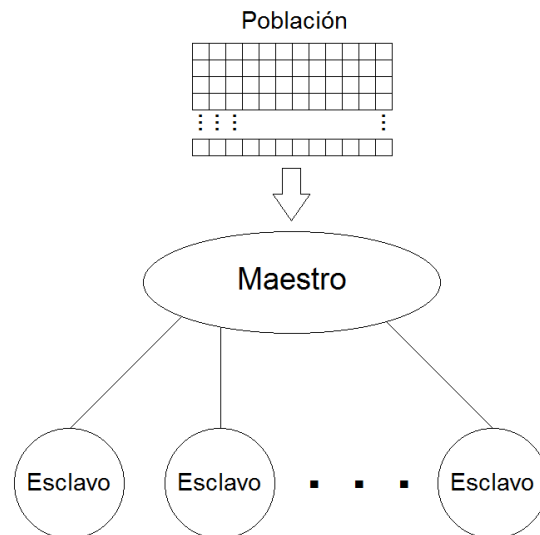
Durante las últimas décadas, el tipo y complejidad de los problemas que se intentan resolver mediante el empleo de cómputo evolutivo, han aumentado, haciendo que el tiempo de ejecución de estos se incremente de forma exponencial debido principalmente, a que su ejecución se realiza generalmente sobre un solo núcleo. La naturaleza poblacional de los algoritmos evolutivos aunado con el actual desarrollo acelerado de las tecnologías de hardware multi-núcleo (CPU y GPU) ha convertido a los algoritmos evolutivos en fuertes candidatos para su ejecución en paralelo [6]. Existen principalmente tres modelos de ejecución paralela en los algoritmos evolutivos:

- Modelo maestro esclavo.
- Modelo distribuido (islas).
- Modelo celular.

### 2.2.1 Modelo Maestro-Eslavo

Los algoritmos evolutivos programados según este modelo, trabajan con una sola población de individuos o población centralizada (Figura 2.3), la cual es gestionada por un nodo maestro. Este modelo, consiste principalmente en distribuir la evaluación de la función de aptitud (y en algunas ocasiones los operadores de cruza y mutación) entre un conjunto de nodos denominados esclavos, mientras que el resto de las operaciones son

ejecutadas por el nodo maestro. Este paradigma es bastante fácil de implementar y la exploración del espacio de búsqueda es conceptualmente idéntica a la exploración realizada por un algoritmo genético serial. Este modelo es más eficiente que la versión serial sólo cuando la evaluación de la función objetivo es costosa.



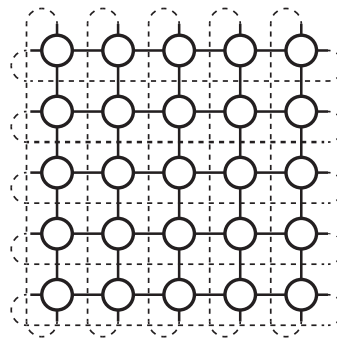
**Figura 2.3:** Modelo Maestro-Esclavo

El intercambio de información entre los nodos procede de la siguiente forma: el nodo maestro envía un subconjunto de individuos a cada nodo esclavo y estos devuelven los valores de la función de aptitud de los individuos procesados. Dicha comunicación puede ser implementada de dos formas: síncrona o asíncrona. Cuando la comunicación es síncrona, el nodo maestro espera a recibir los valores de todos los nodos esclavos antes de continuar con sus operaciones. En la comunicación asíncrona, el algoritmo no espera a que los nodos más lentos envíen sus valores para continuar con sus operaciones. De este modo, se consigue agilizar el proceso pero, en este último caso, el comportamiento del algoritmo maestro-esclavo no es exactamente el mismo que el de un algoritmo genético serial. Para la

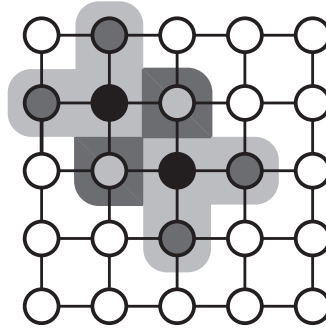
implementación de este tipo de algoritmos, no se ha especificado nada acerca de la arquitectura de hardware subyacente, por lo que pueden ser implementados en computadoras con memoria compartida o distribuida. En el caso de las arquitecturas con soporte para memoria compartida (GPU), la población puede ser almacenada en memoria, donde cada procesador esclavo lee directamente los individuos que le son asignados, mientras que en el caso de las computadoras con memoria distribuida (*cluster*), el nodo maestro es el encargado de distribuir y recoger la información de los nodos esclavos.

### 2.2.2 Modelo celular

La idea esencial del modelo celular, es proveer a la población de una estructura que puede definirse como un grafo conectado en el que cada vértice es un individuo que se comunica con sus vecinos más cercanos. En concreto, los individuos se encuentran dispuestos conceptualmente en una rejilla (ver Figura 2.4), y sólo se permite realizar la recombinación entre individuos topológicamente conectados de acuerdo a la previa determinación de un vecindario. Esto nos lleva a un nuevo concepto conocido como: “*aislamiento por distancia*” (ver Figura 2.5).



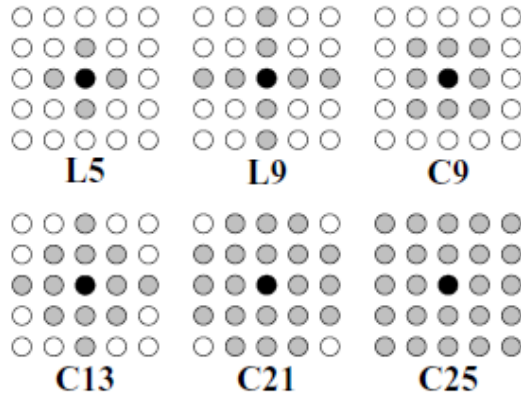
**Figura 2.4:** Estructura de la población de un cGA



**Figura 2.5:** Solapamiento entre vecindarios

El vecindario de un individuo, se define en términos de la distancia Manhattan entre este y el resto de la población. Como es evidente, los vecindarios de los individuos se superponen, lo cual provee un mecanismo implícito de migración entre vecindarios, ocasionando la diseminación de la información al resto de la población. Debido a dicho mecanismo de migración, queda claro que es posible controlar el balance entre *exploración* y *explotación* variando el tamaño y tipo del vecindario.

En cuanto a los tipos de vecindarios (ver Figura 2.6), la etiqueta “ $L_n$ ” (lineal) se utiliza para definir a los vecindarios formados por los individuos (vecinos) que pueden ser alcanzados en  $n$  o menos pasos tomados en una dirección axial determinada (norte, sur, este u oeste), mientras que la etiqueta “ $C_n$ ” (compacto) se emplea para denominar los vecindarios que contienen los  $n - 1$  individuos más cercanos al elemento considerado. Los dos vecindarios más comúnmente utilizados son: (i)  $L_5$  [19, 26, 28], también llamado vecindario de NEWS (por sus siglas en inglés: *North, East, West y South*), o de Von Neumann; y (ii)  $C_9$ , al que también se le conoce como vecindario de Moore.



**Figura 2.6:** Tipos de vecindarios

En función de cómo se aplique el ciclo reproductor a los individuos, existen dos tipos distintos de cGA. Si el ciclo es aplicado a todos los individuos simultáneamente, se dice que el cGA es *síncrono*, puesto que los individuos que formarán la población de la siguiente generación son generados formalmente todos a la vez, de forma paralela. Por otro lado, si se actualizan los nuevos individuos en la población secuencialmente (individuos que formarán parte de la población de la siguiente generación) siguiendo alguna política determinada [12] en lugar de actualizarlos todos a la vez, se tendrá un cGA *asíncrono*. Las principales políticas de actualización de los individuos en los cGAs asíncronos son:

- *Barrido Lineal (Line Sweep - LS)*. Este es el método más simple, consiste en ir actualizando los individuos de la población secuencialmente por filas  $(1, 2, \dots, n)$ .
- *Barrido Aleatorio Fijo (Fixed Random Sweep - FRS)*. En este caso, la siguiente celda a actualizar se selecciona con una probabilidad uniforme sin reemplazo (es decir, que no puede visitarse la misma celda dos veces en una misma generación); esto producirá una secuencia de actualización  $(C_1^j, C_2^k, \dots, C_n^m)$  donde  $C_q^p$ , significa que la celda número  $p$  se



actualiza en el momento  $q$  y  $(j, k, \dots, m)$  es una permutación de las  $n$  celdas. La misma permutación se utiliza entonces para todas las generaciones.

- *Barrido Aleatorio Nuevo (New Line Sweep - NRS)*. Es similar a FRS, con la diferencia de que se utiliza una nueva permutación aleatoria de las celdas en cada generación.
- *Elección Uniforme (Uniform Choice - UC)*. En el caso de este último método, el siguiente individuo a visitar se elige aleatoriamente con probabilidad uniforme de entre todos los componentes de la población con reemplazo (por lo que se puede visitar la misma celda más de una vez en la misma generación). Esto se corresponde con una distribución binomial para la probabilidad de actualización.

---

**Algoritmo 2.3:** Pseudocódigo del cGA canónico

---

```
1   $p$  = GenerarPoblaciónInicial()
2  Evaluar( $p$ )
3  while no CondiciónDeParada do
4      for each  $individuo$  in  $p$ 
5           $vecinos$  = ObtenerVecinos( $p$ ,  $individuo$ )
6           $padres$  = SeleccionarPadres( $vecinos$ )
7           $descendientes$  = Recombinar( $padres$ )
8           $descendientes$  = Mutar( $descendientes$ )
9          Evaluar( $descendientes$ )
10          $paux$  = Reemplazar( $individuo$ ,  $descendientes$ )
11     end for
12   $p$  =  $paux$ 
13  done
```

---

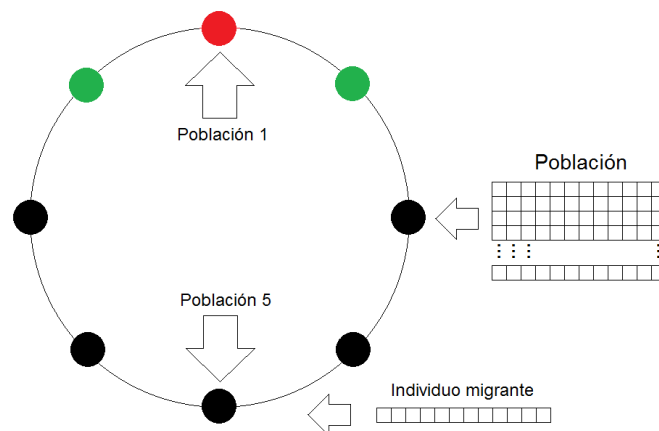
En el Algoritmo 2.3, se muestra el pseudocódigo del cGA. Como se puede observar, en las líneas 1 y 2, el algoritmo genera y evalúa a la población inicial. Posteriormente en la línea 5, se obtiene el vecindario del individuo actual. En la línea 6, se seleccionan los padres que serán recombinados (línea 7) para posteriormente (línea 8) ser mutados. En la línea

10, el individuo actual es remplazado con el mejor de los individuos mutados empleando elitismo. Todo este proceso se ejecuta mientras no se cumpla la condición de paro (línea 3).

### 2.2.3 Modelo distribuido (islas)

Las características más importantes de estos algoritmos [13], son el uso de múltiples poblaciones y la migración (intercambio) de individuos entre ellas. Dado que cada una de las poblaciones evoluciona independientemente, el radio de migración será muy importante para obtener resultados satisfactorios.

La principal motivación del uso de estos algoritmos, es obtener ganancias en velocidad de ejecución a través del uso de islas débilmente acopladas. En la mayoría de los trabajos con algoritmos genéticos distribuidos, se utilizan islas cuya evolución básica es secuencial, esto supone que cada isla ejecuta un número dado de generaciones antes de llegar a la fase de comunicación de individuos con su(s) vecino(s), (Figura 2.7).

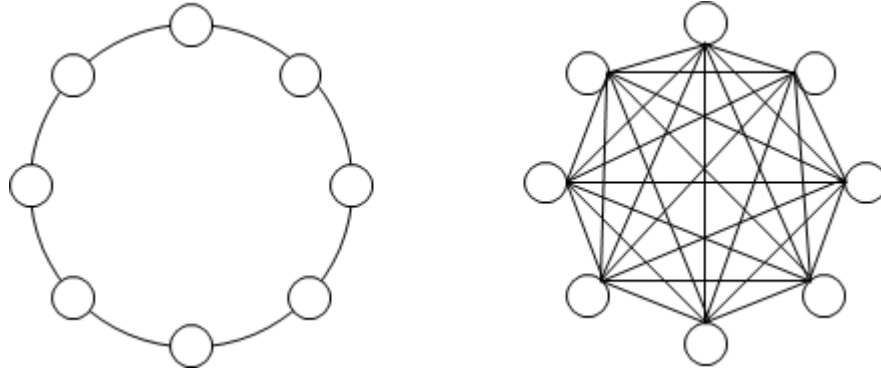


**Figura 2.7:** Modelo distribuido (islas)

### 2.2.3.1 Topología de comunicación

Aunque muchas veces no se tiene demasiado en cuenta, la topología de comunicación es un factor muy importante a la hora de implementar un algoritmo genético distribuido [3] (Figura 2.8), ya que determina la velocidad con la que las soluciones se propagan hacia el resto de poblaciones. Si el grado de conectividad es alto o el diámetro de la red es pequeño (o ambas cosas), las soluciones buenas se expandirán rápidamente a través de todas las poblaciones, favoreciendo así la evolución. En caso contrario, las poblaciones estarán más alejadas las unas de las otras, por lo que evolucionarán más despacio. Por otro lado, es necesario tener en cuenta que la densidad (número) de conexiones puede influir negativamente en el desempeño del algoritmo, ya que supone una sobrecarga importante en el tráfico de la red además, se corre el riesgo de perder la diversidad de las poblaciones (islas) rápidamente. Por eso, elegir correctamente la topología en función del tipo de problema para obtener el máximo rendimiento del algoritmo es una tarea delicada [3].

Por lo general, se suelen utilizar topologías estáticas, es decir, topologías predefinidas que no cambian según avanza la ejecución del algoritmo. Las más utilizadas suelen ser las topologías de anillo y de hipercubo de cuatro dimensiones [12]. Sin embargo, algunas implementaciones proponen que las conexiones entre poblaciones no estén definidas desde el principio, sino que se establezcan sobre la marcha cuando alguna población cumpla algún criterio, como la diversidad de la población [13] o la distancia entre los genotipos de las dos poblaciones [14].



**Figura 2.8:** Topologías de anillo y malla

Finalmente, con relación a la topología de interconexión entre las islas podemos distinguir dos características importantes en un algoritmo evolutivo paralelo distribuido:

- **Comportamiento temporal**, que determina la existencia de un vecindario *estático* o *dinámico*, según el conjunto de vecinos definido por una función  $\xi$  ya sea que cambie o no en el tiempo. En realidad, es el comportamiento de una función componente  $v$  quien determina esta característica al decidir directamente el vecindario [15].
- **Topología del vecindario:** anillo uni/bi-dimensional, árbol, estrella, malla, totalmente conectado, etc. según la disposición de las islas determinada.

### 2.2.3.2 Migración

Hay que tener en cuenta varios factores en la migración de los individuos entre poblaciones, el primero de ellos es la frecuencia con la que estas se llevarán a cabo. La mayor parte de las implementaciones existentes programan las migraciones a intervalos fijos o, dicho de otra manera, de una forma síncrona. Por otro lado, se puede introducir asincronía en la política de migraciones, haciendo que éstas sean efectuadas sólo cuando se

produzca un evento. En el estudio descrito en la tesis de Grosso [16], éste programó las migraciones para que fueran efectuadas únicamente cuando la población estuviera cerca de converger. Braun [17] hizo algo parecido, aunque en este caso las migraciones se realizaban una vez que la población había convergido completamente con el propósito de restaurar la diversidad. Éste es un aspecto muy importante, ya que una migración prematura de individuos puede acarrear serios problemas, debido a que la calidad de los individuos enviados podría no aportar ninguna mejora en el resto de poblaciones, con lo que estaríamos desperdiciando valiosos y costosos recursos de comunicación. Sin embargo, una migración muy tardía puede afectar también negativamente y aumentar considerablemente el tiempo de convergencia al óptimo global o, incluso, propiciar que ésta no se produzca. Por tanto, encontrar la frecuencia adecuada para la migración de los individuos es un aspecto muy importante a tener en cuenta a la hora de diseñar un algoritmo de este tipo.

Otro aspecto indispensable, es determinar qué individuo(s) será(n) enviado(s) en las migraciones. Pettey, Leuze y Grefenstette [18] proponen enviar el mejor individuo de cada población a todas sus adyacentes. Los resultados experimentales que obtuvieron mostraron que los resultados obtenidos eran de una calidad similar a los obtenidos por algoritmos genéticos con una única población. Una aproximación diferente es la que desarrollaron Marin, Trelles-Salazar y Sandoval [19]. Estos propusieron que cada subpoblación (transcurridas las generaciones necesarias para realizar la migración), enviaran su mejor individuo a un nodo maestro, que se encargaría de elegir a los mejores individuos entre todos los recibidos y de reenviarlos de nuevo a todas las poblaciones, con un número pequeño de nodos (alrededor de seis). En sus experimentos, obtuvieron resultados que propiciaban “*speed-ups*” casi lineales en los tiempos de ejecución. En conclusión, la política de migración permite determinar el tipo de intercambio de datos que se está produciendo entre las islas del algoritmo distribuido. Una caracterización de este

concepto debería englobar la mayoría de políticas existentes y permitir proponer nuevas ideas.

### **2.2.3.3 Sincronización**

Cuando se desea implementar un algoritmo evolutivo paralelo, del tipo que sea, una duda que surge inmediatamente es: ¿deben las migraciones efectuarse de forma síncrona o asíncrona? La mayoría de los algoritmos evolutivos distribuidos existentes son *síncronos*.

Esto significa que la fase de migración, entendida como fase de envío y recepción de individuos, está localizada en la misma porción del algoritmo evolutivo distribuido. En este sentido, tanto la implementación como la descripción y el estudio formal se simplifican al asegurarse que todas las islas se encuentran en el mismo estado de evolución en un momento dado.

Por el contrario, el funcionamiento *asíncrono* suele resultar más eficiente en la práctica [20] [21] [22] [23]. Su desventaja es una implementación más complicada y también los problemas que surgen de separar la fase de envío de la de recepción de individuos. Ya que una isla puede recibir en cualquier instante de su evolución un individuo cuando trabaja en modo asíncrono, las islas emisoras y receptoras pueden encontrarse en distintas generaciones y sus individuos en diferentes estados de evolución.

Básicamente, la diferencia radica en la interpretación del operador de migración. En la migración síncrona se realiza el envío de emigrantes sin espera alguna e inmediatamente después el algoritmo se bloquea hasta obtener los inmigrantes de las islas vecinas. Sin embargo, en el caso asíncrono, se realiza el envío de migrantes pero no se bloquea el algoritmo en espera de individuos de entrada, por el contrario, cada paso de la evolución

decide si existen individuos esperando y si es así los inserta de acuerdo a la política de reemplazo usada

Un experimento interesante a este respecto, es el efectuado por Alba y Troya [24], para el que usaron varias implementaciones paralelas de algoritmos genéticos, tanto síncronas como asíncronas, distintas frecuencias de migraciones y número de nodos. Probaron los algoritmos con varias pruebas de diferente complejidad. Los resultados obtenidos pueden verse con más detalle en [24], aunque se puede adelantar que, por regla general, las implementaciones asíncronas obtienen soluciones de igual calidad que las síncronas, evaluando un número similar de individuos, pero en un tiempo inferior. Además, observaron que esta mejora dependía también de la complejidad del problema a resolver y del número de nodos utilizados, ya que si el problema resultaba demasiado simple, o el número de nodos era excesivo, la disminución del tiempo para encontrar una solución óptima era despreciable.

Un hallazgo curioso relacionado con el uso de una política de comunicaciones asíncrona, es el hecho de que estos algoritmos presentan una mayor resistencia al uso de frecuencias de migración erróneas, es decir, si para un problema se eligen frecuencias inapropiadas, tanto si éstas son excesivas como si son insuficientes, la versión síncrona del algoritmo tendrá problemas para encontrar soluciones en un tiempo razonable, llegando en algunos casos a no ser capaces de hacerlo. Por el contrario, parece que esto afecta en menor medida a los algoritmos asíncronos.

En resumen, la calidad de las soluciones encontradas no se ve afectada por el uso de algoritmos síncronos o asíncronos, aunque en general, los tiempos de búsqueda serán inferiores en el caso de estos últimos.

## 2.3 Modelos de búsqueda basados en trayectorias

En esta sección, se describirá brevemente los modelos de búsqueda basados en trayectorias que sirvieron como inspiración para el diseño de dos de los tres modelos desarrollados en esta tesis.

### 2.3.1 Búsqueda por fuerza bruta

Como su nombre indica, la búsqueda por fuerza bruta, es método dirigido que emplea un solo punto para realizar la búsqueda. Este método, divide el espacio de búsqueda en una malla formada por un conjunto de celdas de tamaño fijo, las cuales son visitadas una a una en un cierto orden durante el proceso de búsqueda (ver Figura 2.9).

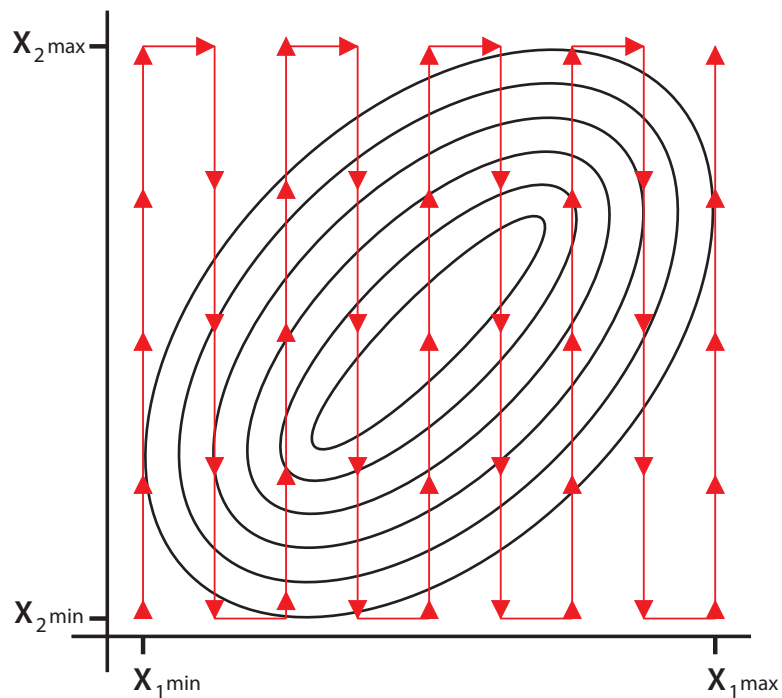


Figura 2.9: Búsqueda por fuerza bruta



Como se puede observar en la Figura 2.9, durante el proceso de búsqueda, se forma una trayectoria de búsqueda, razón por la cual, a este tipo de algoritmos, se les denomina algoritmos basados en trayectorias. Aunque la generación de la secuencia de puntos de búsqueda es una tarea relativamente trivial, el método de búsqueda por fuerza bruta se basa en visitar cada uno de estos puntos, y queda claro que en función de cómo se encuentren generados, habrá zonas del espacio de búsqueda que no serán visitadas.

Por lo tanto, la probabilidad de pasar por alto un punto donde se encuentra el valor óptimo, se encuentra en función del tamaño del paso, es decir, como se han generado los puntos dentro de la malla. Finalmente, para aumentar la probabilidad de encontrar el valor deseado, se debe de disminuir el tamaño del paso, sin embargo, esto produce un aumento del número de puntos por visitar y aunado al espacio dimensional del problema puede ocasionar que la búsqueda tome tiempos prohibitivos.

La explicación del método anterior, ha ayudado a definir dos problemas clásicos al momento de diseñar un algoritmo de búsqueda, que son: Como definir los puntos a visitar dentro del espacio de búsqueda y cual será la trayectoria para visitar dichos puntos.

### **2.3.2 Método de Hooke-Jeeves**

El método de Hooke-Jeeves, ataca el problema del “paso” definido en la sección anterior. Este algoritmo, comienza con un punto inicial  $x_0$ , posteriormente, realiza un proceso de exploración dentro del vecindario de  $x_0$  para obtener la mejor solución dentro de este denominada  $x_1$  (el tamaño del vecindario  $h$  es establecido inicialmente de forma fija ). Si  $x_1$  es mejor que  $x_0$ , entonces se genera un nuevo punto  $x_2$  llamado patrón de movimiento diferencial, el cual se calcula como se muestra en la Ecuación 2.6. Si  $x_2$  es mejor que  $x_1$ , entonces  $x_0 = x_2$  en otro caso  $x_0 = x_1$ . En el caso de que  $x_1$  no sea mejor

que  $x_0$  entonces se reduce el paso de búsqueda. En el pseudocódigo 2.3 se muestra el método de Hooke-Jeeves.

$$x_2 = x_1 + (x_1 - x_0) \quad (2.6)$$

Como se muestra en el pseudocódigo 2.4, en la línea 2, se realiza la búsqueda de soluciones sobre el vecindario de  $x_0$  con una longitud (paso)  $h$ , si la nueva solución  $x_1$  no mejora a la solución actual  $x_0$  entonces se modifica el tamaño del paso multiplicando el paso actual por un factor de reducción ( $h = h * ReductionFactor$ ). En la Figura 2.10 se muestra gráficamente el método de Hooke y Jeeves.

---

**Algoritmo 2.4:** Pseudocódigo para el método de Hooke-Jeeves

---

```
1  while  $h > h_{min}$  do
2       $x_1 = \text{explore}(x_0, h)$ 
3      if  $f(x_1) < f(x_0)$  then
4           $x_2 = x_1 + (x_1 - x_0)$ 
5          if  $f(x_2) < f(x_1)$  then
6               $x_0 = x_2$ 
7          else
8               $x_0 = x_1$ 
9          end if
10     else
11          $h = h * ReductionFactor$ 
12     end if
13 end while
```

---

Uno de los principales problemas del método de Hooke-Jeeves radica en que el tamaño del paso siempre se ve reducido en un factor constante lo que ocasiona que bajo ciertas condiciones el método no converja. El método de búsqueda por poliedros de Nelder y Mead busca precisamente, expandir y contraer el tamaño del paso dinámicamente como sea necesario, para poder solventar el problema del método de Hooke-Jeeves.

### 2.3.3 Método de Nelder y Mead

El método de búsqueda basada en poliedros de Nelder y Mead, intenta resolver el problema del paso, permitiendo que su tamaño se expanda o contraiga dinámicamente. El algoritmo inicia formando un poliedro de dimensión  $(D + 1)$  denominado "*simplex*", donde  $D$ , representa la dimensión del espacio de búsqueda. Los  $D + 1$  puntos que conforman el poliedro, son generados aleatoriamente y se identifican por medio de un índice (en el intervalo comprendido entre 0 y  $D$ ).

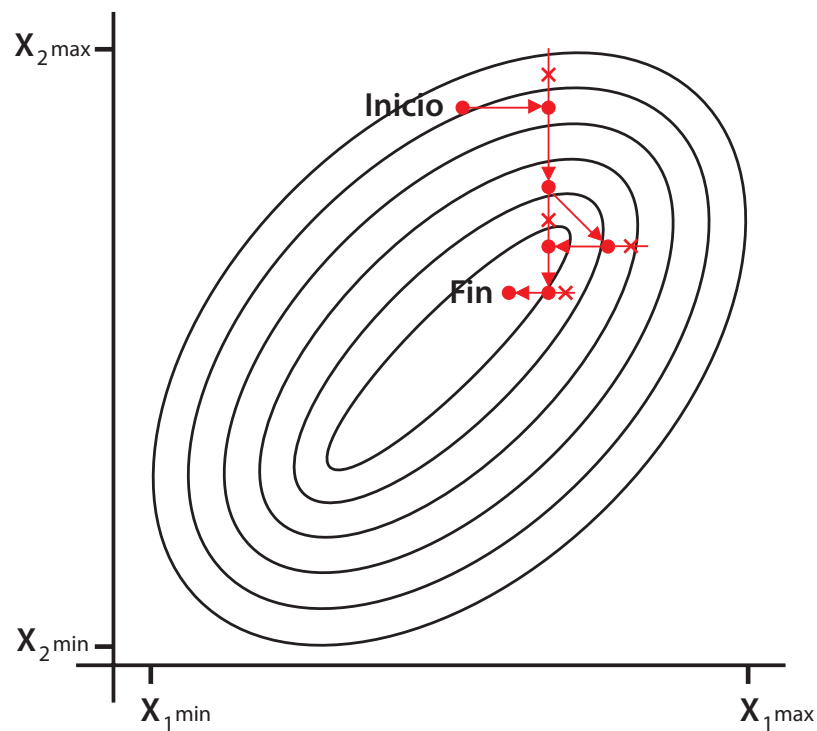


Figura 2.10: Método de Hooke-Jeeves

Por ejemplo, si  $D = 2$ , el *simplex* es un triángulo. Posteriormente, los índices de los puntos que conforman los poliedros, son ordenados ascendentemente en función de su

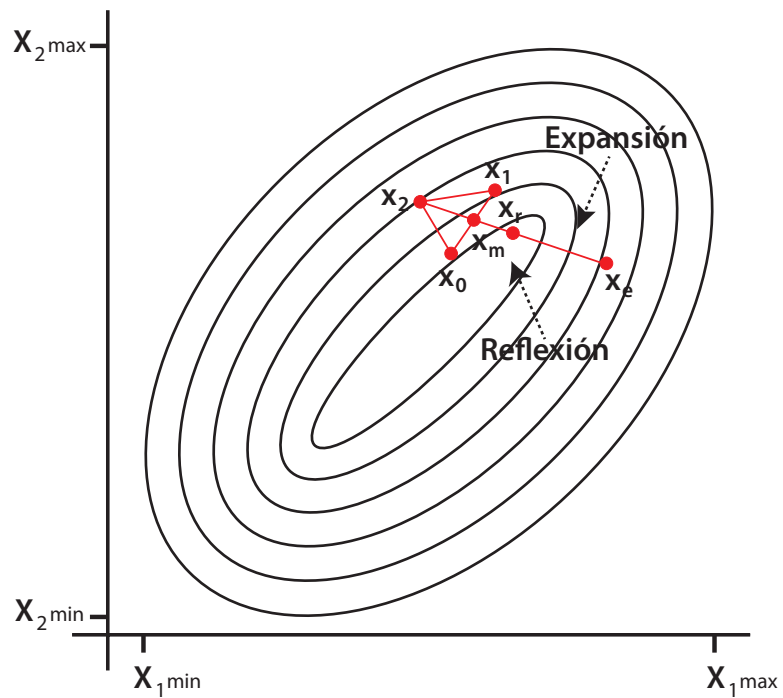
valor de aptitud, por lo que el punto  $x_0$  tendrá el mejor valor de aptitud mientras que  $x_D$  tendrá el peor valor (se supone un problema de minimización).

Para obtener un nuevo punto ( $x_r$ ), el peor punto del poliedro ( $x_D$ ) es reflejado a través de su cara opuesta en el poliedro, usando un factor de escalamiento  $F_1$  (ver ecuación 2.7)

$$x_r = x_m + F_1(x_m - x_D) \quad (2.7)$$

Donde el vector  $x_m$ , es el centroide de la cara opuesta de  $x_D$  como se opuesta en la Ecuación 2.8 (La Figura 2.11, muestra la operación de reflexión definida en la Ecuación 2.7)

$$x_m = \frac{1}{D} \left( \sum_{i=0}^{D-1} x_i \right) \quad (2.8)$$



**Figura 2.11:** Reflexión y expansión de Nelder-Mead

---

**Algoritmo 2.5:** Pseudocódigo para el algoritmo de Nelder-Mead

---

```
1  while (Convergence Criterion no yet met) do
2       $x = \text{sort}(x, D + 1)$ 
3       $x_m = \text{getCentroid}(x)$ 
4       $x_r = x_m + F_1(x_m - x_D)$ 
5      if  $f(x_r) < f(x_0)$  then
6           $x_e = x_r + F_2(x_m - x_D)$ 
7          if  $f(x_e) < f(x_0)$  then
8               $x_D = x_e$ 
9          else
10              $x_D = x_r$ 
11         end if
12     else if  $f(x_r) < f(x_{D-1})$ 
13          $x_D = x_r$ 
14     else
15         if  $f(x_r) < f(x_D)$ 
16              $x_D = x_r$ 
17              $x_c = x_m + F_3(x_m - x_D)$ 
18         else
19              $x_c = x_m + F_3(x_m - x_D)$ 
20         end if
21         if  $f(x_c) < f(x_D)$ 
22              $x_D = x_c$ 
23         else
24              $x = \text{contract}(x, x_0)$ 
25         end if
26     end if
27 end while
```

---

Como se muestra en el Algoritmo 2.5, en la línea 2, el método comienza ordenando los puntos que conforman al *simplex* para poder realizar la proyección y así obtener el nuevo punto  $x_r$  (líneas 2, 3 y 4). Si el nuevo punto  $x_r$  representa una mejor solución que  $x_0$  (la mejor solución hallada hasta el momento), entonces, se obtiene un nuevo punto  $x_e$ , el cual es calculado realizando una expansión en la misma dirección de  $x_r$  y utilizando un factor de escalamiento  $F_1$  (líneas 5 y 6), lo anterior, implica realizar una exploración del

espacio de búsqueda de forma controlada, ya que todas las operaciones se realizan tomando como base al punto  $x_0$ . Si  $x_e$  mejora la solución encontrada hasta el momento, entonces se substituye  $x_D$  (peor solución) por  $x_e$  (línea 8), en caso contrario, se substituye el peor punto por el mejor  $x_D$  por  $x_r$  (línea 10). Si ninguno de los puntos anteriores mejoran la solución, entonces se realiza una contracción en torno al centroide  $x_m$  lo que claramente representa una explotación del espacio de búsqueda (línea 17), finalmente si aun así no se logra mejorar la solución, se intensifica el proceso de búsqueda en torno a la mejor solución encontrada ( $x_0$ ) realizando nuevamente una contracción del *simplex* tomando como referencia a dicho punto.

Nuevamente, como ocurre con el caso del método de Hooke-Jeeves, el conjunto de soluciones encontradas, forman una trayectoria de búsqueda sobre el espacio de búsqueda. Si bien los métodos basados en trayectorias ofrecen soluciones aceptables en tiempos relativamente cortos, dichas soluciones dependen fuertemente de múltiples factores dentro de los cuales destacan la posición del ó los puntos iniciales y de los factores de escalamiento y contracción para el caso del algoritmo de Nelder y Mead.

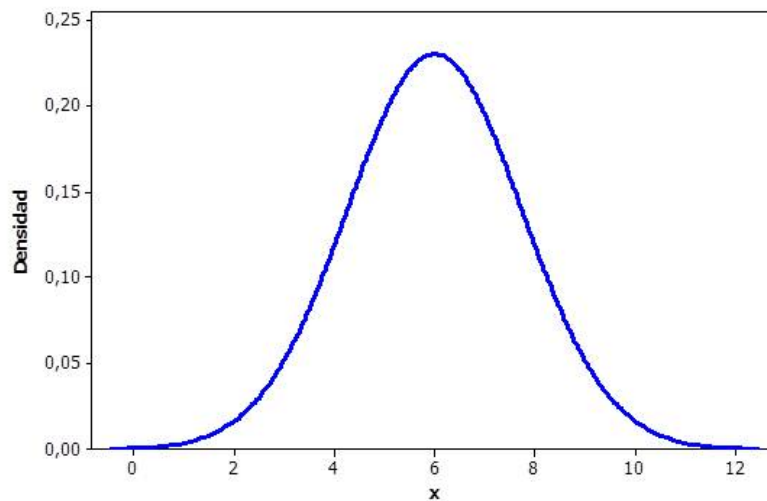
## **2.4 Tests estadísticos y pruebas *Post-Hoc***

El diseño de experimentos es una tarea de importancia capital en el desarrollo de nuevas propuestas. La validación de nuevos algoritmos frecuentemente requiere la definición de un marco experimental exhaustivo, incluyendo un amplio abanico de problemas y algoritmos del estado del arte. La parte crítica de estas comparaciones recae en la validación estadística de los resultados, contrastando las diferencias encontradas entre métodos. En este sentido, es importante contar con una metodología estadística robusta que avale las conclusiones obtenidas. Dentro del conjunto de técnicas disponibles, destacan las pruebas paramétricas y no paramétricas [38] y gracias a la flexibilidad de

estas últimas así como a las pocas restricciones de uso que presentan (en contraste a su contrapartida paramétrica, la cual sufre a menudo problemas derivados de la imposibilidad de cumplir las propiedades de independencia, normalidad y homocedasticidad, necesarias para su uso [38]), serán empleados para la validación de los resultados obtenidos.

Para la aplicación de las pruebas paramétricas, se deben de cumplir necesariamente tres condiciones básicas:

- La variable a ser medida debe de ser numérica.
- La población de la que se obtiene la muestra, debe de tener una distribución normal (Figura 2.3) esto último, puede ser comprobado con la aplicación de la prueba de Kolmogorov Smirnov.
- Debe de presentarse Homocedasticidad dentro de los datos es decir, debe de existir homogeneidad de las varianzas lo cual puede ser comprobado mediante el empleo de la prueba de Levene.



**Figura 2.12:** Distribución normal

Las primeras apariciones de metodologías estadísticas se limitan a realizar comparaciones del algoritmos por pares. El Test de Signos y el Test de Wilcoxon (y su contrapartida paramétrica, el t-test) permiten evaluar el rendimiento de dos algoritmos en un entorno múltiple-problema. Un número razonablemente elevado de problemas permite a estos tests establecer una comparación fiable entre pares de técnicas, produciendo un p-valor para reflejar el resultado de dicha comparación. Sin embargo, es habitual encontrar situaciones en que las comparaciones por pares son insuficientes. Por ejemplo, la presentación de una nueva propuesta generalmente requerirá la realización de una comparación simultánea contra varios métodos del estado del arte. En éste tipo de casos, las comparaciones por pares suelen ser insuficientes, ya que los p-valores obtenidos al realizar cada comparación no pueden acumularse de forma rigurosa. El error cometido por considerar simultáneamente una familia de hipótesis relacionadas en lugar de tomarlas individualmente, hace deseable la utilización de técnicas de comparación múltiple para obtener un análisis más preciso.

### 2.4.1 Test de Friedman

El Test de Friedman [7], [8] trabaja asignando rankings  $r_{ij}$  a los resultados obtenidos por cada algoritmo  $j$  en cada problema  $i$ . Esto es, para cada problema, se asigna un ranking  $1 \leq r_{ij} \leq k$ , donde  $k$  es el número de algoritmos a comparar. Estos rankings, se asignan de forma ascendente, es decir, 1 al mejor resultado, 2 al segundo, etc. (en caso de haber empates, se asignan rankings medios).

El Test de Friedman requiere el cálculo de los rankings medios de los algoritmos sobre los  $n$  problemas (ver ecuación 2.9).

$$R_j = \frac{\sum_{i=1}^n r_{ij}}{n} \quad (2.9)$$



La hipótesis nula que indica que todos los algoritmos se comportan similarmente, por lo que sus rankings  $R_j$  deben ser similares. Siguiendo esta hipótesis, el estadístico de Friedman se distribuye de acuerdo a una distribución  $\chi^2$  con  $k - 1$  grados de libertad (ecuación 2.10).

$$F_F = \frac{12n}{k(k+1)} \left[ \sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \quad (2.10)$$

Este estadístico fue mejorado a su vez por Iman y Davenport, quienes mostraron que el estadístico de Friedman presenta un comportamiento demasiado conservativo. Para evitar éste problema, propusieron otro estadístico más ajustado que se distribuye de acuerdo a una distribución  $F$  con  $k - 1$  y  $(k - 1)(n - 1)$  grados de libertad (ecuación 2.11).

$$F_{ID} = \frac{(n-1)F_F}{n(k+1) - F_F} \quad (2.11)$$

## 2.4.2 Test de Friedman Alineado

El Test de Friedman está orientado a realizar comparaciones intra-conjunto (comparaciones entre rendimientos de algoritmos en un solo conjunto), sin considerar las interrelaciones que puedan existir entre los conjuntos de la prueba completa. Cuando el número de algoritmos en la comparación es pequeño (3,5, . . .), este procedimiento tiene cierta desventaja. En estos casos, en los que la comparación entre conjuntos de datos es más deseable, es recomendable emplear el Test de Friedman Alineado [38]. En esta técnica, se calcula el rendimiento medio alcanzado por cada algoritmo en cada problema (valor de localización). Después, se calculan las diferencias entre el rendimiento obtenido por cada

algoritmo con respecto al valor de localización. Este paso se repite para todos los algoritmos y problemas.

Las diferencias resultantes (observaciones alineadas) se ordenan desde 1 hasta  $k \cdot n$  de forma relativa unas con otras. A partir de ahí, el esquema de ranking es el mismo que el empleado por un procedimiento de comparaciones múltiples con muestras independientes, como el test de Kruskal-Wallis. De este modo, los rankings asignados a las observaciones alineadas se denominan rankings alineados.

El estadístico del Test de Friedman Alineado se define como se muestra en la ecuación 2.12, donde  $\hat{R}_i$  es igual al ranking total del  $i$ -ésimo problema y  $\hat{R}_j$  es el ranking total del  $j$ -ésimo algoritmo.

## **2.5 Estimación del número de clases dentro de un conjunto de datos**

Un algoritmo de agrupamiento [31][41][44][53][54] (en inglés denominado clustering), es un procedimiento que agrupa una serie de vectores de acuerdo con un criterio de cercanía. Esta cercanía se puede definir en términos de una función de distancia como la euclidiana, aunque también existen otras más robustas que permiten extenderla a variables discretas [37].

Por lo general, los vectores de un mismo grupo (clase o clúster), comparten propiedades comunes [61][62][68]. El conocimiento de los grupos ó clases puede permitir una descripción sintética de un conjunto de datos multidimensional complejo. Esta descripción se consigue sustituyendo la descripción de todos los elementos de un grupo por la de un representante característico del mismo [50][30][36].

Uno de los problemas que durante mucho tiempo ha llamado la atención de los investigadores dentro del área del reconocimiento de patrones, ha sido el poder estimar el número de clases presentes dentro de un conjunto de datos (típicamente a dicho número de clases se le denomina valor de 'k'). Si bien queda claro que no existe una respuesta única ó correcta a este problema, también queda claro que una adecuada estimación de dicho valor e 'k' es indispensable para el correcto funcionamiento de muchos de los algoritmos de agrupamiento no supervisado más potentes de hoy en día.

Dado que un problema de clasificación puede ser definido en términos de un problema de optimización, el área del reconocimiento de patrones no ha sido indiferente a las herramientas del computo evolutivo y como consecuencia, se han realizado numerosas investigaciones donde se utilizan dichos algoritmos como métodos de solución para los problemas de clasificación.

A diferencia de lo que ocurre en la mayoría de los trabajos que utilizan algoritmos evolutivos para la clasificación [51][33] donde se utiliza la suma del error cuadrático medio o SSE como función de aptitud, en [42][55] se emplea una medida de calidad o validación del agrupamiento realizado por el algoritmo como función de aptitud es decir, el problema de clasificación es planteado como un problema combinatorio. Algunas funciones típicas de validación de clusters son el Índice de Davies-Bouldin, Xle-Beni e Índice-I.

### **2.5.1 Índices de validación de agrupamientos**

Diferentes métodos de agrupamiento, pueden entregar resultados completamente distintos [34][48][58] incluso para un mismo conjunto de datos. El objetivo de la validación del agrupamiento, consiste en encontrar el particionamiento que mejor se ajuste a los datos subyacentes. Hoy en día, existen muchos índices de validación de agrupamiento de los

cuales solamente tres fueron utilizados en la presente investigación: el índice de Davis-Bouldin (DB), el índice de Xie-Beni (XB) y el índice  $\bar{I}$ .

En el índice DB (Ecuación 2.1), se define una medida de similitud entre las clases  $C_i$  y  $C_j$ . Dicha similitud, se encuentra basada en la medida de dispersión de las clases  $C_i$  y  $C_j$  así como la separación entre ambas clases.

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{j=1..k} \left\{ \frac{Disp(C_i) + Disp(C_j)}{Sep(C_i, C_j)} \right\} \quad (2.12)$$

Donde  $Disp(C_i)$  es la dispersión de la clase  $C_i$  la cual es calculada como se indica en la Ecuación 2.13.

$$Disp(C_i) = \sqrt{\frac{1}{|C_i|} \sum_{o_i, o_j \in C_i} \|o_i, o_j\|^2} \quad (2.13)$$

Y  $Sep(C_i, C_j)$  es la separación entre la clase  $i$  y la clase  $j$  y puede ser medida con alguna función de unión adecuada (simple, completa, promedio, etc).

En el índice Xie-Beni [67](Ecuación 2.14), la pertenencia difusa del patrón  $x_i$  a la clase  $j$  se define como la distancia entre la clase  $x_i$  al centro de la clase, multiplicado por su pertenencia difusa a la misma clase  $j$ .

$$XB = \frac{\sum_{j=1}^k \sum_{i=1}^n \mu_{i,j} \|O_i, u_j\|^2}{n \left( \min_{p=1..k} \{\|u_j, u_p\|^2\} \right)} \quad (2.14)$$

Donde  $\mu_{i,j}$  es la pertenencia del objeto  $i$  a la clase  $j$  y  $u_j$  es el centroide de la clase  $j$ . Finalmente, en el Índice-I (Ecuación 2.15), el primer factor normaliza cada valor del índice por el número total de grupos, el segundo término establece el suma del error cuadrático global para todo el conjunto de datos en relación con el error intra-clase y el tercer término incorpora la máxima diferencia observada entre dos de las  $k$  clases. El cálculo del índice incluye el parámetro  $p \in \mathbb{R}$  para controlar el contraste entre las diferentes configuraciones de las clases.

$$I = \left( \frac{1}{k} \cdot \frac{E_1}{E_k} \cdot D_k \right)^2 \quad (2.15)$$

# Revisión del estado del arte

# 3

El cómputo evolutivo, ha surgido como una herramienta para la solución de problemas complejos para los cuales en muchas ocasiones no existen métodos analíticos para su solución. Aun así, la dificultad de dichos problemas ha aumentado a un grado tal, que por un lado algunos algoritmos evolutivos son insuficientes para la solución de dichos problemas[7][9][50][65][70] y por otro lado, el tiempo requerido para encontrar una solución aceptable a veces resulta prohibitivo. Por tal razón, los investigadores han buscado nuevas formas de resolver los problemas, en algunos casos diseñando nuevos tipos de algoritmos como por ejemplo: la optimización basada en colonias de abejas, la optimización basada en el comportamiento de las luciérnagas, hasta la optimización basada en esquemas musicales [1] y por otro lado, mejorando el comportamiento de los algoritmos existentes [33][26][47], ya sea diseñando técnicas que permitan encontrar los valores adecuados para los parámetros de operación bajo ciertas circunstancias o realizando algunas modificaciones que cambian el comportamiento de los algoritmos de forma tal, que sean capaces de mejorar su comportamiento ante ciertos problemas que en sus versiones canónicas u originales no podían resolver.

## 3.1 Algoritmos evolutivos distribuidos

En la última década, las tecnologías de procesamiento en paralelo (como son los CPUs multinúcleo y las GPUs), ha evolucionado a tal grado, que prácticamente todos los quipos de cómputo de uso cotidiano disponen de ellos lo cual, abre una oportunidad importante, ya que se pueden plantear nuevos algoritmos masivamente paralelos lo cual

aunado al desarrollo de nuevos mecanismos que permitan optimizar el comportamiento de los algoritmos actuales, dañ como resultado la posibilidad de resolver problemas en tiempos tales que antes no eran posibles [27][32][46][59].

Los algoritmos computacionales inspirados en la Teoría de Evolución de Darwin (los algoritmos evolutivos), han sido utilizados, como ya se mencionó, para la solución de problemas de optimización dada su gran versatilidad y simplicidad de implementación [21][35][43][49][69]. Estos algoritmos, utilizan múltiples parámetros para su operación y la selección de un valor adecuado para cada parámetro es una tarea que no resulta trivial.

En el caso de los algoritmos genéticos (uno de los tipos de algoritmo evolutivo más estudiados[1][3][6][9][52], dichos parámetros son principalmente, el tamaño de la población y las probabilidades de cruce y mutación además de la función de selección. La adecuada selección de estos parámetros impacta de manera significativa en la velocidad de convergencia del algoritmo. Muchos algoritmos genéticos, establecen dichos parámetros *a priori*, es decir, son elegidos sin un conocimiento previo de la población y del problema al inicio de la ejecución, lo cual influye en los tiempos de convergencia y rendimiento del algoritmo genético.

Durante esta sección se revisarán los principales trabajos de selección de parámetros para algoritmos evolutivos que de alguna forma han impactado en el estado del arte y que representan una influencia definitiva para este trabajo [20][21][18][23].

Como consecuencia del impacto que tiene la selección de los parámetros en los algoritmos bioinspirados [35][43] y, tomando como ejemplo a los algoritmos genéticos, se han realizado numerosas investigaciones que buscan definir la manera adecuada de seleccionar dichos parámetros.

De forma clásica, en los algoritmos genéticos, la probabilidad de mutación es establecida como un parámetro de entrada al algoritmo, el cual permanece constante a lo largo de la ejecución del algoritmo. En [2] se presenta un algoritmo de estrategias evolutivas donde, la probabilidad de mutación es modificada y adaptada de forma monótona durante el proceso de búsqueda. Los primeros resultados experimentales indican que la auto-adaptación de la probabilidad de mutación es una buena alternativa como opción al manejo de estos parámetros de forma estática en los algoritmos evolutivos [2].

En [4] se propone el uso de sistemas de control difuso (FLC) con el objetivo de incorporar el conocimiento y la experiencia de los expertos para redefinir los operadores de variación del algoritmo, algunos otros trabajos que se han realizado, buscan desde el punto de vista de la modificación de los operadores genéticos, mejorar el rendimiento, tratando de encontrar un balance entre los espacios de exploración y explotación con los que opera el algoritmo [5], también se han realizado trabajos en los que se proponen nuevos métodos de cruce e incluso métodos en los cuales el comportamiento de los operadores genéticos es dinámico con respecto a la evolución del algoritmo [6].

La mayoría de los algoritmos evolutivos, usan una sola población (panmixia) [37][39][45] de individuos y aplican operadores de recombinación sobre éstos. En contraste, existe cierta tradición en el uso de algoritmos evolutivos estructurados donde la población es descentralizada (la población original es dividida en subpoblaciones), este tipo de algoritmos distribuidos son fuertes candidatos a ser implementados utilizando técnicas paralelas [6].

Cuando se habla de nicho [6][9], nos referimos a la «ocupación» o a la función que desempeña cierto individuo dentro de una comunidad. El fenómeno de la formación de nichos es común en la biología. Los nichos imponen restricciones en el apareamiento o



cruza de los individuos y de esta forma promueven la diferenciación de las especies. Por ejemplo, en la naturaleza las islas y los valles aislados son ambientes que han favorecido la evolución de las especies como lo observó Darwin en su famoso tratado de la evolución de las especies. Los biólogos, han observado que aunque la diversidad tiende a ser baja en cada isla o valle, esta tiende a ser mucho mayor entre éstas. Cuando los individuos de dichas islas tienen la posibilidad de interactuar con individuos de otras islas, el intercambio de información genética aumenta dada la diversidad entre éstos, permitiendo la evolución del sistema [8].

En los algoritmos evolutivos, el uso de poblaciones distribuidas está basado en la idea de que las poblaciones aisladas propician la especiación (mayor diferenciación genética entre los individuos) [9]. En muchos casos, [9] los algoritmos que utilizan este tipo de poblaciones descentralizadas obtienen un mejor análisis del espacio de búsqueda mejorando el comportamiento numérico, sin mencionar que su de ejecución, en comparación con los algoritmos centralizados, es menores.

La principal idea de los algoritmos evolutivos distribuidos [1][2][3] es subdividir una población aislada de tamaño  $N$ , en  $s$  subpoblaciones de menor tamaño, a las que generalmente nos referimos como islas o nichos, tales que:

$$N = \sum_{i=1}^s n_i$$

Cada una de estas subpoblaciones evoluciona de forma aislada [5] y, a periodos regulares de tiempo[4] (frecuencia) realizan el intercambio o dispersión de información (individuos) con la finalidad de poder introducir diversidad en las subpoblaciones. Dicho intercambio se realiza empleando una topología que define la forma y dirección en la cual las diferentes islas pueden realizar el intercambio de información [10].

En los algoritmos evolutivos distribuidos (a partir de ahora modelos de islas), además de los parámetros propios del algoritmo con el que evoluciona cada subpoblación, como son: tamaño de la población, tipo de operadores de recombinación, etc. utilizan parámetros adicionales para definir su comportamiento, siendo los siguientes los más utilizados [10]:

- Número de subpoblaciones (islas)
- Frecuencia de migración
- Número de individuos migrados
- Criterio de selección de los individuos a ser migrados
- Criterio de selección de los individuos a ser remplazados

### **3.2 Selección de parámetros en algoritmos distribuidos**

Para reducir el número de parámetros que se deben de establecer, Hiroyasu T. et al [10] propone un algoritmo llamado DuDGA (por sus siglas en inglés, algoritmo genético distribuido dual), éste define a las islas con sólo dos individuos, esos dos individuos se utilizan para realizar las operaciones de recombinación y mutación, finalmente, sólo los dos mejores individuos sobreviven para la siguiente generación. Desafortunadamente, la calidad de la solución final así como el rendimiento global de este algoritmo evolutivo, resulta ser muy pobre ya que dependen de una selección adecuada de los individuos iniciales.

En [11], H. Sawai et al, describen un algoritmo llamado PfGA (por sus siglas en inglés, Algoritmo Genético Libre de Parámetros), en éste, el tamaño de la población no se mantiene constante ya que varía en función de la aptitud o fitness de los individuos. Este algoritmo tiene la característica de no necesitar los parámetros clásicos de los algoritmos genéticos y siempre mantiene poblaciones pequeñas lo cual ocasiona un alto rendimiento.

Aún cuando el propio PfGA no requiere ningún parámetro, la capa de ejecución en paralelo ,provoca que la migración juege un papel esencial como herramienta para mejorar la búsqueda global. Lo anterior se muestra en [12], sin embargo, los autores demostraron que el modelo de migración óptima es diferente para cada problema.

El mismo tipo de población reducida (dos individuos) se utiliza en [13], donde los autores introducen un algoritmo llamado VIGA (por sus siglas en inglés, Algoritmo Genético de Islas Variables). Este algoritmo trabaja con un número dinámico de islas y un mecanismo explícito para cambiar el número de estas. Aunque los resultados de VIGA muestran que es computacionalmente eficiente, se basan en una topología de interconexión completamente conectada, lo cual satura el intercambio de individuos y tiene como consecuencia que se disminuya peligrosamente la diversidad de las poblaciones de las islas aumentando de esta forma el riesgo de una convergencia prematura.

Durante los últimos años, se han realizado una gran cantidad de experimentos para ilustrar los efectos específicos y la importancia que tienen los parámetros utilizados para la ejecución de los modelos de islas. Los principales parámetros utilizados para dichos estudios son [14]:

- Cantidad de individuos migrados
- Frecuencia de migración
- Estrategia de selección de los individuos a ser migrados
- Número de islas
- Tamaño de las poblaciones de las islas

En [1][2][3][15] se propone un modelo de migración el cual establece, que para tener un intercambio eficiente de información genética, en lugar de migrar individuos como se realizaría de forma clásica, se debe iniciar con un mecanismo de extracción de

esquemas, dichos esquemas son enviados al resto de las poblaciones. Cuando los esquemas llegan a las poblaciones destino, se deben de generar “*n*” individuos tomando como base los esquemas recibidos. El número *n* de individuos que se deben de generar está dado por:

$$n = \left[ a * \frac{|RAF - \bar{F}|}{RAF} * RSize \right]$$

Dónde:

*RSize* . Es el número de individuos de la isla que recibe los datos

*RAF* . Es el fitness promedio de la isla que recibe los datos

$\bar{F}$  . Es el valor del fitness promedio de la población de donde fue extraído el esquema

*a* . Es una constante tal que

$$0 < a * \frac{|RAF - \bar{F}|}{RAF} < 1$$

En esa investigación se concluye que el costo de comunicación en el caso de la migración clásica de esquemas se encuentra dado por:

$$T = \left( \frac{g}{MI} - 2 \right) * \sum_{i=1}^m r_i S_i t$$

Donde

*g* . Es el número total de generaciones

*MI* . Es la frecuencia de migración

- $r_i$ . Es la velocidad de migración
- $s_i$ . Es el tamaño de la  $i$ -ésima subpoblación
- $t$ . Es el costo de comunicación para migrar un solo individuo
- $m$ . Es el número de subpoblaciones

Después de aplicar el criterio propuesto por los autores, se concluye que el costo de comunicación del método propuesto está dado por:

$$T = \left( \frac{g}{MI} - 2 \right) * mt$$

### **3.3 Impacto de la migración en los algoritmos evolutivos**

En [16] se estudia el beneficio de la migración en los algoritmos evolutivos distribuidos, en ese trabajo se concluye después de realizar un análisis riguroso de los modelos de islas en las cuales las fases de evolución son independientes y las fases de comunicación son esenciales, que un modelo de islas con un sistema de migración simple, encuentra el óptimo global en un tiempo polinomial mientras que un modelo de islas simple sin migración, requiere un tiempo exponencial.

En [17] se estudia la influencia de la cantidad de individuos migrados así como los intervalos de migración en los modelos de islas. Una de las observaciones más sorprendentes que realizan los autores durante sus experimentos, es que el intervalo de migración parece ser un factor dominante, mientras que el número de individuos juega un papel de menor importancia. En sus experimentos observaron que los intervalos muy

frecuentes de migración, causan que unas islas dominen a otras ocasionando la pérdida de diversidad global antes de que estas puedan intercambiar información para producir mejores soluciones. También se observó que con intervalos menores de migración se obtienen impactos más significantes en el modelo de islas. El mejor resultado se obtiene cuando se realizan migraciones de forma moderada con pocos individuos.

Los autores concluyen que las migraciones frecuentes deben de evitarse debido a que ellas hacen que el intercambio de individuos entre las islas sea muy rápido por lo que las islas terminarían compartiendo a los mismos individuos, lo cual ocasiona la pérdida de diversidad, misma que puede ocasionar una convergencia prematura.

En [18] se estudian los efectos de la migración jerárquica en los algoritmos genéticos paralelos distribuidos, en este, se describen los efectos de los métodos de migración con diferentes esquemas de arquitectura. Se observa, que la arquitectura uniformemente distribuida es mejor que la arquitectura maestro esclavo y el método de migración directa es mejor que el método jerárquico.

En [19] se estudia la elaboración de políticas de migración adaptativas para algoritmos genéticos distribuidos, en ese trabajo se modela a los algoritmos genéticos distribuidos a través de un conjunto de agentes que lo encapsulan como una unidad autónoma que debe mantener en todo momento un conocimiento de la búsqueda. El conjunto de agentes creados, se deben de coordinar entre sí a través de un conjunto de reglas, que pueden ser fijadas a priori o pueden variar en el tiempo. Las políticas de migración se establecen a partir de tres tipos de mecanismos:

- Conciencia social
- Retroalimentación del ambiente
- Orientadas por memoria

Para evaluar las mejoras en los algoritmos genéticos distribuidos, los autores realizaron un conjunto de simulaciones donde ratificaron que la eficiencia del algoritmo aumenta cuando se utilizan políticas de migración dinámicas en lugar de estáticas, ya que éstas emplean el conocimiento de los diversos agentes, para adaptar los parámetros tales como frecuencia y cantidad de individuos migrados.

### **3.4 Topologías de comunicación en los algoritmos evolutivos distribuidos**

Una propiedad importante en los algoritmos genéticos distribuidos es la topología de migración [3], la cual determina la correlación entre las subpoblaciones y la forma en como las buenas soluciones se propagan a través del sistema completo. En un esquema clásico de migración, los intercambios de individuos solo se pueden realizar entre los vecinos que se encuentran restringidos por una determinada topología de conexión [20].

En el caso de contar con una topología completa de conexión entre subpoblaciones y si no se cuenta con ninguna restricción o estrategia de migración, la convergencia del algoritmo tiende a ser prematura debido a la gran cantidad de información intercambiada y la consecuente pérdida de diversidad en las poblaciones [21]. En el caso de contar con una topología de migración clásica como es la topología de anillo, entonces el intercambio de información se vuelve lento lo cual puede ocasionar que no se migren los individuos en los momentos adecuados para poder obtener beneficios al momento del intercambio de información [20]. La pregunta que queda abierta entonces es: ¿Cuál es la topología correcta?, ¿Cuál es la cantidad de individuos que se deben migrar? ¿Cuál es la frecuencia de migración conveniente?

En [21] los autores muestran un nuevo algoritmo evolutivo multiobjetivo llamado SMPGA (por sus siglas en inglés Algoritmo Genético Paralelo con Migración Selectiva), en dicho trabajo se diseña una nueva estrategia de migración basada en una red adaptativa. En el SMPGA la población llamada de búsqueda (PB) y una población denominada de elite (PE) evolucionan al mismo tiempo. Al término de cada ciclo evolutivo, SMPGA decide si un individuo no dominado de la población PB puede ser migrado o no a PE. Si el número de individuos migrados de PB a PE es cero, después de varias generaciones, se remplazan los peores individuos (10%) de PB con nuevos individuos generados de forma aleatoria. Al final, se selecciona el mejor individuo de PE. Los autores concluyeron que esta nueva estrategia de migración ayuda a mantener buenos tiempos de convergencia así como a no perder la diversidad en la población tan rápido.

Tomando como base la noción biológica de la evolución en aislamiento y el intercambio de información en los algoritmos genéticos paralelos, Los autores en [22] proponen un nuevo algoritmo denominado CMBMGA por sus siglas en inglés (Algoritmo Genético Multipoblacional Basado en Migración Caótica). En este algoritmo, la migración de individuos es asíncrona y es guiada por una secuencia denominada “migración caótica”. Según los autores, la información intercambiada entre las poblaciones es eficiente debido a que la secuencia es caótica. El algoritmo es aplicado a un problema de inventarios y muestra ser superior a los algoritmos clásicos en cuanto a su habilidad para no quedar atrapado en óptimos locales.

### **3.5 Algoritmos evolutivos celulares**

El primer cGA del que se tiene conocimiento fue propuesto por Robertson en 1987 [22] y fue aplicado a un sistema de clasificación. La idea original de Robertson, era implementar de forma paralela (utilizando una computadora CM1), un algoritmo genético en el cual cada



individuo estuviera interconectado con otros individuos formando **vecindarios**. Los resultados fueron alentadores debido a que el cGA propuesto arrojó mejores resultados que el AG clásico.

Collins y Jefferson en 1991 [25], caracterizaron la diferencia entre los GA y los cGA de acuerdo con factores como la diversidad del genotipo, fenotipo así como la velocidad de convergencia. En su trabajo, los autores concluyeron que la búsqueda local realizada por el cGA es más apropiada para la evolución artificial que la realizada por el AG con población centralizada.

Ese mismo año, Davidor [27], desarrolló un estudio sobre los cGA con mallas bidimensionales como mecanismo para la **estructuración de la población** utilizando vecindarios con 8 individuos. En dicho estudio, los padres fueron seleccionados utilizando un método de selección proporcional, para después de realizar su recombinación, reinsertar a los descendientes en el vecindario, con una probabilidad basada en su valor de fitness. Davidor concluyó, que el cGA mostraba una velocidad de convergencia mayor que el GA clásico al emplear dichos mecanismos.

En 1994, Gordon et al. [28] estudiaron el comportamiento del cGA utilizando diferentes tipos de vecindarios. Gordon aplicó el cGA a diversos problemas de optimización continuos y discretos. En su trabajo, concluyó que los vecindarios con radios grandes trabajan mejor con problemas simples sin embargo, en el caso de problemas complejos, es mejor el uso de vecindarios de radio pequeño.

En 2002, Eklund [28] realizó un estudio empírico para determinar cuál es el método de selección apropiado así como la forma y tamaño óptimo para el vecindario empleado por los cGAs. Eklund concluyó, que el tamaño y forma del vecindario dependen directamente del tamaño de la población, mientras que para el método de selección, mostró

que todos los métodos se comportan bien siempre y cuando, se utilice elitismo para conservar a las mejores soluciones.

Finalmente, ese mismo año, Alba et al. [28] realizó un estudio comparativo sobre el comportamiento de los cGA con políticas de actualización de los individuos síncronas y asíncronas. Los resultados obtenidos, muestran que los cGA asíncronos tienen una presión de selección mayor que los cGA síncronos de tal forma que su convergencia es más rápida. En dicho estudio, también se observó que las soluciones aparecen más rápidamente en los problemas menos complejos, mientras que en los problemas más complicados, los cGA síncronos muestran una mejor convergencia sobre los modelos asíncronos.

Como se verá más adelante en los resultados, el comportamiento del algoritmo se ve fuertemente afectado por la política de actualización de los individuos además del tamaño y la forma del vecindario.

# Modelo de fusión de poblaciones

# 4

Como ya se comentó en el capítulo 2, un programa evolutivo es un algoritmo estocástico que mantiene una población de individuos,  $P(t) = \{x_1^t, x_2^t, \dots, x_n^t\}$  donde cada uno de estos, representa una solución potencial al problema que se desea resolver, y es representado mediante alguna estructura de datos  $S$ . Cada solución  $x_i^t$  es evaluada para obtener alguna medida de su aptitud. Una nueva población es formada seleccionando a los mejores individuos de la iteración  $t$ , algunos miembros de la nueva población se someterán a transformaciones con operadores genéticos (como la cruce y la mutación), para formar finalmente nuevas soluciones.

La mayoría de los algoritmos evolutivos, tradicionalmente usan un esquema centralizado con respecto al manejo de la población (panmixia), donde aplican operadores de recombinación y selección para la formación de la nueva población. Sin embargo, el uso de algoritmos con poblaciones descentralizadas es decir, aquellos algoritmos donde la población tiene alguna estructuración con respecto a la distribución de los individuos (subdivisión de la población), ha ido cobrando cada vez más popularidad entre los investigadores y dadas las facilidades actuales gracias al acelerado crecimiento y bajos costos en la producción de hardware multinúcleo (CPU y GPU), ha llevado a los investigadores al desarrollo de este tipo de algoritmos.

Nuevamente, también como ya se mencionó en el capítulo 2, el fenómeno de la formación de nichos es común en la biología. Los nichos imponen restricciones en el apareamiento o recombinación de los individuos y de esta forma, promueven la diferenciación de las especies. En el caso de los algoritmos evolutivos descentralizados, el uso de dichas poblaciones distribuidas ha inspirado la idea de que las poblaciones aisladas propician la especiación (mayor diferenciación genética entre los individuos) [1]. En algunos casos [2] los algoritmos que utilizan este tipo de poblaciones descentralizadas, consiguen un mejor análisis del espacio de búsqueda mejorando con ello, la calidad de las soluciones encontradas sin mencionar, que los tiempos de ejecución en comparación con los algoritmos centralizados son menores.

## **4.1 Modelo de fusión de poblaciones**

Es un hecho bien conocido, que la calidad de la solución encontrada por cualquier EA depende directamente del valor de sus parámetros lo que consigue un adecuado balance entre exploración y explotación. Como se comentó en el apartado anterior, los algoritmos genéticos distribuidos (también conocidos como modelos de islas) requieren para su operación (además de los parámetros propios de la heurística que corre dentro de la isla) de los siguientes parámetros:

- Tamaño de la población de las islas.
- Topología de interconexión.
- Frecuencia de migración.

Además de los parámetros indicados, existe la necesidad de establecer una política de migración es decir, determinar el número y la selección de los individuos que migran sin dejar de lado por supuesto la política de remplazo es decir, determinar que individuos

serán substituidos por los individuos provenientes de otras islas (migrantes). El modelo de fusión de poblaciones propuesto en este trabajo es definido por la tupla mostrada en la Ecuación 4.1 y el significado de sus parámetros se encuentra indicado en la Tabla 4.1.

$$mfp = (n, nk, m, f_{island}, f_{population}) \quad (4.1)$$

<b>Parámetro</b>	<b>Significado semántico</b>
$n$	Tamaño de la población de la isla
$nk$	Número inicial de islas
$m$	Frecuencia de fusión
$f_{island}$	Criterio de selección de las islas a ser fusionadas
$f_{population}$	Criterio de selección de individuos después de la fusión

**Tabla 4.1:** Parámetros del modelo de islas

El proceso de migración, es definido por la tupla de la Ecuación 4.2 y el significado de sus parámetros se encuentra indicado en la Tabla 4.2.

$$M = (n, f_m, P_s, P_r) \quad (4.2)$$

<b>Parámetro</b>	<b>Significado semántico</b>
$n$	Número de individuos a migrar
$f_m$	Frecuencia de migración
$P_s$	Política de selección
$P_r$	Política de remplazo

**Tabla 4.2:** Parámetros del proceso de migración

Como se estudió en el estado del arte, claramente se puede observar que el modelo de fusión de poblaciones propuesto, es una modificación al modelo clásico, donde se substituye el proceso de migración por el proceso de fusión y ello implica, que el modelo propuesto no requiera de una topología específica para realizar la fusión (lo cual es una de las principales contribuciones del modelo de fusión de islas), en lugar de ello, se vale de diversos mecanismos para la selección de las islas que serán fusionadas y de los cuales se hablara más adelante.

Lo más relevante hasta este punto, está relacionado con el hecho de que como ya se mostró en el estado del arte, la selección del tipo de topología empleado para la comunicación de las islas, además de no ser una tarea simple, también impacta directamente tanto en la velocidad de convergencia del algoritmo como en la calidad de los resultados obtenidos. En el Algoritmo 4.1, se muestra el pseudocódigo del modelo de fusión propuesto.

---

**Algoritmo 4.1:** Pseudocódigo del modelo de fusión

---

```

1   $P = \text{CreatePopulation}(NP)$ 
2   $K = \text{CreateIslands}(P, nk, n)$ 
3  while ( $|K| > 1$ ) do
4      for each  $k_i$  in  $K$  do in parallel
5           $\text{Evol}(k_i)$ 
6      end for
7       $\text{Island}_{\text{Sel}} = f_{\text{island}}(K)$ 
8       $K = K - \text{Island}_{\text{Sel}}$ 
9       $k_{\text{temp}} = \text{Join}(\text{Island}_{\text{Sel}})$ 
10      $k_{\text{temp}} = f_{\text{population}}(k_{\text{temp}})$ 
11      $K = K + k_{\text{temp}}$ 
12
13 done
14  $\text{Evol}(K)$ 

```

---

Como se observa en el Algoritmo 4.1, en la línea 1, se crea una población centralizada compuesta por  $NP$  individuos, posteriormente (línea 2), dicha población es dividida en  $nk$  subpoblaciones con  $n$  individuos cada una y se debe cumplir con la igualdad de la Ecuación 4.3.

$$P = \bigcup_{i=1}^{nk} k_i, \quad k_1 \neq k_2 \neq \dots \neq k_{nk} \quad (4.3)$$

Lo que la ecuación 4.3 significa, es que todos los individuos deberán de ser tomados en cuenta y solo podrán pertenecer a una isla a la vez. Posteriormente, ya que se han creado las subpoblaciones (línea 2) también llamadas islas, se realiza la evolución en paralelo de cada una de estas (línea 4). Cabe aclarar que la función *Evol* no se encuentra limitada a algún tipo de meta heurística en particular, por lo tanto, esta función puede emplear cualquier mecanismo para la evolución como por ejemplo: algoritmos genéticos, estrategias evolutivas, evolución diferencial, optimización por cúmulos de partículas (PSO), etc. Una vez que todas las islas han pasado por el proceso de evolución, deben ser seleccionadas aquellas islas (línea 8) que serán fusionadas (línea 10), una vez realizada la fusión se seleccionan aquellos individuos que permanecerán en la nueva isla (línea 11). Finalmente, se actualiza el conjunto de islas del modelo (línea 12). Todo el proceso descrito, se realiza mientras el número de islas sea mayor a 1 (línea 3) para evolucionar por última vez a la población resultante (línea 15).

Una característica importante del modelo de fusión expuesta en el Algoritmo 4.1 y que forma parte de las contribuciones, radica en que, debido a que el número de islas disminuye de forma monótona durante la ejecución del algoritmo, también lo hace el tamaño de la población global (producto de la suma de todos los individuos de las poblaciones locales),

por lo que en consecuencia, también disminuye número de evaluaciones de la función de aptitud.

En aquellos casos donde la función de aptitud tiene una complejidad considerable, la disminución del número de evaluaciones conseguido impacta directamente en el tiempo de ejecución del algoritmo es decir, además del beneficio implícito que se pueda tener gracias a la implementación en paralelo del algoritmo ya sea utilizando múltiples núcleos de CPU o GPU se tiene un ahorro por la disminución de veces que se tiene que realizar la evaluación de la función de aptitud.

En el algoritmo genético distribuido clásico, el número de islas y de individuos se mantiene constante por lo que la función de aptitud se evalúa con exactitud  $g * |k_i| * nk$  veces, donde  $g$  es el número total de generaciones que el algoritmo evoluciona,  $|k_i|$  es el tamaño de la población de la isla y  $nk$  es el número total de islas. En contraste, el modelo propuesto, comienza con el mismo tamaño de la población en cada isla, pero lo mantiene solo durante  $m$  generaciones. Cuando el proceso de fusión se lleva a cabo, la población de todas las islas seleccionadas por  $f_{island}$  se fusionan y luego se reduce su tamaño, seleccionando solo aquellos individuos que cumplan con el criterio de selección establecido por  $f_{population}$ .

Cabe destacar que el modelo de fusión propuesto, cuenta con la posibilidad de poder seleccionar las islas a ser fusionadas, lo que permite en consecuencia, poder modificar la forma en como el espacio de búsqueda es explorado a diferencia de lo que ocurre en el método clásico basado en topología, donde es imposible controlar la forma en la cual las soluciones se dispersan ya que es la misma topología la que define el orden en el cual se pueden intercambiar individuos.

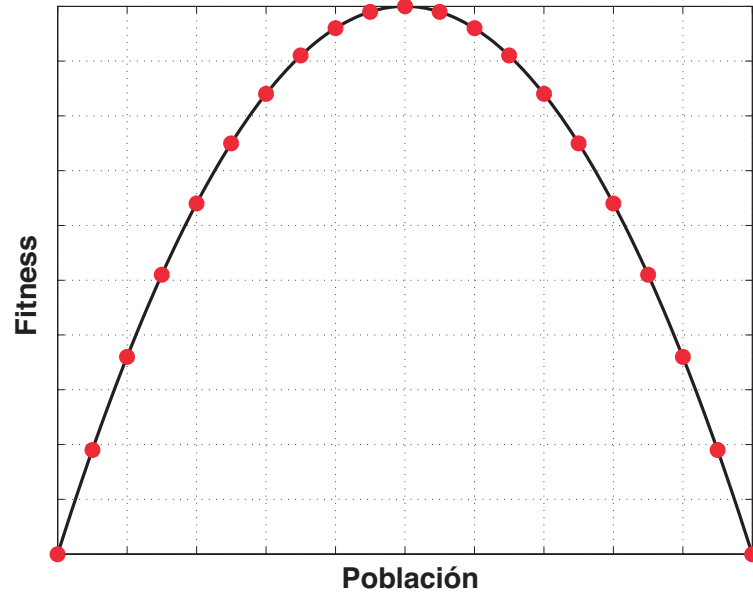


## 4.2 Exploración vs explotación

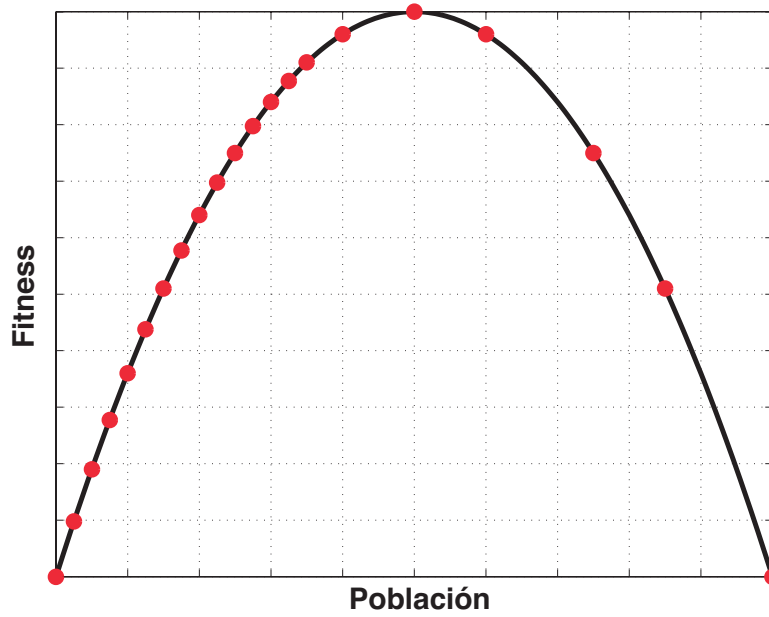
Con la finalidad de realizar un análisis de los mecanismos de exploración y explotación sobre los modelos propuestos, se ha recurrido a una medida que permita obtener la cantidad de información que aporta una determinada población al proceso de búsqueda, dicha medida es obtenida a través del calculo de la entropía de la población (Ecuación 4.4).

$$H(Pob) = \sum_{i=1}^{[Pob]} P(Pob_i) \text{Log}_2 \frac{1}{P(Pob_i)} \quad (4.4)$$

Para mostrar el efecto que tiene la entropía de la población empleada por un algoritmo evolutivo observemos las figuras 4.1 y 4.2. Como se puede observar en la figura 4.1, los individuos de la población, se encuentran dispersos de forma tal, que mapean de mejor forma a la función cuyo valor óptimo se desea obtener a diferencia de lo que ocurre en la figura 4.2, donde los individuos solo mapean efectivamente parte de la función por lo que en términos de entropía de la población se observa que la entropía de la población mostrada en la figura 4.1 es mayor a la entropía de la población mostrada en la figura 4.2.

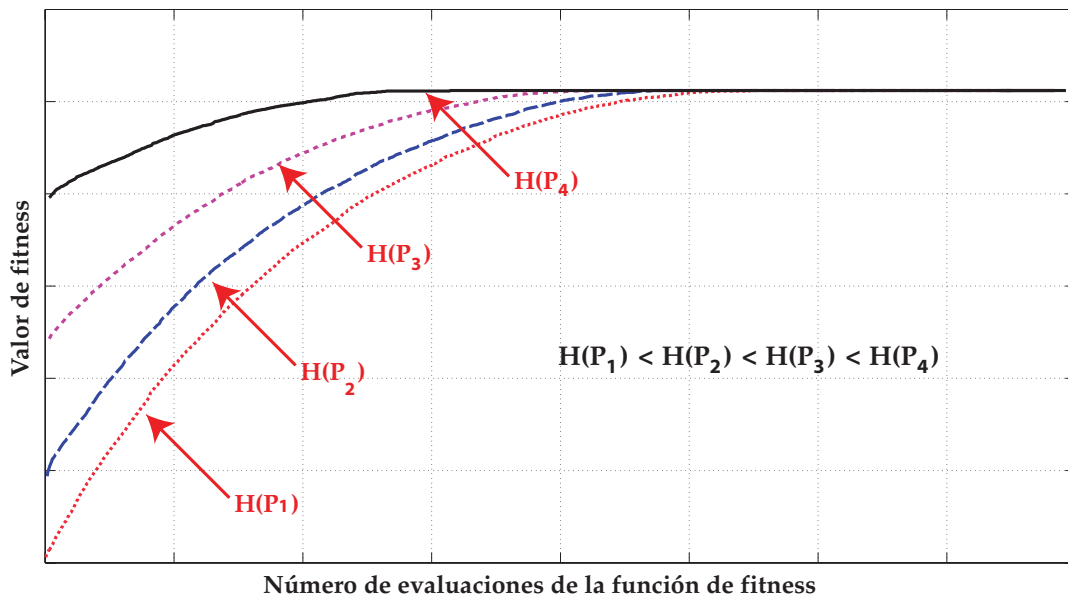


**Figura 4.1:** Población distribuida uniformemente



**Figura 4.2:** Población no distribuida uniformemente

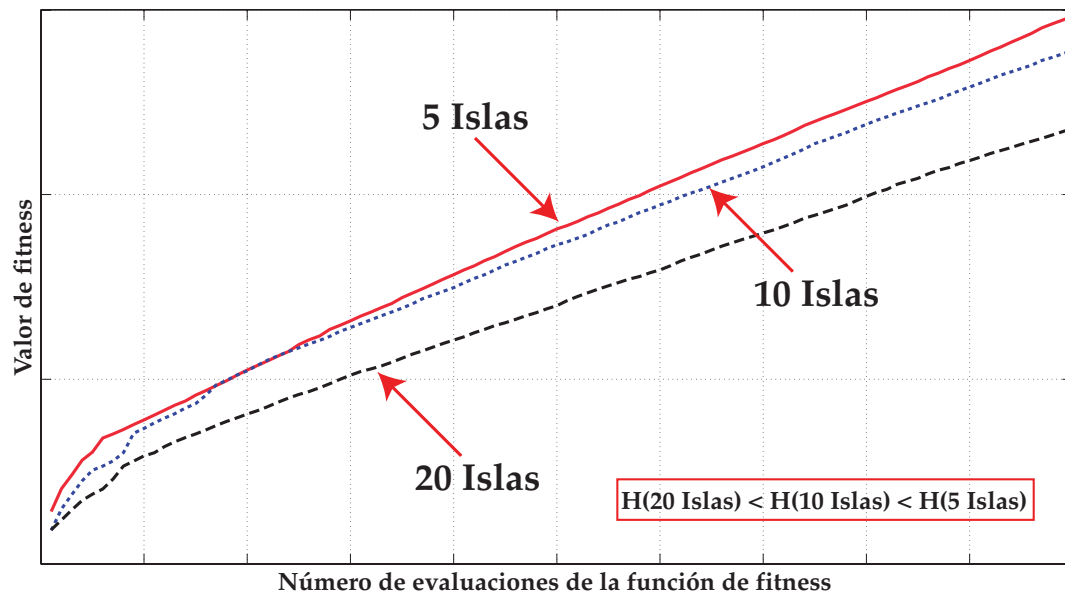
En la figura 4.3, se muestra el comportamiento de un algoritmo evolutivo cuyas condiciones iniciales son las mismas, es decir, las probabilidades de cruce y mutación así como semillas de los generadores aleatorios y todos los parámetros necesarios para su operación son los mismos, sin embargo, como se observa, el comportamiento de los algoritmos es distinto lo cual es debido a la entropía de la población inicial que es utilizada por el algoritmo.



**Figura 4.3:** Algoritmo evolutivo con población inicial con entropía distinta

Como se observa en la figura 4.3, a mayor entropía en la población inicial del algoritmo evolutivo, este muestra una mayor explotación del espacio de búsqueda, y a menor entropía en la población inicial el algoritmo disminuye la intensificación durante el proceso de búsqueda incrementando la exploración de este. Tomando como base los resultados anteriores, se puede concluir que para mantener el balance entre la exploración y explotación del algoritmo, se debe de mantener una entropía media en la población.

Para poder entender los efectos que tienen tanto la topología como el número de migrantes durante el proceso de búsqueda en los modelos basados en islas, se realizaron una serie de experimentos que a partir, de la entropía de las poblaciones, permiten destacar los procesos de exploración y explotación en este tipo de algoritmos.

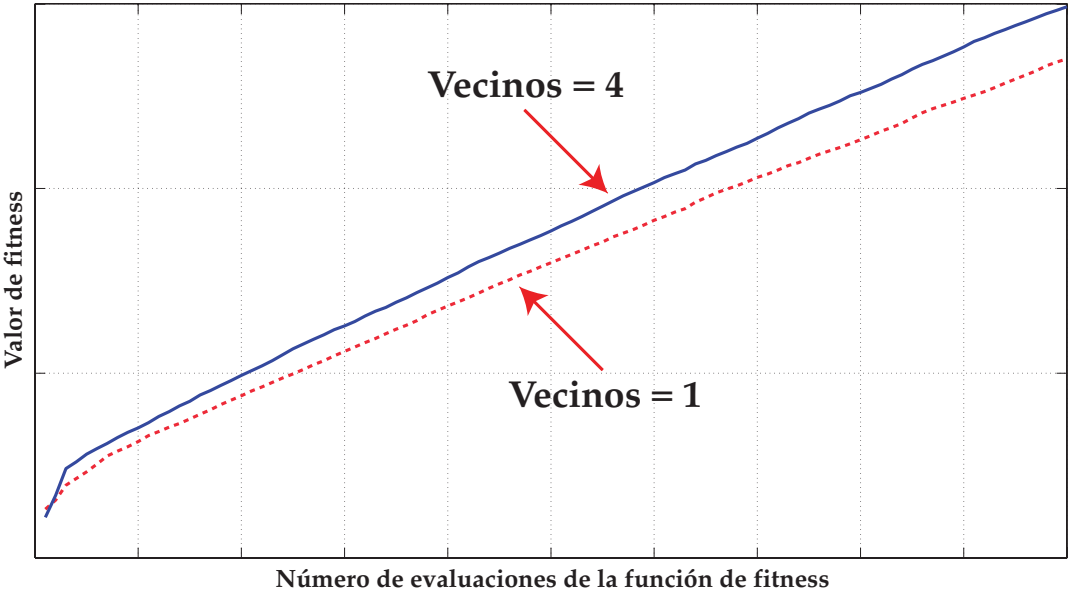


**Figura 4.4:** Exploración vs explotación en función del número de islas

Como se puede observar en la figura 4.4, existe una diferencia clara durante el proceso de búsqueda, cuando el número de islas empleado aumenta o disminuye. Como se puede observar, conforme el número de islas aumenta, la entropía de la población en estas disminuye, lo cual en principio es debido a la pérdida de diversidad dentro de la población y por lo tanto, se produce una intensificación del proceso de búsqueda aumentando la probabilidad de que el algoritmo no converja sin embargo, cuando el número de islas disminuye, la entropía de las poblaciones aumenta, produciendo un incremento en el proceso de exploración del espacio lo cual puede derivar en una convergencia prematura

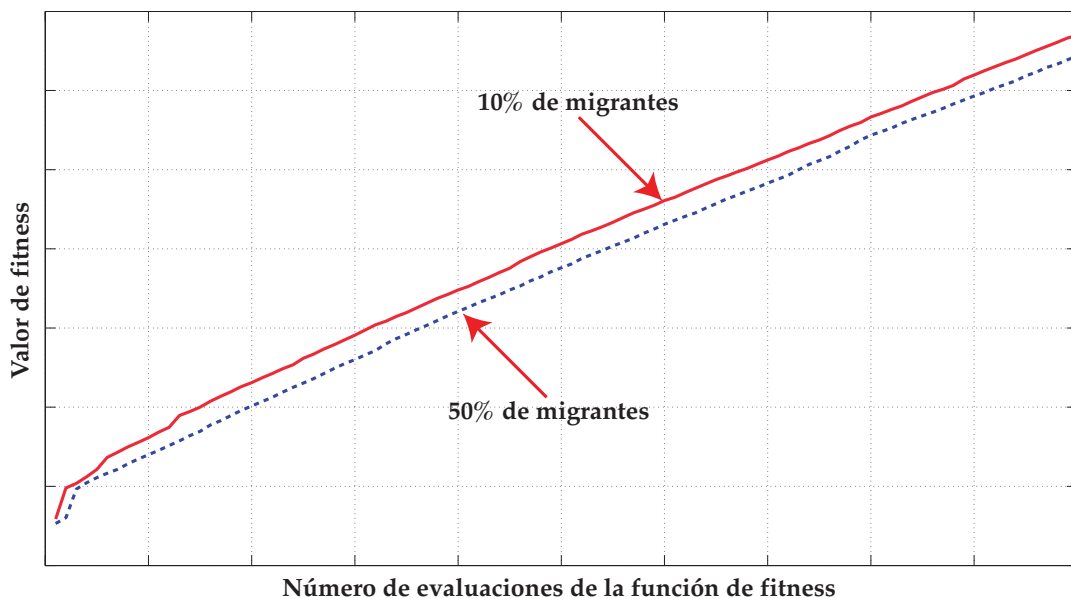
del algoritmo. En la figura 4.4, se concluye que el balance entre exploración y explotación se consigue cuando el número de islas empleadas es aproximadamente de 10.

Para analizar el efecto de la topología sobre el proceso de búsqueda en los modelos de islas, se realizó un conjunto de experimentos donde se varió el número de vecinos de cada una de las islas involucradas en el modelo. Como se puede observar e la figura 4.5, al aumentar el radio de comunicación (disminuyendo el número de vecinos) la propagación de las soluciones es mas lenta, lo cual promueve la explotación del espacio de búsqueda, mientras que al incrementar el número de islas, el radio disminuye, incrementando la comunicación entre las poblaciones y al mismo tiempo las soluciones son compartidas de forma más rápida promoviendo el proceso de exploración sin embargo, esto último puede generar una convergencia prematura sobre todo cuando la complejidad del problema aumenta.



**Figura 4.5:** Exploración vs explotación en función del número de vecinos

Finalmente, para determinar el impacto sobre los procesos de exploración y explotación al variar el número de migrantes entre islas, se realizaron un conjunto de experimentos que permitieron analizar el efecto de dichas migraciones. Como se puede observar en a figura 4.6, al aumentar el número de migrantes entre las poblaciones, la entropía de estas disminuye y como consecuencia la perdida de su diversidad lo cual promueve la explotación del espacio de búsqueda sin embargo, esto puede ocasionar que el algoritmo no sea capaz de converger hacia una solución optima, por otro lado, cuando el número de migrantes disminuye, la entropía de la población no se ve afectada lo cual y al compartir nuevas soluciones, fortalece el proceso de exploración lo cual incrementa la probabilidad de una convergencia prematura. Como resultado de los experimentos realizados, se llego a la conclusión, de que el número de migrantes intercambiados debe ser mínimo cuando el número de islas es grande para asi poder promover el balance entre la exploración y la explotación realizadas por el algoritmo.



**Figura 4.6:** Exploración vs explotación en función del número de migrantes

### 4.3 Experimentos y resultados

Para probar el rendimiento del modelo propuesto, se utilizo un subconjunto de 10 funciones del portafolio propuesto durante la CEC2013. El conjunto completo de funciones junto con una especificación de su intervalo de búsqueda y su valor optimo conocido se muestran en la tabla 4.3.

Etiqueta	Nombre	Función	Óptimo
$f_1$	Sphere	$f_1(x) = \sum_{i=1}^{30} x_i^2$ $-100 \leq x_i \leq 100$	$\min(f_1) = f_1(0, \dots, 0) = 0$
$f_2$	Schwefel's problem 1	$f_2(x) = \sum_{i=1}^{30}  x_i  + \prod_{i=1}^{30}  x_i $ $-10 \leq x_i \leq 10$	$\min(f_2) = f_2(0, \dots, 0) = 0$
$f_3$	Schwefel's problem 2	$f_3(x) = \sum_{i=1}^{30} \left( \sum_{j=1}^i x_j \right)^2$ $-100 \leq x_i \leq 100$	$\min(f_3) = f_3(0, \dots, 0) = 0$
$f_4$	Schwefel's problem 3	$f_4(x) = \max_i \{ x_i , 1 \leq i \leq 30\}$ $-100 \leq x_i \leq 100$	$\min(f_4) = f_4(0, \dots, 0) = 0$
$f_5$	Generalized Rosenbrock's	$f_5(x) = \sum_{i=1}^{29} \left  100(x_{i+1} - x_i^2) + (x_i - 1)^2 \right $ $-30 \leq x_i \leq 30$	$\min(f_5) = f_5(0, \dots, 0) = 0$
$f_6$	Step	$f_6(x) = \sum_{i=1}^{30} ( x_i + 0.5 )^2$ $-100 \leq x_i \leq 100$	$\min(f_6) = f_6(1, \dots, 1) = 0$
$f_7$	Quartic with noise	$f_7(x) = \sum_{i=1}^{30} ix_i^4 + \text{rand}[0,1]$ $-1.28 \leq x_i \leq 1.28$	$\min(f_7) = f_7(1, \dots, 1) = 0$
$f_8$	Generalized Schwefel's problem	$f_8(x) = \sum_{i=1}^{30} \left( x_i \sin(\sqrt{ x_i }) \right)$ $-500 \leq x_i \leq 500$	$\min(f_7)$ $= f_7(420.96, \dots, 420.96)$ $= -12569.5$

$f_9$	Generalized Rastrigin's	$f_9(x) = \sum_{i=1}^{30} [x_i^2 - 10 \cos(2\pi x_i) + 10]$ $-5.12 \leq x_i \leq 5.12$	$\min(f_9) = f_9(0, \dots, 0) = 0$
$f_{10}$	Generalized Griewank's	$f_{10}(x) = \frac{1}{4000} \sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ $-600 \leq x_i \leq 600$	$\min(f_{10}) = f_{10}(0, \dots, 0) = 0$

**Tabla 4.3:** Subconjunto de funciones del CEC2013

Para poder comparar el rendimiento del método de intercambio de individuos clásico contra el propuesto se realizaron los experimentos como se muestra en la tabla 4.4. Nota: Cada ejecución utilizo exactamente la misma población inicial en ambos modelos con el objetivo de que los resultados puedan compararse de forma imparcial.

<b>Número de funciones utilizadas</b>	10
<b>Número de ejecuciones del algoritmo</b>	100
<b>Número de modelos probados</b>	2
<b>Total</b>	2000

**Tabla 4.4:** Número de experimentos realizados

Como se puede observar en la tabla 4.4 el número total de experimentos que se realizaron para poder probar el modelo fueron 2000. En el caso del algoritmo genético distribuido, la topología empleada fue la de anillo debido a que es la más reportada en el estado del arte, para el proceso de migración, se utilizaron los mejores individuos los cuales remplazaron a los peores individuos en la isla destino.



Para el modelo propuesto, el criterio de selección de islas  $f_{island}$  fue aleatorio, el número de islas fusionadas fue de dos y el criterio para seleccionar la población resultante después de la fusión de islas  $f_{population}$  consintió en tomar las mejores dos terceras partes de la población resultante después de la fusión. La Tabla 4.5, muestra los parámetros utilizados por las heurísticas en los experimentos.

<b>Parámetro</b>	<b>AG Distribuido</b>	<b>Fusión</b>
Tamaño de la población global	1000	1000
Tamaño de la población por isla	125	Dinámico
Generaciones por isla	100	100
Generaciones globales	10	Dinámico
Probabilidad de cruce	0.8	0.8
Probabilidad de mutación	0.01	0.01
Número de islas (núcleos)	8	Dinámico
Método de selección de islas	Topología	Aleatorio
Elitismo	10	Dinámico
Número de ejecuciones por experimento	50	50

**Tabla 4.5:** Número de experimentos realizados

En la Figura 4.7, se muestran los tiempos de ejecución para ambos modelos, como se puede observar, los tiempos de ejecución del modelo propuesto mejoran en todos los casos con respecto a los tiempos obtenidos en el modelo clásico.

En la Figura 4.8, se puede observar claramente, una disminución en el número de veces que se evalúa la función de aptitud, cabe aclarar que si bien el algoritmo tiene como condición de terminación un número específico de generaciones lo cual aumenta el número de evaluaciones de la función de aptitud en el modelo clásico, también queda claro, que si la condición de terminación fuera con base en algún umbral como por ejemplo un error, por

la forma del modelo propuesto, se deriva que el número de evaluaciones también sería menor.

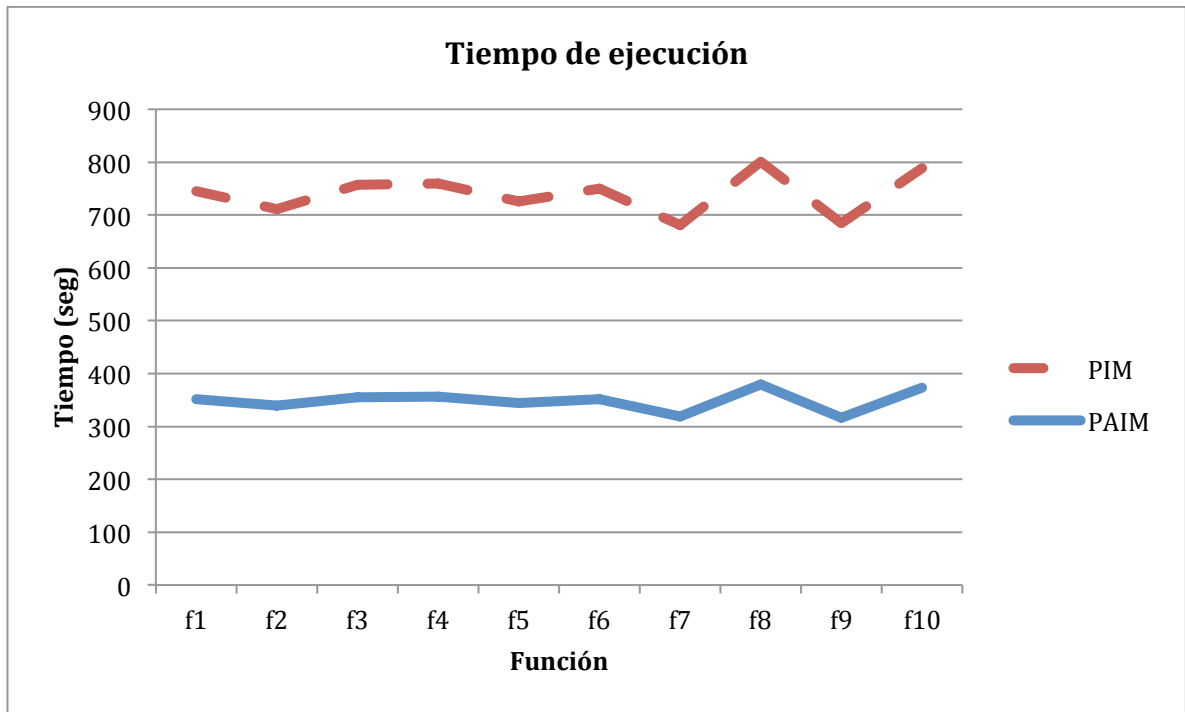
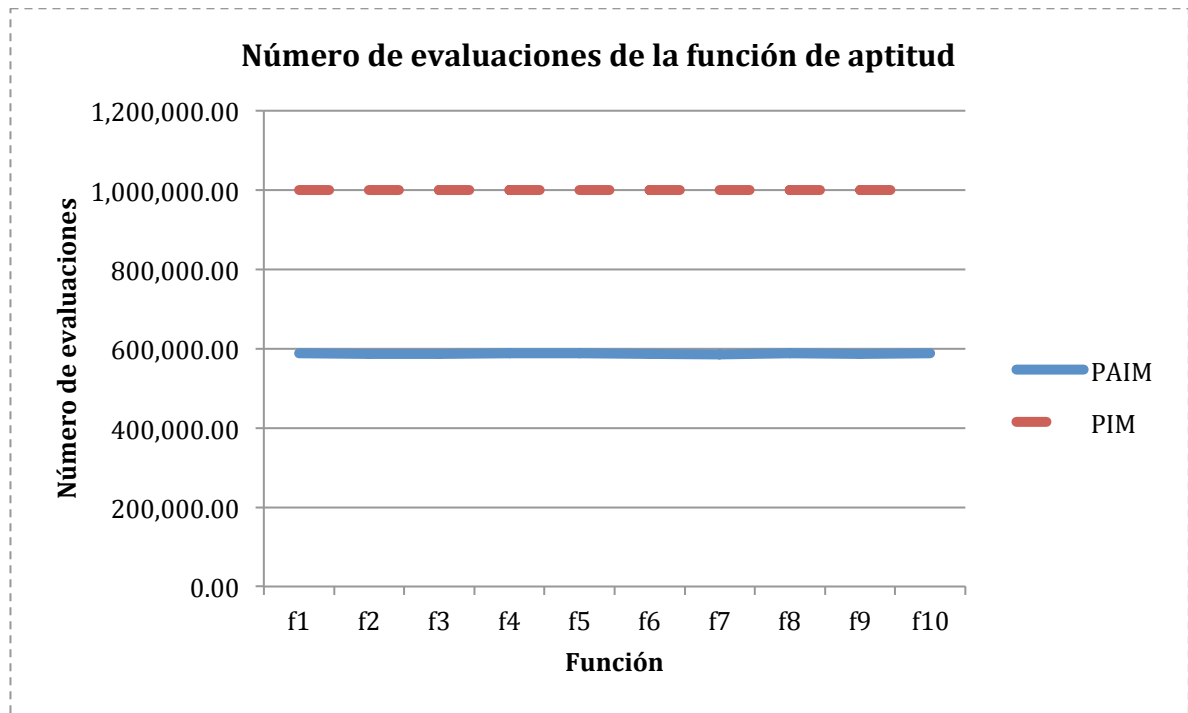


Figura 4.7: Tiempos de ejecución



**Figura 4.8:** Número de evaluaciones de la función de aptitud

La Tabla 4.6, muestra la comparación entre la calidad de las respuestas obtenidas por ambos modelos, como se puede observar, en la mayoría de los casos la calidad de la respuesta obtenida por el modelo propuesto mejora a la calidad del modelo clásico.

Función	PAIM	PIM
$f_1$	0.000003	0.000019
$f_2$	0.000008	0.000017
$f_3$	0.000004	0.00001
$f_4$	0.000007	0.00001
$f_5$	0.000005	0.000008
$f_6$	0.000005	0.000008
$f_7$	0	0

$f_8$	-12568.8972	-12554.2173
$f_9$	0.000002	0.000005
$f_{10}$	0.000003	0.000006

**Figura 4.6:** Soluciones encontradas

## 4.4 Conclusiones parciales

El modelo propuesto denominado *Modelo de Fusión de Islas*, sustituye la operación de migración de individuos utilizada en los modelos de islas clásicos por un sistema de fusión de poblaciones. Dicho sistema de fusión, es empleado como el principal mecanismo de dispersión de soluciones utilizado por el modelo propuesto. De este modo, el modelo de fusión de islas, elimina la necesidad de elegir una topología de interconexión lo que permite entre otras cosas, contar con un abanico de mecanismos que permitan seleccionar de forma objetiva y basada en reglas, las islas que deberán ser seleccionadas para realizar la mezcla de información y por ende la forma en como se explora y explota el espacio de búsqueda.

Las principales ventajas del modelo propuesto son:

- No se requiere de una topología de interconexión
- El número de islas es variable
- El tamaño de la población es variable con un decremento monótono
- El algoritmo puede ser tanto síncrono como asíncrono
- Se disminuye el numero de evaluaciones de la función de aptitud

Como se muestra en los experimentos realizados, el algoritmo propuesto mejora los tiempos de ejecución, esto, es una consecuencia directa de la disminución del número de

veces que se evalúa la función de aptitud, gracias a la disminución sistemática del tamaño de la población (resultado de la fusión y selección de los individuos sobrevivientes).

Otro aspecto importante a destacar, se presenta debido a que al mantener un cambio en los tamaños de las poblaciones de forma dinámica, convierte al modelo en un algoritmo asíncrono lo cual, promueve que el espacio de búsqueda se analice de forma diferente y que como se presenta en el estado del arte, promueve que sea más eficiente.

Finalmente, los resultados muestran que además de mejorar las velocidades de convergencia a partir de los tiempos obtenidos en los experimentos, el modelo propuesto también mejora la calidad de las soluciones encontradas, lo que explica que el balance entre exploración y explotación sea más consistente debido al sistema de reglas y criterios empleados durante la selección de islas e individuos dentro del algoritmo.

## **4.5 Trabajo Futuro**

Hasta el momento, el modelo propuesto solo presenta la posibilidad de tener fusión de islas, sin embargo, se ha considerado que bajo ciertas condiciones (como por ejemplo la pérdida de diversidad en la población) se puedan copiar ciertos mecanismos de otros algoritmos que permitan ya sea reiniciar las poblaciones de las islas o mejor aun dividir las islas y reinicializar el proceso. Otra modificación que se desea trabajar consiste en establecer un sistema de reglas para realizar la fusión variando de esta forma las condiciones bajo las cuales se realizan las diferentes operaciones del algoritmo y que permitan controlar de mejor forma el balance entre la exploración y la explotación.

# Modelo Celular Híbrido

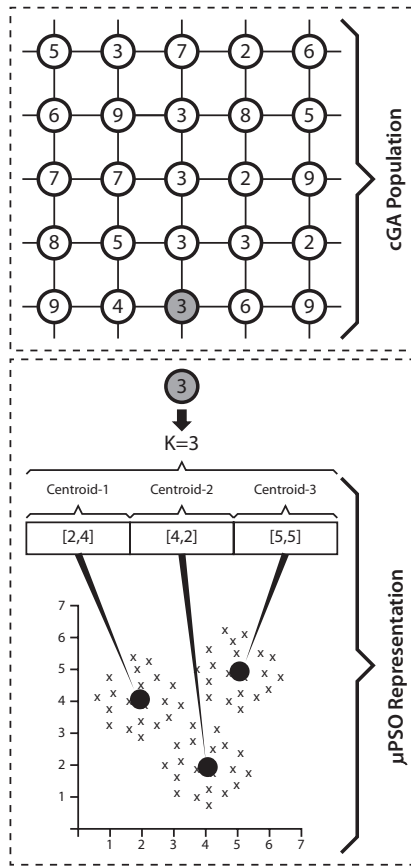
# 5

Típicamente, en la literatura se denomina; Algoritmo Evolutivo Híbrido (hEA por sus siglas en inglés Hybrid Evolutionary Algorithm) [40][60], a todos aquellos algoritmos que combinan dos o mas algoritmos evolutivos (incluidas algunas heurísticas no evolutivas) para la implementación de una nueva meta heurística especializada en la solución de un problema determinado.

En este capítulo, se presenta un hEA con una población espacialmente estructurada (también conocido como algoritmo con población descentralizada), que emplea un cGA como método de exploración y un micro algoritmo [64] de optimización por cumulo de partículas ( $\mu$ PSO) con búsqueda local como método de explotación, capaz de estimar el número de clases presentes en un conjunto de datos.

## 5.1 cGA Híbrido

El algoritmo que se ha desarrollado para estimar el número de clases presentes en un conjunto de datos, está planteado en términos de un hEA compuesto por dos heurísticas: un *cGA* y un  $\mu$ PSOLS. De manera general, el algoritmo está estructurado como se muestra en la Figura 5.1.

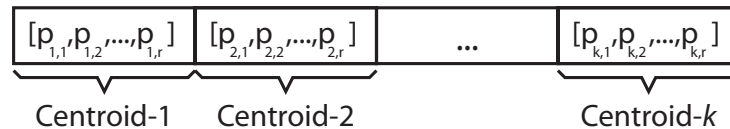


**Figura 5.1:** Algoritmo Propuesto

Como se puede observar en la parte superior de la Figura 5.1, se ha utilizado un *cGA* con una población estructurada en forma de malla completamente conectada (toroide), donde cada individuo representa una posible solución al problema (valor de 'k'). En la parte inferior de la misma figura, podemos observar que el algoritmo emplea un  $\mu$ *PSO* con búsqueda local (LS) para determinar el agrupamiento óptimo de los datos en función del valor del individuo proporcionado por el *cGA*.

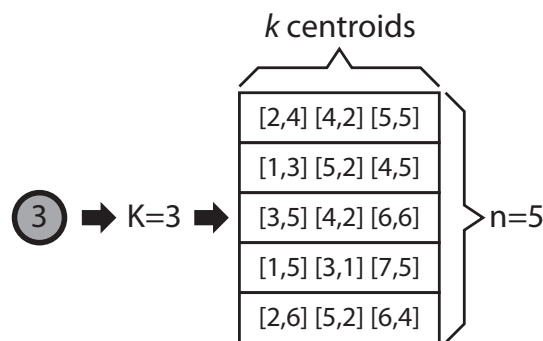
### 5.1.1 Representación

Como se mencionó en el párrafo anterior, el *cGA* utiliza una población estructurada en forma de malla completamente conectada. El valor que pueden tomar los individuos del *cGA*, es un número entero comprendido en el intervalo de búsqueda donde se pretende encontrar el valor de 'k'. Por otra parte, el  $\mu$ PSO utiliza un arreglo de longitud 'k' para representar a cada individuo. Cada una de las posiciones de dicho arreglo, codifica el centroide para cada una de las 'k' clases representadas por el individuo del *cGA* (Figura 5.2).



**Figura 5.2:** Representación de los individuos del  $\mu$ PSO

En la Figura 5.3, se muestra un ejemplo de una población con 5 individuos para el  $\mu$ PSO cuando se tiene como entrada un valor para  $k=3$ .

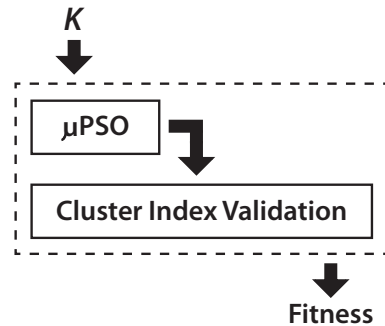


**Figura 5.3:** Ejemplo de población del  $\mu$ PSO



### 5.1.2 Función de aptitud

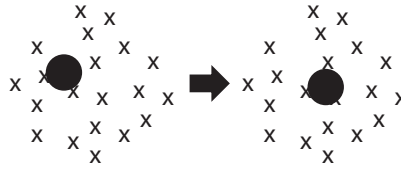
La función de aptitud del algoritmo propuesto, se encuentra representada gráficamente en la Figura 5.4.



**Figura 5.4:** Función de aptitud

Como se puede observar en la figura 5.4, la función de aptitud se encuentra estructurada de la siguiente forma: él  $\mu PSO$  toma como entrada el valor de ' $k$ ' proveniente del  $cGA$ , éste estima una partición para dicho valor. Dicha partición, será utilizada como entrada a un índice de validación de agrupamiento lo que producirá finalmente el valor de aptitud utilizado por el  $cGA$ .

Para obtener un mejor desempeño por parte del  $\mu PSO$ , se ha implementado un método de ajuste local (LA) dentro del proceso evolutivo de dicho algoritmo. Como se puede observar en la parte izquierda de la Figura 5.5, se muestra la representación de un determinado centroide para una clase en particular durante el proceso evolutivo del  $\mu PSO$ , después de realizar el proceso de búsqueda local, el centroide es ajustado (desplazado) dentro de la clase, lo que ocasiona que la velocidad de convergencia del  $\mu PSO$  aumente de forma considerable como se aprecia en la sección de resultados.



**Figura 5.5:** Método de ajuste local

### 5.1.3 Algoritmo *cGA*/ $\mu$ *PSOLA*

A continuación, se describe el pseudocódigo del algoritmo propuesto:

---

**Algoritmo 5.1:** Pseudocódigo del *cGA* canónico

---

```

1  p = GenerarPoblaciónInicial()
2  Evaluar(p)
3  while no CondiciónDeParada do
4      recorrido = GenerarRecorrido(p)
5      for each individuo in recorrido
6          vecinos = ObtenerVecinos(p, individuo)
7          padres = SeleccionarPadres(vecinos)
8          descendientes = Recombinar(padres)
9          descendientes = Mutar(descendientes)
10         Evaluar(descendientes)
11         paux = Reemplazar(individuo, descendientes)
12     end for
13  p = paux
14  done

```

---

Como se puede apreciar en el Algoritmo 5.1, en la línea 1, se genera la población inicial con una distribución aleatoria uniforme, en la línea 2, se evalúa dicha población empleando el método que se explicó en la sección 5.1.2. Posteriormente, en la línea 4 se genera el tipo de recorrido que se va a realizar sobre la población es decir, el orden en el cual van a ser visitados los diferentes individuos de la población. En este trabajo, se empleó

un recorrido lineal (ver marco teórico) en donde los individuos se recorren en orden partiendo de la esquina superior izquierda. Para cada uno de los individuos dentro del *recorrido* (línea 5), se selecciona un vecindario (línea 6) del cual serán seleccionados los padres (línea 7) para ser recombinados y mutados (líneas 8 y 9). Finalmente, se almacenará en una estructura temporal al mejor de los individuos resultantes del proceso anterior (línea 13). Todo el procedimiento anterior, se realiza mientras no se cumpla el criterio de parada (línea 3).

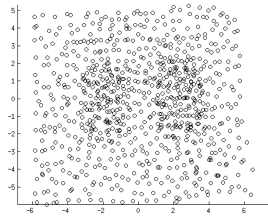
## 5.2 Experimentación

Para mostrar el rendimiento del algoritmo propuesto, se utilizaron 10 conjuntos de datos sintéticos. Cada conjunto de datos fue probado con 3 diferentes índices de validación de agrupamiento y 5 configuraciones diferentes del algoritmo evolutivo. Los conjuntos de datos sintéticos (ver Figura: 5.6) fueron tomados del repositorio de la pagina personal de S. Bandyopadhyay y se encuentran disponibles en <http://www.isical.ac.in/~sanghami/data.html>. Los conjuntos de datos sintéticos fueron diseñados para mostrar diferentes configuraciones de la geometría de los grupos así como la compactación y separación de estos. La tabla 5.1 muestra las características de cada conjunto de datos.

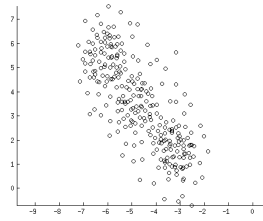
Conjunto de datos	Número de Patrones	Número de Atributos	Número de Clases
2.2	824	2	2
3.2	268	2	3
4.2	1288	2	4
5.2	250	2	5
6.2	300	2	6
7.2	417	2	7
8.2	1005	2	8
9.2	900	2	9
10.2	500	2	10

**Tabla 5.1:** Características de los conjuntos de datos

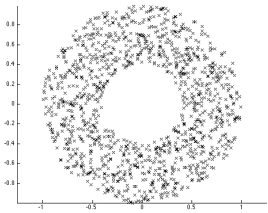
Todos los experimentos fueron realizados empleando un cluster de 16 nodos corriendo el sistema operativo Ubuntu 12.10 de 64 bits y utilizando como gestor de procesos distribuidos al sistema Condor. Cada nodo del cluster, estaba compuesto por un procesador Intel® Core(TM)2Quad a 2.66 GHz y 4 GB de memoria RAM. Para cada conjunto de datos y cada configuración del algoritmo, se corrieron 100 experimentos dando un total de 19,500 experimentos. Finalmente todos los algoritmos fueron programados utilizando Matlab 2012b para UNIX.



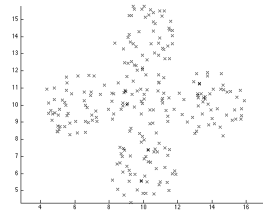
a) Data 2\_2



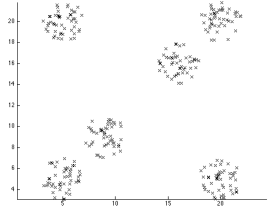
b) Data 3\_2



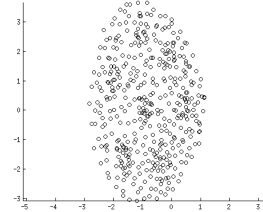
c) Data 4\_2



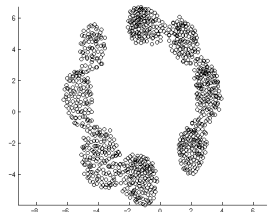
d) Data 5\_2



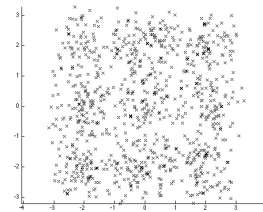
e) Data 6\_2



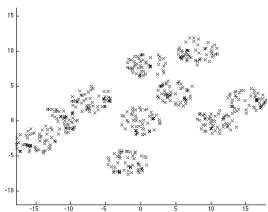
f) Data 7\_2



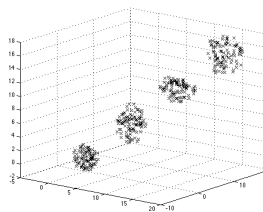
g) Data 8\_2



h) Data 9\_2



i) Data 10\_2



j) Data 4\_3

**Figura 5.6:** Conjunto de datos sintéticos

Los parámetros empleados por las diferentes configuraciones de los algoritmos se encuentran resumidas en la Tabla 5.2. Específicamente, se estudiaron dos tipos de cGAs híbridos: el cGACenter-BT y el cGABT-BT. Ambos difieren uno del otro en el método de selección, esto es debido a que en el cGACenter-BT el primer padre seleccionado corresponde siempre con el elemento activo lo cual garantiza, que todos los individuos serán tomados en cuenta durante el proceso de recombinación mientras que el segundo padre será seleccionado de entre los individuos restantes del vecindario empleando el método de selección por torneo binario. Para el caso del cGABT-BT, ambos padres son seleccionados por el método de torneo binario adicionalmente a lo anterior, también se realizaron pruebas empleando diferentes tipos de vecindarios como por ejemplo el vecindario compacto de 9 vecinos (C9) o el vecindario lineal de 5 vecinos (L5).

<b>Parámetro</b>	<b>Valor</b>
CGA grid size	5x5
CGA generations	50
CGA recombination probability	0.8
CGA mutation probability	0.01
CGA neighborhood tour	lineal
CGA neighborhood type	Compact, lineal
CGA neighborhood size	1,2,5
CGA parent selection	center, binary tournament
PSO inertia	[0.1 ... 0.9]
PSO cognitive weight	0.8
PSO Social weight	1.6
PSO population size	5
PSO generations	30
PSO velocity	[-0.1 ... 0.1]
PSO survivors	2

**Tabla 5.2:** Configuración de los parámetros para los experimentos

El primer resultado importante que se muestra, (tal como ocurre con algunos algoritmos del estado del arte), consiste en que todos los cGAs híbridos propuestos, fueron capaces en el 100% de los casos, de encontrar el número correcto de clases dentro de los conjuntos de datos empleados. La Tabla 5.3, muestra el resumen de los resultados

obtenidos por los algoritmos al emplear los tres índices de validación de clases más comúnmente utilizados en la literatura. Como se puede observar, los mejores resultados se obtienen al emplear el índice de validación I.

Conjunto de datos	Clases	I-Index	Xie-Beni	DB
2.2	2	2	2	2
3.2	3	4	3	3
4.2	4	4	4	4
5.2	4	5	5	5
6.2	6	6	4	4
7.2	7	7	7	6
8.2	8	8	8	8
9.2	9	9	9	4
10.2	10	10	10	8
4.3	4	4	4	4
Iris	3	3	2	2
Wine	3	3	3	2

**Tabla 5.3:** Número e clases encontradas con el algoritmo propuesto

### 5.2.1 Método de selección de los padres

Como se discutió anteriormente, durante el presente estudio, fueron utilizados dos métodos por el cGA para la selección de los padres: centro (C) y torneo binario (BT). Todos aquellos algoritmos denominados **cGACenter-BT** seleccionan al primer padre del centro del vecindario, mientras que el segundo padre es seleccionado mediante torneo binario. La inclusión del individuo central como ya se mencionó, asegura que todos los individuos del vecindario sean seleccionados al menos una vez para formar parte del proceso de recombinación. Por otro lado, todos aquellos algoritmos denominados **cGABT-BT**, seleccionan a ambos padres utilizando torneo binario lo cual no asegura que todos los individuos sean seleccionados durante el proceso de recombinación relajando la presión de selección del algoritmo. Como se puede observar en las graficas mostradas en la Figura 5.7, el método de selección de los padres tiene el efecto de incrementar o disminuir la presión

de selección del cGA. Tal cambio en el método de selección incrementa o decrementa la prioridad sobre la exploración y la explotación del algoritmo causando como consecuencia, una convergencia mas o menos rápida.

Los mejores resultados durante los experimentos realizados, se obtuvieron cuando se empleo el método de torneo binario para seleccionar a ambos padres (cGABT-BT) lo que significa que incrementando la presión de selección por medio del método de selección de los padres y tomando como base el aislamiento por distancia del algoritmo, el cGA presenta una mejor capacidad para recorrer el espacio de búsqueda (vea la tabla 5.4 de resultados estadísticos).

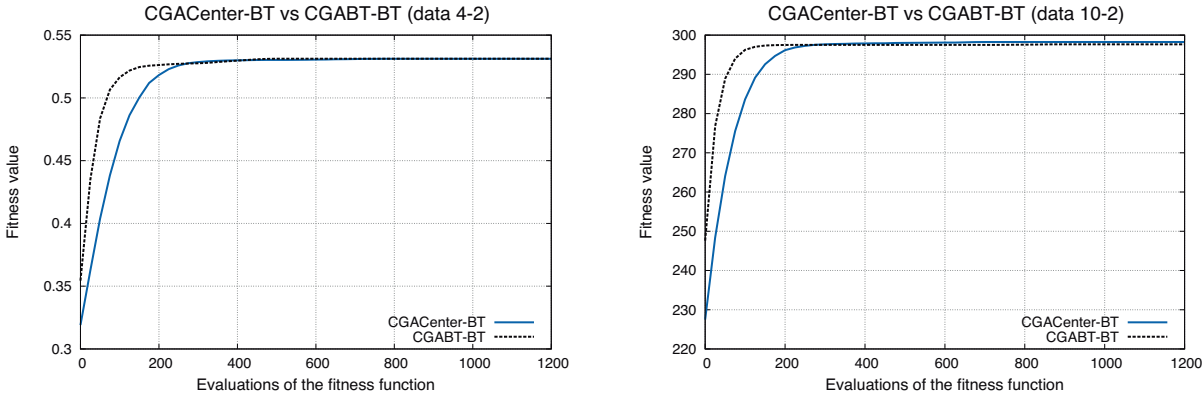


Figura 5.7: Método de selección de los padres

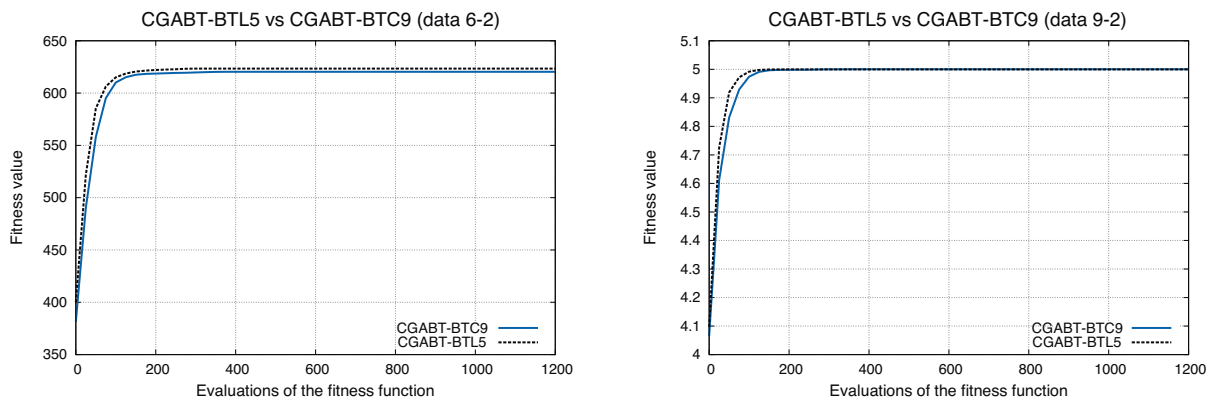
### 5.2.2 Tipo de vecindario

Una vez que se ha determinado el mejor método de selección de los padres (sección 5.2.1), se procedió a determinar el mejor tipo de vecindario para el problema en cuestión. Se probaron dos de los tipos mas comunes de vecindarios: vecindario compacto y vecindario lineal, como se puede observar en las graficas de la Figura 5.8, los mejores



resultados (convergencia más rápida) para los conjuntos de datos 6.2 y 9.2, se alcanzaron cuando se empleo un vecindario de tipo lineal. Los mismos experimentos fueron realizados para todos los conjuntos de datos disponible y los resultados fueron siempre los mismos es decir para el problema que se esta solucionando la mejor alternativa con respecto al tipo de vecindario es el lineal.

Al realizar todos los experimentos anteriores, se puede observar empíricamente que cuando los vecindarios usados para el problema estudiado son de tipo compacto (C), la presión de selección promueve a la explotación sobre la exploración, y como consecuencia, se reduce la velocidad de convergencia del algoritmo. También es importante observar, que mientras que la probabilidad de seleccionar un padre de dos vecindarios traslapados aumente, la probabilidad de seleccionar un individuo que promueva la exploración en el mismo espacio de búsqueda disminuirá lo que nos permite concluir, que la velocidad de convergencia del algoritmo se encuentra relacionada directamente con el tamaño del vecindario y este ultimo se encuentra en función de la complejidad del problema como se vera mas adelante.



**Figura 5.8:** Tipo de vecindario

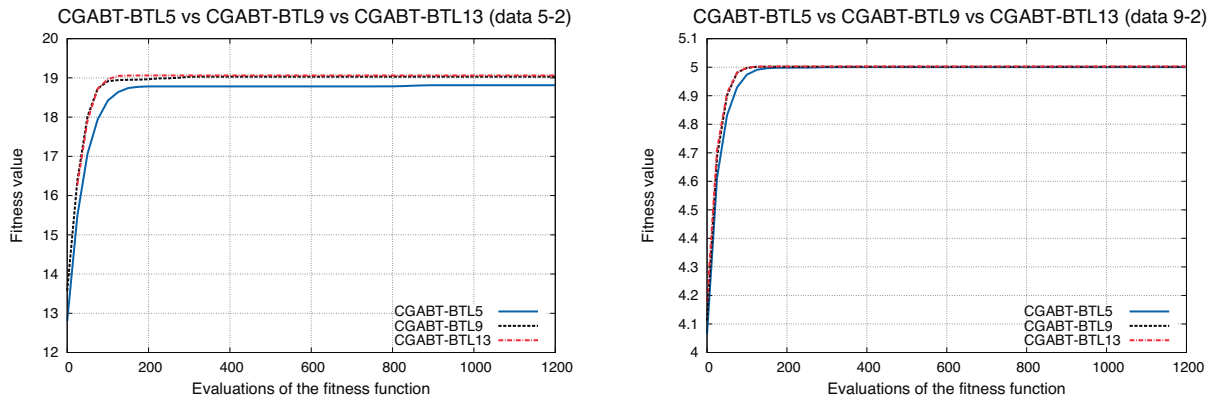
### 5.2.3 Tamaño del vecindario

Hasta este punto y basado en los experimentos y resultados obtenidos hasta el momento, se ha podido determinar el método de selección de los padres así como el tipo el tipo y tamaño del vecindario a utilizar. Finalmente y como se concluyo en la sección anterior, es muy importante determinar el tamaño adecuado del vecindario, ya que este será un factor decisivo a la hora de buscar un equilibrio entre la exploración y la explotación del algoritmo.

Para concluir con el algoritmo propuesto, se realizaron diversos experimentos tomando como base diferentes valores para el radio del vecindario. Considerando que la población del algoritmo celular que se ha diseñado es apenas de 5x5, los radios elegidos para realizar el último conjunto de experimentos fueron de 5, 9 y 13 individuos.

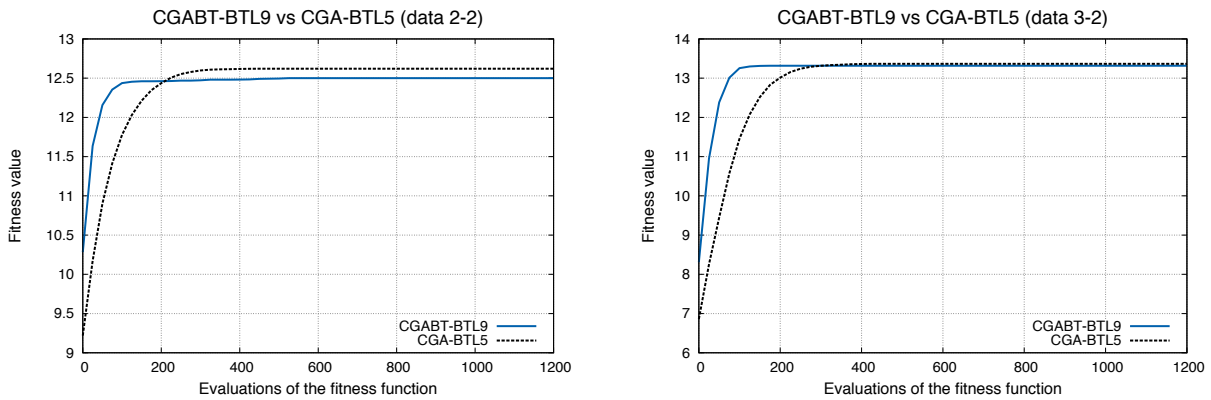
Como se puede observar en las graficas de la Figura 5.9, existe una clara diferencia al emplear vecindarios de 5 individuos con respecto a los formados por 9 y 13, sin embargo, al observar las diferencias en el comportamiento del algoritmo cuando se emplean vecindarios de 9 y 13 individuos respectivamente, el algoritmo se comporta se forma muy similar por lo que se ha concluido que el tamaño óptimo para el vecindario es de 9 individuos lo cual como se puede observar, decremento en casi un 30% el número de cálculos que tiene que realizar por el algoritmo.

Finalmente y tomando en cuenta el hecho de que la estructura de los individuos dentro de cGA es de tipo toroide, al usar vecindarios cuya longitud es mayor a 13 individuos, trae como consecuencia, que dichos individuos se traslapen de manera continua lo cual incrementaría la presión de selección acarreando una convergencia prematura debido a que la probabilidad de seleccionar dos padres que sean idénticos aumenta.



**Figura 5.9:** Tamaño del vecindario

Finalmente y tomando como base todos los resultados obtenidos en los apartados anteriores, en la Figura 5.10, se muestra el resultado al combinar de forma adecuada el método de selección de los padres, tipo de vecindario y longitud de este.



**Figura 5.10:** Tamaño y tipo de vecindario

### 5.3 Análisis de Resultados

Para comparar los resultados experimentales de todos los algoritmos probados de forma exhaustiva, se realizaron dos tipos de análisis estadístico. El primero, involucra indicadores estadísticos clásicos (máx., min, mediana, media, desviación estándar) los

cuales permiten una primer comparación. El segundo tipo de análisis involucra una serie test estadísticos no paramétricos basados en un pruebas de hipótesis.

### **5.3.1 Medidas estadísticas clásicas**

Como primer paso para realizar la validación de los resultados obtenidos , se procedió a calcular los valores: mínimo, máximo, media, mediana y desviación estándar para todo el conjunto de los 19,500 experimentos realizados con la finalidad de tener una primera visión global del comportamiento de los algoritmos estudiados.

Como se puede observar en la Tabla 5.4, el algoritmo con el peor comportamiento fue el cGACenter-BTL5 ya que este presentó el valor de aptitud mas bajo para todos los experimentos. El algoritmo con menor desviación estándar fue el cGABT-BTC9 lo cual es resultado debido a la presión inducida por el tipo de vecindario. El mejor rendimiento a lo largo de todos los experimentos, fue obtenido por los algoritmos cGABT-BT19 y el cGABT-BTL13,. Dichos resultados pueden también ser explicados por el tipo y tamaño de los vecindarios que fueron empleados por estos algoritmos. Por lo tanto, tomando como base los datos mostrados en la Tabla 5.4 se puede concluir, que el mejor algoritmo para solucionar el problema que ha sido estudiado es aquel que selecciona ambos padres para su recombinación, es decir, el método de torneo binario, así mismo para controlar la presión de selección y como consecuencia poder lograr un balance adecuado entre la exploración y la explotación del espacio de búsqueda se deben emplear vecindarios lineales y que para el caso en concreto de la estructuración población del algoritmo propuesto (5 x 5 individuos) los mejores tamaños oscilan entre 9 y 13 individuos. Como se vera mas adelante, la diferencia entre estos dos tamaños de población no representan un comportamiento estadísticamente diferente por lo que se concluye que es mejor seleccionar vecindarios de 9 individuos ya que estos involucran un menor manejo de memoria por parte del HCEA.

Data	Std. Measure	CGACenter-BTL5	CGABT-BTC9	CGABT-BTL5	CGABT-BTL9	CGABT-BTL13
Data 2.2	Min	<b>9.6996</b>	10.7210	10.6140	10.7250	10.7570
	Max	11.9420	12.3580	12.3840	<b>12.4530</b>	12.3500
	Mean	11.0423	11.8928	11.7644	<b>11.9444</b>	11.8782
	Median	11.2480	12.2260	12.0930	<b>12.2870</b>	12.2210
	Std	0.8884	0.06868	0.7379	0.7267	<b>0.06750</b>
Data 3.2	Min	<b>5.8558</b>	8.3908	7.4255	8.3745	8.1257
	Max	12.0140	13.2310	13.1560	<b>13.3050</b>	13.2550
	Mean	9.4085	11.8076	11.4243	<b>11.8671</b>	11.7975
	Median	9.9641	12.5370	12.5850	12.7970	<b>12.8150</b>
	Std	2.4608	<b>1.9978</b>	2.3922	2.0563	2.1430
Data 4.2	Min	<b>0.3065</b>	0.3744	0.3491	0.3775	0.3806
	Max	0.4782	0.5236	0.5167	<b>0.5294</b>	0.5280
	Mean	0.4032	0.4779	0.4643	<b>0.4846</b>	0.4823
	Median	0.4190	0.5035	0.4952	<b>0.5155</b>	0.5124
	Std	0.0703	<b>0.0612</b>	0.0693	0.0638	0.0622
Data 5.2	Min	<b>11.0240</b>	13.3710	12.6400	13.7200	13.2270
	Max	16.8240	18.8230	18.5080	18.9530	<b>18.9970</b>
	Mean	14.2902	17.2230	16.5132	<b>17.3764</b>	17.3008
	Median	14.6990	18.1450	17.4400	<b>18.5630</b>	18.5230
	Std	2.3241	2.2519	2.3847	<b>2.2277</b>	2.4392
Data 6.2	Min	<b>304.3400</b>	395.4500	375.5600	400.0800	393.1600
	Max	554.6600	613.9900	611.8800	<b>619.8500</b>	619.0700
	Mean	440.6040	550.9820	534.7240	<b>558.0380</b>	557.2960
	Median	449.2300	593.4200	582.0900	604.3500	<b>612.2000</b>
	Std	101.0624	<b>90.9416</b>	98.7067	92.7730	97.0172
Data 7.2	Min	<b>3.3119</b>	4.0321	3.9258	4.0462	4.0192
	Max	4.6544	4.8993	4.8910	4.8908	<b>4.9028</b>
	Mean	4.1541	4.6446	4.5765	4.6343	<b>4.6442</b>
	Median	4.3240	4.8061	4.7338	4.8086	<b>4.8151</b>
	Std	0.5332	0.3608	0.3941	<b>0.3541</b>	0.3706
Data 8.2	Min	<b>27.5870</b>	32.6540	31.4500	33.1810	33.3160
	Max	35.7800	<b>35.9710</b>	35.9660	35.9700	35.9680
	Mean	32.9782	35.2594	34.8874	35.3622	<b>35.4172</b>
	Median	34.4910	35.9400	35.8610	35.9450	<b>35.9600</b>
	Std	3.4458	1.4589	1.9441	1.2229	<b>1.1753</b>
Data 9.2	Min	<b>3.7629</b>	4.2465	4.2168	4.2525	4.3572
	Max	4.8965	4.9932	4.9823	5.0017	<b>5.0026</b>
	Mean	4.4627	4.7928	4.7415	4.8018	<b>4.8267</b>
	Median	4.5980	4.9452	4.8690	4.9681	<b>4.9705</b>
	Std	0.4548	0.3143	0.3147	0.3197	<b>0.2745</b>
Data 10.2	Min	<b>234.2100</b>	257.9200	254.1300	258.9300	255.7700
	Max	289.1000	299.1100	296.5500	299.5700	<b>299.7000</b>
	Mean	266.2940	287.8440	283.6300	<b>288.8460</b>	287.7800
	Median	270.5600	295.5000	291.7100	<b>297.8500</b>	297.7400
	Std	21.8989	<b>17.2526</b>	17.7088	17.3374	18.7831

**Tabla 5.4** : Resultados estadísticos para todas las instancias

### 5.3.2 Pruebas estadísticas no paramétricas y pruebas post-hoc

Como primer paso en esta segunda fase de validación de resultados, lo primero que se buscó fue determinar si existe o no un comportamiento estadísticamente diferente entre los algoritmos estudiados, para ello se empleó una técnica basada en rankings. En el test de Friedman Alineado [5], se calcula el rendimiento medio alcanzado por cada algoritmo en cada problema (valor de localización). Después, se calculan las diferencias entre el rendimiento obtenido por cada algoritmo con respecto al valor de localización. Este paso se repite para todos los algoritmos y problemas. Las diferencias resultantes (observaciones alineadas) se ordenan desde 1 hasta  $k - n$  de forma relativa unas con otras. A partir de ahí, el esquema de ranking es el mismo que el empleado por un procedimiento de comparaciones múltiples con muestras independientes, como el test de Kruskal-Wallis. De este modo, los rankings asignados a las observaciones alineadas se denominan rankings alineados.

En la tabla 5.5 se observa que después de aplicar el Test de Friedman Alineado a los resultados experimentales, el algoritmo con mejor ranking fue el cGABT-BTL9 seguido por el cGABT-BTL13 con un margen de diferencia muy pequeño. Asimismo, también se puede observar que el algoritmo con el peor desempeño fue el cGACenter-BTL5. Estos resultados concuerdan perfectamente con los resultados obtenidos de manera empírica en la sección anterior.

<b>Algoritmo</b>	<b>Ranking</b>
CGABT-BTL9	1.7778
CGABT-BTL13	1.8889
CGABT-BTC9	2.6667
CGABT-BTL5	3.6667
CGACenter-BTL5	5.0000

**Tabla 5,5:** Rankings obtenidos por el test de Friedman Alineado

Tanto el Test de Friedman como sus dos propuestas avanzadas (Friedman Alineado y Quade) no identifican las diferencias existentes entre el mejor algoritmo encontrado (denominado algoritmo de control) y el resto. Estos test se limitan a detectar la existencia o no de diferencias en todo el conjunto de resultados. Si se encuentran dichas diferencias, es necesario proceder con un procedimiento post-hoc de identificación de diferencias (basados en la distribución normal).

Una vez obtenido el valor del estimado  $Z$ , es posible obtener a partir de él, el p-valor no ajustado correspondiente. La utilidad de estos procedimientos post-hoc radica en que son capaces de calcular el P-Valor Ajustado (PVA) considerando la familia de hipótesis completa para cada pareja de algoritmos comparados.

Como se puede observar en la Tabla 5.6, tomando como algoritmo de control al cGABT-BTL9 y comparándolo con el resto de los algoritmos, se observa que en el 95% de los casos ( $\alpha=0.5$ ) el algoritmo presenta un comportamiento estadísticamente superior a los algoritmos cGACenter-BTL5, cGABT-BTL5 y cGABT-BTC9 mientras que cuando se comparo el resultado con el cGABT-BTL13 el comportamiento es estadísticamente similar por lo que estadísticamente, no hay diferencia alguna entre emplear uno u otro, lo que nuevamente coincide con lo observado en la sección anterior.

<i>i</i>	Algoritmo	$Z=(R_0-R_i)/SE$	<i>p</i>	<i>Holm</i>
1	cGACenter-BTL5	4.323065	0.000015	<b>0.012500</b>
2	cGABT-BTL5	2.534210	0.011270	<b>0.016667</b>
3	cGABT-BTC9	1.192570	0.233038	<b>0.025000</b>
4	cGABT-BTL13	0.149071	0.881497	0.050000

**Tabla 5.6:** Comparación Post Hoc para  $\alpha=0.5$

## 5.4 Conclusiones parciales

En este capítulo, se presentó un nuevo modelo de heurística espacialmente estructurada de población descentralizada que tiene como objetivo estimar el número de clases contenidas en un conjunto de datos, para validar esto último, se realizaron 19,500 experimentos sobre 10 conjuntos de datos sintéticos con diferentes configuraciones en las clases, variando la compactación y separación así como la geometría lo cual, permitió probar el balance entre la exploración y la explotación de los algoritmos presentados ante diferentes escenarios.

El algoritmo propuesto se denomina *HCEA* y es una hibridación entre un algoritmo genético celular (*cGA*) y un micro algoritmo de optimización por cumulo de partículas con un método de ajuste local ( *$\mu$ PSOLA*). Para realizar la validación de los resultados obtenidos experimentalmente, se emplearon dos metodologías. La primera consistió en obtener las medidas estadísticas clásicas como son: mínimo, máximo, media, media y desviación estándar para todo el conjunto de resultados mientras que la segunda consistió en el empleo de un análisis estadístico basado en rankings y pruebas no paramétricas post-hoc para determinar si estadísticamente existen comportamientos diferentes por parte de los algoritmos.

Como se mencionó en la sección 5.3, los resultados obtenidos son completamente consistentes al aplicar las diferentes pruebas estadísticas y por lo tanto se concluye que la mejor configuración para el algoritmo propuesto con el problema estudiado se obtiene cuando ambos padres son seleccionados empleando torneo binario, topología lineal, y para una población de 5 x 5 individuos estructurada en forma de toroide, el tamaño óptimo para el vecindario es de 5 individuos.



## 5.5 Trabajo futuro

Como ya se menciona en los apartados anteriores, todos los experimentos fueron realizados empleando conjuntos de datos sintéticos porque permitieron probar el comportamiento del algoritmo así como su balance entre exploración y explotación ante diversas situaciones preestablecidas sin embargo, ahora se desea probar el algoritmo con conjuntos de datos reales, donde uno de los máximos retos a vencer, será la alta dimensionalidad de dichos conjuntos así como la gran cantidad de patrones presentes en dichos conjuntos.

Finalmente, una vez que haya sido probado el algoritmo en dichos conjuntos de datos, por la naturaleza y estructura del algoritmo desarrollado, se pretende realizar su implementación en una GPU lo cual permitirá estudiar los efectos de diversos criterios de actualización de la población y por lo tanto el comportamiento de diversos recorridos dentro de su estructura.

# Modelo Diferencial Descentralizado

# 6

En este capítulo se expondrá el tercer y último modelo desarrollado, a diferencia del capítulo 5 en el cual se presentó un modelo celular que simula la evolución del sistema desde el punto de vista del individuo con aislamiento por distancia, ahora, se expondrá un modelo poblacional que evoluciona a partir del conjunto de creencias de todos los individuos que conforman a la población, esto elimina el aislamiento por distancia presentado anteriormente. El modelo meta-heurístico propuesto, ha sido enriquecido empleando un método de búsqueda local basado en un micro algoritmo evolutivo y ajuste local.

Para probar la eficiencia del modelo desarrollado, se ha vuelto a emplear el problema relacionado con la búsqueda del número de clases contenidas en un conjunto de datos que fue presentado en el capítulo anterior, para ello, se ha utilizado una serie de conjuntos de datos con diferentes características especialmente diseñadas para probar el comportamiento del algoritmo y su balance entre exploración y explotación ante dichas situaciones además, de un par de conjuntos de datos clásicos en el estudio del reconocimiento de patrones.

## 6.1 Algoritmo de Evolución Diferencial con Ajuste Local

El algoritmo de evolución diferencial con ajuste local (DELA por sus siglas en inglés), es un algoritmo evolutivo híbrido entre DE y  $\mu$ DE [57][63] y se encuentra inspirado en los algoritmos de Hooke-Jeeves y Nelder y Mead. Para su funcionamiento, DELA emplea dos

mecanismos durante el proceso de búsqueda denominados: convergencia global y convergencia local.

Para realizar una estimación del agrupamiento generado para  $K$  clases, DELA emplea un  $\mu$ DE, proceso al cual se le ha denominado como convergencia local. Una vez que se tiene el valor de aptitud para la mejor estimación generada por la  $\mu$ DE, DELA, emplea un algoritmo de Evolución Diferencial para guiar la convergencia global es decir, determinar el número de clases dentro de un conjunto de datos. A este último mecanismo se le ha llamado, método de convergencia global.

En el Algoritmo 6.1 se muestra el pseudocódigo de DELA. En la línea 1, se crea una población inicial de ' $Np$ ' individuos para la generación cero ( $x_{0,g}$ ), posteriormente en la línea 4, se generan los vectores de prueba que serán empleados durante el proceso de recombinación (línea 5) . En las líneas 6 y 7, se realiza la estimación de la mejor partición posible para el conjunto de datos y el número de clases codificados por los vectores 'u' y 'x' . Para dicho proceso de estimación, se emplea un  $\mu$ DE como ya se comento anteriormente (ver marco teórico). Finalmente en la línea 8, se calcula el valor de aptitud para las particiones generadas lo cual, ayudara a seleccionar al vector resultante para la siguiente generación, todo este proceso, se realiza mientras no se alcance la condición de terminación.

---

**Algoritmo 6.1:** Pseudocódigo para DELA

---

```
1  $x_{0,g} = \text{GenerateInitialPopulation}(Np)$ 
2 while (Termination Criterion no yet met)
3     for ( $p=0; i < NP; i++$ ) do
4          $t_{p,g} = \text{GenerateTrialVectors}()$ 
5          $u_{p,g} = \text{Recombine}(t_{p,g}, x_{p,g})$ 
6          $\text{partition}_u = \text{EstimatePartition}(u_{p,g})$ 
7          $\text{partition}_x = \text{EstimatePartition}(x_{p,g})$ 
```

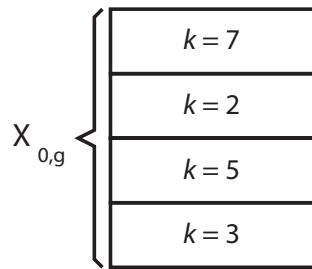
```

8      If  $f(\text{partition}_u) < f(\text{partition}_x)$  then
9           $x_{p,g+1} = u_{p,g}$ 
10     else
11          $x_{p,g+1} = x_{p,g}$ 
12     end if
13 end for
14 end while

```

---

Como se puede observar en la figura 6.1, la población del algoritmo DELA se encuentra codificada por un vector de valores enteros los cuales representan valores para 'k'.



**Figura 6.1:** Ejemplo de población inicial en DELA

La función de aptitud empleada por DELA, mide la calidad del agrupamiento generado por el  $\mu$ DE empleando el índice de validación de agrupamientos 'I'. En el Índice-I (Ecuación 6.1), el primer factor normaliza cada valor del índice por el número total de grupos, el segundo termino, establece el suma del error cuadrático global para todo el conjunto de datos en relación con el error intra-clase y el tercer termino, incorpora la máxima diferencia observada entre dos de las  $k$  clases. El calculo del índice, incluye el parámetro  $p \in \mathbb{R}$  para controlar el contraste entre las diferentes configuraciones de las clases.

$$I = \left( \frac{1}{k} \cdot \frac{E_1}{E_k} \cdot D_k \right)^2 \quad (6.1)$$

En las líneas 6 y 7 del Algoritmo 6.1, se realiza la estimación de la partición para el valor de 'k' generado por el algoritmo de *DE*. El propósito de la función *EstimatePartición* consiste como su nombre lo indica, en estimar una partición factible para el conjunto de datos y de esta forma poder evaluar el índice de validación de agrupamiento que es utilizado como función de aptitud. En el Algoritmo 6.2, se muestra el pseudocódigo de la función *EstimatePartición*.

---

**Algoritmo 6.2:** Pseudocódigo para el algoritmo de EstimatePartition

---

```

1   $x_g = GenerateInitialPopulation()$ 
2  while (Termination Criterion no yet met)
3      while (Nominal Convergence not reached)
4          for  $p=1$  to PopulationSize do
5               $t_{p,g} = GenerateTrialVectors()$ 
6               $u_{p,g} = Recombine(t_{p,g}, x_{p,g})$ 
7              if  $SSE(u_{p,g}) \leq SSE(x_{p,g})$  then
8                   $x_{p,g+1} = Adjustment(u_{p,g})$ 
9              else
10                  $x_{p,g+1} = Adjustment(x_{p,g})$ 
11             end if
12         end for
13     end while
14     if ReplacementCycle then
15          $best = GetBestIndividuals(x_{p,g+1})$ 
16          $x_p = RestartPopulation(best)$ 
17     end if
18 end while

```

---

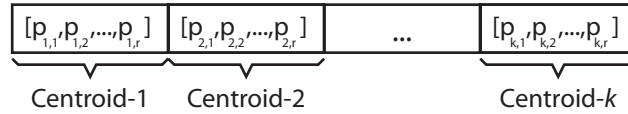
Como se puede observar en el Algoritmo 6.2, la función *EstimatePartición* hace uso de un algoritmo de  $\mu DE$  como meta heurística de optimización para la búsqueda de una partición factible sin embargo, a diferencia de la DE clásica, en este algoritmo se emplean dos ciclos durante el proceso de búsqueda, el primero de ellos (línea 2), será ejecutado mientras no se alcance la convergencia global (obtener una partición factible), mientras que el segundo ciclo (línea 3), será ejecutado mientras no se alcance la convergencia nominal.

La convergencia nominal, puede ser entendida como la convergencia alcanzada por el algoritmo cuando la diversidad de la micro población se pierde, es decir, cuando la diferencia entre los individuos que la conforman no supera un cierto umbral.

Durante el proceso del ciclo interno, se puede observar que la función de aptitud empleada (ver Ecuación 6.2) corresponde con el error cuadrático medio (*SSE* por sus siglas en inglés) lo que automáticamente lleva a la conclusión de que el problema de estimar una partición válida está siendo modelado como un problema no combinatorio.

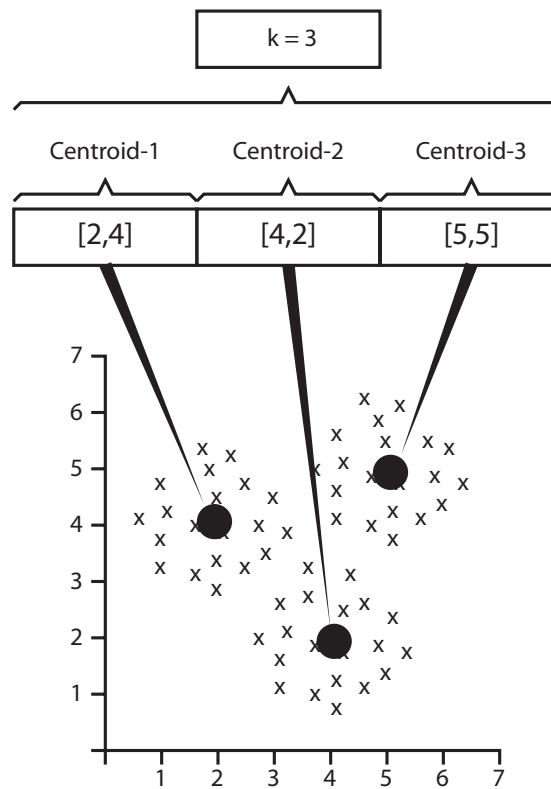
$$SSE = \frac{1}{k} \sum_{j=1}^k \sum_{i=1}^N D_E(o_i, m_j) \quad (6.2)$$

Donde ' $k$ ' es el número de clases, ' $N$ ' el número de patrones en el agrupamiento ' $j$ ', ' $D_E$ ' la distancia Euclidiana, ' $o_i$ ' es el  $i$ -ésimo patrón del grupo ' $j$ ' y ' $m_j$ ' el centroide del grupo ' $j$ '. Los individuos del Algoritmo 6.2, son representados por medio de vectores de dimensión ' $D$ ' donde ' $D$ ' representa el número de centroides que codificara el individuo (ver figura 6.2).



**Figura 6.2:** Codificación del individuo utilizada por *GetPartition*

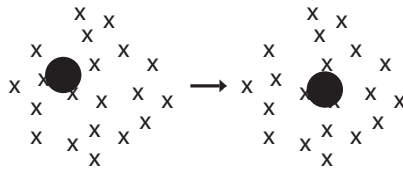
En la Figura 6.3, se muestra el ejemplo de un individuo y su representación gráfica cuando el valor de 'k' es igual a 3. Como se puede observar, cada elemento del vector que representa al individuo, indica la posición del centroide dentro del conjunto de datos.



**Figura 6.3:** Ejemplo de un individuo cuando  $k=3$

Finalmente, como se puede observar en las líneas 8 y 10 del Algoritmo 6.2, el método *GetPartition* emplea una función denominada Adjustment cuya finalidad es realizar un

último ajuste de los centroides en función de los patrones que integren a su clase. La figura 6.4, muestra gráficamente dicho ajuste.



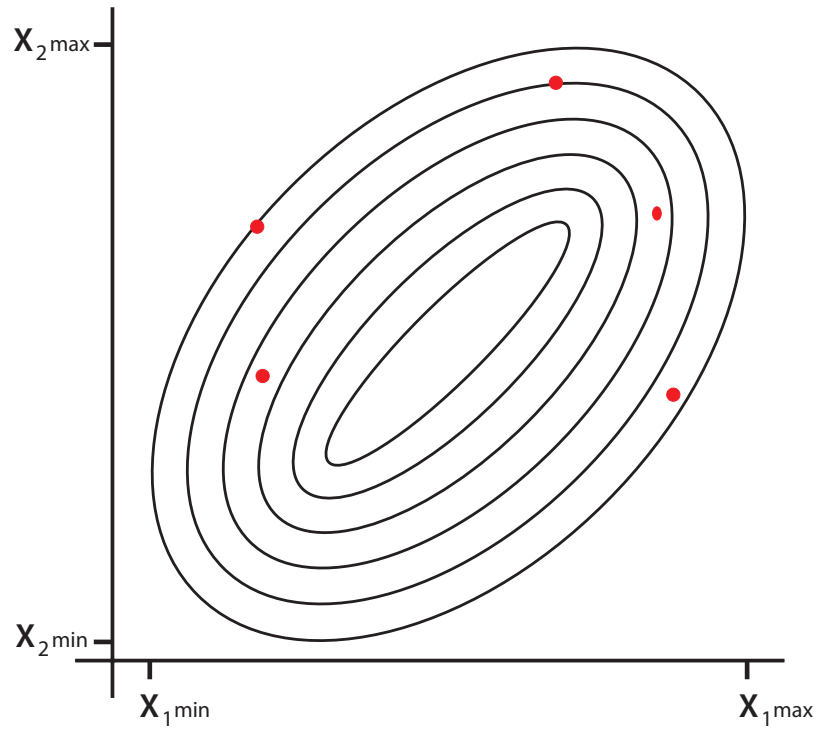
**Figura 6.4:** Efecto de la función *Adjustment*

### 6.1.1 Exploración VS Explotación

Después de la experiencia de los dos modelos anteriores, se ha aprendido que en definitiva, el éxito o fracaso de las meta heurísticas, recae en el correcto balance entre los mecanismos de exploración y explotación que el algoritmo emplee. En el caso del algoritmo DELA, el balance de estos dos mecanismos se consigue mediante la estructuración de la búsqueda y por lo tanto de la población dentro del mismo proceso de búsqueda (Algoritmo 6.1).

El primer paso de DELA, consiste en la generación de múltiples puntos los cuales se encuentran distribuidos uniformemente dentro del espacio de búsqueda (ver Figura 6.5).

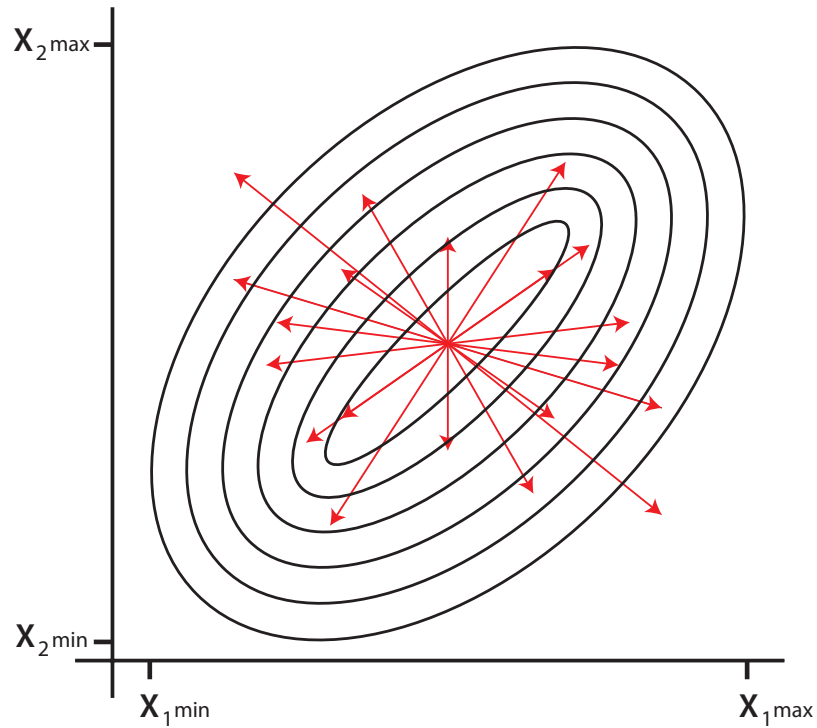




**Figura 6.5:** Uso de múltiples puntos iniciales

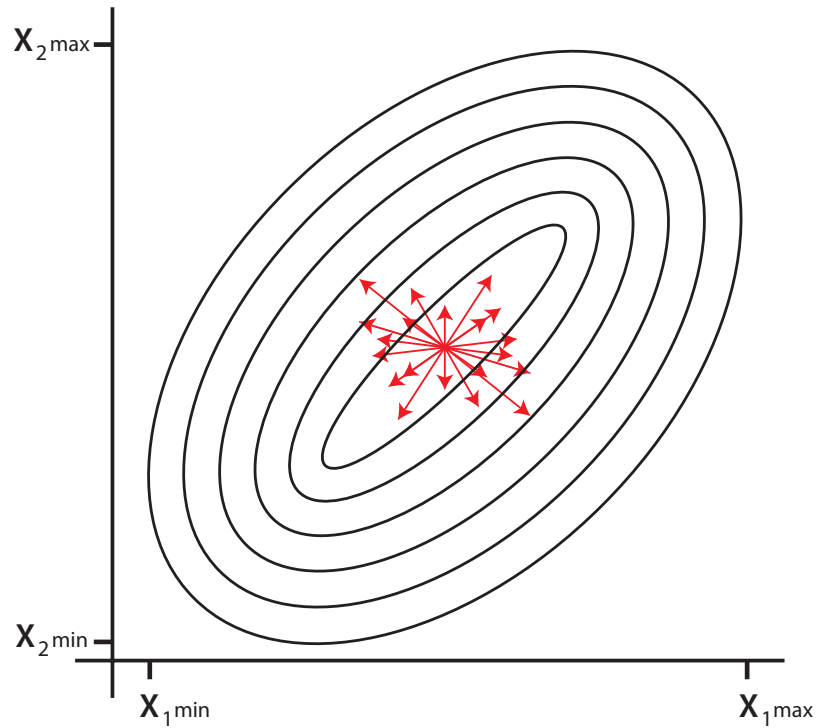
Posteriormente, DELA selecciona (de forma aleatoria) de entre la población, los puntos necesarios para formar el conjunto de *simplex* que será empleado por el algoritmo.

Si se consideran todas las posibles diferencias entre vectores pertenecientes al conjunto de puntos mostrados en la Figura 6.5, y dichas diferencias se desplazan a un punto común (por ejemplo el centro), se obtendrá como resultado el conjunto de vectores mostrados en la Figura 6.6.



**Figura 6.6:** Diferencia de vectores

En la Figura 6.7, se muestra al mismo conjunto de vectores de la Figura 6.6, sin embargo, en esta ocasión las diferencias se encuentran escaladas o ponderadas en un factor  $F$ . En este punto, se pueden realizar dos observaciones importantes: primero, se puede observar que cuando la diferencia de vectores es escalada por un factor alto ( $F = 1$ ), se promueve una mayor exploración dentro del espacio de búsqueda a diferencia de lo que sucede cuando dicho factor de escalamiento es bajo ( $F \leq \varepsilon$ ) donde  $\varepsilon$  es un valor cercano a cero ( $\varepsilon > 0$ ). En segundo lugar, se puede observar que la ventaja de emplear la diferencia de vectores, causa que tanto la orientación como el tamaño del paso se adapten automáticamente al landscape de la función objetivo, lo que permite que los individuos se agrupen en torno a los valores óptimos de dicha función.



**Figura 6.7:** Diferencia de vectores ponderadas en un factor  $F$

En la Figura 6.8, se muestra el comportamiento del algoritmo DELA en términos del proceso de explotación. Como se puede ver, la estructuración de la búsqueda empleando un proceso independiente para la búsqueda local ( $\mu$ DE), permite una eficiente y controlada explotación del espacio de soluciones en torno a un punto específico. Una vez más, se puede observar que las diferencias de los vectores empleados durante el proceso de explotación, pueden ser escaladas de forma tal, que intensifique o inhiba el proceso de explotación sobre un determinado punto. Esto último resulta de suma importancia, debido a que en los algoritmos meta heurísticos poblacionales clásicos (algoritmos genéticos, PSO, estrategias evolutivas, etc.), el proceso de explotación del espacio de búsqueda se consigue empleando métodos de recombinación de dos o más individuos, lo que dependiendo del método de selección de dichos individuos (ruleta, torneo binario, aleatorio, etc), puede redirigir el

proceso en alguna dirección distinta del espacio que se quiere explotar (ocasionando un proceso de exploración en lugar de explotación) cosa que no ocurre en el caso de DELA, lo que ayuda a tener una convergencia más rápida y segura como se muestra mas adelante en los resultados obtenidos.

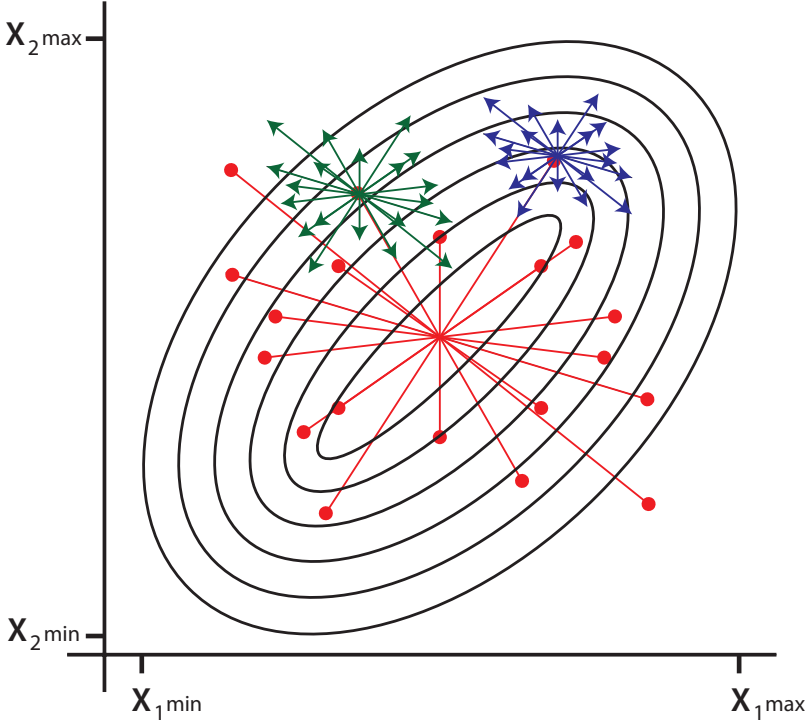


Figura 6.8: Explotación con base en un punto específico

## 6.2 Experimentos y Resultados

Una búsqueda exhaustiva en el estado del arte (ver capítulo del estado del arte) de los últimos años, mostro que las técnicas con mejor desempeño empleadas para buscar el número de clusters dentro de un conjunto de datos son; la evolución diferencial clásica y la

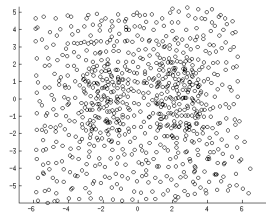
optimización por cúmulo de partículas (PSO). En consecuencia, el algoritmo DELA fue comparado con los resultados obtenidos por las técnicas antes mencionadas.

## 6.2.1 Conjuntos de datos

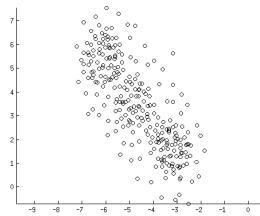
Para mostrar el rendimiento del algoritmo propuesto, se utilizaron 11 conjuntos de datos sintéticos y 2 conjuntos de datos reales. Cada conjunto de datos fue probado con 3 diferentes índices de validación de agrupamiento y 10 configuraciones diferentes del algoritmo evolutivo. Los conjuntos de datos sintéticos (Figura 6.9) fueron tomados del repositorio de la pagina personal de S. Bandyopadhyay y se encuentran disponibles en <http://www.isical.ac.in/~sanghami /data.html>. Los 11 conjuntos de datos sintéticos fueron diseñados especialmente, para mostrar diferentes configuraciones de la geometría de los grupos así como la compactación y separación de estos.. La tabla 6.1 muestra las características de cada conjunto de datos sintético.

Conjunto de datos	Número de Patrones	Número de Atributos	Número de Clases
2.2	824	2	2
3.2	268	2	3
4.2	1288	2	4
5.2	250	2	5
6.2	300	2	6
7.2	417	2	7
8.2	1005	2	8
9.2	900	2	9
10.2	500	2	10
4.3	400	3	4
Iris	150	4	3
Wine	178	13	3

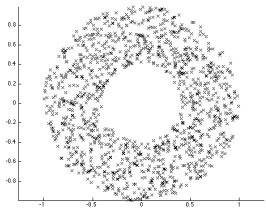
**Tabla 6.1:** Características de los conjuntos de datos



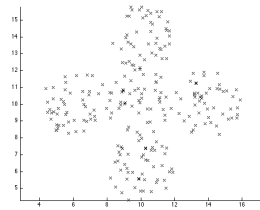
a) Data 2\_2



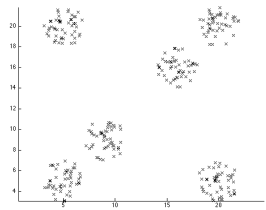
b) Data 3\_2



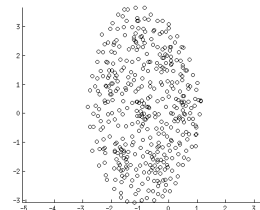
c) Data 4\_2



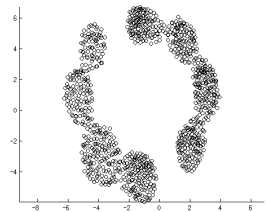
d) Data 5\_2



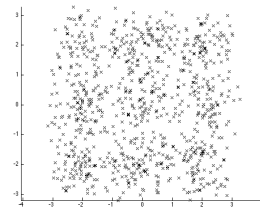
e) Data 6\_2



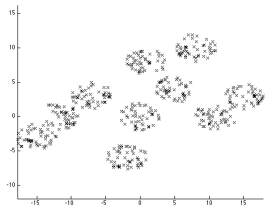
f) Data 7\_2



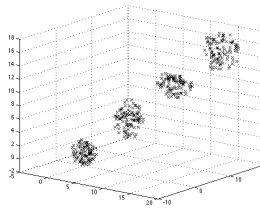
g) Data 8\_2



h) Data 9\_2



i) Data 10\_2



j) Data 4\_3

**Figura 6.9:** Conjunto de datos sintéticos

Los conjuntos de datos reales de Iris y Wine, fueron tomados del '*Machine Learning Repository*' de la Universidad de California e Irvin (UCI) y se encuentran disponibles en <http://archive.ics.uci.edu/ml>. Estos conjuntos de datos fueron utilizados, debido a que sin lugar a dudas, son los conjuntos de datos reales mas empleados para la experimentación en torno al mundo del Reconocimiento de Patrones. El conjunto de datos de Iris, esta conformado por 150 patrones de plantas de Iris para tres tipos distintos: Iris Setosa, Iris Versicolor e Iris Virginica. Cada patrón a su vez, contiene cuatro características que describen la longitud de la flor así como también, el ancho pétalos y los sépalos. El conjunto de datos de Wine, esta compuesto por 178 patrones con 13 características cada uno (Alcohol, ácido málico, cenizas, alcalinidad de cenizas, magnesio, fenoles totales, flavonoides, fenoles, Proantho-cianinas, intensidad del color, tonalidad, OD280/OD315 de vinos diluidos y prolina). Cada patrón, es el resultado de un análisis químico para un conjunto de vinos italianos, que se realiza para determinar la cantidad de cada uno de los componentes antes indicados.

## **6.2.2 Entorno de experimentación**

Todos los experimentos fueron realizados empleando un cluster de 16 nodos corriendo el sistema operativo Ubuntu 12.10 de 64 bits y utilizando como gestor de procesos distribuidos al sistema Cándor. Cada nodo del cluster, estaba integrado por un procesador Intel® Core(TM)2Quad a 2.66 GHz y 4 GB de memoria RAM. Para cada conjunto de datos y cada configuración del algoritmo (DE, PSO y DELA) y empleando los tres índices de validación de clustering comentados anteriormente (Índice-I, Xie Beni y DB), se realizaron un total de 100 experimentos para cada uno, dando un total de 11,700 experimentos. Finalmente todos los algoritmos fueron programados utilizando Matlab 2012b para UNIX.

La Tabla 6.2, muestra los valores empleados para los parámetros de cada uno de los algoritmos empleados durante el proceso de experimentación.

Parámetro	PSO	DE	DELA
Tamaño de la población	30	60	5
F	---	0.9	0.9
CR	---	0.0001	0.0001
Factor de Inercia	0.9	---	---
Factor Cognitivo	1.8	---	---
Factor Social	1.2	---	---
Generaciones	100	100	100

**Tabla 6.2:** Parámetros de configuración para los experimentos

A continuación, se mostraran los resultados obtenidos mismos que serán analizados haciendo uso de un conjunto de pruebas estadísticas no paramétricas. El análisis realizado a los resultados, muestra claramente que el algoritmo DELA tiene un comportamiento superior que los algoritmos con los que fue comparado.

### 6.2.3 Resultados experimentales

El primer resultado importante que se muestra, es que al igual que los resultados mostrados en el estado del arte, el algoritmo DELA también fue capaz de alcanzar el 100% de precisión en todos los experimentos reportados. La Tabla 6.3, muestra el resumen de los resultados obtenidos, con cada uno de los índices utilizados como funciones de aptitud para encontrar el número óptimo de clases. Como se puede ver, los mejores resultados fueron obtenidos cuando se empleo el *Índice-I*.

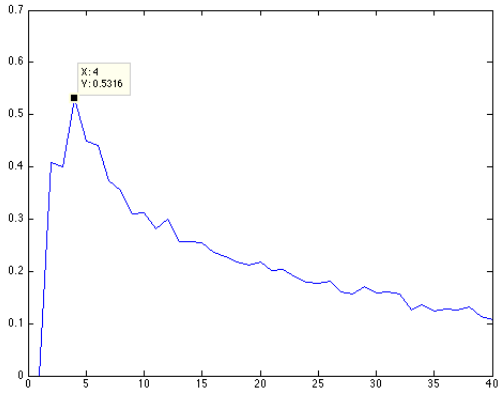


Conjunto de datos	Número de Clases	Índice-I	Xie-Beni	DB
2.2	2	<b>2</b>	2	2
3.2	3	<b>3</b>	3	3
4.2	4	<b>4</b>	4	4
5.2	5	<b>5</b>	5	5
6.2	6	<b>6</b>	4	4
7.2	7	<b>7</b>	7	6
8.2	8	<b>8</b>	8	8
9.2	9	<b>9</b>	9	4
10.2	10	<b>10</b>	10	8
4.3	4	<b>4</b>	4	4
Iris	3	<b>3</b>	2	2
Wine	3	<b>3</b>	3	2

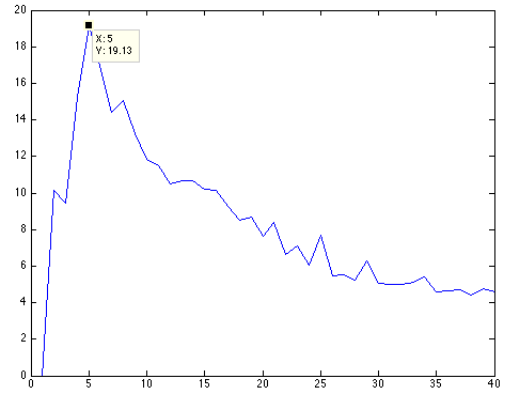
**Tabla 6.3:** Número de clases encontradas con el algoritmo DELA

Durante el proceso de experimentación, se midió la velocidad de convergencia de los diferentes algoritmos ante conjuntos de datos sintéticos y reales. Estos experimentos, permitieron observar el comportamiento del algoritmo DELA ante diversas situaciones de compactación y separación. Como se mostro en la Tabla 6.2, el Índice-i, fue capaz de identificar de forma correcta en un 100% de los casos, el número de clases presentes dentro de los conjuntos de datos empleados. En la figura 6.10, se muestra el comportamiento del Índice-i al ser empleado por el algoritmo DELA con algunos de los conjuntos de datos utilizados.

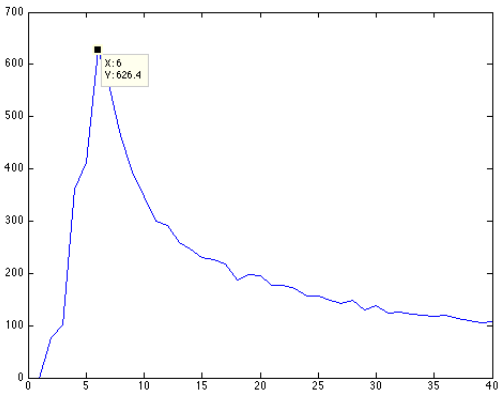
Como se puede observar en la figura 6.8, el Índice-i muestra un comportamiento estable al ser empleado por el algoritmo DELA y en todos los casos, se muestra como se maximiza la función (valor del Índice-i) cuando el número de clases presentes en el conjunto de datos coincide con las clases detectadas por DELA (gracias a la estimación de la función *EstimatePartition* mostrada en el Algoritmo 6.2).



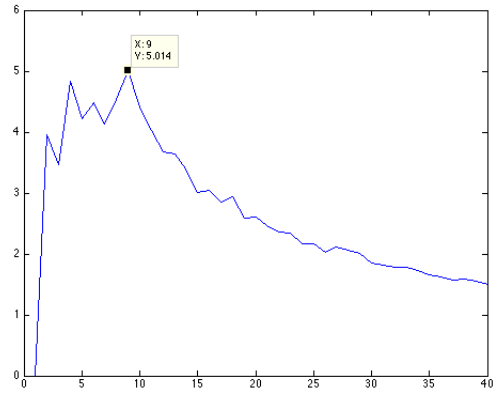
a) data 4.2



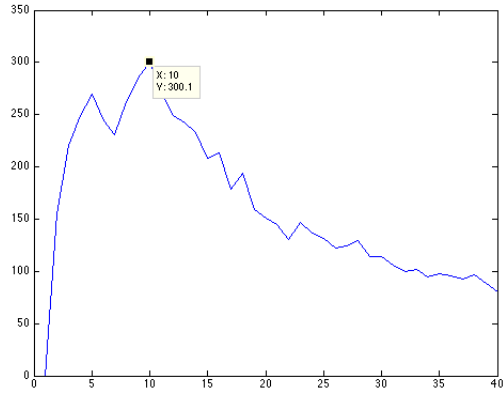
b) data 5.2



c) data 6.2

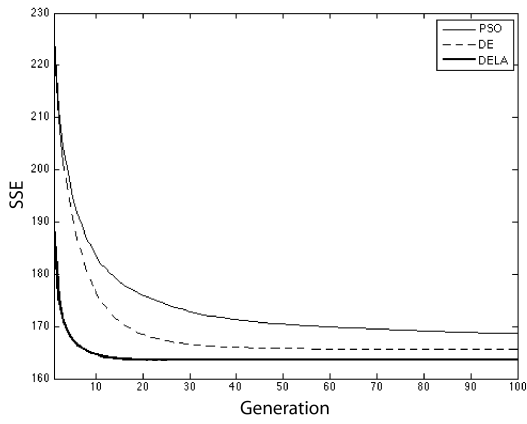


d) data 9.2

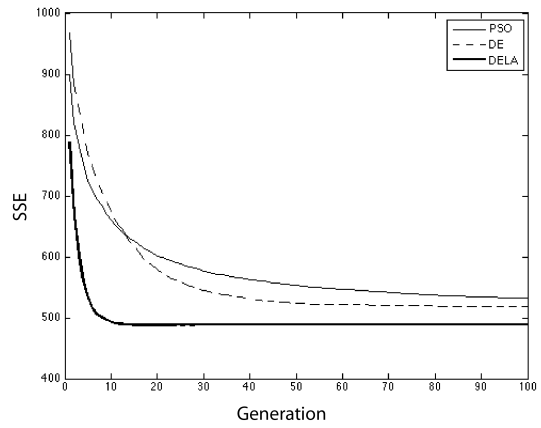


d) data 10.2

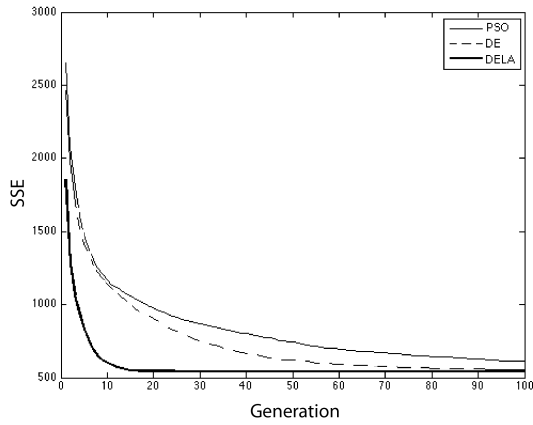
**Figura 6.10:** Comportamiento del Índice-i con el algoritmo DELA



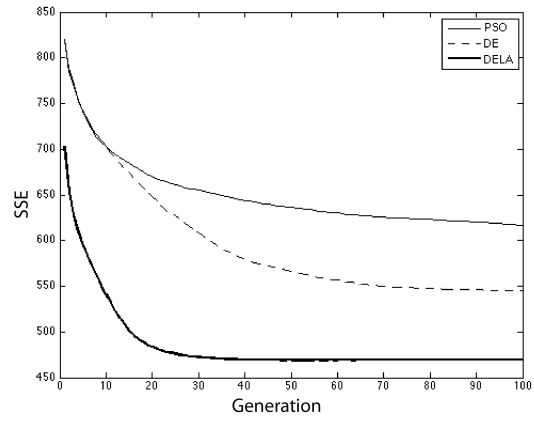
a) data 4.2



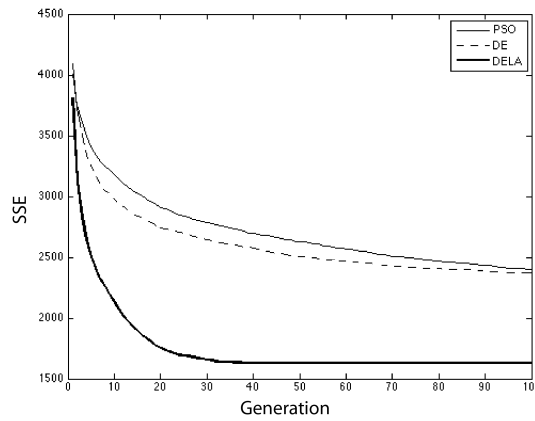
b) data 5.2



c) data 6.2



d) data 9.2



e) data 10.2

**Figura 6.11:** Velocidad de convergencia y calidad de la solución de DELA

Como se puede observar en la Figura 6.11, en los cuatro conjuntos de datos, el algoritmo DELA tiene una velocidad de convergencia muy superior a los algoritmos DE y PSO ya que en promedio DELA alcanza la convergencia antes de las primeras 20 generaciones mientras que el resto de los algoritmos alcanza la convergencia después de la generación 50. También se puede observar que en todos los casos, la calidad de las soluciones encontradas por DELA son mejores que las encontradas por el resto de los algoritmos.

Hasta el momento, se han mostrado algunas de las gráficas de convergencia que muestran el rendimiento de los algoritmos que están siendo comparados, también, se han visto las graficas que muestran el comportamiento del Índice-i al ser empleado por el algoritmo DELA, sin embargo, esto no es suficiente para concluir que DELA es una mejor alternativa con respecto a los algoritmos de DE y PSO, por lo que en la siguiente sección, se realizaran un conjunto de test estadísticos basados en rankings y pruebas de hipótesis, que ayudaran a ratificar lo visto en las Figuras 6.11 y 6.12.

#### **6.2.4 Análisis estadístico de los resultados experimentales**

Como se comento en el capítulo del estado del arte, los test estadísticos no paramétricos, han emergido como una metodología eficaz, robusta y asequible para la evaluación de nuevas propuestas de meta heurísticas y algoritmos evolutivos, alcanzando gran popularidad en la literatura [16]. En particular, se ha considerado la aplicación de la prueba de clasificación de Friedman no alineado, y el uso de pruebas multicompare de Holm como procedimiento *post-hoc* para saber qué algoritmos presentan estadísticamente un peor comportamiento que el algoritmo de referencia (control) propuesto (DELA).

Como primer paso en esta fase de validación de resultados, se buscó determinar si existe o no un comportamiento estadísticamente diferente entre los algoritmos estudiados, para ello, se empleó una técnica basada en rankings. En el test de Friedman Alineado [5], se calcula el rendimiento medio alcanzado por cada algoritmo en cada problema (valor de localización). Después, se calculan las diferencias entre el rendimiento obtenido por cada algoritmo con respecto al valor de localización. Este paso se repite para todos los algoritmos y problemas. Las diferencias resultantes (observaciones alineadas) se ordenan desde 1 hasta  $k - n$  de forma relativa unas con otras. A partir de ahí, el esquema de ranking es el mismo que el empleado por un procedimiento de comparaciones múltiples con

muestras independientes, como el test de Kruskal-Wallis. De este modo, los rankings asignados a las observaciones alineadas se denominan rankings alineados.

Como se puede observar en la tabla 6.4, después de aplicar el test de Friedman Alineado a los resultados experimentales, el algoritmo con mejor ranking fue el DELA seguido por la DE con un margen de diferencia muy grande. Asimismo, también se puede observar que el algoritmo con el peor desempeño fue el PSO. Estos resultados concuerdan perfectamente con los resultados obtenidos de manera empírica en la sección anterior.

<b>Algoritmo</b>	<b>Friedman</b>	<b>Friedman Alineado</b>
DELA	0.9999	3.5000
DE	2.3333	11.8333
PSO	2.6667	13.1666

**Tabla 6.4:** Rankings obtenidos por el test de Friedman y Friedman Alineado

Tanto el Test de Friedman como sus dos propuestas avanzadas (Friedman Alineado y Quade) no identifican las diferencias existentes entre el mejor algoritmo encontrado (denominado algoritmo de control) y el resto. Estos test se limitan a detectar la existencia o no de diferencias en todo el conjunto de resultados. Si se encuentran dichas diferencias, es necesario proceder con un procedimiento post-hoc de identificación de diferencias (basados en la distribución normal).

Una vez que se ha determinado (a través de los algoritmos expuestos en los párrafos anteriores) que los algoritmos DELA, DE y PSO presentan un comportamiento estadísticamente diferente entre ellos, se procederá a realizar una prueba post-hoc que ayudara a determinar cual de los algoritmos tiene el mejor desempeño.. Una vez obtenido el valor del estimado  $Z$ , es posible obtener a partir de él, el p-valor no ajustado correspondiente. La utilidad de estos procedimientos post-hoc radica en que son capaces

de calcular el P-Valor Ajustado (PVA) considerando la familia de hipótesis completa para cada pareja de algoritmos comparados.

Como se puede observar en la Tabla 6.5, se ha realizado un análisis donde se muestra la comparación del comportamiento de todos los algoritmos contra todos (prueba multicompare). Como se puede apreciar en la misma tabla, para el caso de la comparación DELA vs PSO, el comportamiento de DELA es muy superior ( $p_{\text{value}} = 0.003892$ ) al PSO con un valor de  $\alpha = 0.5$  lo que significa que se estima que DELA tendrá un comportamiento superior en más del 95% de los casos sin embargo dado el valor de  $\alpha$  anterior se puede apreciar que DELA será superior en un 99.97% de las veces. Para el caso de la comparación de DELA vs DE, vemos que nuevamente el algoritmo DELA tiene un mejor comportamiento que su oponente ( $p_{\text{value}} = 0.020921$ ) lo que tomando el mismo valor de  $\alpha$  significa que el algoritmo DELA tendrá un mejor comportamiento para el 99.8% de los casos. Finalmente al realizar la comparación entre PSO y DE podemos observar que cumplen la prueba de hipótesis que muestra que el comportamiento de ambos algoritmos es estadísticamente similar ( $p_{\text{value}} = 0.563705 > \alpha$ ).

<i>i</i>	Hipótesis	<i>p</i> value
1	PSO vs DELA	0.003892
2	DE vs DELA	0.020921
3	DE vs PSO	0.563702

**Tabla 6.5:** Comparación Post Hoc para  $\alpha=0.5$

### 6.3 Conclusiones parciales

En este capítulo, se presentó un nuevo modelo de heurística espacialmente estructurada de población descentralizada denominada DELA (Evolución Diferencial con Ajuste Local) que tiene como objetivo estimar el número de clases contenidas en un

conjunto de datos, para validar esto último, se realizaron un total de 11,700 experimentos en un total de 11 conjuntos de datos sintéticos con diferentes configuraciones en las clases, variando la compactación y separación así como la geometría lo cual, permitió probar el balance entre la exploración y la explotación del algoritmo presentado ante diferentes escenarios. También se realizaron pruebas con dos conjuntos de datos reales (Iris y Wine) los cuales son ampliamente utilizados para la validación de algoritmos de agrupamiento. Para realizar la comparación del rendimiento del algoritmo propuesto fueron utilizados dos de las meta heurísticas más utilizadas en el estado del arte para realizar la misma tarea siendo DELA el algoritmo con el mejor desempeño de todos.

## 6.4 Trabajo Futuro

Durante la etapa de experimentación se emplearon solamente dos conjuntos de datos reales (Iris y Wine) por lo que se desea probar el algoritmo con un conjunto de datos reales más extenso así mismo, se desea realizar una comparación entre múltiples algoritmos tanto basados en trayectorias como en poblaciones para realizar la estimación de la partición de tal forma que se pueda obtener la mejor configuración del algoritmo DELA.

Finalmente, dado que la naturaleza del algoritmo DELA es 100% paralela, se desea estudiar el comportamiento del algoritmo empleando distintos métodos de explotación asíncronos con base en un conjunto de reglas es decir, con lo estudiado hasta el momento, aún no se tienen elementos suficientes para descartar el empleo de métodos de explotación en función de las condiciones del espacio de búsqueda lo que significa que no hay razón para siempre utilizar un mismo método ( $\mu DE$ ) por lo que creemos que valdría la pena iniciar con el estudio de las hiper-heurísticas durante la fase de explotación.



# Conclusiones

# 7

Durante el presente trabajo, se realizó un amplio estudio del estado del arte con respecto a los algoritmos evolutivos con poblaciones descentralizadas los cuales son referenciados también por algunos autores como algoritmos evolutivos con poblaciones estructuradas. Las poblaciones estructuradas, son aquellas poblaciones en las cuales, para cualquier individuo, existe un conjunto de individuos que son sus vecinos, dicho conjunto de vecinos, es mas pequeño, que el tamaño de la población completa. En otras palabras, en lugar de que todos los individuos de la población sean considerados como posibles parejas para realizar los procesos de recombinación del algoritmo, solo aquellos individuos que se encuentre en el mismo vecindario serán considerados para tal fin.

Lo anterior, lleva a un nuevo concepto denominado *aislamiento por distancia* el cual implica, que todos los individuos de una población estructurada, solo son capaces de intercambiar información (interactuar) con un segmento de esta (vecindario) mientras que para poder acceder a la información del resto de los individuos de la población, es necesario recurrir a un proceso denominado *migración*.

Para poder establecer la comunicación entre los vecindarios y por ende, tener la capacidad de realizar el intercambio de información e individuos a través de diversas zonas o vecindarios, los algoritmos evolutivos con poblaciones estructuradas, hacen uso de una topología de interconexión la cual establece los mecanismos para el intercambio de individuos entre los vecindarios que conforman a la población global, de esta manera, se agrega un nuevo elemento (topología) en la etapa de diseño este tipo de algoritmos.

Una vez que se ha establecido la topología de interconexión entre vecindarios el paso final consiste en definir una política de migración de individuos la cual esta compuesta principalmente por tres etapas:

- Frecuencia de migración
- Política de selección de los individuos a ser migrados
- Política de selección de los individuos a ser reemplazados

La frecuencia de migración, establece las condiciones que se deben de cumplir para que se pueda realizar el intercambio de individuos. Una de las condiciones mas simples y que son ampliamente reportadas en el estado del arte dicta que la migración se debe de realizar cada cierto número de iteraciones o generaciones del algoritmo es decir, después de transcurrido cierto tiempo de evolución de los individuos, se debe de realizar de forma síncrona, el intercambio de individuos entre los vecindarios para poder continuar con el proceso evolutivo sin embargo como también se estudio en el estado del arte, se pueden considerar otros factores como disparadores para realizar el intercambio de individuos tales como: perdida de diversidad en las poblaciones, atasco en óptimos locales, etc.

Las políticas de selección y remplazo de los individuos durante el proceso de migración, define, como su nombre lo indica, cuales son aquellos individuos que deben ser seleccionados para ser enviados a otros vecindarios, incluso, en muchos trabajos del estado del arte, se reporta que los individuos que son enviados son clones es decir, que el vecindario solo envía una copia del individuo y no el individuo en si, lo que permite seguir trabajando con el individuo en ambas partes.

Con respecto a cuales individuos deben de ser seleccionados y enviados, existen varias posibilidades, tantas, como combinaciones de situaciones se presenten, por ejemplo, en la mayoría de los casos del estado del arte, se opta por enviar aquellos individuos con los

mejores valores de aptitud y por lo tanto, los que mas información pueden aportar durante los procesos de recombinación del algoritmo, sin embargo, en otros casos, se suele enviar a individuos seleccionados de forma aleatoria, lo cual busca contribuir en aumentar la diversidad de la población receptora.

Finalmente, con respecto a la política de remplazo, generalmente son remplazados aquellos individuos con el peor valor de aptitud, debido a que estos individuos ya no son capaces de aportar nueva información durante la etapa de recombinación.

Los dos principales modelos de algoritmos con poblaciones estructuradas que fueron estudiados y a partir de los cuales se plantearon tres nuevos modelos son: los algoritmos evolutivos distribuidos a los cuales también suele llamárseles algoritmos de islas y los algoritmos evolutivos celulares.

## **7.1 Algoritmo de fusión de islas**

En el modelo propuesto denominado Modelo de Fusión de Islas, se substituye al proceso de migración entre poblaciones, con un nuevo sistema de fusión el cual sirve como mecanismo para la dispersión de soluciones. De este modo, el modelo propuesto elimina la necesidad de elegir la topología de interconexión de las islas utilizadas por los modelos clásicos y como consecuencia, permite contar con mecanismos que permiten seleccionar de forma objetiva, las poblaciones que deben ser seleccionadas para realizar el intercambio de información. Las principales ventajas del modelo propuesto son:

- No se requiere de una topología de interconexión
- El número de islas es variable
- El tamaño de la población es variable
- Algoritmo síncrono/asíncrono

Como se mostró en la experimentación realizada en el capítulo 4, el algoritmo propuesto mejoró los tiempos de ejecución al ser comparado con el modelo clásico de islas debido a la disminución del número de veces que fue evaluada la función de aptitud originado por la disminución sistemática del tamaño de la población que resulta de la fusión y selección de los individuos sobrevivientes.

Los resultados también muestran, que además de mejorar las velocidades de convergencia con base en los tiempos obtenidos durante los experimentos, el modelo propuesto también es capaz de mejorar la calidad de las soluciones encontradas, resultado de la eliminación de la topología de conexión que como ya se explicó, no se ve limitada a los vecinos definidos según el criterio topológico. En la tabla 7.1, se muestran las mejoras alcanzadas por el modelo de fusión propuesto al ser comparado contra el modelo descentralizado original en un conjunto de 10 funciones extraídas del portafolio del CEC2013.

<b>Función</b>	<b>Calidad de la solución (%)</b>	<b>Número de evaluaciones de la función de aptitud (%)</b>	<b>Tiempos de ejecución (%)</b>
$f_1$	76.00	41.2	10.91
$f_2$	22.66	41.33	8.86
$f_3$	27.22	41.27	11.67
$f_4$	13.88	41.12	11.18
$f_5$	16.60	41.17	9.95
$f_6$	15.30	41.29	11.55
$f_7$	16.25	41.47	11.88
$f_8$	9.99	41.16	9.83
$f_9$	27.77	41.3	14.12
$f_{10}$	23.7	11	10.34

**Tabla 7.1:** Mejoras del algoritmo de fusión propuesto

Después de haber estudiado el estado del arte, quedo claro, que para los algoritmos evolutivos distribuidos se debe de prestar especial atención a la hora de seleccionar los siguientes parámetros:

- Cantidad de individuos migrados
- Frecuencia de migración
- Estrategia de selección de los individuos a ser migrados
- Número de islas
- Tamaño de las poblaciones de las islas

En el caso del modelo propuesto, la eliminación de la topología de interconexión entre subpoblaciones (como se puede observar en los resultados del capítulo 4 y en el resumen de la tabla 7.1) tubo un fuerte impacto en el rendimiento del algoritmo ya que dicha eliminación de la topología en conjunto con la fusión y selección de individuos sobrevivientes permitió disminuir los tiempos sin afectar la calidad de los resultados obtenidos lo cual ayuda a conformar que una mala selección de la topología de interconexión puede ocasionar la falsa convergencia de los algoritmos distribuidos.

Finalmente, es importante observar, que dentro de los algoritmos evolutivos distribuidos, el proceso de migración tiene como objetivo principal, apoyar el mecanismo de exploración del algoritmo ya que es precisamente el intercambio de individuos el factor que en la mayoría de los casos ayuda a sacar de los óptimos locales a las subpoblaciones mientras que por otro lado, la evolución de la población en si, esta intensificando la búsqueda en un cierto vecindario y como consecuencia apoyando al proceso de explotación.

Para el caso del algoritmo propuesto, el balance entre exploración y explotación, ha sido ampliado, ya que es precisamente el criterio de selección de las subpoblaciones a fusionar así como la selección de los individuos sobrevivientes, lo que permite redirigir la búsqueda del algoritmo y así potenciar o inhibir dichos procesos. Queda claro que una adecuada política de selección y fusión (con ayuda del experto) de las subpoblaciones ayudara a convergir el tan necesario balance entre exploración y explotación.

## 7.2 Modelo Celular Híbrido

En el capítulo 5, se presentó un nuevo modelo de heurística espacialmente estructurada de población descentralizada que tiene como objetivo estimar el número de clases contenidas en un conjunto de datos, para validar dicho modelo, se realizaron 19,500 experimentos sobre 10 conjuntos de datos sintéticos con diferentes configuraciones, las cuales varían la compactación y separación así como la geometría de las clases contenidas y que ayudó probar el balance entre la exploración y la explotación de los algoritmos presentados.

El algoritmo propuesto, se denominó *HCEA* y consiste en una hibridación entre un algoritmo genético celular (*cGA*) y un micro algoritmo de optimización por cúmulo de partículas con un método de ajuste local ( *$\mu$ PSOLA*). Para realizar la validación de los resultados obtenidos experimentalmente, se emplearon dos metodologías. La primera, consistió en obtener las medidas estadísticas clásicas como: mínimo, máximo, media, mediana y desviación estándar para todo el conjunto de resultados y así poder realizar una comparación a priori del rendimiento de los algoritmos mientras que la segunda metodología, se empleó un análisis estadístico basado en rankings, que ayudo a determinar, si estadísticamente existe un comportamiento diferente entre los algoritmos

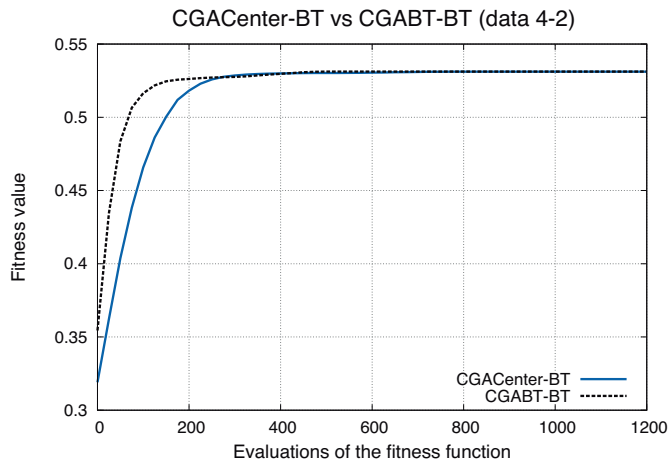
comparados y de ser así, entonces aplicar una prueba post-hoc para identificar cual de entre los algoritmos estudiados, es el que tiene el mejor desempeño.

Para la realización de los experimentos, se utilizaron diferentes configuraciones para del algoritmo propuesto y así determinar cual de ellas arroja los mejores resultados. Durante de las pruebas realizadas, se analizó el comportamiento de los siguientes parámetros:

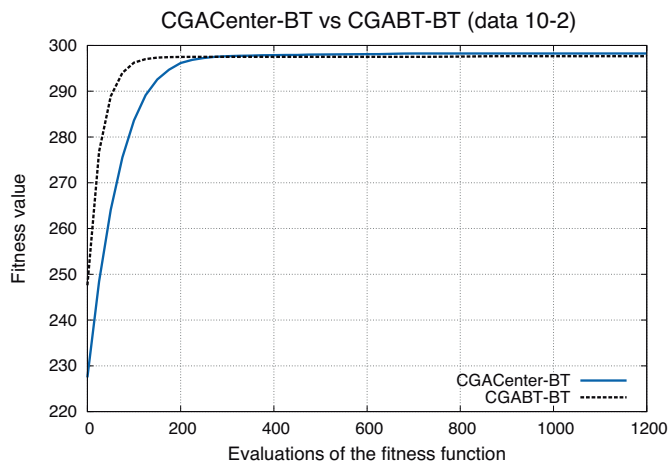
- Método de selección de los padres
- Tipo de vecindario
- Tamaño de vecindario

Como se discutió en la sección 5.2.1 del capítulo 5, durante la fase de experimentación, fueron utilizados dos métodos por el cGA para la selección de los padres: centro (C) y torneo binario (BT). Todos aquellos algoritmos denominados **cGACenter-BT** seleccionaron al primer padre del centro del vecindario, mientras que el segundo padre es seleccionado mediante torneo binario. La inclusión del individuo central, aseguró que todos los individuos del vecindario, fueran seleccionados al menos una vez durante el proceso de recombinación. Por otro lado, todos aquellos algoritmos denominados **cGABT-BT**, seleccionaron a ambos padres, empleando torneo binario lo cual como es claro, no asegura que todos los individuos sean seleccionados durante el proceso de recombinación relajando así, la presión de selección del algoritmo.

Como se puede observar en las graficas mostradas en las Figuras 7.1 y 7.1, el método de selección de los padres tiene el efecto de incrementar o disminuir la presión de selección del cGA. Tal cambio en el método de selección, incrementa o decrementa, la intensidad explotación del algoritmo, causando como consecuencia, una convergencia mas o menos rápida.



**Figura 7.1:** Método de selección de los padres: data 4-2

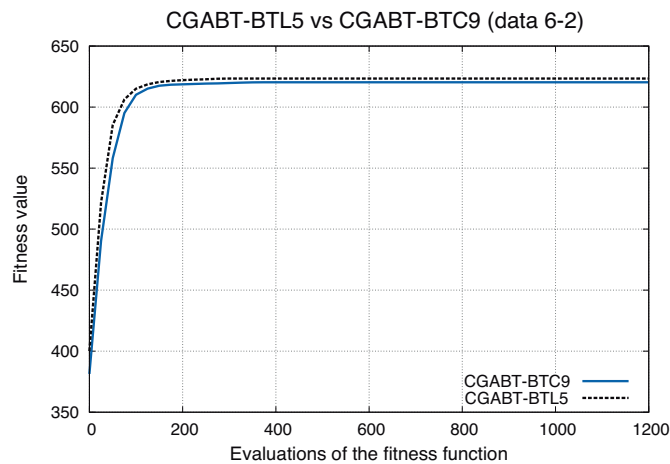


**Figura 7.2:** Método de selección de los padres: data 10-2

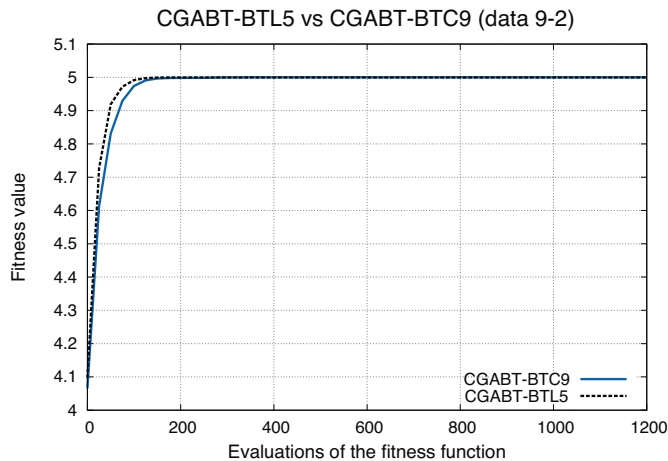
Para el caso de los problemas más complejos, es recomendable realizar una explotación del espacio de búsqueda mas detallada, por lo que partiendo de la base de los experimentos realizados, se debería emplear durante el mecanismo de recombinación de los individuos, la selección de todos los individuos lo que en el caso del modelo propuesto, se consigue empleando un cGACenter-BT. Para el caso de los problemas menos complejos, se puede relajar la intensidad de la explotación empleando para ello un cGABT-BT.



Una vez que se ha determinado que el mejor método de selección de los padres (sección 5.2.1) depende de la complejidad del problema y que para el conjunto de conjuntos de datos analizados es un cGABT-BT, se procedió a determinar el mejor tipo de vecindario. Se probaron dos de los tipos más comunes de vecindarios: vecindario compacto ( $Cn$ ) y vecindario lineal ( $Ln$ ), como se puede observar en las gráficas de las Figura 7.3 y 7.4, la convergencia más rápida tomando como base los conjuntos de datos 6.2 y 9.2, se alcanzaron cuando se empleó un vecindario de tipo lineal. Los mismos experimentos fueron realizados para todos los conjuntos de datos disponibles y los resultados fueron siempre los mismos, es decir, para el tipo de problema que se está solucionando la mejor alternativa es la elección de un vecindario lineal sin embargo dicha selección, no representa una mejora significativa con respecto a la otra ( $Cn$  vs  $Ln$ ).

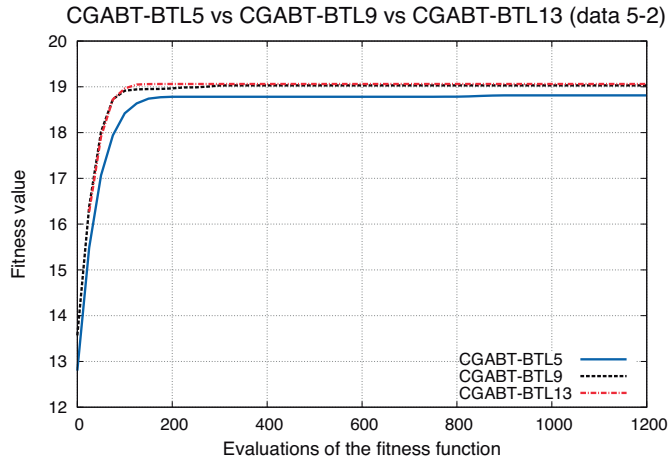


**Figura 7.3:** Tipo de vecindario: data 6-2

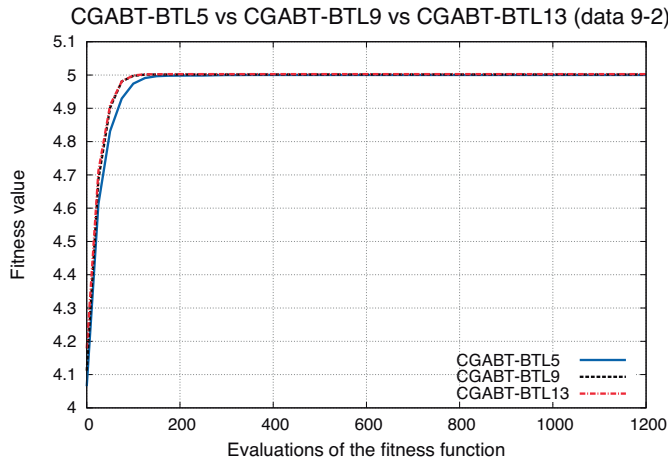


**Figura 7.4:** Tipo de vecindario: data 9-2

Para continuar con el estudio del algoritmo propuesto, se realizaron diversos experimentos tomando como base diferentes valores para el radio del vecindario. Considerando que la población del algoritmo celular que se diseñó fue apenas de 5x5, los radios elegidos para realizar el último conjunto de experimentos fueron de 5, 9 y 13 individuos. Como se muestra en la gráficas de la Figura 7.5 y 7.6, cuando el problema es simple, los vecindarios de radio pequeño tienen un mal rendimiento, lo que mejora si se emplean radio más grandes, para el caso de problemas más complejos, no hay diferencia si se emplean radios chicos o grandes lo que va en contra de lo que empíricamente se podría pensar es decir, para problemas simples radios pequeños y para problemas complejos radios grandes.

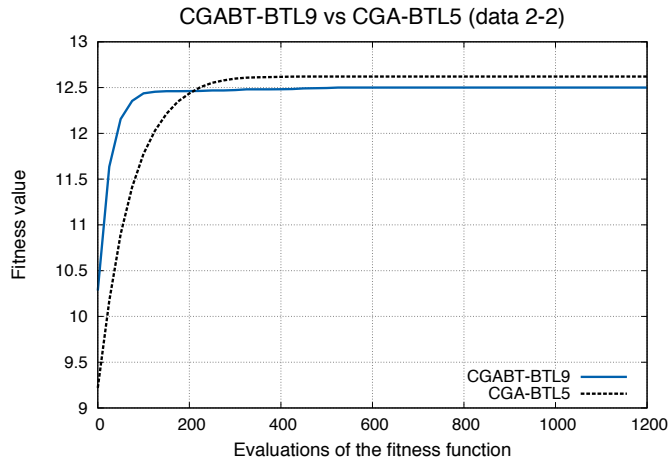


**Figura 7.5:** Tamaño del vecindario: data 5-2

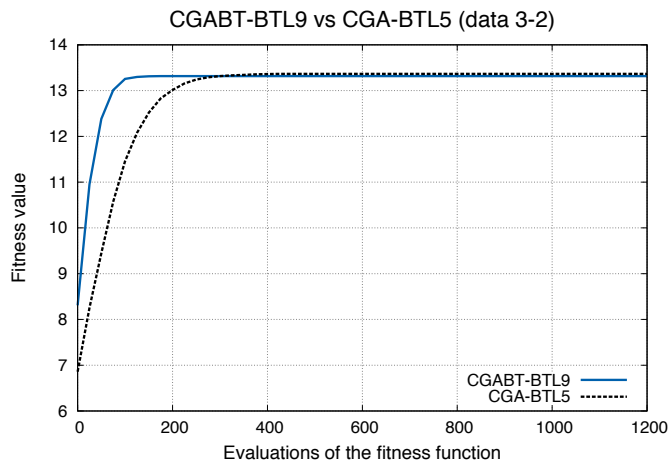


**Figura 7.6:** Tamaño del vecindario: data 9-2

Finalmente y tomando como base todos los resultados obtenidos, se realizo una comparación entre la combinación de los parámetros con mejor y peor desempeño para destacar la importancia de la selección de estos. En la gráficas de las Figuras 7.7 y 7.8, se muestra el resultado al combinar de forma adecuada el método de selección de los padres, tipo de vecindario y longitud de este.



**Figura 7.7:** Método de selección + tipo y tamaño de vecindario: data 2-2



**Figura 7.8:** Método de selección + tipo y tamaño de vecindario: data 3-2

Para dar soporte a los datos y resultados obtenidos experimentalmente, se realizó un conjunto de pruebas estadísticas que ayudaron a confirmar, que para el caso del problema estudiado (estimar el número de clases presentes en un conjunto de datos) el mejor algoritmo es el cGABT-BTL9 y el peor algoritmo es el cGACenter-BTL5. En la Tabla 7.2, se muestran los resultados arrojados por el análisis de Friedman alineado.

Algoritmo	Ranking
CGABT-BTL9	1.7778
CGABT-BTL13	1.8889
CGABT-BTC9	2.6667
CGABT-BTL5	3.6667
CGACenter-BTL5	5.0000

**Tabla 7.2:** Resultados del analisis de Friedman Alineado

Posteriormente cuando se determino que existían diferencias estadísticas significativas entre los diferentes algoritmos estudiados, se realizo una prueba post-hoc tomando como algoritmo de control al cGABT-BTL9 y se comparo con el resto de los algoritmos para determinar si estadísticamente este algoritmo era el que tenia el mejor comportamiento. En la tabla 7.3, se puede observar como la prueba de Holm arrojó una vez mas, que el mejor algoritmo es nuevamente el cGABT-BTL9.

<i>i</i>	Algoritmo	$Z=(R_0-R_i)/SE$	<i>p</i>	<i>Holm</i>
1	cGACenter-BTL5	4.323065	0.000015	<b>0.012500</b>
2	cGABT-BTL5	2.534210	0.011270	<b>0.016667</b>
3	cGABT-BTC9	1.192570	0.233038	<b>0.025000</b>
4	cGABT-BTL13	0.149071	0.881497	0.050000

**Tabla 7.3:** Resultados del analisis *post-hoc*

### 7.3 Modelo Diferencial Descentralizado

Tomando toda la experiencia acumulada durante el estudio y desarrollo de los dos primeros modelos se concluyo:

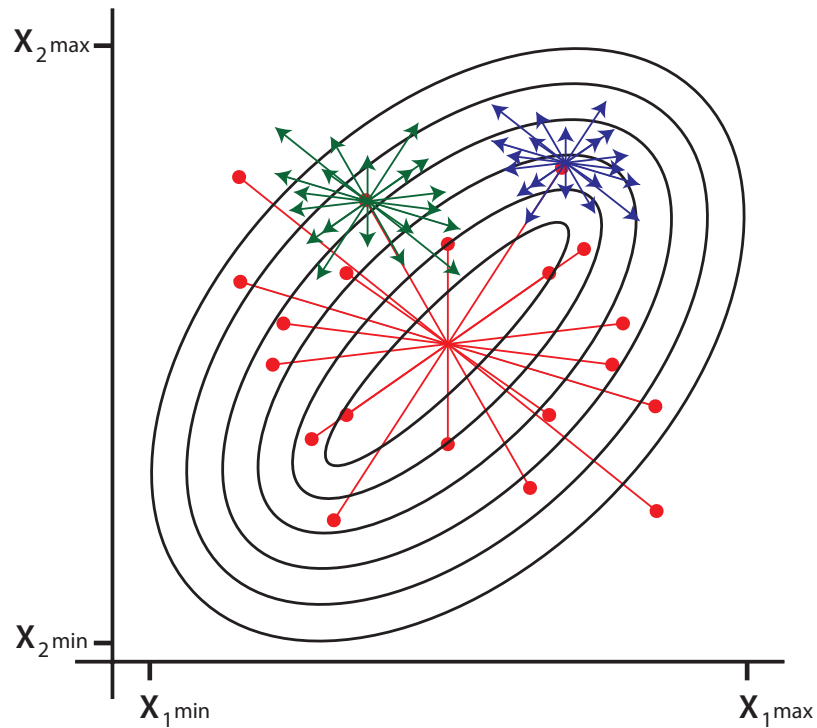
Los algoritmos con poblaciones estructuradas, al tener un mecanismo de aislamiento por distancia, describen (como se vio en el capítulo 4 durante el diseño del modelo basado en fusión de islas), un proceso de exploración y explotación más controlado.

Posteriormente, durante el estudio y diseño del modelo celular híbrido (capítulo 5), quedo claro, que existe un impacto directo entre el tamaño y tipo de vecindario empleado para la estructuración de la población y el balance entre la exploración y explotación del modelo, además, también quedo claro, que en función del método de selección de los padres para generar los nuevos individuos y por consecuencia realizar el intercambio de estos entre poblaciones afecta directamente la presión de selección del algoritmo lo cual puede llevar a una convergencia prematura si no se tiene en cuenta la complejidad del problema.

Tomando como base la experiencia acumulada, se diseñó el tercer y último modelo el cual incorporó todos los conocimientos adquiridos. El algoritmo de evolución diferencial con ajuste local (DELA por sus siglas en inglés), es un algoritmo evolutivo híbrido entre DE y  $\mu$ DE y se encuentra inspirado en los algoritmos de Hooke-Jeeves y Nelder y Mead. Para su funcionamiento, DELA emplea dos mecanismos durante el proceso de búsqueda denominados: convergencia global y convergencia local los cuales ayudan a equilibrar e balance entre a exploración y la explotación.

En la Figura 7.9, se muestra el comportamiento del algoritmo DELA en términos del proceso de explotación. Como se puede ver, la estructuración de la población empleando un proceso independiente para la búsqueda local ( $\mu$ DE), permite una eficiente y controlada explotación del espacio de soluciones en torno a un punto específico. Como se puede observar, las diferencias de los vectores empleados durante el proceso de explotación, pueden ser escalados de forma tal, que se intensifique o inhiba el proceso de explotación sobre un determinado punto. Esto último resulta de suma importancia, debido a que en los algoritmos meta heurísticos poblacionales clásicos (algoritmos genéticos, PSO, estrategias evolutivas, etc.), el proceso de explotación del espacio de búsqueda se consigue empleando métodos de recombinación de dos o mas individuos, lo que dependiendo del método de selección de dichos individuos (ruleta, torneo binario, aleatorio, etc), puede redirigir el

proceso en alguna dirección distinta del espacio que se quiere explotar (ocasionando un proceso de exploración en lugar de explotación) cosa que no ocurre en el caso de DELA, lo que ayuda a tener una convergencia más rápida y segura como se mostró en los resultados del capítulo 6.



**Figura 7.9:** Explotación con base en un punto específico

Para probar el funcionamiento del algoritmo DELA, se realizaron un total de 11,700 experimentos sobre 11 conjuntos de datos sintéticos con diferentes configuraciones en las clases, variando la compactación y separación así como la geometría de estas, se permitió probar el balance entre la exploración y la explotación del modelo presentado. También, se realizaron pruebas con dos conjuntos de datos reales (Iris y Wine) estos, fueron empleados debido a que son ampliamente utilizados para la validación de algoritmos de agrupamiento en el campo del reconocimiento de patrones. Finalmente, para realizar la comparación del

rendimiento del algoritmo propuesto, fueron utilizadas dos de las meta heurísticas mas utilizadas en el estado del arte para realizar la misma tarea y a través del uso de herramientas estadísticas para análisis de datos, se llego a la conclusión que el algoritmo DELA es el que presenta el mejor comportamiento de todos.

Como se puede observar en la Tabla 7.4, después de aplicar la prueba Friedman alineado, el algoritmo DELA mostro un comportamiento mucho muy superior al de sus rivales (DE y PSO), lo cual es una consecuencia como se ha venido comentando a lo largo del trabajo, del correcto balance entre la exploración y la explotación.

Algoritmo	Friedman	Friedman Alineado
DELA	0.9999	3.5000
DE	2.3333	11.8333
PSO	2.6667	13.1666

**Tabla 7.4:** Rankings obtenidos por el test de Friedman y Friedman Alineado

Aquí vale la pena realizar una pausa. Si se observan los análisis estadísticos basados en rankings de los dos últimos métodos diseñados, se puede observar claramente que para el caso del algoritmo DELA la diferencia con el segundo lugar es de casi cuatro ordenes de magnitud lo cual muestra que el balance entre exploración y explotación del algoritmo es muy superior al de otras meta-heurísticas como en este caso son la evolución diferencial y la optimización por cúmulo de partículas.

Como se puede observar en la Tabla 7.5, se realizó un análisis donde se muestra la comparación del comportamiento de todos los algoritmos contra todos (prueba multicompare). Como se puede apreciar en la misma tabla, para el caso de la comparación DELA vs PSO, el comportamiento de DELA es muy superior ( $p_{value} = 0.003892$ ) al PSO con un valor de  $\alpha = 0.5$  lo que significa, que se estima que DELA tendrá un comportamiento



superior en mas del 95% de los casos sin embargo dado el valor de  $\alpha$  anterior se puede apreciar que DELA será superior en un 99.97% de las veces.

Para el caso de la comparación de DELA vs DE, vemos que nuevamente el algoritmo DELA tiene un comportamiento superior al su oponente ( $p_{\text{value}} = 0.020921$ ) lo que tomando el mismo valor para  $\alpha$ , significa que el algoritmo DELA tendrá un comportamiento superior para el 99.8% de los casos. Finalmente al realizar la comparación entre PSO y DE podemos observar que se cumple la prueba de hipótesis que identifica que el comportamiento de ambos algoritmos es estadísticamente similar ( $p_{\text{value}} = 0.563705 > \alpha$ ).

<i>i</i>	Hipótesis	<i>p</i> value
1	PSO vs DELA	0.003892
2	DE vs DELA	0.020921
3	DE vs PSO	0.563702

**Tabla 7.5:** Comparación Post Hoc para  $\alpha=0.5$

Finalmente y para terminar, después de analizar las pruebas post-hoc (ver tabla 7.5), se concluyo definitivamente, que el rendimiento del algoritmo DELA supera claramente a los dos algoritmos más utilizados dentro del estado del arte en los últimos años para la solución del problema de estimar el número de clases contenidas dentro de un conjunto de datos.

## 7.4 Últimas palabras

Después de 4 años de ardua investigación y un amplio estudio y análisis del estado del arte en el campo de los algoritmos evolutivos espacialmente estructurados, se ha llegado a la conclusión de que estos proveen de un conjunto de mecanismos insuperables, que hacen que estas técnicas, sean muy efectivas a la hora de resolver problemas complejos tal y como se observó en los resultados y conclusiones de los capítulos 4, 5 y 6.

Después de la experiencia obtenida durante el diseño, prueba y comprobación de los resultados para los modelos desarrollados, se llegó a la conclusión de que la técnica de combinar dos o mas algoritmos (hibridación), genera buenos resultados, incluso, como se puede observar en los capítulos 5 y 6, no es necesario emplear poblaciones de individuos muy grandes, ya que para el caso de los algoritmos y problemas propuestos, se utilizaron poblaciones de entre 5 y 25 individuos, lo cual, de manera directa, ayudó a disminuir dramáticamente el número de veces que es evaluada la función de aptitud, situación que en muchas ocasiones, es utilizada como métrica para la comparación entre algoritmos.

# Referencias

**[1]** Noda, E.; Coelho, A.L.V.; Ricarte, I.L.M.; Yamakami, A.; Freitas, A.A.; , "Devising adaptive migration policies for cooperative distributed genetic algorithms," Systems, Man and Cybernetics, 2002 IEEE International Conference on , vol.6, no., pp. 6 pp. vol.6, 6-9 Oct. 2002

**[2]** Borovska, P.; Lazarova, M.; , "Migration Policies for Island Genetic Models on Multicomputer Platform," Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on , vol., no., pp.143-148, 6-8 Sept. 2007

**[3]** Hijaze, M.; Corne, D.; , "An Investigation Of Topologies and migration schemes for asynchronous distributed evolutionary algorithms," Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on , vol., no., pp.636-641, 9-11 Dec. 2009

**[4]** Sang-Keon Oh; Cheol Taek Kim; Ju-Jang Lee; , "Balancing the selection pressures and migration schemes in parallel genetic algorithms for planning multiple paths," Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on , vol.4, no., pp. 3314- 3319 vol.4, 2001

**[5]** Ningchuan Xiao and Marc P. Armstrong. 2003. A specialized island model and its application in multiobjective optimization. In Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII (GECCO'03), Erick Cant\&\#250;-Paz (Ed.). Springer-Verlag, Berlin, Heidelberg, 1530-1540.

**[6]** Guangyuan Liu; Jingjun Zhang; Ruizhen Gao; Yang Sun; , "A improved parallel genetic algorithm based on fixed point theory for the optimal design of multi-body model vehicle suspensions," Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on , vol., no., pp.430-433, 8-11 Aug. 2009

**[7]** Yussof, S.; Razali, R.A.; Ong Hang See; Ghapar, A.A.; Din, M.M.; , "A Coarse-Grained Parallel Genetic Algorithm with Migration for Shortest Path Routing Problem," High Performance Computing and Communications, 2009. HPCCC '09. 11th IEEE International Conference on , vol., no., pp.615-621, 25-27 June 2009

**[8]** Ko-Ming Chiu; Jing-Sin Liu; , "Robot routing using clustering-based parallel genetic algorithm with migration," Merging Fields Of Computational Intelligence And Sensor Technology (CompSens), 2011 IEEE Workshop On , vol., no., pp.42-49, 11-15 April 2011

**[9]** Ru Zhongliang; Zhao Hongbo; Zhu Chuanrui; , "A parallel genetic algorithm for optimal designing of frame structure," Environmental Science and Information Application Technology (ESIAT), 2010 International Conference on , vol.1, no., pp.751-754, 17-18 July 2010

**[10]** Feng Liu; Yonghua Zeng; Yousuo Zou; Juan Liu; Huaibei Zhou; , "Parallel island-based estimation of distribution algorithms for wireless network planning," Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on , vol.2, no., pp. 1056- 1059, 23-26 Sept. 2005

**[11]** Hee-Myung Jeong; Hwa-Seok Lee; June-Ho Park; , "Application of parallel particle swarm optimization on power system state estimation," Transmission & Distribution Conference & Exposition: Asia and Pacific, 2009 , vol., no., pp.1-4, 26-30 Oct. 2009

**[12]** Yuan Xiao-lei; Bai Yan; , "Stochastic nonlinear system identification using multi-objective multi-population parallel genetic programming," Control and Decision Conference, 2009. CCDC '09. Chinese , vol., no., pp.1148-1153, 17-19 June 2009

**[13]** Yu, W.; Zhang, W.: , "Study on Function Optimization Based on Master-slave Structure Genetic Algorithm," Signal Processing, 2006 8th International Conference on , vol.3, no., 16-20 2006

**[14]** Czech, Z.J.; Czarnas, P.; , "Parallel simulated annealing for the vehicle routing problem with time windows," Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on , vol., no., pp.376-383, 2002

**[15]** Caprio, Griffin; , "Parallel Metaheuristics," Distributed Systems Online, IEEE , vol.9, no.8, pp.3, Aug. 2008

**[16]** Quagliarella, D.; Vicini, A.; , "Sub-population policies for a parallel multiobjective genetic algorithm with applications to wing design," Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on , vol.4, no., pp.3142-3147 vol.4, 11-14 Oct 1998

**[17]** Mei Yue; Tao Hu; Baoping Guo; Xuan Guo; , "The research of parameters of genetic algorithm and comparison with particle swarm optimization and shuffled frog-leaping algorithm," Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on , vol.1, no., pp.77-80, 19-20 Dec. 2009

**[18]** Lin Gao; Junrong Xia; Yiping Dai; , "Modeling of combined cycle power plant based on a genetic algorithm parameter identification method," Natural Computation (ICNC), 2010 Sixth International Conference on , vol.7, no., pp.3369-3373, 10-12 Aug. 2010

**[19]** T. Hiroyasu, M. Miki, M.Sano, Y.Tanimura, M.Hamasaki: Dual Individual Distributed Genetic Algorithm. CISE, Vol. 38, No. 11, pp. 990-995, (2002)

**[20]** Algorithm(PfGA) with Steady-State Genetic Algorithm. IEICE, Vol. J81-D-II, No. 5, pp. 1455-1459, (1998)

**[21]** S. Adachi, H.Sawai: Effect of Migration Methods in Parallel Distributed Parameter-free Genetic Algorithm. IEICE, Vol. J83-D-I, No. 8, pp. 834-843, (2000)

**[22]** Takuma Jumonji, Goutam Chakraborty, Hiroshi Mabuchi, Masafumi Matsuhara: A novel Distributed Genetic Algorithm Implementation with Variable Number of Islands. IEEE CEC, pp. 4698-4705, (2007)

**[23]** Luis de la Ossa, José A. Gámez and José M. Puerta: Initial approaches to the application of island-based parallel EDAs in continuous domains. vol., no., pp.4698-4705, 25-28 Sept. 2007.

**[24]** Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos A. Coello Coello. 2006. A comparative study of differential evolution variants for global optimization. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06). ACM, New York, NY, USA, 485-492

**[25]** Bandyopadhyay, S. and Maulik, U. (2002a). An evolutionary technique based on k-means algorithm for optimal clustering in rn. Information Sciences, 146(1):221– 237.

**[26]** Bandyopadhyay, S. and Maulik, U. (2002b). Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition, 35(6):1197–1208.

**[27]** Bellis, M. A., Jarman, I., Downing, J., Perkins, C., Beynon, C., Hughes, K., and Lisboa, P. (2012). Using clustering techniques to identify localities with multiple health and social needs. *Health & place*, 18(2):138–143.

**[28]** Cabrera, J. C. F. and Coello, C. A. C. (2007). Handling constraints in particle swarm optimization using a small population size. In *MICAI 2007: Advances in Artificial Intelligence*, pages 41–51. Springer.

**[29]** Cabrera, J. C. F. and Coello, C. A. C. (2010). Micro-mopso: a multi-objective particle swarm optimizer that uses a very small population size. In *Multi-Objective Swarm Intelligent Systems*, pages 83–104. Springer.

**[30]** Chang, L., Duarte, M. M., Sucar, L., and Morales, E. F. (2012). A bayesian approach for object classification based on clusters of sift local features. *Expert Systems With Applications*, 39(2):1679–1686.

**[31]** Correa-Morris, J., Espinosa-Isidron, D. L., and Alvarez- Nadiozhin, D. R. (2010). An incremental nested partition method for data clustering. *Pattern Recognition*, 43(7):2439–2455.

**[32]** Cortina-Borja, M. (2012). Handbook of parametric and nonparametric statistical procedures. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 175(3):829–829.

**[33]** Das, S., Abraham, A., and Konar, A. (2008). Automatic clustering using an improved differential evolution algorithm. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(1):218–237.

**[34]** Davies David L. Bouldin, D. W. (1979). A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2:224–227.

**[35]** Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. Lecture notes in computer science, 1917:849–858.

**[36]** F. Martínez-Álvarez, A. Troncoso, J. C. R. J. S. A.-R. (2011). Energy time series forecasting based on pattern sequence similarity. In IEEE Transactions on Knowledge and Data Engineering, pages 1230–1243 vol.23 No. 8. IEEE.

**[37]** Franek, L., Abdala, D., Vega-Pons, S., and Jiang, X. (2011). Image segmentation fusion using general ensemble clustering methods. Computer Vision–ACCV 2010, pages 373–384.

**[38]** Garcia, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec?2005 special session on real parameter optimization. Journal of Heuristics, 15(6):617–644.

**[39]** Goldberg, D. E. (1989). Sizing populations for serial and parallel genetic algorithms. In Proceedings of the 3rd International Conference on Genetic Algorithms, pages 70–79. Morgan Kaufmann Publishers Inc.

**[40]** Grosan, C., Abraham, A., and Ishibuchi, H. (2007). Hybrid evolutionary algorithms. Springer Publishing Company, Incorporated.



**[41]** Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108.

**[42]** Hong, Y., Kwong, S., Chang, Y., and Ren, Q. (2008). Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. *Pattern Recognition*, 41(9):2742–2756.

**[43]** J Cao, Z Wu, J. W. W. L. (2012). Towards informationtheoretic k-means clustering for image indexing. *Signal Processing*, 39(2):1–12.

**[44]** Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

**[45]** Jarboui, B., Cheikh, M., Siarry, P., and Rebai, A. (2007). Combinatorial particle swarm optimization (cpso) for partitional clustering problem. *Applied Mathematics and Computation*, 192(2):337–345.

**[46]** Kanade, P. M. and Hall, L. O. (2003). Fuzzy ants as a clustering concept. In *Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American*, pages 227–232. IEEE.

**[47]** Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, pages 1942–1948 vol.4. IEEE.

**[48]** Kodratoff, Y. and Michalski, R. S. (1990). *Machine learning: an artificial intelligence approach*, volume 3. Morgan Kaufmann Publishers.

**[49]** Krishnakumar, K. (1989). Micro-genetic algorithms for stationary and non-stationary function optimization. In 1989 Advances in Intelligent Robotics Systems Conference, pages 289–296. International Society for Optics and Photonics.

**[50]** Kwedlo, W. (2011). A clustering method combining differential evolution with the means algorithm. *Pattern Recognition Letters*, 32(12):1613–1621.

**[51]** Lau, R. Y., Li, Y., Song, D., and Kwok, R. C. W. (2008). Knowledge discovery for adaptive negotiation agents in e-marketplaces. *Decision Support Systems*, 45(2):310–323.

**[52]** Lopez-Ortega, O. and Rosales, M.-A. (2011). An agentoriented decision support system combining fuzzy clustering and the ahp. *Expert Systems with Applications*, 38(7):8275–8284.

**[53]** Lu, Y., Lu, S., Fotouhi, F., Deng, Y., and Brown, S. J. (2004). Fgka: a fast genetic k-means clustering algorithm. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 622–623. ACM.

**[54]** Martinez-Trinidad, J. F. and Guzman-Arenas, A. (2001). The logical combinatorial approach to pattern recognition, an overview through selected works. *Pattern Recognition*, 34(4):741–751.

**[55]** Maulik U, B. S. (2002). Performance evaluation of some clustering algorithms and validity indices. *IEEE T. Pattern*, 24(12):1650–1654.

**[56]** Omran, M., Engelbrecht, A. P., and Salman, A. (2005). Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(03):297–321.

**[57]** Parsopoulos, K. E. (2009). Cooperative micro-differential evolution for high-dimensional problems. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pages 531–538. ACM.

**[58]** Rousseeuw, P. J. and Kaufman, L. (1990). Finding groups in data: An introduction to cluster analysis. John, John Wiley & Sons.

**[59]** Saha, I., Maulik, U., and Bandyopadhyay, S. (2009). A new differential evolution based fuzzy clustering for automatic cluster evolution. Advance Computing Conference, 2009. IACC 2009. IEEE International, pages 706–711.

**[60]** Taher Niknam, B. B. F. and Nayeripour, M. (2008). An efficient hybrid evolutionary algorithm for cluster analysis. World Applied Sciences Journal, 4(2):300–307.

**[61]** Vega-Pons, S., Ruiz-Shulcloper, J., and Guerra-Gandon, A. (2011). Weighted association based methods for the combination of heterogeneous partitions. Pattern Recognition Letters, 32(16):2163–2170.

**[62]** Villa, A., Chanussot, J., Benediktsson, J. A., Jutten, C., and Dambreville, R. (2012). Unsupervised methods for the classification of hyperspectral images with low spatial resolution. Pattern Recognition.

**[63]** Viveros-Jiménez, F., Mezura-Montes, E., and Gelbukh, A. (2009). Elitistic evolution: a novel micro-population approach for global optimization problems. In Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on, pages 15–20. IEEE.

**[64]** Viveros Jim´enez, F., Mezura Montes, E., and Gelbukh, A. (2012). Empirical analysis of a micro-evolutionary algorithm for numerical optimization. *Int. J. Phys. Sci*, 7:1235–1258.

**[65]** Wang, X., Yang, C., and Zhou, J. (2009). Clustering aggregation by probability accumulation. *Pattern Recognition*, 42(5):668–675.

**[66]** Wei-Ping Lee, S.-W. C. (2010). Automatic clustering with differential evolution using a cluster number oscillation method. *Intelligent Systems and Applications*, pages 218–237.

**[67]** Xie, X. L. and Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 13(4).

**[68]** Xu, R., Wunsch, D., et al. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678.

**[69]** Y. Yang, Y. Liao, G. M. and Lee, J. (2011). A hybrid feature selection scheme for unsupervised learning and its application in bearing fault diagnosis. *Expert Systems With Applications*, 38(9):1311–1320.

**[70]** Yan, H., Chen, K., Liu, L., and Yi, Z. (2010). Scale: a scalable framework for efficiently clustering transactional data. *Data mining and knowledge Discovery*, 20(1):1–27.