

ALGORITMO DE ESTIMACIÓN DE DISTRIBUCIÓN

**Para la Resolución de Problemas de
Optimización con Restricciones**

**David Esparza Alba
Arturo Hernández Aguirre**

Centro de Investigación en Matemáticas, A.C.

**Tesis de:
Maestría en Ciencias
con Especialidad en
Computación y Matemáticas Industriales**



CIMAT

Guanajuato, Gto. Noviembre de 2009

Centro de Investigación en Matemáticas

Maestría en Ciencias con Especialidad en Computación y Matemáticas
Industriales

EDA para la Resolución de Problemas de Optimización con Restricciones

por

David Esparza Alba

Director de Tesis: Dr. Arturo Hernández Aguirre

Guanajuato, Gto. México. 3 de Noviembre de 2009

*A mis padres Jorge y Concepción
A mi hermano Jorge*

Agradecimientos

A Dios, por permitirme llegar hasta este momento de mi vida.

A mi asesor, el Dr. Arturo Hernández Aguirre, por los valiosos consejos y por el tiempo dedicado para llevar a cabo esta investigación.

A los revisores de este documento, el Dr. Salvador Botello Rionda y el Dr. Alonso Ramírez Manzanares, por todos sus comentarios y sugerencias.

Al CIMAT, por facilitarme las herramientas necesarias para llevar a cabo este proyecto de investigación.

Al CONACyT, por haber financiado en su totalidad mis estudios de maestría.

A mi madre Concepción, por su ayuda y consejos en cada momento de mi vida.

A mi hermano Jorge, por el apoyo incondicional en cada instante de mi vida.

A mis compañeros de generación, por brindarme su amistad, y por todos los momentos que pasamos juntos. Gracias a: Emma, Gustavo, Memo, Roberto, Juan Carlos, Valentin, Miguel y Mario.

A mi amigo, compañero y maestro, Rogelio Salinas, por el tiempo empleado para revisar este documento, y por los valiosos consejos durante los últimos años de mi vida.

A mi amigo y maestro, Oscar Davalos, por ser una parte importante de mi vida.

A Cindy, por su valiosa amistad y por el apoyo brindado en momentos difíciles.

Índice general

Índice general	I
1 Introducción	1
1.1. Justificación	2
1.2. Contenido	3
1.3. Objetivos	3
2 Computación Evolutiva	5
2.1. Algoritmo Genético (AG)	6
2.1.1. Representación de un Algoritmo Genético	7
2.1.2. Población Inicial	8
2.1.3. Función de Aptitud	8
2.1.4. Selección	9
2.1.4.1. Selección Aleatoria	9
2.1.4.2. Selección Proporcional	10
2.1.4.3. Selección por Torneo	11
2.1.4.4. Selección por Rango	11
2.1.5. Reproducción o Cruza	13
2.1.6. Mutación	14
2.1.7. Parámetros de un Algoritmo Genético	15
2.1.8. Bloques Constructores y Teorema de los Esquemas	16
2.1.9. Funciones Deceptivas	17
2.1.10. El Algoritmo	19
2.2. Estrategias Evolutivas (EEs)	19
2.2.1. Representación	20
2.2.2. Selección	20
2.2.2.1. Selección (μ, λ)	20
2.2.2.2. Selección $(\mu + \lambda)$	21
2.2.3. Reproducción	21

2.2.3.1. Reproducción Sexual	21
2.2.3.2. Reproducción Panmítica	22
2.2.4. Mutación	22
3 Algoritmos de Estimación de Distribución	25
3.1. EDAs con Representación Discreta	27
3.1.1. EDAs sin Dependencia	28
3.1.1.1. UMDA	28
3.1.1.2. PBIL	29
3.1.2. EDAs con Dependencias Bivariadas	30
3.1.2.1. MIMIC	30
3.1.2.2. BMDA	32
3.1.2.3. Árbol de Dependencias de Chow y Liu	35
3.1.3. EDAs con Dependencias Múltiples	38
3.1.3.1. Algoritmo PADA	38
3.1.3.2. Algoritmo BOA	40
3.2. EDAs con Representación Real	41
3.2.1. EDAS sin Dependencias	41
3.2.1.1. UMDAc	42
3.2.1.2. PBILc	42
3.2.2. EDAS con Dependencias	43
3.2.2.1. MIMICc	43
3.2.2.2. Algoritmo EMNA	44
4 Manejo de Restricciones en Algoritmos Evolutivos	47
4.1. Funciones de Penalización	50
4.1.1. Pena de Muerte (Death Penalty)	50
4.1.2. Penalización Estática	50
4.1.2.1. Stochastic Ranking	51
4.1.3. Penalización Dinámica	52
4.1.4. Penalización Adaptativa	53
4.1.5. Recocido Simulado	55
4.1.6. Penalizaciones Co-evolutivas	55
4.1.7. Algoritmo Genético Segregado	56
4.1.8. Penalización Difusa	56
4.2. Algoritmos de Reparación	56
4.3. Restricciones y Objetivos	57
4.3.1. Co-evolución	57
4.3.2. Superioridad de Puntos Factibles	57
4.3.3. Memoria Conductista	58
4.3.4. Técnicas Multi-objetivo para Manejo de Restricciones	58

5 Algoritmo EMEDA	59
5.1. Algoritmo EM	62
5.1.1. Mezcla de Gaussianas	63
5.1.2. Algoritmo K-Medias	67
5.2. Evaluación de Individuos	75
5.3. Método de Selección	79
5.4. Control de Diversidad	81
5.4.1. Mutaciones del Mejor Individuo	83
6 Resultados	91
6.1. Resultados de la función g01	92
6.2. Resultados de la función g02	94
6.3. Resultados de la función g03	95
6.4. Resultados de la función g04	98
6.5. Resultados de la función g05	99
6.6. Resultados de la función g06	102
6.7. Resultados de la función g07	103
6.8. Resultados de la función g08	105
6.9. Resultados de la función g09	106
6.10. Resultados de la función g10	108
6.11. Resultados de la función g11	111
6.12. Resultados de la función g12	114
6.13. Resultados de la función g13	115
6.14. EMEDA vs. Stochastic Ranking	118
7 Conclusiones	121
A Funciones de Benchmark	125
B Algunas Variantes Exploradas	131
Bibliografía	135
Índice de figuras	139
Índice de cuadros	141

Capítulo 1

Introducción

La optimización se refiere al hecho de encontrar el mejor elemento entre un conjunto de alternativas disponibles, con la intención de minimizar o maximizar una función objetivo. Las soluciones están compuestas de cadenas de información, llamadas variables, las cuales pueden estar sujetas a restricciones, de manera que el propósito principal consiste en encontrar el valor de dichas variables que optimicen la función objetivo.

El proceso de identificar la función objetivo, las variables y las restricciones de un determinado problema se le conoce como *modelado*. La construcción del modelo es el primer paso en el proceso de optimización. Un modelo demasiado simple puede ser insuficiente para encontrar la solución de problemas complejos, por otro lado, un modelo demasiado complicado puede llegar a ser más difícil de resolver que el problema original.

Una vez que el modelo ha sido definido, un algoritmo de optimización puede ser utilizado para encontrar la solución. Ya que no existe un algoritmo de optimización universal, se han propuesto una gran cantidad de algoritmos capaces de resolver determinados problemas.

La computación evolutiva ha demostrado ser una herramienta eficiente para la resolución de diversos problemas de optimización. Existen técnicas que van desde los algoritmos genéticos (AGs), hasta los algoritmos de estimación de distribución (EDAs); estos últimos representan un área relativamente nueva dentro de la computación evolutiva, por lo que existe un extenso campo de investigación respecto a ellos.

Los AGs funcionan en base a la simulación del proceso evolutivo y se compon-

nen de tres partes fundamentales: selección, reproducción o cruza y mutación. El algoritmo consiste en elegir a los mejores individuos de la población, con la intención de que al recombinarse se produzcan mejores soluciones que formen parte de la siguiente generación; dichas soluciones son susceptibles de sufrir mutaciones, es decir, experimentar pequeños cambios en su información genética, de manera que exista mayor diversidad en la población.

A diferencia de los AGs, los EDAs carecen de operadores de cruza y mutación, por lo que los nuevos individuos son generados mediante un modelo de probabilidad capaz de considerar dependencias entre las variables, el cual se encarga de aproximar la distribución conjunta de una muestra de los mejores individuos de la población.

El objetivo de esta investigación es el de proponer un algoritmo de estimación de distribución (EDA), basado en el concepto de mezclas de distribuciones normales, con la finalidad de resolver problemas de optimización en un espacio de búsqueda multidimensional con restricciones.

1.1. Justificación

Dentro de la computación evolutiva, existen diferentes técnicas capaces de trabajar con problemas de optimización con restricciones en espacios de búsqueda n-dimensionales, lo cual deja una pregunta para la reflexión.

- ¿Por qué trabajar con un EDA y no con otra técnica como los algoritmos genéticos?

Aunque es cierto que el algoritmo genético ha demostrado ser una herramienta poderosa para resolver problemas de optimización, existe un grupo de funciones difíciles de resolver, ya que tienen como objetivo tratar de engañar al algoritmo genético. A estas funciones se les conoce como *funciones deceptivas*. La principal razón por la cual los EDAs han tomado fuerza en los últimos años, y por la cual se eligió como técnica para este trabajo de investigación, es que son capaces de resolver problemas que los algoritmos genéticos no pueden, tal es el caso de las funciones deceptivas. Otra característica de los EDAs es que pueden resolver con mayor facilidad, problemas cuya solución con otros métodos es difícil de encontrar.

Una motivación importante para llevar a cabo este trabajo de investigación, es el hecho de que la resolución de problemas de optimización con restricciones

mediante EDAs, es un área que no ha sido explorada a profundidad, por lo que existe una amplia gama de posibilidades de investigación.

1.2. Contenido

El contenido de esta investigación está organizado en siete capítulos.

En este capítulo se abordan diversos temas, como el planteamiento del problema que se desea resolver, la justificación y los objetivos de esta investigación.

El capítulo 2, trata dos de las técnicas más populares de la computación evolutiva: los algoritmos genéticos y las estrategias evolutivas. Se explica en qué consiste cada una de ellas, sus inicios, cuáles son sus características, sus aplicaciones y sus principales diferencias.

El capítulo 3, contiene lo referente a los EDAs, como cuáles son los diferentes tipos de EDA que existen y cómo funciona cada uno de ellos, así como sus propiedades, sus ventajas y sus desventajas.

En el capítulo 4 se analizan diversos métodos que permiten el manejo de restricciones utilizando algoritmos evolutivos. Se hace énfasis en las formas de penalización y cómo lidiar con espacios de búsqueda multidimensionales con restricciones.

En el capítulo 5 se presenta el algoritmo implementado, se realiza un estudio detallado del algoritmo EM (Expectation Maximization), así como del método utilizado para el manejo de restricciones en el espacio de búsqueda; además se hace un análisis de las diferentes técnicas que fue necesario implementar para el correcto funcionamiento del algoritmo.

En el capítulo 6 se encuentran los resultados obtenidos al aplicar el algoritmo propuesto a diferentes funciones de prueba.

Por último, el capítulo 7 contiene las conclusiones y recomendaciones que se derivan del análisis de los resultados obtenidos.

1.3. Objetivos

Los objetivos que se plantearon al realizar esta investigación fueron:

Objetivo General

Proponer un EDA basado en la mezcla de diferentes distribuciones normales para resolver problemas de optimización con restricciones.

Objetivos Específicos

- Estudiar los diferentes tipos de EDA que existen con la intención de adquirir un conocimiento más detallado sobre su funcionamiento.
- Analizar diferentes técnicas de penalización que permitan trabajar con algoritmos evolutivos en espacios de búsqueda multidimensionales con restricciones.
- Implementar un algoritmo capaz de estimar una función de probabilidad, mediante la mezcla de varias distribuciones normales.

Capítulo 2

Computación Evolutiva

Durante la década de los 50's se comenzaron a aplicar los principios de la teoría de Charles Darwin en la resolución de problemas de optimización. Años después varias corrientes han surgido para conformar a lo que hoy se le conoce como computación evolutiva. Dos de las principales corrientes son:

- Algoritmos Genéticos
- Estrategias Evolutivas

La programación evolutiva fue desarrollada en la década de los 60's por Lawrence J. Fogel. Su trabajo consistía en desarrollar técnicas de inteligencia artificial basadas en la evolución de máquinas de estado finitas. Los algoritmos genéticos fueron propuestos por John Holland en 1975 con la intención de presentar un modelo general de proceso adaptable que siguiera las leyes de la evolución. Las estrategias evolutivas fueron propuestas por Ingo Rechenberg y Hans-Paul Schwefel en la década de los 70's con el objetivo de optimizar parámetros.

Según la teoría Neo-Darwinista, que es el fundamento biológico de la computación evolutiva, el proceso evolutivo está compuesto por cuatro elementos fundamentales:

1. Reproducción
2. Mutación
3. Competencia
4. Selección

Los algoritmos evolutivos tienen dos propiedades fundamentales. Primero, son técnicas basadas en una población de individuos; segunda, los individuos que conforman a la población están en constante comunicación y compartiendo información a través de operadores de reproducción y mutación.

2.1. Algoritmo Genético (AG)

Los algoritmos genéticos surgen de la necesidad de resolver problemas cada vez más complejos que no pueden ser resueltos mediante métodos convencionales. La metodología del algoritmo genético, como su nombre lo indica, asemeja el proceso evolutivo de una población, la cual se va perfeccionando al paso de varias generaciones y basa su funcionamiento en la supervivencia del más apto.

Los primeros casos de algoritmos genéticos aparecieron a finales de los años 50's y a principios de los 60's, desarrollados por biólogos que buscaban realizar modelos de la evolución natural. En 1965 Ingo Rechenberg introdujo una técnica que llamó "Estrategia Evolutiva", la cual consistía en que un solo parente mutaba para producir un descendiente y se conservaba al mejor de los dos, convirtiéndose en el parente de la siguiente generación.

En 1966 L.J. Fogel, A.J. Owens y M.J. Walsh introdujeron una nueva técnica conocida como "Programación Evolutiva", la cual funcionaba mutando aleatoriamente a un individuo de entre varios, conservando al mejor de los dos. No fue sino hasta 1972, cuando John Holland estableció el principio de los algoritmos genéticos proponiendo los conceptos de cruzamiento y otros operadores de recombinación. Sin embargo el trabajo fundamental sobre algoritmos genéticos apareció hasta 1975 con el libro "Adaptación en Sistemas Naturales y Artificiales", que recoge el trabajo realizado por Holland y otros investigadores de la universidad de Michigan. Pero no es sino hasta 1989 con el libro escrito por Goldberg que los algoritmos genéticos se llegan a popularizar como una herramienta de búsqueda y optimización.

En la década de los 80's, el uso de estos algoritmos abarcaba varias disciplinas que iban desde el campo meramente matemático hasta aplicaciones en la ingeniería. Con el inicio de la era de la información y el surgimiento nuevas herramientas como el internet, los algoritmos genéticos han evolucionado en diferentes técnicas que en la actualidad están siendo utilizadas para resolver diversos problemas, ya sea en el campo industrial, científico, didáctico, etc.

Como establece Michalewicz Z. [4] un algoritmo genético para un problema en particular, debe tener los siguientes componentes:

1. Una representación genética para soluciones potenciales del problema.
2. Una manera de crear una población inicial de soluciones potenciales.
3. Una función de evaluación (función de aptitud) para valorar soluciones en base a su aptitud.
4. Operadores genéticos para alterar la composición de la población.
 - Selección.
 - Reproducción o crusa.
 - Mutación.
5. Valores para los diferentes parámetros que el algoritmo genético necesita.

2.1.1. Representación de un Algoritmo Genético

Una población está formada por varios individuos, los cuales generalmente son representados mediante una cadena binaria llamada “cromosoma”, y donde a cada bit de dicha cadena se le conoce como “gen”.

La cadena por la cual son representados los individuos de una población no tiene que ser necesariamente de longitud fija, pues existen individuos que quizás requieran menos bits para ser representados que otros, aunque en muchas ocasiones por simplicidad se considera la misma longitud para todos los individuos de la población para evitar problemas al momento de la implementación.

A la codificación que recibe un determinado individuo y que representa a los cromosomas, se denomina **genotipo**, mientras que el **fenotipo** consiste en interpretar la codificación del cromosoma de manera que pueda ser evaluado en la función de aptitud.

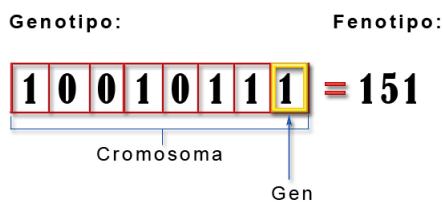


Figura 2.1: Representación de un cromosoma

Entre más extensa sea la cadena para formar los cromosomas, más preciso será el resultado obtenido, aunque computacionalmente será más costoso. Existen

una gran variedad de formas para representar a un individuo, se pueden usar letras, números reales, o bien diferentes bases numéricas además de la binaria; lo verdaderamente importante es saber interpretar y recuperar la información que se encuentra oculta en el cromosoma.

2.1.2. Población Inicial

Antes de iniciar con el proceso iterativo del algoritmo genético, se debe crear una población inicial, lo más común es generar aleatoriamente cada uno de los individuos y cada uno de los genes que componen a los cromosomas, siempre y cuando el valor que representen se encuentre dentro del espacio de búsqueda.

2.1.3. Función de Aptitud

La función de aptitud es aquella que nos dice si un individuo es más o menos apto para sobrevivir a la siguiente generación. El diseño de una función de aptitud es de suma importancia dentro de un algoritmo genético, pues en ella se basa la construcción de las nuevas generaciones, entonces si desde un inicio está mal diseñada, encontrar el óptimo resultará bastante complicado. Es por esta razón que el desarrollo de la función de aptitud debe planearse cuidadosamente y asegurase de que cumpla con los siguientes requerimientos:

- a) Debe ser clara y sencilla.

La elaboración de la función de aptitud debe ser clara y sencilla, pues si contiene demasiada información, será extremadamente selectiva y pocos individuos serán catalogados como aptos para sobrevivir, por otro lado, si dicha función carece de información, quizás nunca se presenten mejoras en la población, ya que muy pocos individuos se extinguirán.

- b) Que represente al problema que se desea resolver.

El problema que se desea resolver debe ser resumido al grado de poder representarlo mediante una función, con la finalidad de evaluar que tan apto es un individuo para sobrevivir. Por ejemplo, para el caso de una función $f(x)$, donde se desea encontrar su mínimo global, la misma función puede emplearse como función de aptitud, de manera que los valores de x dentro del espacio de búsqueda donde $f(x)$ sea pequeño serán considerados más aptos que aquellos valores de x donde $f(x)$ adquiere un valor considerablemente grande.

Existe otro término llamado penalización de la función objetivo, que se presenta cuando se manejan restricciones dentro del espacio de búsqueda. En estos

casos es necesario evaluar tanto la función de aptitud como su penalización, un individuo con un gran nivel de penalización puede ser considerado como no apto y sus posibilidades de sobrevivir serán muy pocas, sin importar que su valor de aptitud sea mejor que el de otros individuos.

Una forma de penalizar a un individuo que viola alguna restricción consiste en influir directamente sobre su valor de aptitud, de tal manera que si el grado de penalización es muy grande, entonces se puede incrementar o disminuir el valor de aptitud de dicho individuo considerablemente, de manera que no sea catalogado como apto para la supervivencia.

2.1.4. Selección

La selección es el proceso en el que ciertos individuos son elegidos para generar una nueva población, con el objetivo de que la nueva generación de individuos sea mejor que la anterior. El operador de selección es de gran importancia para el algoritmo genético, pues de ella depende la convergencia de la solución, una presión de selección muy fuerte ocasionará lo que se llama “convergencia acelerada”, que consiste en una escasa exploración del espacio de búsqueda, lo que provoca en muchas ocasiones, el estancamiento de la población en un mínimo local; por otro lado, si la presión de selección es muy débil, prácticamente no habrá mejoras en la población al paso de las generaciones.

Como en la evolución natural, es más probable que los individuos más aptos sean seleccionados para generar descendientes que sobrevivan a la siguiente generación. Ante esta situación surgen ciertas preguntas: ¿Qué pasa con los individuos considerados como no aptos?, ¿Son simplemente eliminados? Es aquí cuando entra el concepto de diversidad, la cual se define como la variedad o variabilidad que puede mostrar una misma especie en una población. La diversidad es fundamental para el correcto funcionamiento de un algoritmo genético, ya que establece un equilibrio entre los individuos aptos y los demás, realizando una mejor exploración del espacio de búsqueda y evitando una convergencia acelerada.

A través de los años se han propuesto diversos métodos de selección, los cuales se explican a continuación:

2.1.4.1. Selección Aleatoria

Como su nombre lo indica, los individuos son seleccionados de manera aleatoria con una probabilidad uniforme, lo cual provoca una presión de selección muy débil, ya que los elementos no aptos de la población tienen las mismas posibilidades de ser seleccionados que aquellos individuos con un fuerte valor de aptitud.

2.1.4.2. Selección Proporcional

Mejor conocido como el método de la ruleta. Consiste en asignar a cada individuo una porción de una ruleta circular proporcional a su valor de aptitud, donde la probabilidad de que un individuo sea seleccionado está representada por la siguiente expresión:

$$p_k = \frac{f_k}{\sum_{i=1}^n f_i}$$

La figura 2.2 muestra un ejemplo donde se desea maximizar cierta función objetivo $f(x)$, de manera que entre mayor sea el valor de aptitud de un individuo, mayor será la posibilidad de ser seleccionado.

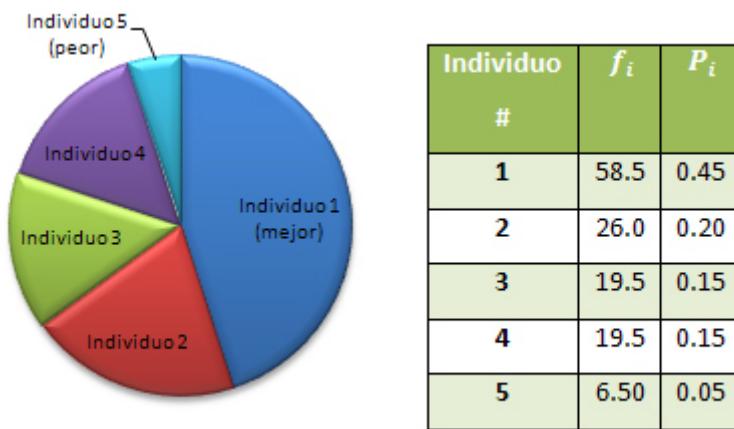


Figura 2.2: Proporción del valor de aptitud de una población

El mejor individuo de la población tiene una mayor probabilidad de ser seleccionado, mientras que el individuo menos apto, será el que tendrá menos posibilidades de influir en la siguiente generación. Dependiendo de su valor de aptitud, un individuo será más o menos propenso a ser seleccionado.

El número esperado de veces que un individuo en particular sea seleccionado, equivale al valor de aptitud de dicho individuo dividido entre el valor de aptitud esperado de toda la población.

$$n_k = \frac{f_k}{\bar{f}} = \frac{f_k}{\frac{\sum_{i=1}^n f_i}{n}} = np_k$$

El método de la ruleta ejerce una presión de selección muy fuerte sobre la población, pues tiende a seleccionar en muchas ocasiones al individuo más fuerte, especialmente cuando existe una notoria diferencia entre los valores de aptitud, lo cual provoca una convergencia acelerada de la población.

2.1.4.3. Selección por Torneo

Consiste en elegir a dos individuos diferentes de la población de manera aleatoria, y se elige aquel que tenga mejor valor de aptitud. Existen algunas variaciones de este método, en ocasiones además de elegir a los dos individuos también se elige un valor de forma aleatoria, y si dicho valor es menor que un cierto número k ($0 \leq k \leq 1$), entonces se procede a seleccionar al individuo que tenga mejor valor de aptitud de los dos, en caso contrario se toma al que tenga peor valor de aptitud. Por lo general se utiliza un valor de k cercano al 0.8.

En ocasiones suele tomarse más de dos individuos, pero el procedimiento es exactamente el mismo, el más fuerte de todos los individuos es aquel que se selecciona para influir en la siguiente generación.

El método de selección por torneo, realiza una muy buena exploración del espacio de búsqueda y en muchas ocasiones evita caer en el problema de tener una convergencia acelerada de la población, no obstante, cuando los valores de aptitud son muy similares en cada uno de los individuos, ocasiona que la población evolucione muy lentamente.

2.1.4.4. Selección por Rango

La selección por rango, es un método diseñado para prevenir una convergencia acelerada de la población, consiste en ordenar a los individuos de acuerdo a su valor de aptitud y donde el rango de cada individuo corresponde a la posición que ocupan en la población, de tal manera que el peor individuo tiene rango 1 y el mejor individuo tiene un rango igual al tamaño de la población.

El valor esperado de ser seleccionado de cada individuo depende de su rango y no del valor de aptitud, evitando que la nueva generación sea generada solo por un pequeño grupo de individuos considerados como aptos.

Si el i -ésimo individuo tiene un rango f_i , entonces la probabilidad de ser elegido como padre es de f_i/F , donde F es la suma de los rangos de toda la población; por lo tanto el peor individuo quien recibe un rango de 1 tendrá menos posibilidades de ser elegido para influir en la siguiente generación.

La selección por ruleta proporciona una mayor presión de selección sobre la población que la que ejerce la selección por rango, siendo esta última menos propensa a presentar una convergencia acelerada.

Individuo #	Aptitud	Rango	P_{ruleta}	P_{rango}
1	1.47	1	0.099	0.048
2	2.52	5	0.169	0.238
3	4.97	6	0.333	0.286
4	1.68	2	0.113	0.095
5	2.45	4	0.164	0.190
6	1.82	3	0.122	0.143

Figura 2.3: Probabilidad de selección por rango y por ruleta de una misma población

Se han propuesto métodos que basan su funcionamiento en la selección por rango, dos de los más utilizados son:

- Rango Lineal
- Rango Exponencial

Rango Lineal

En el método del rango lineal (linear ranking) propuesto por Baker, cada individuo es ordenado de manera ascendente por su valor de aptitud, desde 1 hasta N. El valor esperado del i-ésimo individuo en el tiempo t esta dado por:

$$ExpVal(i, t) = Min + (Max - Min) \frac{rank(i, t) - 1}{N - 1},$$

donde Min es el valor esperado del individuo con rango 1, $1 \leq Max \leq 2$ y $Min = 2 - Max$. En el caso en que $Max = 2$ el peor individuo no tendrá posibilidad de influir en la siguiente generación y de esta manera se presentará una mayor presión de selección sobre la población.

Los valores de aptitud de los individuos intermedios son obtenidos por medio de interpolación a través de la siguiente expresión:

$$f(i) = Max - \frac{2(i - 1)(Max - 1)}{N - 1}$$

Baker recomienda un valor para Max de 1.1, el cual ha sido utilizado en varias pruebas, demostrando una mejor exploración del espacio de búsqueda que la selección proporcional. Una desventaja que tiene la selección por rango lineal es que al momento de disminuir la presión de selección sobre la población, le

será más difícil al algoritmo encontrar los individuos más aptos.

Rango Exponencial

En esta estrategia, el mejor individuo de la población recibe un valor de aptitud de 1, el segundo mejor recibe un valor de aptitud s , normalmente alrededor 0.99; el tercero mejor recibe un valor de aptitud de s^2 y así sucesivamente, hasta que al último y peor individuo se le asigna un valor de aptitud de s^{N-1} .

La presión de selección es proporcional a $1 - s$ y el número esperado de hijos de cada individuo es proporcional a su valor de aptitud dividido entre el valor de aptitud esperado de toda la población.

La principal diferencia de este método con respecto al de selección por rango lineal, es que se da más oportunidad a los peores individuos, lo cual ocasiona que el rango de pérdida del peor individuo sea mucho menor y se realice una mejor exploración del espacio de búsqueda.

2.1.5. Reproducción o Cruza

La reproducción es el proceso en el cual uno o más hijos son creados a partir de los padres escogidos en el proceso de selección. El método de crusa más popular es aquel en el que dos padres producen dos hijos. Para esto se elige un punto de crusa de manera aleatoria entre el primero y el último bit de los cromosomas de los padres; los bits a la izquierda del punto de crusa del *padre₁* son copiados a la parte izquierda del punto de crusa del *hijo₁*, mientras que los bits a la izquierda del punto de crusa del *padre₂* pasan a la izquierda del punto de crusa del *hijo₂*. Una vez hecho esto, los bits a la derecha del punto de crusa del *padre₁* conforman los bits de la parte derecha del punto de crusa del *hijo₂*, y por último, los bits a la derecha del *padre₂* son copiados a la derecha del punto de crusa del *hijo₁*. De esta manera los hijos formados contienen información de ambos padres. El proceso es ilustrado en la figura 2.4.

Al combinar la información de dos individuos, existe una buena posibilidad de obtener una mejoría en la población y acercarse a la solución del problema de optimización. Una variante del operador de reproducción es utilizar varios puntos de crusa, en donde los hijos son generados por la combinación de varios segmentos extraídos de los cromosomas de ambos padres.

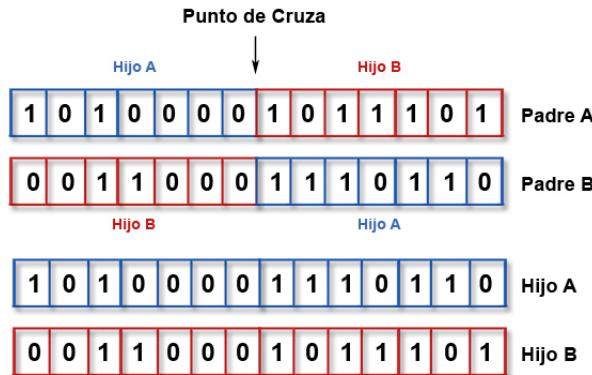


Figura 2.4: Proceso de crusa de dos individuos

Existe otro tipo de reproducción llamada “Reproducción Uniforme”, en donde cada uno de los genes de los hijos es creado al copiar el gen correspondiente de uno de los dos padres elegido aleatoriamente con una probabilidad uniforme. Dado que no existe una dependencia entre bits adyacentes, la reproducción uniforme lleva a la población a una mejor exploración del espacio de búsqueda, sin embargo, cuando se está cerca de la convergencia, los padres dan origen a hijos muy similares a ellos.

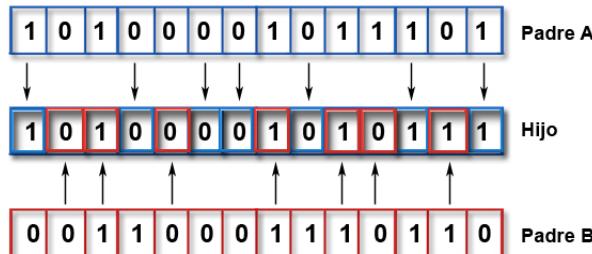


Figura 2.5: Proceso de crusa uniforme

2.1.6. Mutación

Aunque es cierto que la reproducción es el operador que presenta mayor variabilidad en el algoritmo genético, también es cierto que la mutación juega un papel muy importante en la solución al problema que se desea resolver, pues es la que previene que la población presente una convergencia acelerada y que existan múltiples copias de un mismo individuo.

El proceso de mutación consiste en alterar en un grado muy pequeño la información genética de un individuo. En el caso de una representación binaria del

cromosoma, una pequeña cantidad de bits son cambiados, de manera que si un bit originalmente tiene un valor de 0, este es cambiado por 1, y viceversa.

Cada uno de los genes de un cromosoma tiene una probabilidad $p(0 \leq p \leq 1)$ de ser cambiado, por lo general se trata de un valor muy pequeño para que el cromosoma no sea alterado significativamente; por cada uno de los genes se elige un valor aleatorio r con distribución uniforme en $[0,1]$, de tal manera que si $r < p$, el gen correspondiente es alterado, en caso contrario permanece igual.

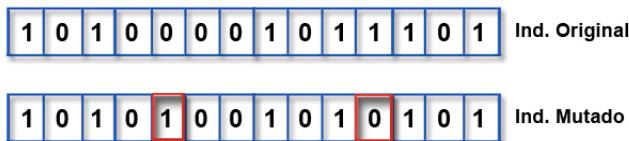


Figura 2.6: Proceso de mutación de un individuo

Es conveniente tener un balance entre cada uno de los operadores para obtener un buen funcionamiento del algoritmo genético y poder solucionar de manera correcta el problema planteado. Cada uno de los operadores (selección, reproducción y mutación), deben ser diseñados cuidadosamente y de manera individual para que el algoritmo funcione de la mejor manera posible y no enfocarse solo en uno, ya que el proceso evolutivo está conformado por varios factores y necesita de la participación de cada uno de ellos para desarrollarse de manera óptima.

2.1.7. Parámetros de un Algoritmo Genético

Existen algunos parámetros que hay que tener en cuenta al momento de implementar un algoritmo genético, tales como el tamaño de la población, probabilidad de cruce y probabilidad de mutación.

El tema de la obtención de dichos parámetros ha sido discutido en muchas ocasiones a través de los años. De Jong (1975) indica que el tamaño de la población debe ser entre 50 y 100 individuos, la probabilidad de reproducción entre dos individuos alrededor de 0.6 y la probabilidad de mutación de 0.001 por bit. Estos valores han sido usados en varios experimentos, pero no significa que siempre tenga que ser así; estos pueden ser modificados para ajustarse al problema que se trata de resolver.

Encontrar los valores adecuados para los parámetros de un algoritmo genético, puede ser visto como un problema de optimización, y aunque parezca contradictorio Greffenstette (1986) mencionó que se puede utilizar un algoritmo genético para encontrar dichos parámetros. El usuario debe ajustar cada uno de estos

parámetros dependiendo del problema que se desea resolver, pues existen valores que funcionan mejor que otros para determinados casos.

2.1.8. Bloques Constructores y Teorema de los Esquemas

Los fundamentos teóricos de los algoritmos genéticos están basados en la representación binaria de soluciones y en el concepto de esquema. Un esquema es un conjunto de bits, donde cada bit puede tomar cualquier valor del conjunto $S = \{0, 1, *\}$, de manera que un esquema representa a todas las cadenas (un hiperplano o subconjunto del espacio de búsqueda) que contienen exactamente los mismos bits, excepto por las posiciones representadas por el símbolo “*”; tales cadenas son conocidas con el nombre de *instancias* del esquema. Considere el siguiente esquema:

$$H = (10 * 11 * 0),$$

donde H es el esquema que representa a todas las cadenas de siete bits con un valor de 0 en el 2^o y 7^o bit, y un valor de 1 en el 1^o , 4^o y 5^o bit. Las instancias de H son cuatro:

$$(10\mathbf{0}11\mathbf{0}0)$$

$$(10\mathbf{0}11\mathbf{1}0)$$

$$(10\mathbf{1}11\mathbf{0}0)$$

$$(10\mathbf{1}11\mathbf{1}0)$$

Cada esquema contiene exactamente 2^r instancias, donde r representa el número de posiciones con el símbolo “*”. Por otro lado, cada cadena de longitud m es representada por 2^m esquemas.

El orden de un esquema H (denotado por $o(H)$) es el número de posiciones fijas, es decir, posiciones que pueden contener 0 o 1 como valor. En otras palabras, el orden de un esquema es igual a la longitud del esquema menos el número de posiciones con el símbolo “*”. Por ejemplo, considere los siguientes esquemas de longitud 6:

$$H_1 = (1 * * * 00)$$

$$H_2 = (*1 * *0*)$$

$$H_3 = (**1010)$$

El orden de cada uno de los esquemas es:

$$\begin{aligned} o(H_1) &= 3 \\ o(H_2) &= 2 \\ o(H_3) &= 4 \end{aligned}$$

El concepto de orden de un esquema es de gran importancia al momento de calcular la probabilidad de supervivencia de un esquema debido a posibles mutaciones.

La longitud de un esquema H (denotado por $\delta(H)$) se define como la distancia entre la primera y la última posición fija. Para los tres esquemas del ejemplo anterior, la longitud de cada uno está dada por:

$$\begin{aligned} \delta(H_1) &= 6 - 1 = 5 \\ \delta(H_2) &= 5 - 2 = 3 \\ \delta(H_3) &= 6 - 3 = 3 \end{aligned}$$

Si un esquema contiene una sola posición fija, se dice que tiene una longitud de 0.

El concepto de longitud se utiliza al momento de calcular la probabilidad de supervivencia de un esquema debido al operador de reproducción.

El teorema de los esquemas fue propuesto por Holland (1975) y establece que aquellos esquemas cortos y de bajo orden que se encuentran por encima de la media, crecen exponencialmente en las generaciones subsecuentes del algoritmo genético, ya que aquellos esquemas cortos y de bajo orden intercambian información durante los procesos de reproducción y mutación, originando mejores individuos.

El algoritmo genético tiene el objetivo de encontrar una solución óptima a un determinado problema a través de la yuxtaposición de esquemas cortos y de bajo orden. A este enunciado se le conoce como la *hipótesis de los bloques constructores* y fue propuesta por Goldberg (1989).

2.1.9. Funciones Deceptivas

Las funciones deceptivas son funciones diseñadas para violar de manera extrema la hipótesis de los bloques constructores, de manera que bloques cortos y de bajo orden conduzcan a bloques largos y de mayor orden. Bethke (1980) estableció que sería de gran dificultad para un algoritmo genético, encontrar el óptimo

de una función de aptitud si esquemas de bajo orden contienen información engañosa acerca de esquemas de mayor orden. Por ejemplo, suponga que el valor de aptitud de un individuo depende del número unos que contenga, de manera que entre mayor cantidad de unos tenga un individuo este será mejor evaluado. ¿Qué sucedería si el óptimo global está representado por la solución con ceros en todos sus bits? En principio, sería difícil encontrar el óptimo 0000..,0 para un algoritmo genético, ya que cada esquema de bajo orden proporciona información engañosa acerca de la posición del óptimo global. A este tipo de funciones se les conoce como *deceptivas completas*.

Existen funciones de aptitud con diferente grado de decepción. Bethke hace uso de la “*Transformada de Walsh*”, la cual es similar a la Transformada de Fourier, para diseñar funciones de aptitud con varios grados de decepción. Refiérase a [3] para más información sobre este tema. Goldberg y sus colaboradores continuaron con el trabajo de Bethke, y han desarrollado un gran número de investigaciones teóricas acerca de las funciones deceptivas, de tal manera que la decepción se ha convertido en uno de los temas principales en la teoría de los algoritmos genéticos.

El concepto de decepción en el algoritmo genético comúnmente esta relacionado con problemas de optimización; ya que la decepción en una función de aptitud ocasiona que sea más difícil de encontrar el óptimo global. La función “*trap₅*”, es una función deceptiva que supone un tamaño de cadena múltiplo de cinco, la cual se encuentra dividida en partes disjuntas de cinco bits cada una, de manera que las posiciones correspondientes a la i-ésima partición están denotadas por los índices $b_{i,1}$ a $b_{i,5}$, tal que:

$$f_{\text{trap}_5}(x) = \sum_{i=1}^{\frac{n}{5}} \text{trap}_5(x_{b_{i,1}} + \dots + x_{b_{i,5}}),$$

donde

$$f_{\text{trap}_5}(u) = \begin{cases} 5 & \text{si } u = 5 \\ 4 - u & \text{otro caso} \end{cases}$$

La variable u representa el número de unos en la partición de 5 bits, de manera que se tiene un óptimo global cuando $u = 5$ y un óptimo local en $u = 0$. Cuando la longitud de la cadena es mayor a cinco bits, existen $2^{\frac{n}{5}} - 1$ óptimos locales, mientras que el óptimo global se encuentra en 111..,1. Lo anterior ocasiona que cualquier partición de cinco bits que contenga uno o más ceros, aleje al AG del óptimo global.

Así como la función $trap_5$, existen un gran número de funciones completamente deceptivas que el algoritmo genético no puede resolver. Por esta razón han surgido diferentes técnicas que tratan de lidiar con el problema de la decepción, una de estas técnicas son los llamados Algoritmos de Estimación de Distribución, los cuales se explican más a detalle en el capítulo 3.

2.1.10. El Algoritmo

El algoritmo genético necesita de una población de individuos representados por una o más cadenas de cromosomas. El proceso de evolución consiste en tres partes fundamentales, la selección, la reproducción y la mutación, en donde cada uno de estos procesos es llevado a cabo en cada generación. A continuación se muestra la estructura del algoritmo genético básico.

Algoritmo 1 Algoritmo Genético Básico

```

1:  $t \leftarrow 0$ 
2: Inicializar  $P^t$ 
3: Evaluar aptitud de  $P^t$ 
4: mientras No se cumpla la condición de paro hacer
5:    $t \leftarrow t + 1$ 
6:   Seleccionar  $P^t$  de  $P^{t-1}$ 
7:   Reproducción  $P^t$ 
8:   Mutación  $P^t$ 
9:   Evaluar aptitud de  $P^t$ 
10: fin mientras
```

2.2. Estrategias Evolutivas (EEs)

Las estrategias evolutivas son una metodología diseñada para la optimización de funciones reales, basadas en una población perteneciente a un espacio multi-dimensional. Al igual que el AG, las estrategias evolutivas utilizan operadores de reproducción y mutación, así como de una representación y una función objetivo; con la diferencia de que en las EEs los individuos evolucionan a través de sus fenotipos, es decir, por medio de organismos constituidos a partir de las variables que se desean optimizar. En las EEs la exploración progresá por medio de la mutación de los individuos más aptos, en vez de por la reproducción, como lo hacen los AGs.

Las EEs pueden dividirse en dos tipos:

EEs Simples

Son procedimientos estocásticos de optimización paramétrica con paso adaptativo. Consisten en la evolución de un solo individuo utilizando solo la mutación como operador genético.

EEs Múltiples

Surgen ante las debilidades de las EEs simples. Se basan en la existencia de múltiples individuos que se reproducen en cada generación, dando origen a nuevos individuos capaces de sufrir mutaciones que reemplazan a la población anterior.

2.2.1. Representación

Cada individuo de la población consta de dos tipos de variables: las variables objeto y las variables estratégicas. Las variables objeto son aquellos valores capaces de ser representados en el espacio de búsqueda; mientras que las variables estratégicas son los parámetros mediante los que se rige el proceso evolutivo, dicho de otra manera, las variables estratégicas indican la manera en que las variables objeto son afectadas por la mutación. Un individuo se puede representar de la siguiente manera:

$$\langle x_1, \dots, x_n; \sigma_1, \dots, \sigma_n; \theta_1, \dots, \theta_n \rangle,$$

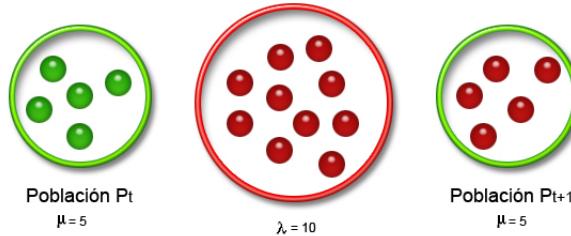
donde cada x_i representa una variable objeto y cada σ_i y θ_i es una variable de control que afecta la operación de mutación.

2.2.2. Selección

A diferencia de los AGs donde la selección de individuos es de forma aleatoria, en las EEs se efectúa de forma determinística, de manera que no es necesario argumentar la selección mediante probabilidades. Se espera que aquellos individuos que mejor adapten sus parámetros estratégicos tengan mejor valor de aptitud y por lo tanto sean más propensos a ser seleccionados para poblar la siguiente generación. Existen dos formas canónicas de selección: (μ, λ) y $(\mu + \lambda)$.

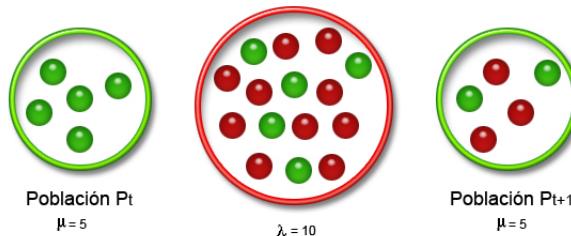
2.2.2.1. Selección (μ, λ)

Consiste en seleccionar μ individuos del grupo formado por los λ hijos generados, sin importar si son o no mejores que sus respectivos padres, con el objetivo de que se conviertan en los padres de la siguiente generación. En este método, la presión de selección se ve determinada por la razón μ/λ la cual aumenta o disminuye de manera proporcional a este factor.

Figura 2.7: Selección (μ, λ)

2.2.2.2. Selección $(\mu + \lambda)$

Los μ padres junto con sus λ hijos son agrupados en un mismo conjunto del cual se seleccionan los μ individuos más aptos para formar la siguiente generación. La presión de selección aumenta considerablemente respecto a la selección (μ, λ) y la presencia de los mejores padres en la población disminuye en gran medida la complejidad del algoritmo.

Figura 2.8: Selección $(\mu + \lambda)$

2.2.3. Reproducción

Es el operador por el cual la información contenida en una población de individuos es compartida para crear nuevos individuos. Pueden existir diferentes reglas de recombinación para cada uno de los parámetros que conforman al individuo. En las EEs existen dos estrategias para la reproducción:

2.2.3.1. Reproducción Sexual

Dos padres son elegidos al azar para cada uno de los parámetros que conformarán al nuevo individuo $\langle X, \sigma, \theta \rangle$.

2.2.3.2. Reproducción Panmítica

Un padre elegido de manera aleatoria se reproduce con otro elegido de la misma manera para cada uno de los parámetros $\langle X, \sigma, \theta \rangle$.

Existen cuatro diferentes variantes de las estrategias de recombinación, que aplican tanto a la reproducción sexual como a la panmítica:

1. **Sin Recombinación:** El hijo es igual a uno de los dos padres.
2. **Discreta:** Para cada uno de los parámetros, se toma el valor de uno de los padres elegido aleatoriamente.
3. **Intermedia:** La variable toma como valor el punto intermedio del parámetro correspondiente de ambos padres.
4. **Generalizada Intermedia:** El nuevo individuo adquiere un valor aleatorio dentro del intervalo formado por los valores del parámetro correspondiente de ambos padres.

2.2.4. Mutación

Es el operador principal de las EEs y es el encargado de realizar la explotación del espacio de búsqueda. Consiste en la alteración de las variables objeto y estratégica mediante parámetros elegidos aleatoriamente. Las mutaciones se realizan primero sobre las variables estratégicas y después sobre las variables objeto. Las reglas para las variables estratégicas son:

$$\sigma'_i = \sigma_i e^{(\tau' N(0,1) + \tau N_i(0,1))}$$

$$\theta'_j = \theta_j + \beta N_j(0, 1)$$

donde $N_i(0, 1)$ y $N_j(0, 1)$ son números aleatorios provenientes de una distribución normal estándar ($\mu = 0, \sigma = 1$) que son generados para cada variable estratégica σ_i y θ_j . $N(0, 1)$ es un numero aleatorio de una distribución normal estándar ($\mu = 0, \sigma = 1$) que permanece constante durante el proceso de mutación. Los parámetros τ'_i, τ y β son constantes que dependen del número k de variables objeto.

$$\tau = \frac{1}{4k^{1/4}}$$

$$\tau' = \frac{1}{(2k)^{1/2}}$$

$$\beta \approx 0.0873$$

Después de haber mutado las variables estratégicas, se procede a mutar las variables objeto alterando cada variable por medio de mutaciones correlacionadas. La regla de mutación correlacionada se representa con la siguiente expresión:

$$X = X + N(0, C(\sigma, \theta)),$$

donde el término $N(0, C(\sigma, \theta))$ representa un número aleatorio que proviene de una distribución normal con media cero y varianza igual a la contenida en la matriz de covarianzas $C(\sigma, \theta)$.

Capítulo 3

Algoritmos de Estimación de Distribución

El correcto funcionamiento de los algoritmos genéticos depende en gran medida de los parámetros utilizados en los operadores de reproducción y de mutación. El hecho de no considerar la interacción entre las variables que conforman a cada uno de los individuos de la población, provoca que el movimiento de la solución sea difícil de predecir. Holland mencionó que tomar en cuenta la interacción entre las variables podría ser beneficioso para el algoritmo genético. Siguiendo esta idea, en los últimos años se han propuesto algoritmos basados en modelos probabilísticos, en donde los operadores de selección y recombinación han sido reemplazados por modelos de probabilidad, mismos que son obtenidos al tomar una muestra de la población conformada por individuos favorables. Estos algoritmos son llamados Algoritmos de Estimación de Distribución (EDA, por sus siglas en inglés).

Los operadores de cruce y mutación no existen en los EDAs, de tal manera que las nuevas generaciones de individuos son generadas a través de un modelo probabilístico, en donde la interacción entre las variables puede ser tomada en consideración. Mientras que en la computación evolutiva, las interrelaciones entre las diferentes variables son representadas implícitamente, en los EDAs las interrelaciones pueden ser representadas explícitamente por medio de la distribución de probabilidad conjunta entre las variables.

Al igual que en el algoritmo genético, los individuos tienen diversas formas de representación y son evaluados según su valor de aptitud. Aquellos individuos con mejor valor de aptitud son más propensos a sobrevivir a la siguiente generación, que aquellos individuos con un bajo valor de aptitud.

El proceso que realiza el EDA es un proceso iterativo, en donde cada generación presenta una mejoría de la generación anterior. En cada iteración se realizan tres tareas fundamentales:

1. Selección de una muestra.
2. Creación del modelo de probabilidad, basado en la muestra.
3. Generar la nueva población en base a la simulación del modelo obtenido.

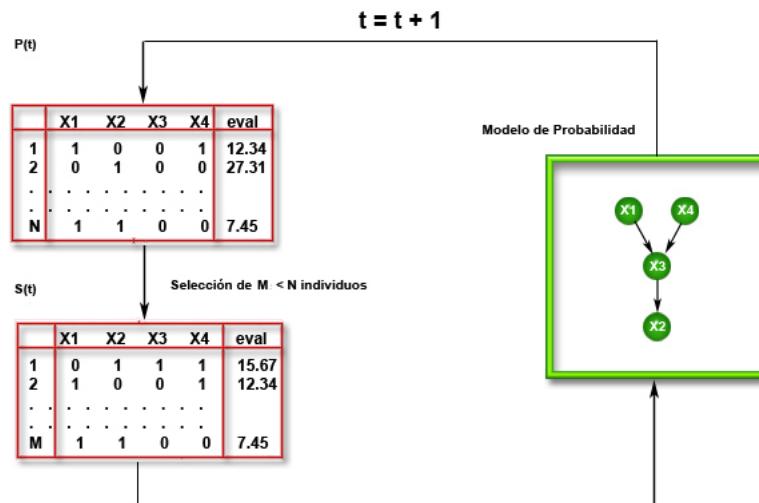


Figura 3.1: Funcionamiento de un EDA básico. 1) Selección de una muestra S de M individuos. 2) Crear el modelo de probabilidad en base a la muestra S . 3) Generar la nueva población en base a la simulación del modelo de probabilidad.

El modelo de distribución de probabilidad depende de la muestra de individuos seleccionada, en caso de que los individuos seleccionados no sean favorables, es probable que la siguiente generación de individuos generada en base al modelo tampoco lo sea y se aleje de la solución. Por esta razón es importante seleccionar individuos aptos para generar el modelo de distribución de probabilidad, y que las siguientes generaciones tengan más oportunidad de mejorar y encontrar la solución al problema planteado.

A continuación se muestra el algoritmo general de un EDA

Algoritmo 2 EDA básico

```

1:  $t \leftarrow 0$ 
2: Inicializar  $P^t$  con  $N$  individuos
3: Evaluar aptitud de  $P^t$ 
4: mientras No se cumpla la condición de paro hacer
5:    $t \leftarrow t + 1$ 
6:   Seleccionar una muestra  $S$  de tamaño  $M \leq N$  de  $P^{t-1}$ 
7:   Estimar la distribución de probabilidad  $p(x|S)$  de la muestra seleccionada
8:   Generar  $P^t$  utilizando  $p(x|S)$ 
9:   Evaluar aptitud de  $P^t$ 
10:  fin mientras

```

Cuando una variable x_i se puede escribir en función de alguna otra variable x_j , se dice que existe una dependencia entre este par de variables. Por ejemplo, el salario de los profesores de una determinada escuela, se asigna de acuerdo a su grado académico; entonces, podemos decir que el salario de un profesor depende del grado académico que posee.

Los EDAs se pueden clasificar en dos grandes grupos:

1. Algoritmos con representación discreta
2. Algoritmos con representación real

En ambos grupos existen algoritmos sin dependencias o con dependencias, las cuales pueden ser bivariadas o múltiples. A continuación se analizan algunos de los algoritmos con representación discreta y real más populares.

3.1. EDAs con Representación Discreta

En el caso de modelos discretos, la estimación de la función de probabilidad conjunta se puede representar mediante modelos gráficos, donde los vértices simbolizan a las variables, y las aristas constituyen la dependencia entre ellas, de manera que si una arista se dirige del vértice x_1 al vértice x_2 , significa que la variable x_2 depende de la variable x_1 . En el caso de no existir dependencia alguna entre las variables, la distribución de probabilidad conjunta se puede estimar mediante el producto de distribuciones marginales.

Las instancias de las variables son generadas de forma secuencial. Sea $pa_{\pi(i)}$ el conjunto de padres de la variable $x_{\pi(i)}$, entonces, las variables deben ser ordenadas de tal manera que los valores de $pa_{\pi(i)}$ sean asignados antes de muestrear a la

variable $x_{\pi(i)}$. Una vez obtenido el conjunto de padres $pa_{\pi(i)}$, el valor de $x_{\pi(i)}$ es simulado utilizando la distribución $p(x_{\pi(i)}|pa_{\pi(i)})$. A este método se le conoce como “*Probabilistic Logic Sampling*” (PLS), y fue propuesto por Henrion (1988).

3.1.1. EDAs sin Dependencia

Los métodos que pertenecen a este grupo no consideran dependencia alguna entre sus variables, por lo que representan una buena opción para resolver problemas que carecen de interacción entre sus variables, sin embargo son insuficientes para problemas que presentan algún grado de dependencia. Estos algoritmos consideran a la probabilidad conjunta como el producto de las probabilidades marginales de cada una de las variables. Dos de los algoritmos más populares son el UMDA y el PBIL, los cuales se explican más a detalle a continuación.

3.1.1.1. UMDA

El algoritmo UMDA (Univariate Marginal Distribution Algorithm) fue propuesto por primera vez por Mühlenbein (1998). Este algoritmo consiste en estimar el comportamiento de los individuos seleccionados por medio de la distribución de probabilidad conjunta $p_i(x)$, la cual se factoriza como el producto de distribuciones marginales independientes, de manera que:

$$p_t(x) = p(x|S_{t-1}) = \prod_{i=1}^n p_t(x_i)$$

donde n representa el número de variables, y S_{t-1} la muestra de individuos en la iteración $t - 1$. Cada distribución marginal se estima a partir de las frecuencias marginales:

$$p_t(x_i) = \frac{\sum_{j=1}^N \delta_j(X_i = x_i|S_{t-1})}{N},$$

tal que

$$\delta_j(X_i = x_i|S_{t-1}) = \begin{cases} 1 & \text{si en el } j\text{-ésimo caso de } S_{t-1}, X_i = x_i \\ 0 & \text{otro caso} \end{cases}$$

Algoritmo 3 UMDA

```

1:  $t \leftarrow 0$ 
2: Inicializar  $P^t$  con  $N$  individuos
3: Evaluar aptitud de  $P^t$ 
4: mientras No se cumpla la condición de paro hacer
5:    $t \leftarrow t + 1$ 
6:   Seleccionar una muestra  $S$  de tamaño  $M \leq N$  de  $P^{t-1}$ 
7:   Estimar la distribución de probabilidad conjunta  $p_t(x) = p(x|S_{t-1}) =$ 
     $\prod_{i=1}^n p_t(x_i)$ 
8:   Generar  $P^t$  de tamaño  $N$  utilizando  $p_t(x)$ 
9:   Evaluar aptitud de  $P^t$ 
10: fin mientras

```

3.1.1.2. PBIL

El algoritmo PBIL (Population Based Incremental Learning) fue introducido por Baluja (1994) [8], y posteriormente por Baluja y Caruana (1995) [14], con el objetivo de encontrar el óptimo de una función n-dimensional en el espacio binario. Los individuos son representados como un vector de probabilidades $p_t(x)$.

$$p_t(x) = \{p_t(x_1), \dots, p_t(x_k), \dots, p_t(x_n)\},$$

donde $p_t(x_k)$ indica la probabilidad de obtener el valor 1 en la k-ésima variable de la población.

El algoritmo consiste en generar N nuevos individuos en cada generación utilizando el vector de probabilidades $p_t(x)$. Una vez generados los N individuos, se ordena la población en base a su valor de aptitud, seleccionando los mejores M individuos ($M \leq N$) para actualizar el vector de probabilidades, de acuerdo a la siguiente regla:

$$p_{t+1}(x) = (1 - \alpha)p_t(x) + \alpha \frac{1}{M} \sum_{k=1}^M x_k^t,$$

donde x_k^t representa el k-ésimo elemento de la muestra tomada en el tiempo t , y $\alpha \in (0, 1]$ es un parámetro del algoritmo, de manera que en el caso en que $\alpha = 1$ el algoritmo PBIL es equivalente al algoritmo UMDA.

Algoritmo 4 PBIL

```

1:  $t \leftarrow 0$ 
2: Inicializar  $P^t$  con  $N$  individuos
3: Evaluar aptitud de  $P^t$ 
4: Obtener un vector inicial de probabilidades  $p_0(x)$ 
5: mientras No se cumpla la condición de paro hacer
6:    $t \leftarrow t + 1$ 
7:   Usando  $p_{t-1}(x)$ , generar la población  $P^t$  de  $N$  nuevos individuos
8:   Evaluar aptitud de  $P^t$ 
9:   Ordenar  $P^t$  de acuerdo al valor de aptitud
10:  Seleccionar una muestra  $S$  de los mejores  $M$  individuos de  $P_t$  :
     $x_1^t, \dots, x_i^t, \dots x_M^t$ 
11:  para  $i = 1$  to  $n$  hacer
12:     $p_t(x_{i+1}) = (1 - \alpha)p_{t-1}(x_i) + \alpha \frac{1}{M} \sum_{k=1}^M x_k^t$ 
13:  fin para
14: fin mientras

```

3.1.2. EDAs con Dependencias Bivariadas

Los algoritmos que pertenecen a este grupo, consideran dependencias entre cada par de variables, lo que proporciona una mejor solución que el caso univariado, aun así, el modelo resultante está muy alejado de solucionar problemas del mundo real, donde se presenta una interacción múltiple entre las variables.

A través de los años se han presentado diversos algoritmos que basan su funcionamiento en el hecho de que existe una dependencia entre cada par de variables de un espacio n-dimensional. A continuación se mencionan algunos de los algoritmos más populares de este género.

3.1.2.1. MIMIC

El algoritmo MIMIC (Mutual Information Maximization for Input Clustering) fue propuesto por De Bonet (1997) [19]. Consiste en realizar una búsqueda de una permutación de las variables, con la intención de obtener la función de probabilidad conjunta que más se asemeje a la distribución empírica del conjunto de datos seleccionados. La función de probabilidad conjunta de un grupo de variables aleatorias se define como:

$$P(X) = p(X_1|X_2 \dots X_n)p(X_2|X_3 \dots X_n) \dots p(X_{n-1}|X_n)p(X_n)$$

Dado que solo existen dependencias condicionales entre cada par de variables, $p(X_i|X_j)$, resulta imposible generar un modelo exacto que se ajuste a la distribu-

ción de los datos, aun así, es posible generar un modelo que asemeje a la función de probabilidad conjunta, utilizando solo dependencias bivariadas y univariadas.

Dada una permutación de los números entre 1 y n , $\pi = i_1 i_2 \dots i_n$, se puede definir a $\hat{p}_\pi(x)$ como una *aproximación de la función de probabilidad conjunta*, expresada por:

$$\hat{p}_\pi(x) = p(X_{i_1}|X_{i_2})p(X_{i_2}|X_{i_3})\dots p(X_{i_{n-1}}|X_{i_n})p(X_{i_n})$$

El objetivo es encontrar la permutación π que minimice la diferencia entre la función $\hat{p}_\pi(x)$, y la verdadera función de probabilidad conjunta $p(X)$. Para conocer la diferencia entre dos distribuciones se utiliza la distancia Kullback-Liebler.

$$\begin{aligned} D_{KL}(p\|p_\pi) &= \int_x p[\log p - \log \hat{p}_\pi] dx \\ &= E[\log p] - E[\log \hat{p}_\pi] \\ &= -h(p) - E[\log p(X_{i_1}|X_{i_2})p(X_{i_2}|X_{i_3})\dots p(X_{i_{n-1}}|X_{i_n})p(X_{i_n})] \\ &= -h(p) + h(X_{i_1}|X_{i_2}) + h(X_{i_2}|X_{i_3}) + \dots + h(X_{i_{n-1}}|X_{i_n}) + h(X_{i_n}) \end{aligned}$$

La distancia Kullback-Liebler nunca es negativa, y solo es igual a cero cuando ambas distribuciones son idénticas, por lo tanto, la permutación óptima π se define como aquella que minimiza esta distancia. La permutación óptima es aquella que produce la menor entropía respecto a la distribución verdadera.

La estructura del modelo de estimación se puede ver como un grafo conexo compuesto de $n - 1$ aristas, tal que dos variables i, j ($i \neq j$), no pueden depender de una misma variable.



Figura 3.2: Estructura del algoritmo MIMIC

El algoritmo de aprendizaje consiste en encontrar la permutación óptima π que minimice la distancia Kullback-Liebler, para esto se lleva a cabo un proceso iterativo que consiste en encontrar la variable X_{i_j} que minimice la entropía condicional respecto a la variable $X_{i_{j+1}}$ encontrada en la iteración anterior. El algoritmo 5 muestra el algoritmo de aprendizaje del MIMIC.

Algoritmo 5 MIMIC

- 1: $\pi \leftarrow$ vacío
 - 2: $S \leftarrow \{x_1, x_2, \dots, x_{n-1}, x_n\}$
 - 3: Encontrar la variable X_{i_n} con menor entropía
 - 4: Eliminar la variable X_{i_n} del conjunto S
 - 5: Agregar la variable X_{i_n} al conjunto π
 - 6: **mientras** S no se encuentre vacío **hacer**
 - 7: $n \leftarrow n - 1$
 - 8: Encontrar la variable X_{i_n} en S cuya entropía condicional media con respecto a la variable $X_{i_{n+1}}$ seleccionada en el paso anterior sea mínima
- $$i_n = \arg \min_j h(X_j | X_{i_{n+1}}), \text{ donde } j \neq i_{n+1}, \dots, i_n$$
- 9: Eliminar la variable X_{i_n} del conjunto S
 - 10: Aregar la variable X_{i_n} al conjunto π
 - 11: **fin mientras**
-

3.1.2.2. BMDA

El algoritmo BMDA (Bivariate Marginal Distribution Algorithm) propuesto por Pelikan y Mühlenbein (1999) [25] es una extensión del algoritmo UMDA. Considera dependencias entre cada par de variables y propone una factorización de la función de distribución conjunta que requiere de estadísticos de segundo orden.

El primer paso consiste en generar una población inicial de manera aleatoria dentro del espacio de búsqueda, para después hacer una selección de los mejores individuos y calcular las frecuencias marginales univariadas y bivariadas.

Sea un individuo x un cromosoma de longitud n , donde cada gen es representado de manera binaria, de manera que $x_i \in \{0, 1\}$. Se define a la frecuencia marginal univariada $p_i(x_i)$ como el número de individuos en la población que tienen el valor de x_i en la i -ésima posición. De manera similar para cada dos índices $i \neq j \in \{0, \dots, n - 1\}$, se define a la frecuencia marginal bivariada $p_{i,j}(x_i, x_j)$ como el número de individuos en la población con los valores x_i y x_j en la posición i y en la posición j respectivamente.

Una vez obtenidas las frecuencias marginales univariadas y bivariadas, se procede a la construcción del llamado “grafo de dependencias”, que representa el modelo de estimación de probabilidad necesario para crear la nueva generación de individuos.

El grafo de dependencias G es definido por tres conjuntos, V , E y R , de manera que $G = (V, E, R)$. V representa al conjunto de vértices, es decir, cada una de las variables del espacio n-dimensional, $E \in V \times V$ es el conjunto de aristas en el grafo y R es el conjunto que contiene el nodo raíz de cada una de las componentes conexas de G .

Existen dos importantes características del grafo de dependencias:

- El grafo de dependencias no tiene que ser necesariamente conexo, es decir, no tiene que ser un árbol, sino que puede ser un bosque.
- El grafo de dependencias es siempre acíclico.

Se denota a A como el conjunto de vértices que no han sido procesados, de manera que al inicio del algoritmo, $A = V$. El algoritmo de termina cuando A se encuentre vacío, indicando que todos los vértices han sido procesados. El conjunto D , es aquel que representa a todos los pares de $V \times V$ que son dependientes en un 95 %, tal que:

$$D = \{(i, j) | i \neq j \in \{0, \dots, n - 1\} \wedge X_{i,j}^2 \geq 3,84\},$$

donde $X_{i,j}^2$ representa al estadístico chi-cuadrada de Pearson. En términos de frecuencias univariadas y bivariadas y el número total de individuos tomados en consideración, para las posiciones $i \neq j$, se tiene que:

$$X_{i,j}^2 = \frac{(Np_{i,j}(x_i, x_j) - Np_i(x_i)p_j(x_j))^2}{Np_i(x_i)p_j(x_j)}$$

de manera que si las variables i y j son independientes en un 95 %, el estadístico chi-cuadrada de Pearson cumple con la siguiente desigualdad:

$$X_{i,j}^2 < 3,84 \quad (3.1)$$

El algoritmo 6 muestra el *algoritmo de aprendizaje* del BMDA, el cual consiste en encontrar el grafo de dependencias que maximice la suma de los estadísticos chi-cuadrada. El primer paso consiste en elegir de manera aleatoria una variable r que funcione como nodo raíz, tal que $r \in V \setminus A$, después se procede a buscar todos los pares de vértices v y v' que cumplan con la condición de dependencia (3.1), de manera que $v \in A$ y $v' \in V \setminus A$; en caso de no encontrar al menos un par de vértices que cumplan con dicha condición, se elige aleatoriamente una variable r ($r \in V \setminus A$) como raíz de un nuevo subgrafo. El proceso se repite hasta que todas las variables se encuentren dentro del grafo de dependencias.

Algoritmo 6 BMDA

```

1:  $V \leftarrow \{0, \dots, n - 1\}$ 
2:  $A \leftarrow V$ 
3:  $E \leftarrow$  vacío
4:  $v \leftarrow$  Cualquier vértice de  $A$ 
5: Añadir  $v$  a  $R$ 
6: Eliminar  $v$  de  $A$ 
7: si  $A$  está vacío entonces
8:   Terminar
9: fin si
10: Encontrar el par de vértices  $v$  y  $v'$  en  $D$  que maximice  $X_{v,v'}^2$ , donde  $v \in A$  y
     $v' \in V \setminus A$ 
11: si  $X_{v,v'}^2 < 3,84$  entonces
12:   Ir al paso 4
13: fin si
14: Eliminar  $v$  de  $A$ 
15: Añadir la arista  $(v, v')$  al conjunto  $E$ 
16: Ir al paso 7

```

Para formar la nueva generación de individuos, es necesario hacer uso del grafo de dependencias $G = (V, E, R)$. Para cada individuo, los valores para las variables contenidas en R son generados por frecuencias marginales univariadas. Para toda variable v que no ha sido generada y se encuentra conectada por medio de una arista a alguna variable v' previamente generada, se utiliza el concepto de probabilidad condicional para obtener su valor correspondiente, de manera que:

$$p_{v,v'}(x_v | x_{v'}) = \frac{p_{v,v'}(x_v, x_{v'})}{p_{v'}(x_{v'})}$$

Entonces, el modelo para estimar la función de probabilidad conjunta esta dado por:

$$p(x) = \prod_{r \in R} p_r(x_r) \prod_{i \in V \setminus R} p_{i,e(i)}(x_i | x_{e(i)}),$$

donde el vértice $e(i)$ corresponde al padre del i -ésimo vértice.

El proceso se repite hasta que todas las variables han sido generadas, formando así, el grafo acíclico, donde la suma de los estadísticos chi-cuadrada es máxima.

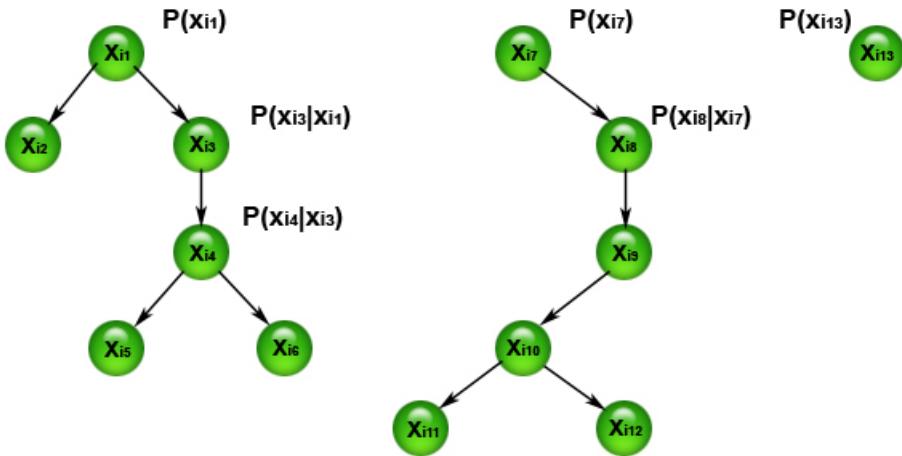


Figura 3.3: Árbol de Dependencias BMDA

3.1.2.3. Árbol de Dependencias de Chow y Liu

El árbol de dependencias propuesto por C.K. Chow y C.N. Liu (1968) [1], es un método que trata de aproximar de manera óptima la distribución de probabilidad para variables discretas en un espacio n-dimensional, por medio del producto de distribuciones de segundo orden, de manera que *el modelo de probabilidad conjunta* está dado por:

$$p(x) = \prod_{i=1}^n p(x_{\pi i} | x_{\pi j(i)}), \quad 0 \leq j(i) \leq i,$$

donde $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ representa una permutación de los números del 1 hasta n, de manera que $x_{\pi j(i)}$ es la variable que funge como padre de la variable x_i .

Para conocer la eficiencia de una distribución de probabilidad aproximada $p(x)$, se utiliza el concepto de información mutua entre las variables que conforman el árbol de dependencias. La información mutua entre un par de variables se define como:

$$I(x, y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

La información mutua entre dos variables es siempre no negativa ($I(x, y) \geq 0$) y simétrica ($I(x, y) = I(y, x)$). El árbol de dependencias óptimo T es aquel que tiene el mayor peso posible. El peso de un árbol equivale a la suma de las informaciones mutuas entre cada par de variables conectadas entre sí, de manera que el árbol de dependencias óptimo se define como:

$$T = \max \left\{ \sum_{i=1}^n I(x_{\pi^{k_i}}, x_{\pi^{k_j(i)}}), \forall k \in \{1, 2, \dots, n!\} \right\},$$

donde π^k representa una permutación de las variables del espacio multidimensional. La información mutua entre dos variables discretas se puede expresar en términos de la entropía, por medio de la siguiente relación:

$$\begin{aligned} I(x, y) &= H(x) - H(x|y) \\ &= H(y) - H(y|x) \\ &= H(x) + H(y) - H(x, y) \\ &= H(x, y) + H(x|y) - H(y|x) \end{aligned}$$

donde

$$\begin{aligned} H(x) &= - \sum_{i=1}^n p(x_i) \log p(x_i) \\ H(y|x) &= - \sum_{x \in X} \sum_{y \in Y} p(y, x) \log p(y|x) \end{aligned}$$

El algoritmo de aprendizaje consiste en encontrar el árbol de dependencias óptimo T , el cual se define como el conjunto de aristas E y vértices V , que maximiza $\sum_{i=1}^n I(x_{\pi i}, x_{\pi j(i)})$, donde cada vértice representa una variable del espacio de búsqueda. Sea A el conjunto de vértices que no han sido procesados, de manera que al inicio del algoritmo $A = V$. En cada paso, se busca el par de vértices v y v' con mayor información mutua, de manera que $v \in A$ y $v' \in V \setminus A$. El algoritmo termina cuando se han procesado todos los vértices. El algoritmo 7 muestra los pasos que hay que seguir para crear el árbol de dependencias óptimo, dado un conjunto de variables en un espacio de búsqueda n-dimensional.

Algoritmo 7 Árbol de Dependencias de Chow y Liu

- 1: $V \leftarrow \{0, \dots, n - 1\}$
 - 2: $A \leftarrow V$
 - 3: $E \leftarrow$ vacío
 - 4: $(v, v') \leftarrow$ Par de vértices con mayor información mutua
 - 5: Añadir la arista (v, v') al conjunto E
 - 6: Eliminar v y v' del conjunto A
 - 7: **si** A está vacío **entonces**
 - 8: Terminar
 - 9: **fin si**
 - 10: Encontrar el par de vértices v y v' que maximice $I(v, v')$ donde $v \in A$ y $v' \in V \setminus A$
 - 11: Añadir la arista (v, v') al conjunto E
 - 12: Eliminar v de A
 - 13: Ir al paso 6
-

El grafo que representa el modelo de probabilidad conjunta, como se muestra en la figura 3.4, es un grafo conexo y acíclico, compuesto de $n - 1$ aristas, con un sólo nodo raíz, donde dos variables i, j ($i \neq j$) pueden depender de la misma variable.

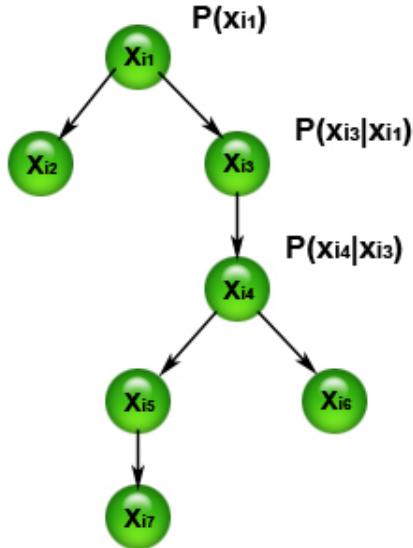


Figura 3.4: Árbol de Dependencias de Chow y Liu

3.1.3. EDAs con Dependencias Múltiples

Existen varios algoritmos EDAs que proponen como método de factorización de la distribución de probabilidad, estadísticos de orden a superior a dos, es decir, que exista una mayor interacción entre las variables.

El primer trabajo del que se tiene registro, en el que se contempla la posibilidad de adaptar métodos usados en la inducción de modelos gráficos probabilísticos a partir de datos de un algoritmo EDA, fue presentado por Baluja y Davies (1998). Sin embargo no se llegó a proponer ningún método de implementación.

Los algoritmos que consideran dependencias múltiples entre las variables en un espacio n-dimensional, pueden codificar la probabilidad de distribución mediante el uso de redes Bayesianas.

3.1.3.1. Algoritmo PADA

Propuesto por Soto y col. (1999) , el algoritmo PADA (Polytree Approximation of Distribution Algorithm) consiste en realizar una factorización de la distribución de probabilidad por medio de una red Bayesiana con estructura de poliárbol, es decir, sin que exista más de un camino no dirigido entre cada par de variables.

Este algoritmo puede ser considerado un híbrido entre un método de detección de dependencias e independencias condicionales y un procedimiento basado en aptitud y búsqueda. Una distribución de poliárbol puede escribirse como el producto de diferentes distribuciones condicionales, de manera que *el modelo de probabilidad conjunta* está dado por:

$$p(x) = \prod_{i=1}^n p(x_i | x_{j_1(i)}, x_{j_2(i)}, \dots, x_{j_m(i)}),$$

donde $\{x_{j_1(i)}, x_{j_2(i)}, \dots, x_{j_m(i)}\}$ es el conjunto de padres de la variable x_i , los cuales son mutuamente independientes, tal que:

$$p(x_{j_1(i)}, x_{j_2(i)}, \dots, x_{j_m(i)}) = \prod_{k=1}^m p(x_{j_k(i)})$$

en los poliárboles existen tres tipos de tripletas adyacentes:

1. Secuencial: $X_i \rightarrow X_k \rightarrow X_j$
2. Divergente: $X_i \leftarrow X_k \rightarrow X_j$
3. Convergente: $X_i \rightarrow X_k \leftarrow X_j$

La tripleta $X_i \rightarrow X_k \leftarrow X_j$ se llama patrón cabeza-cabeza, y el nodo al cual convergen los arcos se conoce como nodo cabeza-cabeza. Tanto en la tripleta secuencial como en la divergente, las variables X_i y X_j son marginalmente dependientes y condicionalmente independientes dado X_k . Sin embargo, en la tripleta convergente las variables X_i y X_j son marginalmente independientes y condicionalmente dependientes dado X_k .

El algoritmo de aprendizaje parte de un grafo sin aristas, las cuales se van insertando según dos medidas definidas sobre los arcos $\langle X_i, X_j \rangle$, la dependencia marginal $Dep(X_i, X_j)$ y la dependencia global $Dep_g(X_i, X_j)$. Las dos medidas son valores no negativos y se basan en los conceptos de información mutua marginal y condicional. A lo más se pueden insertar $n - 1$ aristas, lo cual garantiza que el grafo sea conexo.

La información mutua marginal $I(X_i, X_j)$ de las variables aleatorias X_i y X_j se define como:

$$I(X_i, X_j) = \sum_{x_i, x_j} p(x_i, x_j) \log_2 \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \geq 0$$

La información mutua condicional $I(X_i, X_j|X_k)$ de las variables aleatorias X_i y X_j dado X_k se define como:

$$I(X_i, X_j|X_k) = \sum_{x_i, x_j, x_k} p(x_i, x_j, x_k) \log_2 \frac{p(x_i, x_j, x_k)p(x_k)}{p(x_i, x_k)p(x_j, x_k)} \geq 0$$

Las aristas insertadas son aquellas cuyos valores $Dep(X_i, X_j)$ y $Dep_g(X_i, X_j)$ superan ciertos pesos umbrales y que además cumplen con las restricciones correspondientes al modelo utilizado.

El cálculo de las medidas $Dep(X_i, X_j)$ y $Dep_g(X_i, X_j)$ tiene complejidad $O(n^2)$ y $O(n^3)$ respectivamente, donde n es el número de variables.

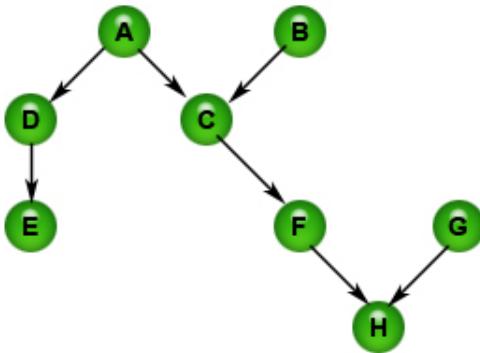


Figura 3.5: Estructura del algoritmo PADA

3.1.3.2. Algoritmo BOA

Pelikan y Goldberg (2000) propusieron el algoritmo BOA (Bayesian Optimization Algorithm), el cual es una extensión de los algoritmos UMDA y BMDA que permite interacciones de mayor orden entre las variables. Este algoritmo utiliza la métrica BDe (Bayesian Dirichlet equivalence) para evaluar la bondad de cada estructura. Esta métrica tiene la propiedad de asignar el mismo valor a estructuras que reflejan las mismas dependencias o independencias condicionales.

El algoritmo de aprendizaje consiste en encontrar la red Bayesiana que maximice la métrica BDe. La búsqueda de la estructura se basa en una estrategia glotona y comienza cada generación con un grafo sin arcos. Con el objetivo de reducir la cardinalidad del espacio de búsqueda se le asigna a cada nodo de la red bayesiana la restricción de que no puede tener más de k padres.

Las redes Bayesianas son frecuentemente utilizadas para modelar datos con variables tanto discretas como continuas. Una red Bayesiana se encarga de codificar las relaciones entre las variables, de manera que sea posible generar nuevas instancias con propiedades similares a las de los datos dados. Cada vértice en la red corresponde a una variable, y las aristas representan la relación entre dos variables. Las aristas pueden ser dirigidas o no dirigidas, para el caso del BOA, la red Bayesiana se representa mediante un grafo acíclico dirigido.

Para el caso de una red Bayesiana sin ciclos y con aristas dirigidas, *la distribución de probabilidad conjunta* se puede expresar como:

$$p(X) = \prod_{i=1}^n p(X_i|\Pi_{X_i}),$$

donde $X = (X_1, \dots, X_n)$ representa al conjunto de soluciones disponibles, y Π_{X_i} es el conjunto de padres de la variable X_i , de manera que la distribución es capaz de generar nuevos individuos utilizando sólo probabilidades marginales y condicionales.

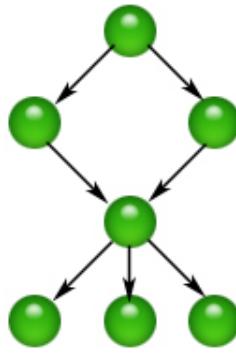


Figura 3.6: Estructura del algoritmo BOA

3.2. EDAs con Representación Real

Al igual que en el caso discreto, existen diferentes tipos de algoritmos, desde aquellos que no consideran dependencia alguna entre las variables, hasta los que presentan una dependencia bivariada o múltiple. La mayoría de los algoritmos hacen uso de distribuciones Gaussianas para estimar la función de densidad conjunta.

3.2.1. EDAS sin Dependencias

Este tipo de algoritmos asumen que la función de densidad conjunta sigue una distribución normal n-dimensional, la cual es factorizada como el producto de distribuciones normales unidimensionales independientes, tal que:

$$f(x; \mu, \Sigma) = \prod_{i=1}^n f(x_i; \mu_i, \sigma_i) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left(\frac{x_i - \mu_i}{\sigma_i}\right)^2}$$

Dos de los algoritmos más populares de éste género son el $UMDA_c$ y el $PBIL_c$, los cuales son extensiones de los algoritmos $UMDA$ y $PBIL$ utilizados para variables discretas. A continuación se explica más a detalle en qué consiste cada uno de estos métodos.

3.2.1.1. UMDAc

El algoritmo $UMDA_c$ (Univariate Marginal Distribution Algorithm for continuous domains) ejecuta en cada iteración y por cada variable, una serie de pruebas estadísticas para poder encontrar la función de densidad que mejor se ajusta a la variable. La función de densidad conjunta se estima mediante el producto de las distribuciones marginales, de manera que:

$$f(x; \theta^t) = \prod_{i=1}^n f(x_i, \theta_i^t)$$

La estimación de los parámetros se lleva a cabo una vez que las densidades han sido identificadas mediante los estimadores de máxima verosimilitud. Sea $\theta_i^t = (\theta_i^{t,1}, \dots, \theta_i^{t,k})$ los k parámetros en que la función de densidad de la i -ésima variable depende en la generación t . Los estimadores de máxima verosimilitud para cada uno de los parámetros es la solución del siguiente sistema de ecuaciones:

$$\frac{\partial}{\partial \theta_i^{j,t}} \sum_{w=i}^N \ln f_{X_i}(x_{i,w}, \theta_i^t) = 0; \quad j = 1, \dots, k.$$

En el caso en que todas las distribuciones univariadas se traten de distribuciones normales, los dos parámetros que deben ser calculados en cada iteración son la media, μ_i^t , y la desviación estándar, σ_i^t , a partir de los estimadores de máxima verosimilitud.

$$\hat{\mu}_i = \bar{X}_i = \frac{1}{N} \sum_{w=1}^N x_{i,w}$$

$$\hat{\sigma}_i = \sqrt{\frac{1}{N} \sum_{w=1}^N (x_{i,w} - \bar{x}_i)^2}$$

3.2.1.2. PBILc

Propuesto por Sebag y Ducoulombier (1998), consiste en asumir que la función de densidad conjunta sigue una distribución gaussiana que puede ser factorizada como el producto de densidades marginales independientes y unidimensionales. En cada iteración es necesario calcular la media para cada una de las variables del espacio, para lo cual se utiliza la siguiente fórmula:

$$\mu_i^{t+1} = (1 - \alpha)\mu_i^t + \alpha(X_{i,best,1}(t) + X_{i,best,2}(t) - X_{i,worst}(t)),$$

donde $X_{i,best,1}(t)$ y $X_{i,best,2}(t)$ representan al mejor y al segundo mejor individuo respectivamente; $X_{i,worst}(t)$ representa al peor individuo de la población, y α es una constante.

Para calcular el vector de varianzas se proponen cuatro diferentes heurísticas:

1. Usar un valor constante para todas las marginales en cada generación.
2. Ajustarlo como en la estrategia evolutiva $(1, \lambda)$.
3. Calcular la varianza muestral de los K mejores individuos en cada generación.

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^K (X_i^j - \bar{X}_i)^2}{K}}$$

4. Memorizando la diversidad de los K mejores individuos

$$\sigma_i^{(t+1)} = (1 - \alpha)\sigma_i^t + \alpha\sqrt{\frac{\sum_{j=1}^K (X_i^j - \bar{X}_i)^2}{K}}$$

3.2.2. EDAS con Dependencias

Conforme la complejidad de los problemas aumenta y la dependencia entre las variables es mayor, es necesario hacer uso de algoritmos que sean capaces de considerar dichas dependencias para llevar a cabo una buena estimación de la distribución. Existen algoritmos que consideran dependencias entre pares de variables como el *MIMIC_c*, el cual es una extensión del caso discreto, por otro lado, algunos algoritmos como el *EMNA* utilizan distribuciones normales multivariadas, con la intención de que exista una mayor interacción entre las variables, y de esta forma poder realizar una mejor aproximación de la función de densidad conjunta.

3.2.2.1. MIMIC_c

El algoritmo *MIMIC_c* es una adaptación del algoritmo *MIMIC* aplicado a espacios continuos, donde el modelo de probabilidad para cada par de variables consiste en una Gaussiana bivariada.

La idea principal consiste en estimar la función de densidad conjunta utilizando una sola densidad marginal univariada y $n - 1$ funciones de densidad bivariadas. Dada una permutación $\pi = (i_1, i_2, \dots, i_n)$, se define a la función de densidad $F_\pi(x)$ como:

$$F_\pi(x) = \{f_\pi(x)|f_\pi(x) = f(x_{i_1}|x_{i_2})f(x_{i_2}|x_{i_3}), \dots, f(x_{i_{n-1}}|x_{i_n})f(x_{i_n})\}$$

donde $f(x_{i_n})$ y $f(x_{i_j}|x_{i_{j+1}})$, $j = 1, \dots, n-1$ siguen funciones de densidad normales. El objetivo es encontrar la permutación π^* , de manera que $f_{\pi^*}(x)$ minimice la

distancia Kullback-Leibler entre la verdadera función de densidad $f(x)$ y la función estimada $f_\pi(x)$.

De acuerdo con el teorema de Whitaker (1991), si X es una función de densidad normal n-dimensional, entonces la entropía de X está dada por:

$$h(X) = \frac{1}{2}n(1 + \log 2\pi) + \frac{1}{2}\log |\Sigma|$$

Aplicado el concepto de entropía a las funciones de densidad normal univariadas y bivariadas que componen el modelo de probabilidad conjunta del algoritmo $MIMIC_c$, se obtiene que:

$$h(X) = \frac{1}{2}(1 + \log 2\pi) + \log \sigma_X$$

$$h(X|Y) = \frac{1}{2} \left[(1 + \log 2\pi) + \log \left(\frac{\sigma_X^2 \sigma_Y^2 - \sigma_{X,Y}^2}{\sigma_Y^2} \right) \right]$$

donde σ_X^2 y σ_Y^2 denotan la varianza de la variable X y Y respectivamente, mientras que $\sigma_{X,Y}$ representa la covarianza entre las variables X y Y .

El algoritmo $MIMIC_c$ trabaja como un algoritmo glotón compuesto de dos pasos fundamentales. El primero consiste en elegir la variable con menor varianza. En el segundo paso se elige aquella variable X , cuya estimación $\frac{\sigma_X^2 \sigma_Y^2 - \sigma_{X,Y}^2}{\sigma_Y^2}$ con respecto a la variable anterior Y , sea menor.

3.2.2.2. Algoritmo EMNA

El algoritmo EMNA (Estimation of Multivariate Normal Algorithm) propuesto por Larrañaga y Lozano (2001), realiza una aproximación de la función de distribución normal multivariada en cada generación, mediante la estimación del vector de medias $\mu^t = \mu_1^t, \dots, \mu_d^t$ y la matriz de covarianzas Σ .

El algoritmo se basa en la suposición de que la población tiene una distribución normal multivariada $N(\mu, \Sigma)$ con una densidad definida como:

$$f(x) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Para generar una nueva generación de individuos a partir de una distribución normal multivariada, es posible utilizar la descomposición de Cholesky. Dado que la matriz de covarianzas Σ es simétrica positiva definida, esta puede ser factorizada en una matriz triangular inferior L , de manera que $LL^T = \Sigma$. Si Z_1, Z_2, \dots, Z_n son individuos provenientes de distribuciones normales independientes ($Z_i \sim N(0, 1)$), se puede generar la nueva población $X \sim N(\mu, \Sigma)$ utilizando la siguiente ecuación:

$$X = \mu + LZ \quad (3.2)$$

Es importante mantener a los nuevos individuos dentro del espacio de búsqueda, pues el resultado arrojado por la ecuación (3.2), puede ocasionar que algunos individuos caigan fuera de los límites establecidos.

Algoritmo 8 EMNA

- 1: $t \leftarrow 0$
 - 2: Inicializar P^t con N individuos
 - 3: Evaluar P^t
 - 4: **mientras** No se cumpla la condición de paro **hacer**
 - 5: $t \leftarrow t + 1$
 - 6: Seleccionar una muestra S de tamaño $M \leq N$ de P^{t-1}
 - 7: Calcular a partir de S el vector de medias μ^t y la matriz de covarianzas Σ^t
 - 8: Factorizar Σ^t en la matriz triangular inferior L con $LL^T = \Sigma^t$, por medio de la factorización de Cholesky
 - 9: Generar un conjunto de individuos independientes $Z_i \sim N(0, 1)$
 - 10: Hacer $P^t = \mu^t + LZ$
 - 11: Evaluar P^t
 - 12: **fin mientras**
-

Capítulo 4

Manejo de Restricciones en Algoritmos Evolutivos

Los algoritmos evolutivos son técnicas de búsqueda para espacios multidimensionales sin restricciones. De ahí surge la necesidad de implementar técnicas que permitan el manejo de restricciones tanto lineales como no lineales. Estas técnicas buscan penalizar a los individuos que se encuentran fuera de la zona factible, de tal manera que se preserven aquellos individuos que posean un mejor valor de aptitud.

El proceso de optimizar se refiere al hecho de minimizar o maximizar una función sujeta a restricciones en sus variables.

Considere:

- x : Vector de variables de decisión, donde $x = x_1, \dots, x_d$
- f : Función objetivo que se desea maximizar o minimizar
- g_i : Restricciones de desigualdad lineales y no lineales a las que están sujetas todas aquellas soluciones dentro del espacio de búsqueda.
- h_i : Restricciones de igualdad que pueden ser tanto lineales como no lineales; a las que están sujetas todas aquellas soluciones dentro del espacio de búsqueda.

Dentro del área de optimización computacional, existen dos campos: Optimización sin restricciones, y con restricciones.

Si se desea resolver un problema de optimización sin restricciones, cualquier solución x dentro del espacio de búsqueda, es una solución factible y no debe ser penalizada. Un problema de optimización puede definirse como:

$$\min_x f(x)$$

Un punto x^* es un óptimo global si $f(x^*) \leq f(x)$ para todo $x \in \mathbb{R}^n$.

En la práctica, el óptimo global puede ser difícil de encontrar, ya que los algoritmos son susceptibles de converger a un óptimo local, los cuales son puntos en los que se alcanza el menor valor de f en una vecindad.

Un punto x^* es un mínimo local si existe una vecindad N alrededor de x^* tal que $f(x^*) \leq f(x)$ para todo $x \in N$.

Por otro lado, un problema de optimización en un espacio con restricciones puede ser escrito como:

$$\min_x f(x) \text{ sujeto a } \begin{cases} g_i(x) \leq 0 & i = 1, \dots, n \\ h_j(x) = 0 & j = 1, \dots, m \end{cases}$$

de manera que una solución factible es aquella que cumple con la siguiente expresión:

$$\sum_{i=1}^n \max[0, g_i(x)] + \sum_{j=1}^m |h_j(x)| = 0$$

La mayoría de los problemas están sujetos a restricciones de desigualdad, de manera que cuando se presentan restricciones de igualdad, estas pueden ser transformadas en la forma:

$$|h_j(x)| - \epsilon \leq 0,$$

donde ϵ es un valor pequeño que representa la tolerancia permitida para decidir si un individuo es factible o no lo es. Una restricción de desigualdad $g_i(x)$ es activa en x si para un vector solución x dentro de la zona factible del espacio de búsqueda $S(F \subseteq S)$, cumple con la condición $g_i(x) = 0$.

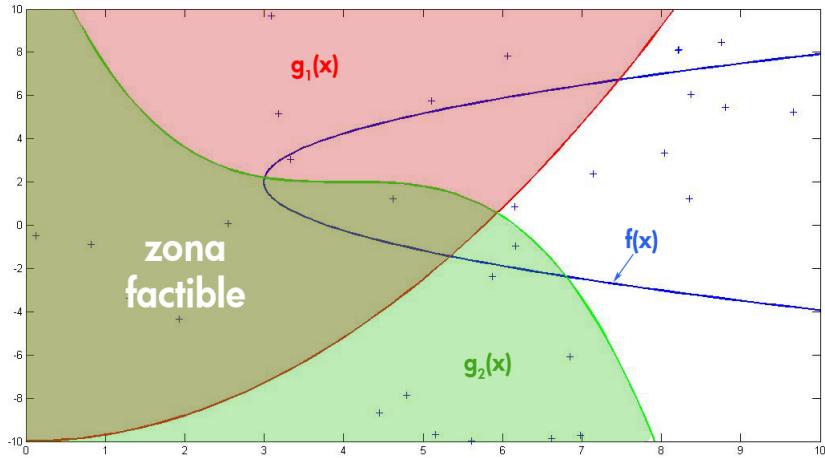


Figura 4.1: Espacio de búsqueda con restricciones

A continuación se muestra una tabla con las principales propiedades de las funciones de benchmark definidas en el apéndice A. El valor de n representa la dimensión del problema, DL y DN indican el número de restricciones de desigualdad lineales y no lineales respectivamente, IN es el número de restricciones no lineales de igualdad, y a representa el número de restricciones activas en el óptimo.

$f(x)$	n	$f(x)$ Tipo	DL	IN	DN	a
g01	13	cuadrático	9	0	0	6
g02	20	no lineal	1	0	1	1
g03	10	polinomial	0	1	0	1
g04	5	cuadrático	0	0	6	2
g05	4	cubico	2	3	0	3
g06	2	cubico	0	0	2	2
g07	10	cuadrático	3	0	5	6
g08	2	no lineal	0	0	2	0
g09	7	polinomial	0	0	4	2
g10	8	lineal	3	0	3	3
g11	2	cuadrático	0	1	0	1
g12	3	cuadrático	0	0	9^3	0
g13	5	exponencial	0	3	0	3

Cuadro 4.1: Principales Propiedades de las Funciones de benchmark

4.1. Funciones de Penalización

La penalización consiste en la modificación del valor de aptitud de un individuo, de acuerdo al número de violaciones de las restricciones. La forma general de la aptitud de cada individuo de la población está dada por:

$$\phi(x) = f(x) + \left[\sum_{i=1}^n r_i g_i(x) + \sum_{j=1}^m c_j h_j(x) \right],$$

donde r_i y c_j son constantes positivas denominadas “factores de penalización”, n es el número de restricciones de desigualdad y m el número de restricciones de igualdad.

El principal problema que se presenta al implementar una función de penalización, consiste en elegir los valores adecuados de los factores de penalización, de manera permitan orientar la exploración del espacio de búsqueda a la zona factible y poder encontrar el óptimo global.

A través de los años se han propuesto varios métodos para disminuir la dependencia de las funciones de penalización hacia los factores de penalización. A continuación se describen los métodos más populares.

4.1.1. Pena de Muerte (Death Penalty)

Consiste en asignar un valor de aptitud de cero a todos aquellos individuos que estén fuera de la zona factible. Se recomienda su uso para espacios de búsqueda convexos y con zonas factibles de tamaño considerable con respecto al espacio de búsqueda. Su principal desventaja consiste aquellos casos donde la población inicial no tiene individuos factibles; en este caso se presenta un estancamiento de algoritmo, pues todos los individuos tienen la misma aptitud.

Este método solo funciona con problemas de optimización con restricciones de desigualdad, debido a la dificultad de generar soluciones que satisfagan restricciones de igualdad.

4.1.2. Penalización Estática

Los factores de penalización permanecen sin cambio durante la ejecución del algoritmo. Existen diversas propuestas para penalizar a un individuo utilizando penalización estática. Kuri [23] propone que el valor de aptitud sea calculado de la siguiente manera:

$$\text{fitness}(x) = \begin{cases} f(x) & \text{si la solución es factible} \\ K - \sum_{i=1}^s \left(\frac{K}{m}\right) & \text{otro caso} \end{cases}$$

donde s es el número de restricciones satisfechas, m es el número total de restricciones y K es una constante (1×10^9). Todos aquellos individuos que violen el mismo número de restricciones son penalizados de la misma manera, sin importar si un individuo es mejor que otro.

Otro método de penalización estática es el propuesto por Homafair, Lai y Qi [10]. En este método se establecen diferentes niveles de penalización, tal que a cada nivel se asignan coeficientes de penalización específicos de acuerdo a la cantidad de violaciones para cada una de las restricciones. La evaluación de un individuo se realiza mediante la siguiente expresión:

$$\text{fitness}(x) = f(x) + \sum_{i=1}^m (R_{k,i} \max[0, g_i(x)]^2),$$

donde $R_{k,i}$ son los coeficientes estáticos de penalización correspondientes al k -ésimo nivel y m representa el número total de restricciones en el espacio de búsqueda.

La gran cantidad de parámetros que se necesitan manipular para poder penalizar cada uno de los individuos, hace de este método uno muy difícil de controlar cuando se presenta un gran número de restricciones.

Hoffmeister y Sprave [18] propusieron la siguiente función de penalización:

$$\text{fitness}(x) = f(x) + \sqrt{\sum_{i=1}^m \{H[-g_i(x)]g_i(x)^2\}},$$

donde H se define como:

$$H(x) = \begin{cases} 0 & x > 0 \\ 1 & x \leq 0 \end{cases}$$

El problema con el método de Hoffmeister y Sprave consiste en el hecho en que los individuos no factibles son evaluados de peor manera que aquellos que si son factibles, lo cual no siempre conduce a encontrar la solución óptima al problema planteado.

4.1.2.1. Stochastic Ranking

Técnica propuesta por Runarsson y Yao [31]. Consiste en crear un balance entre la función objetivo y la función de penalización. No se requiere la definición

de un factor de penalización, en su lugar, se ocupa un proceso de selección basado en la jerarquía o rango de los individuos. La población es ordenada por medio de un algoritmo similar al método de la “burbuja”, tomando en consideración el parámetro $P_f(0 \leq P_f \leq 1)$ definido por el usuario, el cual permite que ciertos individuos penalizados sobrevivan a la siguiente generación, con el objetivo de realizar una mejor exploración del espacio de búsqueda. Cada individuo es penalizado mediante la suma de las restricciones que infringen. Un valor recomendado para P_f es $0,4 < P_f < 0,5$. El algoritmo 9 muestra la estructura del stochastic ranking.

Algoritmo 9 Stochastic Ranking

```

1: para  $i = 1$  to  $N$  hacer
2:    $c \leftarrow 0$ 
3:   para  $j = 1$  to  $\lambda - 1$  hacer
4:      $u = random(0, 1)$ 
5:     si ( $\phi(x_j) = \phi(x_{j+1}) = 0$ ) ó  $u < P_f$  entonces
6:       si  $f(x_j) > f(x_{j+1})$  entonces
7:         Intercambia( $x_j, x_{j+1}$ )
8:        $c \leftarrow c + 1$ 
9:     fin si
10:    si no
11:      si  $\phi(x_j) > \phi(x_{j+1})$  entonces
12:        Intercambia( $x_j, x_{j+1}$ )
13:         $c \leftarrow c + 1$ 
14:      fin si
15:    fin si
16:  fin para
17:  si  $c = 0$  entonces
18:    Terminar
19:  fin si
20: fin para

```

donde N es el tamaño de la población, $f(x)$ corresponde a la función de aptitud evaluada en x , y $\phi(x)$ es la función de penalización evaluada en x .

4.1.3. Penalización Dinámica

Los métodos de penalización dinámica se basan en la idea de utilizar el tiempo (número de generaciones transcurridas) para el cálculo del factor de penalización de un individuo. Entre más generaciones pasen, la penalización será más severa, ya que es más probable de que el algoritmo haya alcanzado la zona factible del espacio de búsqueda.

Joines y Houck [11] utilizan el número de generaciones para asignar el valor del factor de penalización, de manera que la penalización aumenta conforme transcurren las generaciones.

$$\text{fitness}(x) = f(x) + (C \times t)^\alpha \times SVC(\beta, x),$$

donde C , α y β son constantes definidas por el usuario (los autores utilizan $C = 0,5$, $\alpha = 1$ ó 2 y $\beta = 1$ ó 2), y $SVC(\beta, x)$ se define como:

$$SVC(\beta, x) = \sum_{i=1}^n D_i^\beta(x) + \sum_{j=1}^m D_j^\beta(x)$$

$$D_i(x) = \begin{cases} 0 & g_i(x) \leq 0 \\ |g_i(x)| & \text{En otro caso } 1 \leq i \leq n \end{cases}$$

$$D_j(x) = \begin{cases} 0 & -\epsilon \leq h_i(x) \leq \epsilon \\ |h_j(x)| & \text{En otro caso } 1 \leq j \leq n \end{cases}$$

Kazarlis y Petridis [24], propusieron un método para penalizar a un individuo dependiendo del número de generaciones transcurridas, de manera que el valor de aptitud de un individuo queda determinado por la siguiente expresión:

$$\text{fitness}(x) = f(x) + V(g) \times \left(A \sum_{i=1}^m (\delta_i, w_i \Phi(d_i(S))) + B \right) \times \delta_s$$

donde A es un factor de “severidad”, m es el número total de restricciones, δ_i es 1 si la i -ésima restricción es infringida y 0 en otro caso. Cada restricción es asociada con un peso w_i .

La función $V(g)$ se ajusta dinámicamente durante la ejecución del algoritmo, y adquiere un valor cada vez mayor conforme transcurren las generaciones, los autores sugieren:

$$V(g) = \left(\frac{g}{G} \right)^2,$$

donde g es el número de la generación actual y G es el número total de generaciones.

4.1.4. Penalización Adaptativa

La penalización adaptativa utiliza información del mismo proceso evolutivo para actualizar los valores de los factores de penalización. Bean y Hadj-Alouane [5, 21] desarrollaron una función de penalización que utiliza una retroalimentación del mismo proceso de búsqueda, donde cada individuo es evaluado de la siguiente manera:

$$\text{fitness}(x) = f(x) + \lambda(t) \left[\sum_{i=1}^n g_i^2(x) + \sum_{i=1}^m |h_i|^2(x) \right],$$

donde $\lambda(t)$ se actualiza en cada generación t de acuerdo con:

$$\lambda(t+1) = \begin{cases} \left(\frac{1}{\beta_1}\right)\lambda(t) & \text{caso \#1} \\ \beta_2\lambda(t) & \text{Caso \#2 } \beta_1, \beta_2 > 1, \beta_1 > \beta_2 \text{ y } \beta_1 \neq \beta_2 \\ \lambda(t) & \text{otro caso} \end{cases}$$

El caso **#1** ocurre si el mejor individuo en las últimas k generaciones fue siempre factible. Mientras que el caso **#2** se presenta cuando el mejor individuo en las últimas k generaciones nunca fue factible.

El factor de penalización $\lambda(t+1)$ decrece si los últimos k mejores individuos fueron factibles, e incrementa si fueron no factibles. La principal desventaja de este método consiste en la definición de los parámetros β_1, β_2 y $\lambda(0)$.

Uno de los métodos más robustos para el manejo de restricciones es el propuesto por Hamida y Schoenauer [33], denominado **ASCHEA** (Adaptive Segregational Constraint Handling Evolutionary Algorithm). Este método se conforma por 3 componentes principales:

1. Una función de penalización adaptativa

$$\text{fitness}(x) = \begin{cases} f(x) & \text{si la solución es factible} \\ f(x) - p(x) & \text{en otro caso} \end{cases}$$

donde

$$p(x) = \alpha \sum_{j=1}^q g_j^+(x) + \alpha \sum_{j=q+1}^m |h_j(x)|$$

donde $g_j^+(x)$ es la parte positiva de $g_j(x)$ y α es el coeficiente de penalización para todas las restricciones del espacio de búsqueda. El valor de α se adapta de acuerdo al porcentaje deseado de soluciones factibles τ_{target} y al porcentaje actual de soluciones factibles en la generación t , τ_t , de manera que:

$$\alpha(t+1) = \begin{cases} \frac{\alpha(t)}{c} & \text{si } \tau_t > \tau_{target} \\ \alpha(t)c & \text{otro caso} \end{cases}$$

tal que $c > 1$ y τ_{target} son parámetros definidos por el usuario. El valor de $\alpha(0)$ se estima mediante la siguiente fórmula:

$$\alpha(0) = \left| \frac{\sum_{i=1}^n f_i(x)}{\sum_{i=1}^n V_i(x)} \right| \times 1000$$

donde $V_i(x)$ es la suma de las restricciones infringidas por el i-ésimo individuo.

2. Recombinación de restricciones (cruzamiento)

Combina una solución no factible con una factible cuando existe un número bajo de soluciones factibles con respecto a τ_{target} .

3. Selección de segregación basada en factibilidad

Se define un porcentaje de soluciones factibles τ_{select} basado en su valor de aptitud. El resto de los individuos son seleccionados por medio de algún método de selección basado en el valor de aptitud penalizado.

4.1.5. Recocido Simulado

Es un método para controlar los diversos factores de penalización, de manera que la penalización aumenta con respecto al tiempo y se construye en relación a las restricciones activas. Michalewicz y Attia [13], proponen utilizar la siguiente función de penalización:

$$fitness(x) = f(x) + \frac{1}{2\tau} \left[\sum_{i=1}^n \{\max[0, g_i(x)]\}^2 + \sum_{j=1}^m h_j^2(x) \right],$$

donde el valor de τ representa el “horario de enfriamiento”, el cual cambia cada determinado número de generaciones.

4.1.6. Penalizaciones Co-evolutivas

Consiste en el uso de dos sub-poblaciones. En una evolucionan los factores de penalización, mientras que en la otra evolucionan las soluciones codificadas del problema original. Requiere el cálculo de los parámetros de penalización para cada una de las dos sub-poblaciones, de manera que la complejidad del algoritmo aumenta. La función de penalización de un individuo se expresa por:

$$fitness(x) = f(x) - (c \times w_1 + v \times w_2),$$

donde c es la suma de las restricciones que han sido infringidas por el individuo x , y v es la cantidad de restricciones infringidas por el mismo individuo.

El alto grado de complejidad del algoritmo y el gran número de parámetros requeridos, hacen que éste método sea difícil de implementar.

4.1.7. Algoritmo Genético Segregado

Propuesto por Le Riche y colaboradores [16]. Consiste en utilizar dos diferentes factores de penalización, la población es entonces separada en dos partes, una basada en una penalización moderada y la otra en un valor de aptitud severamente penalizado. Las mejores soluciones de ambas partes son seleccionadas para la reproducción. La siguiente generación se conforma de los mejores individuos de entre el grupo de formado por padres e hijos.

La principal desventaja de este método es la necesidad de definir dos diferentes factores de penalización para una población.

4.1.8. Penalización Difusa

La penalización difusa fue propuesta por primera vez por Wu y Yu [34]. Esta técnica hace uso de reglas difusas para actualizar el valor del factor de penalización de un individuo. El valor de aptitud de una solución esta dado por la siguiente fórmula:

$$\text{fitness}(x) = \begin{cases} f(x) & \text{si la solución es factible} \\ f(x) - r_f \times G(x) & \text{otro caso} \end{cases}$$

donde $G(x) = \sum_{i=1}^n g_j^+(x) + \sum_{j=1}^m |h_i(x)|$ y r_f es definido por una función miembro que actualiza el factor de penalización utilizando información de la función objetivo y la cantidad de restricciones infringidas.

4.2. Algoritmos de Reparación

Los algoritmos que entran en este grupo, tienen como tarea principal la de convertir soluciones no factibles en soluciones factibles. Se puede presentar el caso en que la reparación de un individuo sea utilizada exclusivamente para la evaluación, o bien, puede llegar a reemplazar al individuo original. Esta técnica ha sido ampliamente utilizada en problemas de optimización combinatoria.

Existen diversos algoritmos de reparación como el GENOCOP III, propuesto por Michalewicz [17], que utiliza dos diferentes poblaciones de individuos, donde las soluciones de una población influyen en la manera de evaluar a los individuos de la otra población. Las soluciones de la primera población son reparadas de forma que asemejen a las soluciones de la segunda población.

El principal problema que presentan los algoritmos de reparación radica en que el proceso para reparar una solución no factible suele ser más costoso que resolver el problema original.

4.3. Restricciones y Objetivos

A diferencia de las funciones de penalización que combinan la función objetivo con las restricciones para asignar un valor de aptitud a un individuo. Existen métodos que son capaces de manejar restricciones y objetivos de forma separada.

4.3.1. Co-evolución

Propuesto por Paredis [12], consiste en tener dos poblaciones, la primera contiene las restricciones del espacio de búsqueda, mientras que la segunda contiene las soluciones potenciales y no necesariamente factibles del problema que se desea resolver. La presión de selección sobre los miembros de una población depende del valor de aptitud de los individuos de la otra población.

Una solución en la segunda población con alto valor de aptitud, representa una solución que satisface muchas restricciones, por otro lado, un individuo en la primera población con alto valor de aptitud, representa una restricción infringida por muchas soluciones pertenecientes a la segunda población.

4.3.2. Superioridad de Puntos Factibles

Consiste en asignar valores de aptitud más altos a aquellas soluciones que son factibles. Powell y Skolnick [6], propusieron una técnica que agrupa las soluciones factibles en el intervalo $(-\infty, 1)$ y las soluciones no factibles en el intervalo $(1, \infty)$. Los individuos son evaluados de acuerdo a la siguiente expresión:

$$\text{fitness}(x) = \begin{cases} f(x) & \text{si la solución es factible} \\ 1 + r \left(\sum_{j=1}^n g_j(x) + \sum_{j=1}^m h_j(x) \right) & \text{en otro caso} \end{cases}$$

donde $f(x)$ se encuentra dentro del intervalo $(-\infty, 1)$, mientras que $g_j(x)$ y $h_j(x)$ se encuentran en el intervalo $(1, \infty)$.

La exploración del espacio de búsqueda utiliza un torneo binario como proceso de selección, el cual se basa en tres aspectos fundamentales:

- Una solución factible es siempre preferible a una no factible.
- Entre dos soluciones factibles, aquella con mejor valor de aptitud es seleccionada.
- Entre dos soluciones no factibles, aquella con menor valor de penalización es seleccionada.

La principal desventaja de este método es la poca diversidad de la población, lo que ocasiona una pobre exploración del espacio de búsqueda.

4.3.3. Memoria Conductista

Propuesto por Schoenauer y Xanthakis [7]. Este método propone satisfacer secuencialmente cada una de las restricciones del espacio de búsqueda. Se utiliza la pena de muerte para eliminar de la población aquellas soluciones que no cumplen con al menos una restricción. Además requiere que las restricciones estén ordenadas linealmente de manera que el orden en que las restricciones son procesadas influya directamente en el resultado final del algoritmo.

4.3.4. Técnicas Multi-objetivo para Manejo de Restricciones

Consiste en convertir un problema mono-objetivo a uno multi-objetivo. Existen varios métodos que utilizan este concepto. Camponogara y Talukdar [22], proponen utilizar dos objetivos, uno referente a la función objetivo original, y otro referente a la suma de violación de restricciones. Jiménez y Verdegay [30], proponen un enfoque min-max con comparaciones simples durante el proceso de selección.

Un fenómeno que hay que tener en consideración es la existencia de conflictos entre objetivos, que ocasiona que el mejoramiento de alguno de ellos dé lugar al empeoramiento de algún otro. Para tratar este problema se han utilizado diversas técnicas:

- Métodos basados en el concepto de eficiencia de Pareto.
- Métodos basados en la combinación de objetivos, por ejemplo, el método de la suma ponderada, donde se trata de optimizar el valor obtenido mediante la suma de los valores correspondientes a los distintos objetivos, multiplicados cada uno por un coeficiente.

$$\min \sum_{i=1}^k w_i f_i(x),$$

donde $w_i \geq 0$ es el coeficiente de peso correspondiente al objetivo i .

- Métodos basados en la asignación de prioridades, donde se establecen prioridades entre los distintos objetivos de acuerdo a su jerarquía durante la ejecución del algoritmo.

Capítulo 5

Algoritmo EMEDA

En este capítulo se propone un método para resolver problemas de optimización con restricciones, mediante un algoritmo de estimación de distribución, basado en el concepto de mezcla de distribuciones normales. El algoritmo EMEDA (Expectation Maximization EDA) consiste en estimar la distribución de la población para cada una de las dimensiones del espacio de búsqueda; de manera que es necesario encontrar en cada iteración, los parámetros w_i , μ_i y Σ_i , para cada una de las dimensiones, tal que:

$$f(x; w, \mu, \Sigma) = \sum_{i=1}^g w_i \delta(x; \mu_i, \Sigma_i),$$

donde

$$\delta(x; \mu_i, \Sigma_i) = (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)}$$

Es importante mencionar que se implementaron diversas técnicas para llevar a cabo una buena estimación de la distribución, sin obtener muy buenos resultados. Dichas técnicas se encuentran sintetizadas en el apéndice B.

Uno de los principales problemas que afrontan los EDAs es la alta velocidad de convergencia que presentan. Por esta razón, en este apartado se hace un análisis de los diferentes métodos de selección y de control de diversidad implementados en el algoritmo propuesto, los cuales ayudan a realizar una mejor exploración del espacio de búsqueda, evitando caer en el problema de una convergencia acelerada. Además se describe a detalle el comportamiento de diversos algoritmos que afectan directamente en el control de diversidad, y se presentan algunos resultados obtenidos acerca del funcionamiento de ciertos parámetros de control.

Dado que el propósito principal del algoritmo es el de resolver problemas con restricciones, la población se puede dividir en individuos factibles y no factibles, de manera que aquellos individuos no aptos deben ser penalizados para poder encontrar el óptimo global. Lo anterior requiere de un método de evaluación eficiente que sea capaz de catalogar de manera correcta a los individuos de una población. El algoritmo EMEDA hace uso de un método de penalización estática para llevar a cabo dicha evaluación, durante este capítulo se analizan las principales características de este método, así como sus ventajas y desventajas. Además se presentan los resultados de diversas pruebas realizadas, con el objetivo de observar el comportamiento de ciertos parámetros que influyen en el proceso de penalización.

El algoritmo 10 muestra la estructura del algoritmo EMEDA.

Algoritmo 10 Algoritmo EMEDA

```

1:  $t \leftarrow 1$ 
2:  $N \leftarrow 200, s \leftarrow N/2, m \leftarrow 20$ 
3:  $\sigma^t \leftarrow \sigma_{start}$ 
4:  $\epsilon \leftarrow \epsilon_{start}$ 
5: Inicializar  $P^t$  de tamaño  $N$ 
6: Evaluar( $P^t, \epsilon$ )
7:  $X_{best} \leftarrow$  Mejor Individuo en  $P^t$ 
8:  $X_{prev} \leftarrow X_{best}$ 
9: mientras No se cumpla la condición de paro hacer
10:    $S \leftarrow$  Seleccion_por_Torneo( $P^t, s$ )
11:   para  $j = 1$  to  $D$  hacer
12:      $(W^j, \mu^j, \Sigma^j) \leftarrow$  EM( $S, G$ )
13:   fin para
14:    $H \leftarrow$  PDF( $W, \mu, \Sigma$ )
15:    $H^* \leftarrow$  Control_de_Diversidad( $H$ )
16:    $P_M, X_{prev}, \sigma^{(t+1)} \leftarrow$  Mutaciones_del_Mejor_Individuo( $X_{best}, X_{prev}, t, m, \sigma^t$ )
17:    $P^* \leftarrow \{P^t, H^*, P_M\}$ 
18:   Evaluar( $P^*, \epsilon$ )
19:    $P^{(t+1)} \leftarrow$  Selección_por_Truncamiento( $P^*, N$ )
20:   si  $fitness(P_0^t) < fitness(X_{best})$  entonces
21:      $X_{best} \leftarrow P_0^t$ 
22:   fin si
23:    $P_0^t \leftarrow X_{best}$ 
24:    $t \leftarrow t + 1$ 
25: fin mientras

```

La constante G es un parámetro definido por el programador, indicando el número de distribuciones gaussianas necesarias para construir el modelo de probabilidad. Los parámetros ϵ_{start} y σ_{start} , como se menciona más adelante en este capítulo, son necesarios para tener control sobre la diversidad de la población y también son definidos por el usuario.

El algoritmo EM se encarga de estimar un modelo de probabilidad basado en la suma de distribuciones normales a partir de una muestra significativa de los datos, lo cual permite generar nuevos y mejores individuos para las siguientes generaciones.

La función $PDF(W, \mu, \Sigma)$ tiene la función de generar nuevos individuos mediante el modelo estimado por el algoritmo EM, de forma que las nuevas soluciones se añaden a la población actual, para después pasar por un proceso de selección. A continuación se muestra el pseudocódigo de la función PDF.

Algoritmo 11 Algoritmo PDF

Entrada: $W = (w_1, \dots, w_G)$, $\mu = (\mu_1, \dots, \mu_G)$, $\Sigma = (\Sigma_1, \dots, \Sigma_G)$

Salida: H

```

1: para  $i = 1$  to  $N$  hacer
2:   para  $j = 1$  to  $D$  hacer
3:      $u \leftarrow rnd$ 
4:      $sum \leftarrow 0$ 
5:      $k \leftarrow 0$ 
6:     mientras  $sum < u$  hacer
7:        $k \leftarrow k + 1$ 
8:        $sum \leftarrow sum + W_k^j$ 
9:     fin mientras
10:     $H_j^i \leftarrow Nrnd(\mu_k^j, \Sigma_k^j)$ 
11:  fin para
12: fin para

```

La variable rnd representa un número aleatorio entre 0 y 1 procedente de una distribución uniforme, mientras que la función $Nrnd(\mu, \Sigma)$ devuelve un valor aleatorio proveniente de una distribución normal con media μ y varianza Σ .

Las funciones “Control_de_Diversidad”, y “Mutaciones_del_Mejor_Individuo”, aceleran la búsqueda del óptimo global, realizando una mejor exploración del espacio de búsqueda, y evitando que la población se estanke en un óptimo local.

La forma de evaluación empleada en el algoritmo propuesto se encarga de penalizar a aquellas soluciones fuera de la región factible, de manera que influyan

en menor grado en las generaciones subsecuentes.

Los métodos de selección implementados tienen la función de controlar la diversidad de la población manteniendo un balance entre soluciones factibles y no factibles, esto con la intención de generar mejores soluciones que encaminen la búsqueda del algoritmo hacia el óptimo global.

Así pues, el algoritmo EMEDA está compuesto de cuatro partes fundamentales:

1. Algoritmo EM como modelo para estimar la distribución de probabilidad de la población
2. Una técnica de evaluación
3. Un método de selección
4. Un método de control de diversidad

A continuación se procede a explicar cada una de las partes que componen este algoritmo.

5.1. Algoritmo EM

Propuesto por Dempster, Laird y Rubin (1977), el algoritmo EM (Expectation - Maximization) es un método iterativo utilizado para encontrar estimadores de máxima verosimilitud de parámetros en modelos probabilísticos que dependen de variables no observables.

En cada iteración del algoritmo se observa un parámetro z_i y existe un parámetro oculto x_i . Se denota a $Z = \{z_1, \dots, z_m\}$ como el conjunto observados, a $X = \{x_1, \dots, x_m\}$ al conjunto de datos no observados y a $Y = Z \cup X$ como el conjunto completo de datos. El conjunto X puede considerarse una variable aleatoria cuya distribución de probabilidad depende de los parámetros a estimar θ y los datos observados Z . Del mismo modo, Y se puede considerar como una variable aleatoria que está definida en términos de la variable aleatoria X . Se define a h como la hipótesis actual de los valores de los parámetros θ , y h' como la hipótesis que se estima en cada iteración del algoritmo.

El algoritmo EM busca encontrar la hipótesis h' que maximiza la esperanza $E[\ln p(Y|h')]$, donde $p(Y|\theta)$ es la distribución de probabilidad que define Y y que depende de los parámetros desconocidos θ .

Dado que la distribución de Y es desconocida, el algoritmo EM usa la hipótesis actual h para estimar la distribución de Y . Se define entonces la función $Q(h|h)$ que proporciona $E[\ln p(Y|h)]$ como una función de h , suponiendo que $\theta = h$ y dado el conjunto de observaciones Z .

$$Q(h|h) = E[\ln p(Y|h)|h, Z]$$

La distribución de probabilidad Y definida por Z y h , es aquella que se utiliza para calcular $E[\ln p(Y|h)]$ para una hipótesis cualquiera h . El algoritmo EM realiza los siguientes pasos en cada iteración:

Paso E

Paso de estimación. Calcula $Q(h|h)$ utilizando la hipótesis actual h y los datos observados Z para estimar la distribución de probabilidad Y .

$$Q(h|h) \leftarrow E[\ln p(Y|h)|h, Z]$$

Paso M

Paso maximización. Se sustituye h por la hipótesis h que maximiza la función $Q(h|h)$.

$$h \leftarrow \arg \max Q(h|h)$$

5.1.1. Mezcla de Gaussianas

Dados N puntos en un espacio n-dimensional, $x_1, \dots, x_N \in R^n$, y una familia F de funciones de densidad de probabilidad, se puede estimar una función de probabilidad $f(x, \theta) \in F$ capaz de generar los puntos dados. Si se considera que todas las funciones en F representan una distribución normal multivariada, la función $f(x, \theta)$ puede ser expresada por la siguiente ecuación:

$$f(x, \theta) = \sum_{i=1}^g w_i \delta(x; \mu_i, \Sigma_i),$$

donde

- $\theta = (w_1, \dots, w_g; \mu_1, \dots, \mu_g; \Sigma_1, \dots, \Sigma_g)$
- $0 \leq w_i \leq 1, \sum_{i=1}^g w_i = 1$
- μ_i es el vector de medias de tamaño $n \times 1$, correspondiente a la i-ésima distribución normal.

- Σ_i es la matriz de covarianzas de tamaño $n \times n$, correspondiente a la i-ésima distribución normal.
- $\delta(x; \mu_i, \Sigma_i) = (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} e^{[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)]}$

El primer paso consiste en obtener la función de verosimilitud para el conjunto de parámetros θ , a partir de una muestra de variables independientes e idénticamente distribuidas.

$$L(x, \theta) = \prod_{j=1}^n \left\{ \sum_{i=1}^g w_i \delta(x_j; \mu_i, \Sigma_i) \right\}$$

La función log-verosimilitud está determinada al aplicar logaritmo natural a la función de verosimilitud.

$$l(x, \theta) = \sum_{j=1}^n \ln \left\{ \sum_{i=1}^g w_i \delta(x_j; \mu_i, \Sigma_i) \right\}$$

Asumiendo que las observaciones de la muestra $x_1, \dots, x_N \in R^n$, donde x_i proviene de alguna de las distribuciones normales, y el conjunto $y^j = \{y_1^j, \dots, y_n^j\}$, donde y_i^j indica la distribución de la que proviene x_j , de manera que si la observación x_j proviene de la i-ésima distribución Gaussiana, entonces $y_i^j = 1$, en otro caso $y_i^j = 0$. En el caso en que $y_i^j = 1$, entonces $x_j \sim \delta(x; \mu_i, \Sigma_i)$. Un individuo tiene una probabilidad w_i de ser generado por la i-ésima distribución.

La función de verosimilitud de los datos completos es de la forma:

$$L(\theta, x, y) = \prod_{j=1}^n f(x_j, y_j; \theta) \prod_{j=1}^n f(x_j | y_j; \mu, \Sigma) f(y_j; w)$$

Simplificando

$$L(\theta, x, y) = \prod_{j=1}^n \prod_{i=1}^g [w_i \delta(x_j; \mu_i, \Sigma_i)]^{y_j^i}$$

La función de log-verosimilitud de los datos completos, queda expresada como:

$$l(\theta, x, y) = \sum_{j=1}^n \sum_{i=1}^g y_j^i [\ln w_i + \ln \delta(x_j; \mu_i, \Sigma_i)] \quad (5.1)$$

La función de log-verosimilitud puede ser vista como una variable aleatoria, pues los valores y^j , con $j = 1, \dots, n$, son desconocidos y aleatorios.

Paso E

Los datos no observados son sustituidos por sus valores esperados al calcular la esperanza condicional de (5.1), en base a los datos observados y los valores actuales de los parámetros.

$$Q(\theta|\theta^{(k)}) = E \left[l(\theta, x, y) | x, \theta^{(k)} \right]$$

Dada la linealidad que existe de la función de log-verosimilitud respecto a los datos no observados y_j^i . Es necesario el cálculo de la esperanza condicional actual de y_j^i , dados los datos no observados x y los valores de θ^k .

$$\begin{aligned} E[y_j^i | x, \theta^{(k)}] &= P[y_j^i = 1 | x_j; \mu_i, \Sigma_i] \\ &= \frac{P[y_j^i = 1] f(x_j | y_j^i = 1)}{f(x_j; \mu_i, \Sigma_i)} \\ &= \frac{w_i \delta(x_j; \mu_i, \Sigma_i)}{\sum_{h=1}^g w_h \delta(x_j; \mu_h, \Sigma_h)} \end{aligned}$$

Entonces, la esperanza condicional de (5.1) dados los datos observados y los valores actuales de los parámetros, se define como:

$$Q(\theta|\theta^{(k)}) = \sum_{j=1}^n \sum_{i=1}^g \phi(x_j; \theta^{(k)}) [\ln w_i + \ln \delta(x_j; \mu_i, \Sigma_i)], \quad (5.2)$$

donde

$$\phi(x_j; \theta^{(k)}) = \frac{w_i \delta(x_j; \mu_i, \Sigma_i)}{\sum_{h=1}^g w_h \delta(x_j; \mu_h, \Sigma_h)}$$

Paso M

Una vez reemplazados los valores no observados por sus esperanzas, el siguiente paso consiste en determinar los estimadores de máxima verosimilitud de θ , al maximizar la función (5.2) respecto al conjunto de parámetros contenidos en θ .

$$\begin{aligned} w_i^{(k+1)} &= \sum_{j=1}^n \frac{\phi_i(x_j; \theta^{(k)})}{n} \\ \mu_i^{(k+1)} &= \frac{\sum_{j=1}^n \phi_i(x_j; \theta^{(k)}) x_j}{\sum_{j=1}^n \phi_i(x_j; \theta^{(k)})} \\ \Sigma_i^{(k+1)} &= \frac{\sum_{j=1}^n \phi_i(x_j; \theta^{(k)}) (x_j - \mu_i^{(k+1)}) (x_j - \mu_i^{(k+1)})^T}{\sum_{j=1}^n \phi_i(x_j; \theta^{(k)})} \end{aligned}$$

La implementación del algoritmo consiste básicamente en actualizar continuamente los valores w , μ y Σ para cada una de las distribuciones gaussianas. A continuación se presenta el algoritmo EM utilizando el modelo de mezclas de distribuciones normales.

Algoritmo 12 Algoritmo EM

Entrada: $X = (x_1, \dots, x_n)$, g

Salida: Parámetros w , μ y Σ

- 1: $t \leftarrow 0$
- 2: $(w^t, \mu^t, \Sigma^t) \leftarrow \text{K-medias}(X, g)$
- 3: **mientras** No se cumpla la condición de paro **hacer**
- 4: Obtener la esperanza condicional de la función de log-verosimilitud de los datos observados.

$$Q(\theta|\theta^{(t)}) = \sum_{j=1}^n \sum_{i=1}^g \phi(x_j; \theta^{(t)}) [\ln w_i + \ln \delta(x_j; \mu_i, \Sigma_i)]$$

- 5: Estimar los estimadores de máxima verosimilitud de θ , al maximizar la función $Q(\theta|\theta^{(t)})$ respecto a los parámetros w , μ y Σ , de manera que:

$$\begin{aligned} w_i^{(t+1)} &= \sum_{j=1}^n \frac{\phi_i(x_j; \theta^{(t)})}{n} \\ \mu_i^{(t+1)} &= \frac{\sum_{j=1}^n \phi_i(x_j; \theta^{(t)}) x_j}{\sum_{j=1}^n \phi_i(x_j; \theta^{(t)})} \\ \Sigma_i^{(t+1)} &= \frac{\sum_{j=1}^n \phi_i(x_j; \theta^{(t)}) (x_j - \mu_i^{(t+1)}) (x_j - \mu_i^{(t+1)})^T}{\sum_{j=1}^n \phi_i(x_j; \theta^{(t)})} \end{aligned}$$

- 6: $t \leftarrow t + 1$
 - 7: **fin mientras**
-

Un aspecto que es importante mencionar es la aproximación inicial que se hace de los parámetros w , μ y Σ para cada una de las distribuciones gaussianas empleadas, ya que una mala aproximación puede influenciar negativamente en el desempeño del algoritmo EM. Por otro lado, una buena aproximación inicial ayuda al algoritmo a encontrar en pocas iteraciones una estimación adecuada de los parámetros.

Una opción para obtener una aproximación inicial de los parámetros que influya positivamente en el comportamiento del algoritmo EM, es utilizar el algoritmo "**k-medias**", el cual se describe a continuación:

5.1.2. Algoritmo K-Medias

Dado un conjunto de observaciones $X = (x_1, x_2, \dots, x_n)$ en un espacio n-dimensional, el algoritmo k-medias trata de dividir las n observaciones en k grupos $S = \{S_1, S_2, \dots, S_k\}$, con la intención de minimizar la función objetivo $f(x)$ definida como:

$$f(x) = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$

donde μ_i es la media de las observaciones pertenecientes a S_i .

La idea principal del algoritmo consiste en definir k centroides, uno para cada grupo, de manera que cada una de las observaciones se asocia al centroide más cercano. El siguiente paso consiste en calcular los nuevos k centroides como la media de las observaciones pertenecientes a cada uno de los grupos, tal que:

$$\mu_i = \frac{\sum_{j=1}^m x_j}{m},$$

donde m es el número de observaciones asociadas al i -ésimo grupo

Suponga una población de n individuos en un espacio de búsqueda en 2 dimensiones. El objetivo es el de dividir a la población en dos grupos utilizando el algoritmo k-medias. El primer paso del algoritmo consiste en la asignación de k ($k = 2$) centroides iniciales. Esto se puede realizar de varias maneras, una opción es elegir los puntos de manera aleatoria; una segunda opción es la de elegir aleatoriamente k diferentes puntos del conjunto de observaciones. El resultado depende en cierta forma de la colocación inicial de los centroides en el espacio de búsqueda, por esta razón se recomienda colocar cada centroide lo más alejado posible de los demás.

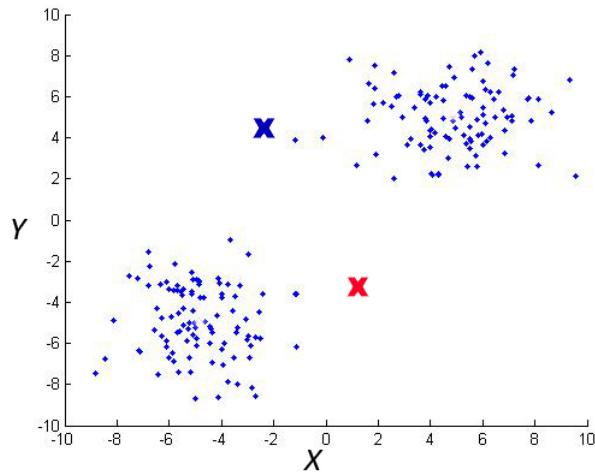


Figura 5.1: Paso 1. Proponer centroides iniciales

Una vez definidas las posiciones iniciales de los k centroides se procede a asociar cada una de las observaciones a su centroide más cercano, dividiendo el espacio de búsqueda en k diferentes regiones, lo que da como resultado un Diagrama de Voronoi.

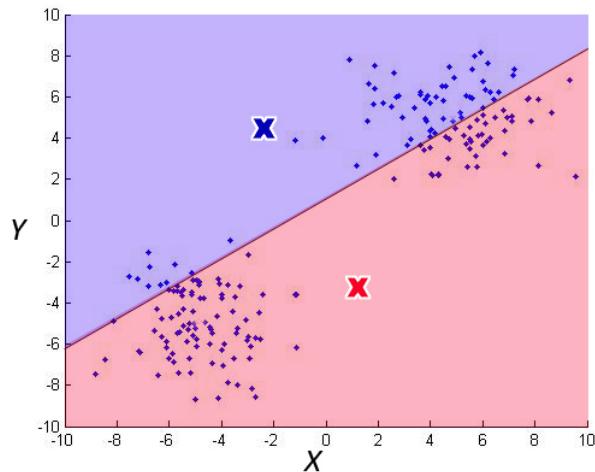


Figura 5.2: Paso 2. Asociar cada observación al centroide correspondiente

El siguiente paso consiste en re-calcular las posiciones de los k centroides. Para esto se obtiene la media de las observaciones asociadas a cada grupo, de manera que los valores obtenidos se convierten en los nuevos centroides.

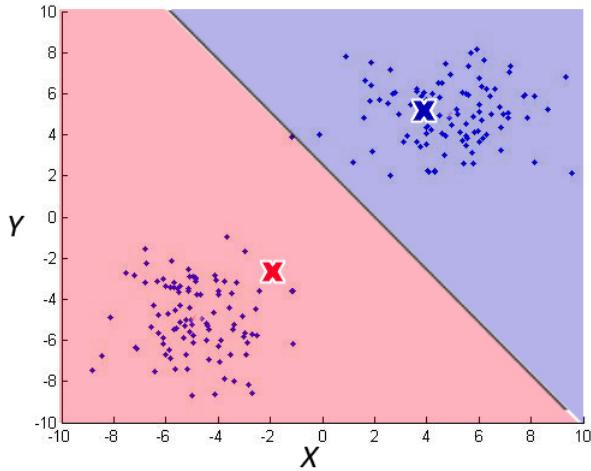


Figura 5.3: Paso 3. Re-calcular las posiciones de los centroides

El proceso se repite hasta que los centroides dejan de moverse, produciendo una separación de las observaciones en distintos grupos. A continuación se presenta el algoritmo k-medias para el cálculo de clusters a partir de una serie de observaciones.

Algoritmo 13 Algoritmo K.medias

Entrada: $X = (x_1, \dots, x_n)$, k

Salida: Parámetros W , μ y Σ

- 1: $t \leftarrow 0$
- 2: Definir centroides iniciales S^0
- 3: **mientras** No se cumpla la condición de paro **hacer**
- 4: Asociar cada una de las observaciones x_i al centroide más cercano, de manera que se minimice la función objetivo $f(x)$.

$$f(x) = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

- 5: Obtener la media de las observaciones asociadas a cada uno de los grupos
 - 6: $S^{(t+1)} \leftarrow \mu_i$
 - 7: $t \leftarrow t + 1$
 - 8: **fin mientras**
 - 9: **para** $i = 0$ to G **hacer**
 - 10: $w_i \leftarrow l(S_i)/n$
 - 11: $\mu_i \leftarrow S_i$
 - 12: $\Sigma_i \leftarrow \Sigma(S_i)$
 - 13: **fin para**
-

El valor de $l(S_i)$ es la cantidad de observaciones pertenecientes al i -ésimo grupo, mientras que $\Sigma(S_i)$ representa la matriz de covarianzas de los individuos asociados al grupo i .

A continuación se muestran algunos ejemplos del funcionamiento del algoritmo EM para estimar la suma de tres distribuciones normales, colocadas aleatoriamente en el espacio de búsqueda y con una determinada varianza. Para esto se generaron 100 individuos de manera aleatoria, en donde cualquier individuo puede provenir de cualquiera de las distribuciones. La línea azul indica la suma de las distribuciones normales, mientras que la línea roja indica la estimación hecha por el algoritmo.

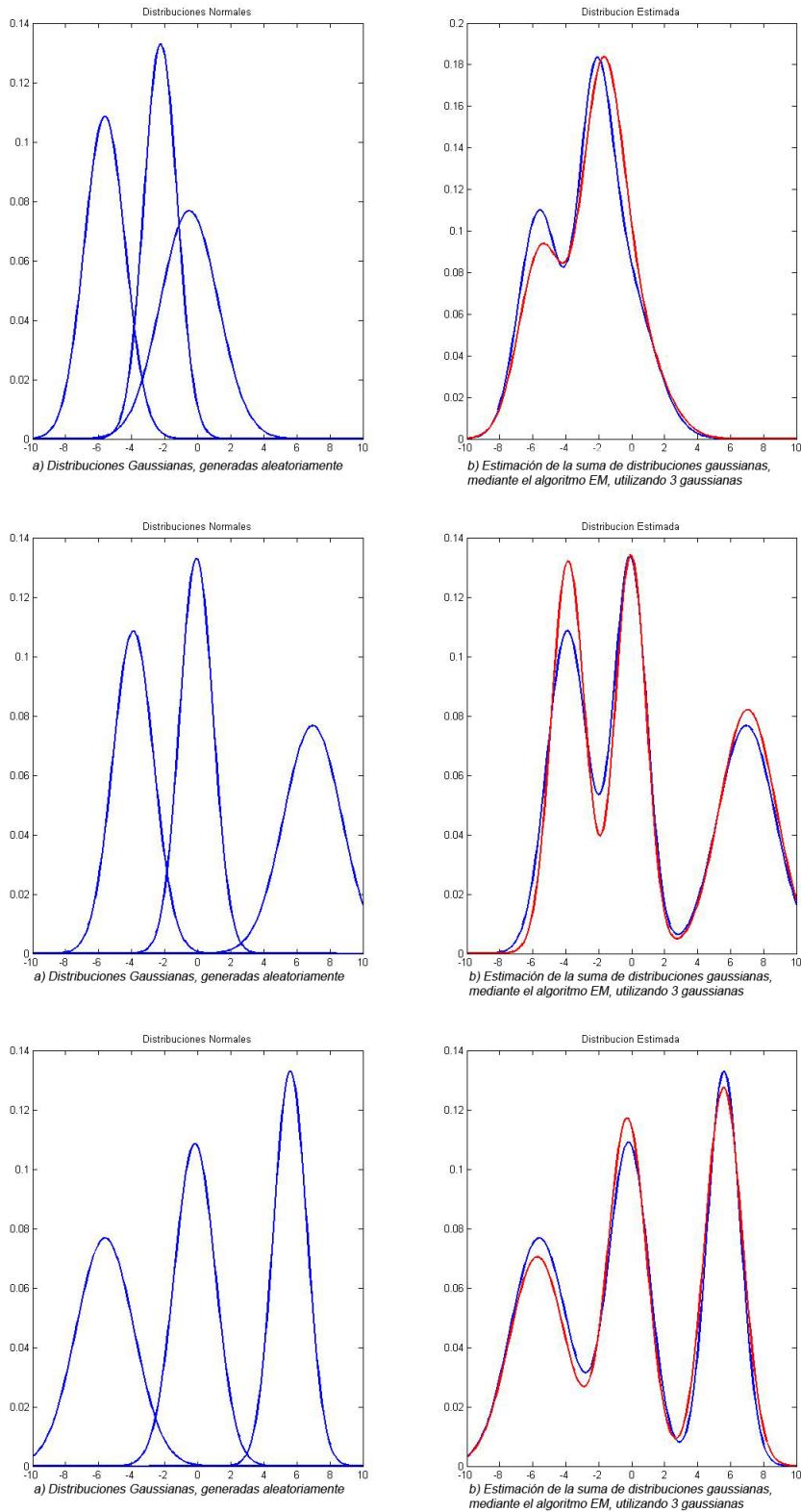


Figura 5.4: Algoritmo EM, utilizando 3 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales

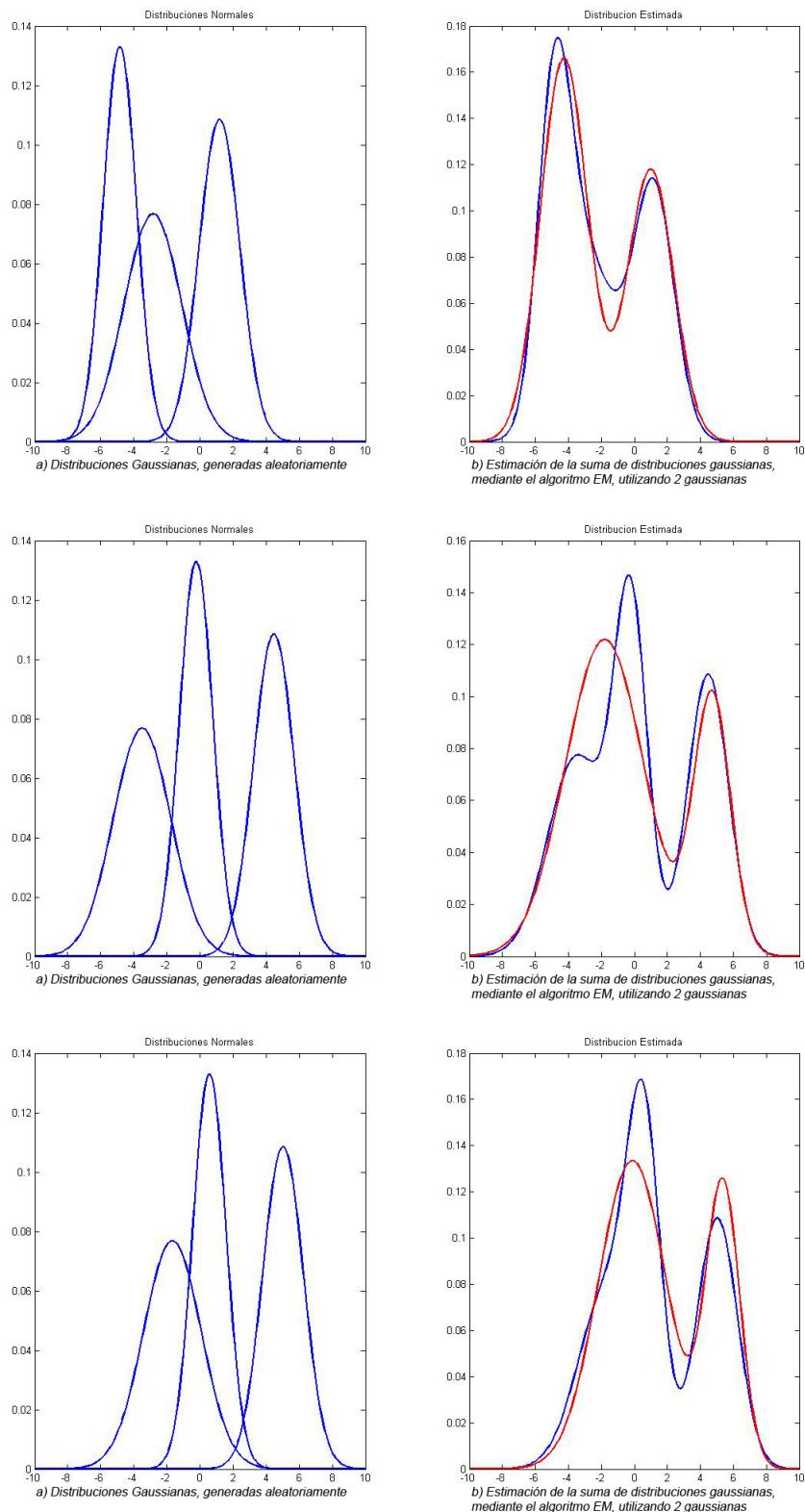


Figura 5.5: Algoritmo EM, utilizando 2 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales

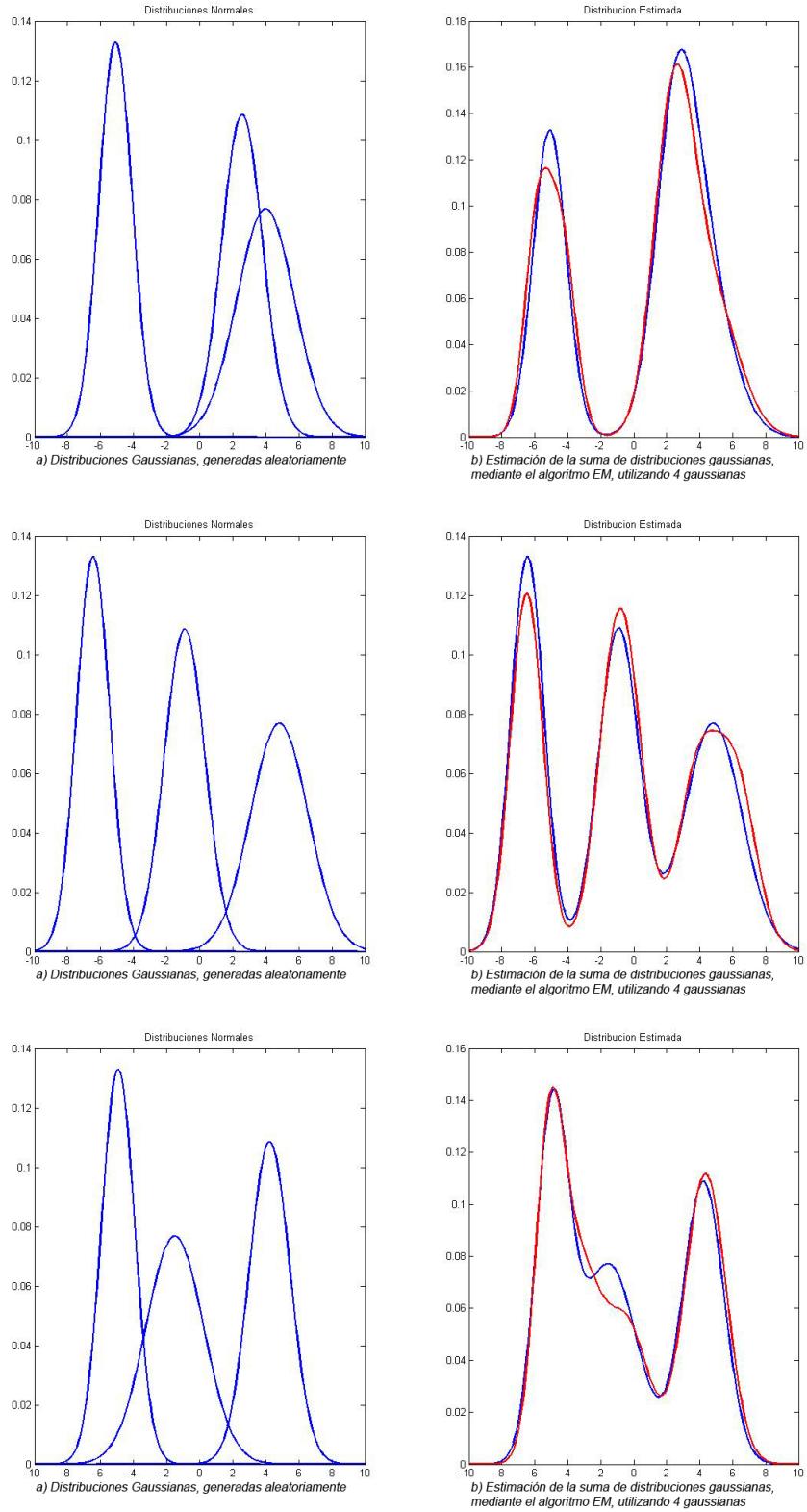


Figura 5.6: Algoritmo EM, utilizando 4 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales

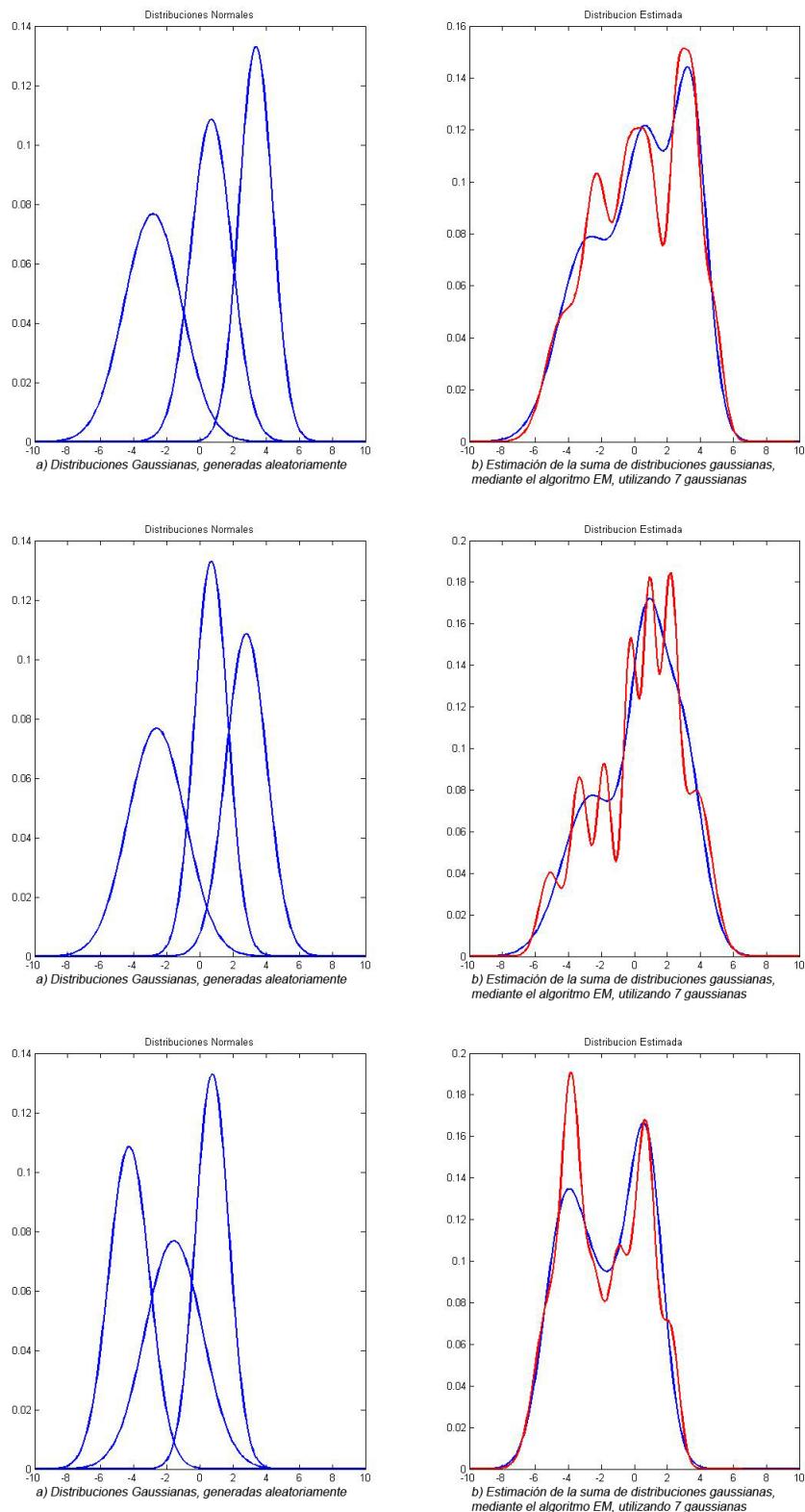


Figura 5.7: Algoritmo EM, utilizando 7 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales

5.2. Evaluación de Individuos

Como se menciona en el capítulo 4, existen dos tipos de restricciones, de desigualdad y de igualdad. Estas últimas, pueden definirse como restricciones de desigualdad de tal manera que:

$$|h_j(x)| - \epsilon \leq 0,$$

donde ϵ es un valor muy pequeño que representa a la tolerancia permitida, pero, ¿Cómo saber que valor de ϵ es más adecuado para el problema que se esta resolviendo? Si se considera un valor de ϵ muy pequeño, se esta limitando el espacio de búsqueda, pues demanda una gran precisión. Por otro lado, si ϵ toma un valor muy grande, se tiene una mayor exploración, pero es susceptible de caer en regiones no factibles.

La figura 5.6 muestra un ejemplo de un espacio de búsqueda, donde $h(x)$ representa una restricción de igualdad, y la región sombreada representa la zona factible, de tal manera que todos aquellos individuos fuera de esta zona son penalizados de una determinada manera.

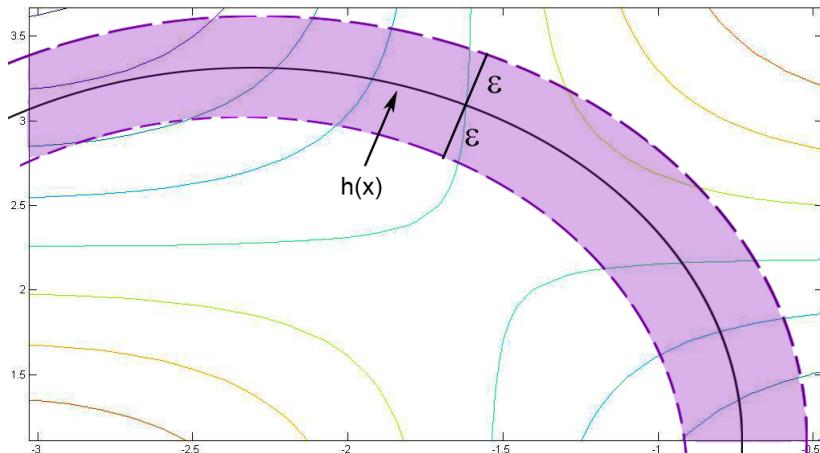


Figura 5.8: Espacio de Búsqueda con Restricciones de igualdad

Una buena forma de lidiar con el problema de la penalización cuando se tienen restricciones de igualdad, es hacer una zona factible adaptable, es decir, que vaya cambiando con el transcurso de las generaciones. Para hacer esto solo hay que cambiar el valor de ϵ , si el valor disminuye, tambien lo hará la zona factible y si aumenta también esta lo hará.

La idea consiste en iniciar con un valor de ϵ relativamente grande ($\epsilon \approx 1$) y disminuirlo gradualmente conforme la población vaya convergiendo a la solución,

de manera que:

$$\epsilon^{k+1} = \begin{cases} \epsilon^k & \text{si } p_f < p_t \\ \beta\epsilon^k & \text{si } p_f \geq p_t \end{cases}$$

donde p_f es el porcentaje de individuos dentro de la región factible, los parámetros p_t ($0 < p_t \leq 1$) y β ($0 < \beta \leq 1$) son definidos por el programador. Varias pruebas han demostrado que $p_t \approx 0.7$ y $\beta = 0.95$ resultan ser una buena opción al momento de la implementación.

Es necesario tener un límite inferior para el valor ϵ , pues conforme avancen las generaciones, el algoritmo demandara una mayor precisión de las soluciones, y la mayoría de ellas serán descartadas para formar parte en las siguientes iteraciones del algoritmo. Un valor mínimo en el orden de 10^{-5} es suficiente para que el algoritmo siga en busca de la solución a través del espacio de búsqueda.

El hecho de contar con un valor de ϵ lo suficientemente grande para las restricciones de igualdad, hace que los individuos que se encuentran lejos de la región factible no sean penalizados y sean catalogados como aptos para las siguientes generaciones, logrando una mejor exploración y evitando llevar a la población a una convergencia acelerada.

Los factores de penalización permanecen sin cambio durante toda la ejecución, de tal manera que el valor de aptitud de cada individuo de la población, esta representado por:

$$fitness(x) = f(x) + p(x), \quad (5.3)$$

donde

$$p(x) = \alpha \sum_{j=1}^q g_j^+(x) + \alpha \sum_{j=q+1}^m H_j^+(x),$$

donde $g_j^+(x)$ es la parte positiva de $g_j(x)$, $H_j^+(x)$ es la parte positiva de $|h(x)| - \epsilon$, y α es el coeficiente de penalización para todas las restricciones del espacio de búsqueda. El parámetro α es constante y se recomienda tomar un valor del orden de 10^4 , para que de esta manera los individuos fuera de la región factible sean fuertemente penalizados.

El valor de α depende en gran medida del problema que se desea resolver. Existen casos donde no es necesario definir un valor de α demasiado alto, en otros casos si no se penaliza fuertemente, el algoritmo no logra converger al óptimo global. A continuación se presenta el pseudocódigo de la función *Evaluar*,

encargada de penalizar a los individuos de la población.

Algoritmo 14 Algoritmo Evaluar

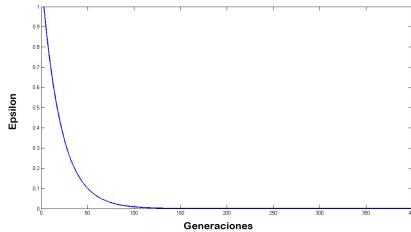
Entrada: $X = (x_1, \dots, x_n)$, ϵ

- 1: Definir α
 - 2: Definir β
 - 3: **para** $i = 1$ to n **hacer**
 - 4: $p(x_i) \leftarrow \alpha \sum_{j=1}^q g_j^+(x_i) + \alpha \sum_{j=q+1}^m H_j^+(x_i)$
 - 5: $fitness(x_i) \leftarrow f(x_i) + p(x_i)$
 - 6: **fin para**
 - 7: $pf \leftarrow$ Porcentaje de soluciones factibles
 - 8: **si** $pf \geq p_t$ **entonces**
 - 9: $\epsilon \leftarrow \beta \times \epsilon$
 - 10: **fin si**
-

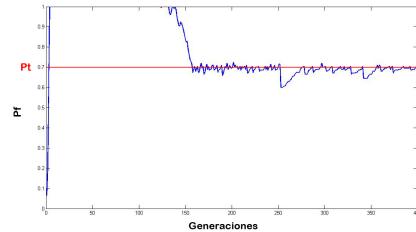
La función de aptitud sirve para clasificar a los individuos como aptos o como no aptos mediante un operador de selección, aunque debido al ajuste del parámetro ϵ , es posible que ciertos individuos que son factibles en una generación, no lo sean en generaciones posteriores, lo cual ayuda en gran medida a la explotación del espacio de búsqueda.

Para resolver problemas de optimización con restricciones, es necesario mantener un balance entre soluciones factibles y no factibles, de manera que se pueda alcanzar el óptimo global. La principal desventaja de hacer del parámetro ϵ un parámetro ajustable, es que en ocasiones la población llega a converger a una solución que está fuera de la región factible.

A continuación se muestra el comportamiento del parámetro ϵ durante el transcurso de varias generaciones, al utilizar el algoritmo EMEDA para optimizar las funciones de benchmark que cuentan con restricciones de igualdad. Estas funciones se encuentran definidas en el apéndice A.

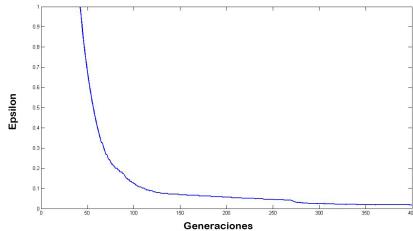
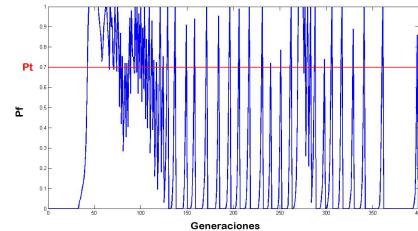


parámetro ϵ para la función $g03$

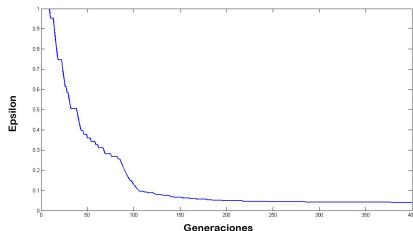
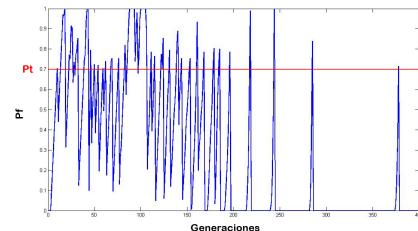


proporción de individuos factibles en $g03$

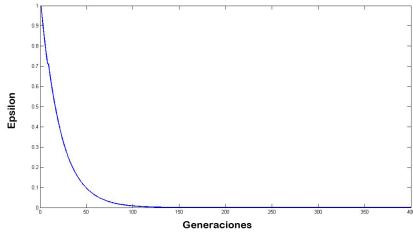
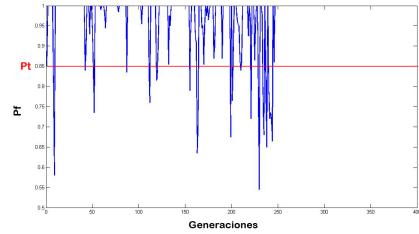
Para la función $g03$, el valor de ϵ disminuye casi inmediatamente, ya que el porcentaje de individuos factibles está por encima del valor P_t desde las primeras generaciones. A medida que transcurre el proceso evolutivo y la población converge al óptimo global, el porcentaje de soluciones dentro de la región factible comienza a oscilar alrededor del valor de P_t , exigiendo una mayor precisión del algoritmo.

parámetro ϵ para la función $g05$ proporción de individuos factibles en $g05$

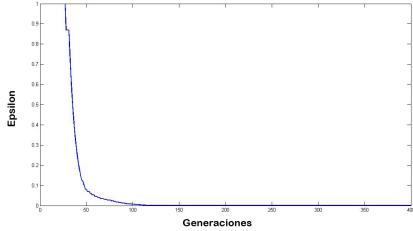
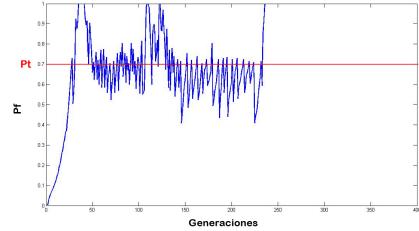
En la función $g05$ el porcentaje de individuos factibles P_f se encuentra por debajo del valor de referencia P_t durante las primeras generaciones; una vez que se rebasa este valor, el parámetro ϵ comienza a disminuir, ocasionando que la región factible reduzca su tamaño de tal forma que soluciones que eran factibles en generaciones anteriores dejan de serlo. Lo anterior afecta directamente en el porcentaje de soluciones factibles, lo que causa que P_f oscile bruscamente durante el proceso evolutivo.

parámetro ϵ para la función $g10$ proporción de individuos factibles en $g10$

Aunque la función $g10$ no cuenta con restricciones de igualdad, se sabe que en el óptimo las tres primeras restricciones son activas. Si se consideran estas restricciones como restricciones de igualdad, se observa como el valor de ϵ disminuye en forma escalonada, debido a que individuos catalogados como aptos dejan de serlo conforme se reduce la zona factible. Una vez que la población comienza a converger a la solución, la oscilación de P_f se vuelve menos frecuente, manteniendo al parámetro ϵ lejos del mínimo permitido, ya que el porcentaje de soluciones factibles permanece por debajo del valor P_t especificado.

parámetro ϵ para la función $g11$ proporción de individuos factibles en $g11$

El porcentaje de soluciones factibles en la función $g11$ se mantiene por encima de P_t en la mayor parte del proceso evolutivo, cuando este porcentaje se encuentra por debajo de P_t , lo hace sólo por un corto lapso de tiempo, pues inmediatamente vuelve a subir. Lo anterior ocasiona que la región factible disminuya desde las primeras generaciones hasta llegar prácticamente al límite inferior permitido, exigiendo así una mayor precisión en la búsqueda del óptimo global.

parámetro ϵ para la función $g13$ proporción de individuos factibles en $g13$

En la función $g13$ el parámetro ϵ permanece sin cambio durante las primeras generaciones del algoritmo, una vez que la variable P_f rebasa el valor de P_t el porcentaje de soluciones factibles en la población comienza a oscilar alrededor de P_t , hasta llegar a un punto en que el valor de P_f se mantiene por encima de P_t durante lo que resta del proceso evolutivo, ocasionando que la zona factible disminuya gradualmente su tamaño hasta llegar al límite inferior permitido, logrando así una exploración más precisa del espacio de búsqueda.

5.3. Método de Selección

Los operadores de selección son una parte fundamental en el comportamiento de los algoritmos evolutivos, ya que influyen de gran manera en la exploración del espacio de búsqueda, de manera que una mala selección de individuos, puede ocasionar que la población converja a un óptimo local, en vez de al óptimo global.

Para el algoritmo EMEDA se requiere de dos procesos de selección, el primero se encarga de tomar una muestra de tamaño m ($m \leq n$) de la población, la cual sirve como referencia para el modelo de estimación de distribución, quien se encarga de obtener la nueva generación de individuos. Para este caso se eligió el método de *selección por torneo*, ya que permite una mejor exploración del espacio de búsqueda. El proceso consiste en elegir dos individuos de manera aleatoria y quedarse con aquel cuyo valor de aptitud sea menor. El valor de aptitud de un individuo quedó previamente definido por la ecuación (5.3).

El siguiente algoritmo presenta el método de selección por torneo implementado, el cual tiene como objetivo elegir una muestra significativa de la población en base a cual se construirá el modelo de probabilidad. Los parámetros de entrada del algoritmo consiste en una población X y la cantidad de individuos m que se desean seleccionar.

Algoritmo 15 Selección_por_Torneo

Entrada: $X = (x_1, \dots, x_n)$, m

Salida: $S = (s_1, \dots, s_m)$

```

1: para  $i = 1$  to  $m$  hacer
2:    $a \leftarrow Irnd(n)$ 
3:    $b \leftarrow Irnd(n)$ 
4:   mientras  $a = b$  hacer
5:      $b \leftarrow Irnd(n)$ 
6:   fin mientras
7:   si  $fitness(x_a) < fitness(x_b)$  entonces
8:      $s_i \leftarrow x_a$ 
9:   si no
10:     $s_i \leftarrow X_b$ 
11:  fin si
12: fin para

```

La función $Irnd(n)$ devuelve un número entero aleatorio entre 1 y n . La salida del algoritmo es el conjunto S de individuos seleccionados.

El segundo proceso de selección se trata de un método similar a la estrategia $(\mu + \lambda)$, en donde los mejores μ individuos son escogidos de la población formada de μ padres y λ hijos, los cuales son generados mediante el modelo de estimación de distribución y por un porcentaje pequeño de mutaciones del mejor individuo. El método consiste en ordenar al conjunto de soluciones que conforman la población de acuerdo a su valor de aptitud, para luego proceder a truncar dicha población, y de esta manera iniciar la nueva generación con los individuos más aptos. Para llevar a cabo este método, se implemento la función

Selección_por_Truncamiento, que al igual que en la selección por torneo, necesita como parámetros de entrada una población y el número de individuos que se quieren seleccionar.

Algoritmo 16 Selección_por_Truncamiento

Entrada: $X = (x_1, \dots, x_n)$, m

Salida: $S = (s_1, \dots, s_m)$

```

1: sort( $X$ )
2: para  $i = 1$  to  $m$  hacer
3:    $s_i \leftarrow x_1$ 
4: fin para

```

La función $\text{sort}(X)$ ordena a la población X en base al valor de aptitud de sus individuos. Existen diversos métodos de selección, por lo que se deja a elección del programador el algoritmo a utilizar.

5.4. Control de Diversidad

El control de la diversidad en la población es un proceso de gran importancia para los algoritmos evolutivos, ya que ayuda a realizar una mejor exploración del espacio de búsqueda, y de esta manera disminuye el riesgo de que la población converja a un óptimo local.

Al implementar un EDA es importante contar con un método eficiente que permita tener cierto control sobre la diversidad de la población, ya que uno de los principales problemas que afrontan estos algoritmos es su alta velocidad de convergencia, lo cual puede ocasionar que la población se estanque en un óptimo local de manera precipitada.

Para lidiar con el problema de la diversidad, el EMEDA emplea un método que consiste en tomar dos individuos (X_{r1} y X_{r2}) de la población de manera aleatoria, con el objetivo de cambiar en cierto grado la dirección del j -ésimo individuo generado por el modelo de distribución de probabilidad, tal que:

$$X_j = X_j + K(X_{r2} - X_{r1})$$

La diferencia escalada entre los dos individuos elegidos aleatoriamente $K(X_{r2} - X_{r1})$, define la dirección y longitud de la búsqueda, donde la constante K es un parámetro de control entre 0 y 1, que se encarga de mantener un balance entre la explotación y la exploración del espacio de búsqueda. Véase la figura 5.6.

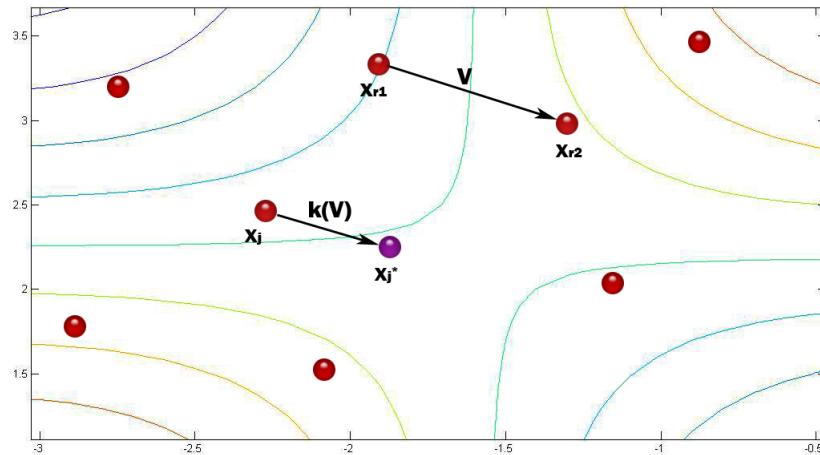


Figura 5.9: Control de Diversidad de la Población

La idea principal de este método consiste en adaptar el tamaño de paso conforme avanza el proceso evolutivo, de manera que en las primeras generaciones el tamaño de paso es demasiado grande, ya que la distancia entre los individuos es considerable. Conforme pasan las generaciones, los individuos se acercan cada vez más los unos a los otros, reduciendo el tamaño de paso.

Un factor que es importante resaltar es el parámetro P_r , que representa la probabilidad de cambio de cada una de las dimensiones que conforman al individuo que se desea modificar X_j . Este parámetro es una constante y es definida por el programador. Si $P_r = 1$, entonces el individuo va a cambiar en todas sus dimensiones, por otro lado, si $P_r = 0$, la nueva solución va a ser exactamente igual a la solución actual. El hecho de definir el valor de P_r por debajo de 1, permitiendo que se conserven ciertos datos de la solución actual, incrementa las capacidades de exploración, lo cual resulta en una mejor convergencia del algoritmo. se aconseja $P_r \approx 0.8$. Entonces el j-ésimo individuo queda definido por:

$$X_i^j = \begin{cases} X_i^{(r2)} + K(X_i^{(r2)} - X_i^{(r1)}) & \text{si } rnd < P_r, \quad i = 1, \dots, D \\ X_i^j & \text{si } rnd \geq P_r, \quad i = 1, \dots, D \end{cases}$$

donde rnd es un valor aleatorio en el intervalo $[0, 1]$, y D es la dimensión del espacio de búsqueda. Este proceso se lleva a cabo para cada uno de los individuos generados por el modelo de probabilidad. Los individuos X_{r1} y X_{r2} provienen de la población original y son elegidos aleatoriamente.

El algoritmo 17 describe el método implementado para llevar a cabo un mejor control sobre la diversidad de la población.

Algoritmo 17 Control_de_Diversidad

Entrada: $X = (x_1, \dots, x_n)$ **Salida:** $S = (s_1, \dots, s_n)$

```

1: Definir  $P_r$ 
2: para  $i = 1$  to  $n$  hacer
3:    $a \leftarrow Irnd(n)$ 
4:    $b \leftarrow Irnd(n)$ 
5:   mientras  $a = b$  hacer
6:      $b \leftarrow Irnd(n)$ 
7:   fin mientras
8:   para  $j = 1$  to  $D$  hacer
9:      $u \leftarrow rnd$ 
10:    si  $u < P_r$  entonces
11:       $s_j^i \leftarrow x_j^i + rnd \times (x_j^b - x_j^a)$ 
12:    si no
13:       $s_j^i \leftarrow x_j^i$ 
14:    fin si
15:  fin para
16: fin para

```

donde $Irnd(n)$ devuelve un número entero aleatorio entre 1 y n , rnd es un número aleatorio entre 0 y 1 proveniente de una distribución uniforme, y la constante D representa la dimensión del espacio de búsqueda.

Aunque este método ayuda en gran medida al algoritmo a realizar una mejor exploración del espacio de búsqueda, no es suficiente, ya que cuando la distancia entre los individuos es demasiado pequeña, el tamaño de paso es prácticamente nulo, de manera que si la población se encuentra atrapada en un óptimo local, le sería imposible dirigirse hacia el óptimo global.

Lo anterior sugiere la necesidad de un método que permita seguir explorando el espacio de búsqueda, aún cuando la población se encuentre totalmente estancada en un óptimo local.

5.4.1. Mutaciones del Mejor Individuo

Crear mutaciones del mejor individuo, permite seguir explorando el espacio de búsqueda, aún cuando la población se encuentre atrapada en un óptimo local, lo cual ayuda en gran medida al algoritmo a encontrar la solución al problema que se desea resolver.

El proceso consiste en generar un porcentaje pequeño de individuos mediante una distribución normal, con media μ igual a la posición del mejor individuo de la población, y varianza σ^2 . Esto ocasiona que las nuevas soluciones se coloquen alrededor del mejor individuo en busca del óptimo global. En la figura 5.7 se muestra un ejemplo del proceso de mutación, donde el punto de color rojo representa al mejor individuo de la población, mientras que los puntos azules representan a las mutaciones de éste.

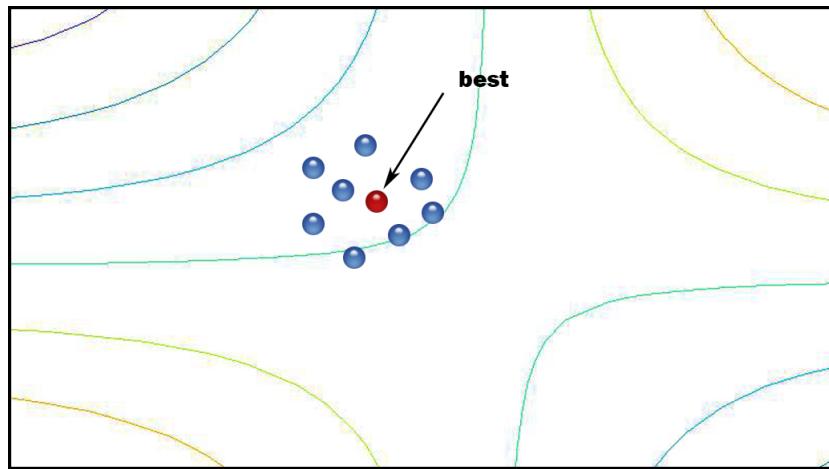


Figura 5.10: Mutaciones del mejor individuo

El principal problema de este método consiste en calcular el parámetro σ , ya que un valor de σ demasiado pequeño, podría no ser suficiente para llevar a cabo una buena exploración, por otro lado, si el valor de σ toma un valor muy grande, debido al pequeño número de individuos generados, estos se dispersarán sobre el espacio haciendo muy difícil encontrar una mejor solución.

La idea principal consiste en hacer del parámetro σ un parámetro adaptable, de tal forma que éste se modifique conforme avance el proceso evolutivo. En las primeras generaciones es conveniente que el valor de σ sea relativamente grande, con la intención de realizar una mejor exploración del espacio de búsqueda. A medida que la población tiende a converger al óptimo global, el valor de σ disminuye, exigiendo una mayor precisión en la búsqueda de nuevas soluciones. Con la finalidad de que las mutaciones se encuentren dentro de un rango determinado, es necesario fijar un límite inferior (σ_l) y un límite superior (σ_u) para el valor de σ . En el algoritmo propuesto se utilizó $\sigma_l = 1 \times 10^{-3}$ y $\sigma_u = 1$.

El ajuste de la varianza consiste en multiplicar el parámetro σ por un factor de escalamiento que permita aumentar o disminuir el rango de búsqueda en el espacio. En caso de que no exista una mejora en la solución, el parámetro

σ se multiplica por un factor α ($1 < \alpha < 2$), ocasionando que las próximas mutaciones se encuentren más alejadas del mejor individuo. Por otro lado, si la nueva solución es mejor que las anteriores, el valor de σ se divide por un factor β ($1 < \beta < 2$, $\beta < \alpha$), ocasionando una reducción en el rango de búsqueda y exigiendo una mayor precisión del algoritmo. Si el valor de σ sobrepasa el límite superior σ_u , éste se reinicia desde un valor más pequeño cercano al límite inferior σ_l , con la intención de comenzar una nueva búsqueda desde las cercanías del mejor individuo.

En la figura 5.8 se muestra la función de densidad para una variable X que proviene de una distribución normal con media 0 y varianza σ^2 . Se puede apreciar el comportamiento del parámetro σ , el cual puede tomar diferentes valores que van desde σ_l (color rojo), hasta σ_u (color verde), de manera que entre mayor sea el valor de σ , mayor será el rango de búsqueda en el espacio. Es importante mencionar el hecho de que la distribución original de los datos no se pierde durante el ajuste de la varianza, lo cual ayuda en gran medida al proceso evolutivo, pues permite conservar al mejor individuo de la población, quien funciona como base para crear las mutaciones.

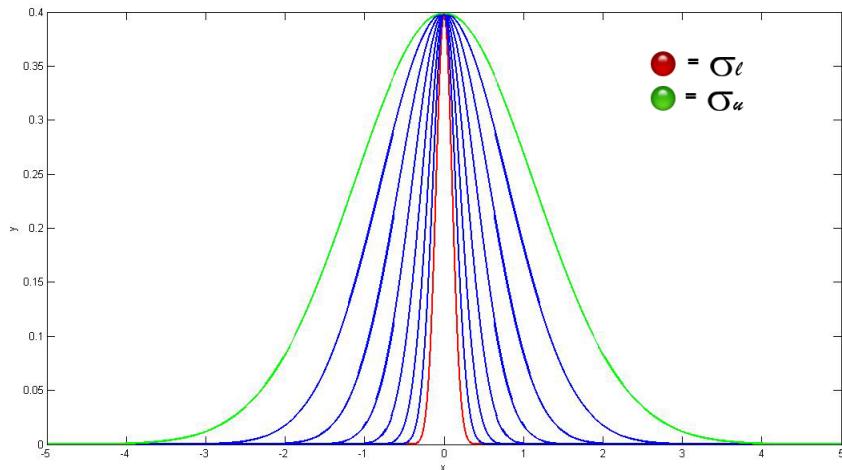


Figura 5.11: Ajuste de varianza

Otro aspecto que es importante considerar es la frecuencia con la que se ajusta el valor de σ para llevar a cabo las mutaciones. Una forma consiste en actualizar dicho parámetro después de transcurrido un número determinado de generaciones, de manera que el algoritmo cuente con el tiempo suficiente para explorar el espacio de búsqueda. No es conveniente modificar el valor de la varianza inmediatamente, pues no se realizaría una búsqueda del espacio lo suficiente-

mente eficiente como para encontrar una mejor solución. Por otro lado, el hecho de mantener constante la varianza por largos periodos de tiempo, no ayuda al algoritmo, ya que se estaría limitando el rango de búsqueda.

En pocas palabras, el proceso de ajuste de varianza consiste en verificar cada k generaciones si la población mejoró, si eso sucede, la varianza disminuye, en caso contrario esta aumenta. A continuación se presenta una descripción del procedimiento utilizado en el algoritmo propuesto para crear mutaciones del mejor individuo y ajustar la varianza.

Algoritmo 18 Mutaciones_del_Mejor_Individuo

Entrada: X_{best} , X_{prev} , m , t , σ

Salida: $S = (s_1, \dots, s_m)$, σ^*

```

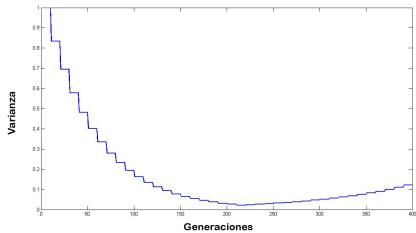
1: Definir  $k$ ,  $\beta$ ,  $\alpha$ 
2: para  $i = 1$  to  $m$  hacer
3:    $s_i \leftarrow Nrnd(X_{best}, \sigma)$ 
4: fin para
5: si  $t \bmod k = 0$  entonces
6:   si  $fitness(X_{best}) < fitness(X_{prev})$  entonces
7:      $\sigma^* \leftarrow \sigma/\beta$ 
8:   si  $\sigma < \sigma_l$  entonces
9:      $\sigma^* \leftarrow \sigma_l$ 
10:  fin si
11:  si no
12:     $\sigma^* \leftarrow \alpha \times \sigma$ 
13:    si  $\sigma > \sigma_u$  entonces
14:       $\sigma^* \leftarrow \sigma_m$ 
15:    fin si
16:  fin si
17:   $X_{prev} \leftarrow X_{best}$ 
18: fin si

```

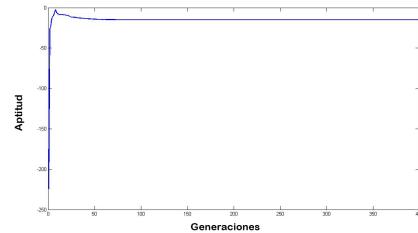
donde X_{best} representa al mejor individuo de la población actual, X_{prev} es el mejor individuo de la k -ésima generación anterior, la constante m determina el número de mutaciones por realizar, t es la generación en la que se encuentra actualmente el algoritmo, y la variable σ representa la desviación estándar con la cual se generan los nuevos individuos. Los parámetros σ_l , σ_u , σ_m y k son parámetros definidos por el programador. La función $Nrnd(\mu, \sigma)$ devuelve un número aleatorio proveniente de una distribución normal con media μ y varianza σ^2 .

A continuación se presenta una serie de gráficas que muestran el comportamiento del parámetro σ durante cierto número de generaciones del proceso

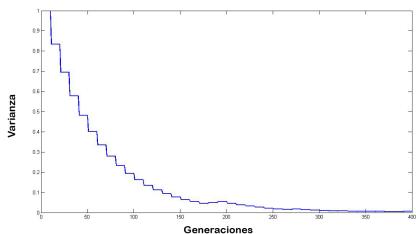
evolutivo para cada una de las funciones de Benchmark.



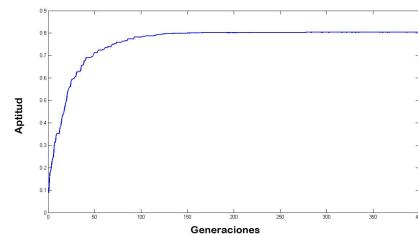
Parámetro σ para la función $g01$



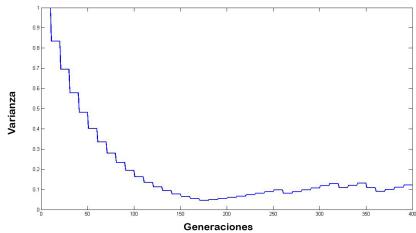
Aptitud del mejor individuo en $g01$



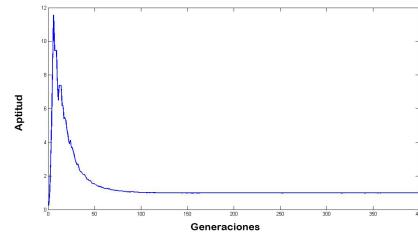
Parámetro σ para la función $g02$



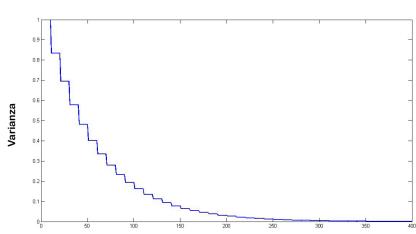
Aptitud del mejor individuo en $g02$



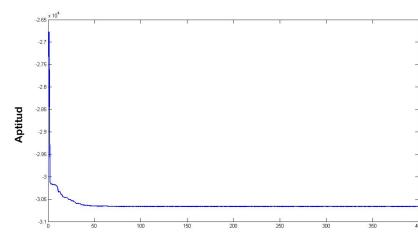
Parámetro σ para la función $g03$



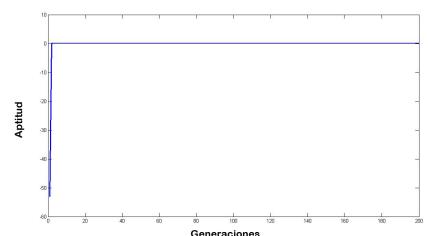
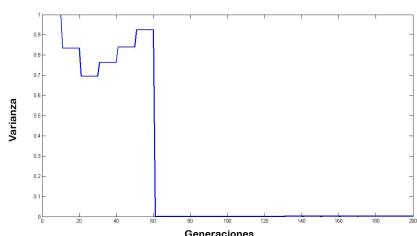
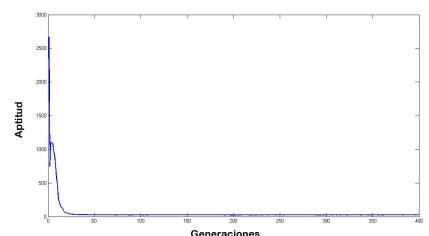
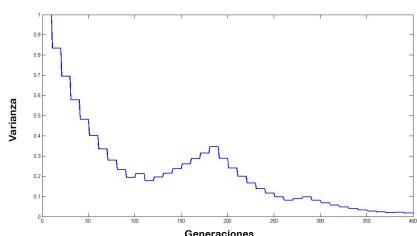
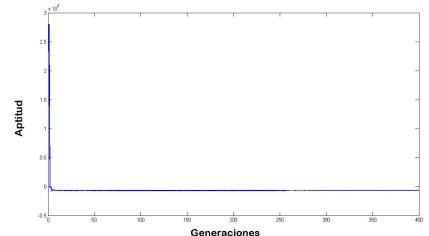
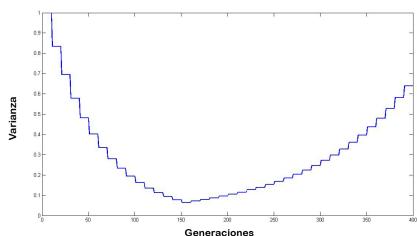
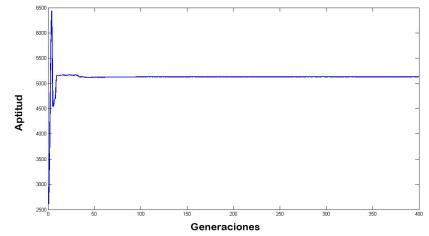
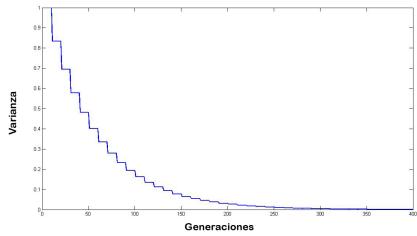
Aptitud del mejor individuo en $g03$

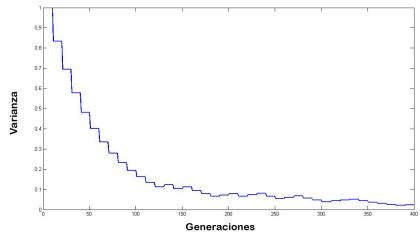
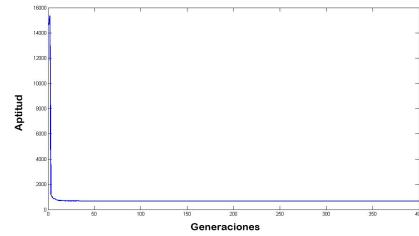
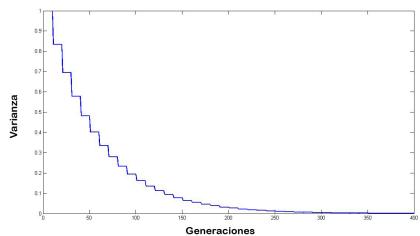
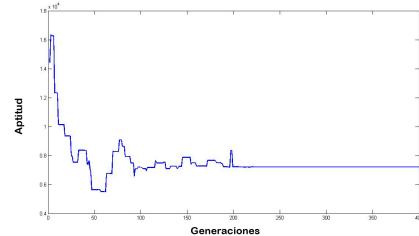
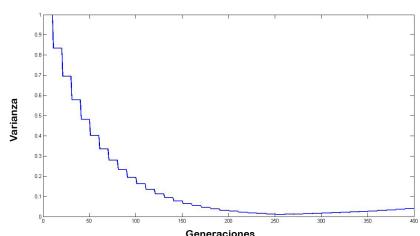
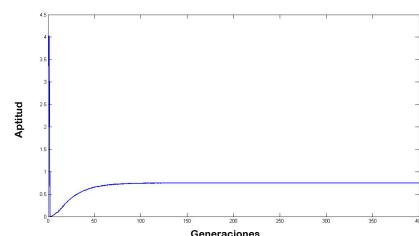
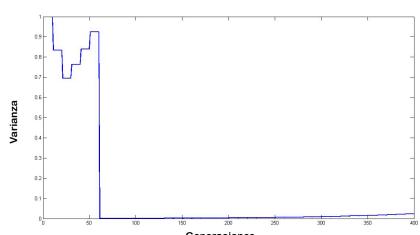
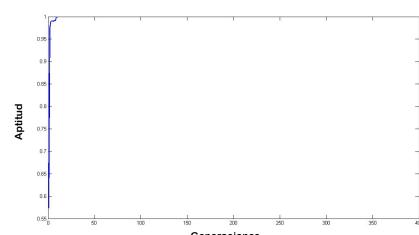


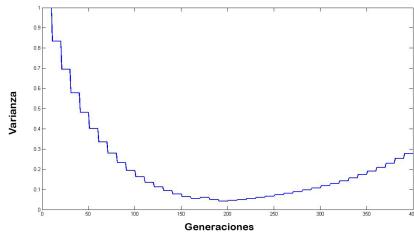
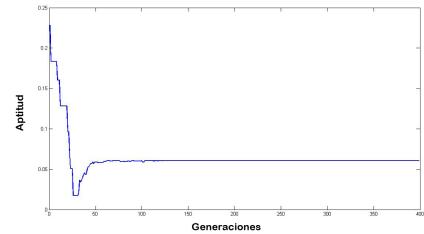
Parámetro σ para la función $g04$



Aptitud del mejor individuo en $g04$



Parámetro σ para la función $g09$ Aptitud del mejor individuo en $g09$ Parámetro σ para la función $g10$ Aptitud del mejor individuo en $g10$ Parámetro σ para la función $g11$ Aptitud del mejor individuo en $g11$ Parámetro σ para la función $g12$ Aptitud del mejor individuo en $g12$

Parámetro σ para la función $g13$ Aptitud del mejor individuo en $g13$

En cada una de las gráficas anteriores se muestra como la varianza disminuye cuando hay cambios en el valor de aptitud del mejor individuo, indicando que se encontró una mejor solución, por otro lado, cuando no hay una mejora en la población, la varianza comienza a aumentar su valor con el objetivo de realizar una mejor exploración del espacio de búsqueda. La forma escalonada del comportamiento de la varianza se debe a que la actualización del parámetro σ se lleva a cabo cada diez generaciones, manteniendo su valor constante entre cada actualización.

En la función $g03$ se observa como después de transcurridas varias generaciones, la varianza comienza a oscilar manteniendo una tendencia a incrementar su valor, indicando que el aumento en la varianza efectivamente ayuda al algoritmo a encontrar una mejor solución.

Como se muestra en la función $g06$ la velocidad con la que aumenta la varianza es menor que la velocidad con la que disminuye, esto se hace con el objetivo de llevar a cabo una exploración cuidadosa del espacio de búsqueda, de manera que la ubicación de las mutaciones generadas se aleje del mejor individuo poco a poco.

La aptitud del mejor individuo en la función $g07$ cae rápidamente, ocasionando que la varianza disminuya su valor. Despues de algunas generaciones donde no se obtiene una mejora en la población, la varianza comienza a aumentar, extendiendo así el rango de búsqueda hasta encontrar una mejor solución, y entonces proceder a disminuir su tamaño hasta converger al óptimo global.

En las funciones $g08$ y $g12$ la población converge de manera rápida hacia el óptimo global, por lo que la varianza, al no detectar una mejora en la población, aumenta su tamaño hasta rebasar el límite superior, ocasionando que esta se reinicie desde un valor pequeño, obligando al algoritmo a llevar a cabo una búsqueda más precisa en las cercanías del mejor individuo.

Capítulo 6

Resultados

En este capítulo se presentan los resultados obtenidos al aplicar el algoritmo EMEDA para optimizar las funciones de benchmark definidas en el apéndice A. Por cada función se realizaron 30 corridas del algoritmo utilizando un máximo de 600,000 evaluaciones de función y una población de 200 individuos. Los parámetros utilizados en cada uno de los problemas fueron los siguientes:

$f(x)$	ϵ_{start}	ϵ_{min}	σ_u	σ_l	α	β
g01	1	1E-5	1	1E-3	10000	0.95
g02	1	1E-5	1	1E-3	10000	0.95
g03	1	1E-5	1	1E-3	10000	0.95
g04	1	1E-5	1	1E-3	10000	0.95
g05	1	1E-5	1	1E-3	10000	0.95
g06	1	1E-5	1	1E-3	10000	0.95
g07	1	1E-5	1	1E-3	10000	0.95
g08	1	1E-5	1	1E-3	10000	0.95
g09	1	1E-5	1	1E-3	10000	0.95
g10	1	1E-5	1	1E-3	30000	0.95
g11	1	1E-5	1	1E-3	10000	0.95
g12	1	1E-5	1	1E-3	10000	0.95
g13	1	1E-5	1	1E-3	10000	0.85

Cuadro 6.1: Parámetros del algoritmo EMEDA

El valor del parámetro σ necesario para llevar a cabo las mutaciones del mejor individuo y poder mantener un control en la diversidad de la población, se actualiza cada 10 generaciones, momento en el cual puede aumentar o disminuir su

valor dependiendo de la convergencia del algoritmo. Los valores de α y β permanecen constantes durante la ejecución del algoritmo, y representan al factor de penalización y al factor de decrecimiento de la zona factible respectivamente. El número de Gaussianas utilizadas en el modelo de probabilidad fue de 2 para todos los casos.

El criterio de paro empleado consiste en agotar el máximo número de generaciones, o bien, identificar si existe una mejora en la población en las últimas 400 generaciones, si no es así, se detiene la búsqueda y se reporta la mejor solución encontrada. A continuación se muestra un análisis de los resultados obtenidos de cada una de las funciones de prueba utilizadas.

6.1. Resultados de la función g01

Óptimo	-15
Mejor	-15
Peor	-15
Media	-15
Mediana	-15
Desviación Estándar	0
Evaluaciones de Función Promedio	72346.67

Cuadro 6.2: Resultados de la función g01

La figura 6.1 muestra la aptitud del mejor individuo durante el transcurso del proceso evolutivo utilizando el algoritmo EMEDA, mientras que la figura 6.2 presenta el comportamiento del parámetro σ , el cual disminuye su tamaño cada vez que existe una mejora en la población.

Durante las primeras generaciones el algoritmo logra entrar en la zona factible, una vez dentro, los métodos implementados para controlar la diversidad de la población ayudan a realizar una exploración más precisa del espacio de búsqueda, y de esta manera encontrar el óptimo global.

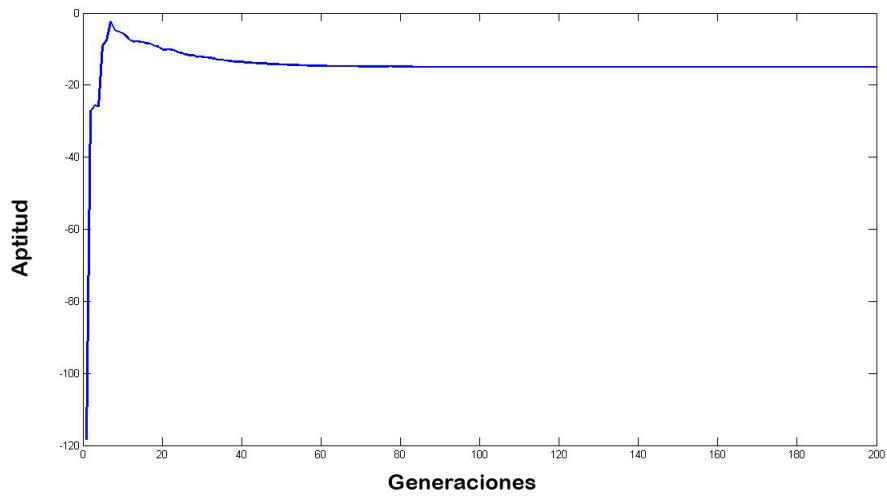


Figura 6.1: Aptitud del mejor individuo para el problema g01

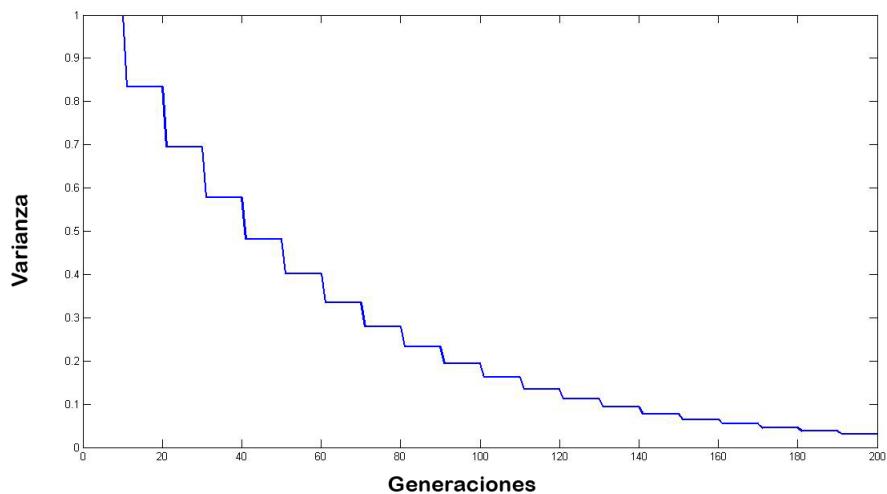


Figura 6.2: Parámetro σ para el problema g01

6.2. Resultados de la función g02

Óptimo	0.803619
Mejor	0.803613
Peor	0.721917
Media	0.7730288
Mediana	0.785468
Desviación Estándar	0.029207827
Evaluaciones de Función Promedio	290560

Cuadro 6.3: Resultados de la función g02

La figura 6.3 muestra la aptitud del mejor individuo de la población durante la ejecución del algoritmo, donde se puede apreciar como la población converge rápidamente hacia el óptimo global, una vez que se encuentra cerca del óptimo, se lleva a cabo una búsqueda más precisa, provocando que la aptitud del mejor individuo ascienda lentamente.

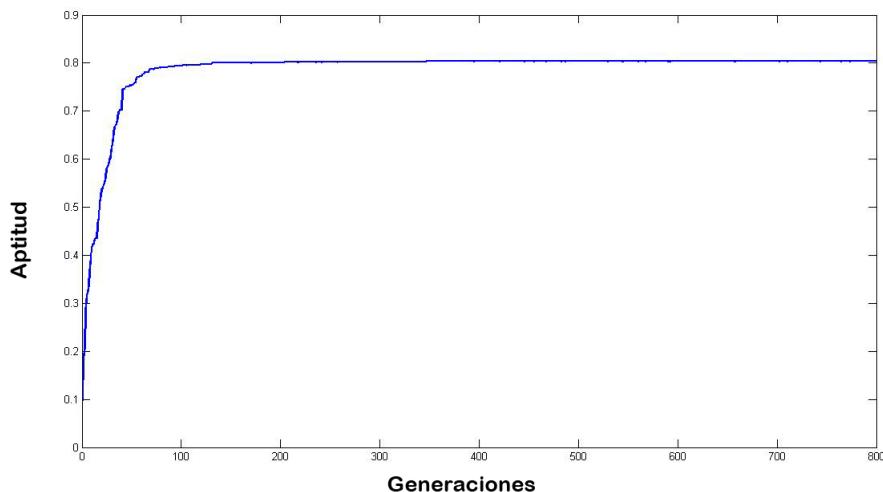


Figura 6.3: Aptitud del mejor individuo para el problema g02

En la figura 6.4 se puede observar como la varianza con la que se llevan a cabo las mutaciones del mejor individuo tiene un valor inicial relativamente grande, lo cual permite realizar una exploración más extensa del espacio de búsqueda. Conforme la población converge al óptimo global, la varianza comienza a disminuir, haciendo más precisa la búsqueda del óptimo; cuando no se presenta una mejora en la población, la varianza aumenta su tamaño con la intención de ampliar el

rango de búsqueda. Esto ocasiona que el parámetro σ oscile suavemente durante las generaciones subsecuentes.

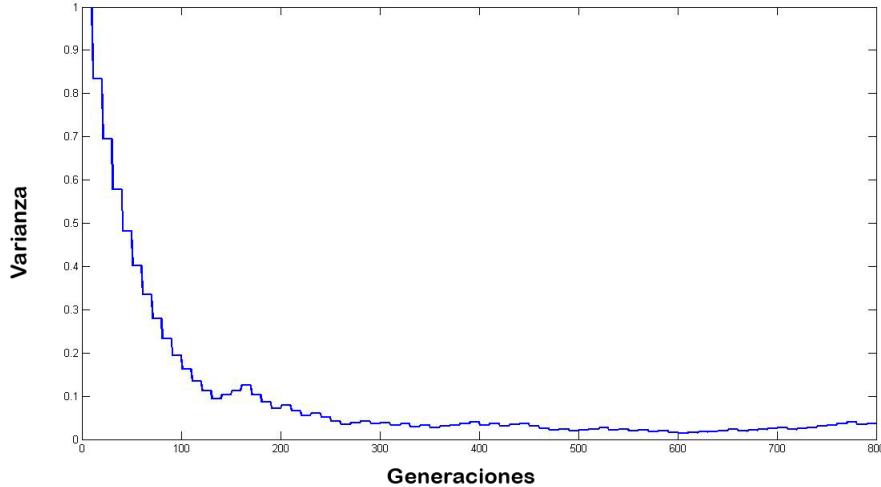


Figura 6.4: Parámetro σ para el problema g02

6.3. Resultados de la función g03

Óptimo	1
Mejor	1
Peor	0.997158
Media	0.9995808
Mediana	0.9998025
Desviación Estándar	0.000651833
Evaluaciones de Función Promedio	520120

Cuadro 6.4: Resultados de la función g03

A continuación se hace un análisis de una corrida del algoritmo EMEDA para resolver el problema g03

En la figura 6.5 se muestra la aptitud del mejor individuo. Al inicio del proceso evolutivo la solución se encuentra fuera de la zona factible, pocas generaciones después la población comienza a converger hacia el óptimo global.

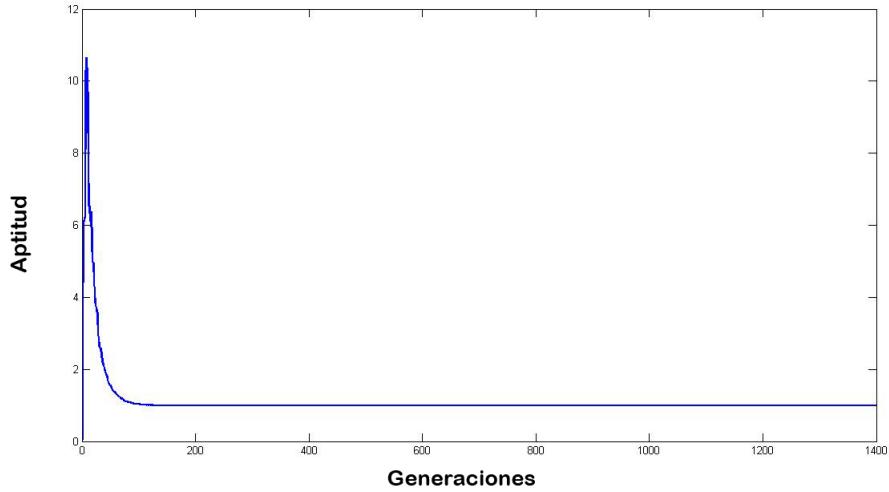


Figura 6.5: Aptitud del mejor individuo para el problema g03

A medida que la población converge hacia la solución, la varianza de las mutaciones del mejor individuo disminuye; una vez que la población se estaciona en un óptimo local, la varianza incrementa su valor, ocasionando que el rango de búsqueda aumente hasta llegar al máximo valor permitido σ_u , lo cual provoca que el valor de σ se reinicie desde un valor pequeño en busca de una nueva solución.

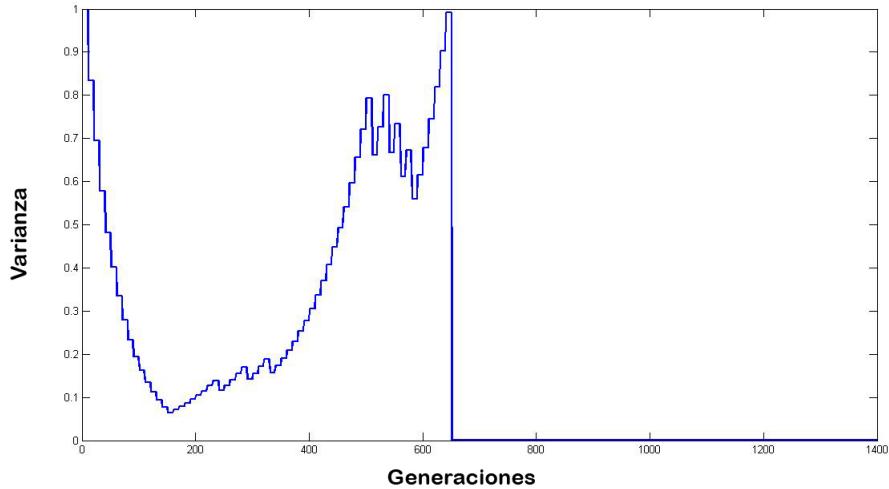


Figura 6.6: Parámetro σ para el problema g03

La zona factible se reduce siempre y cuando el porcentaje de soluciones factibles P_f este por encima del valor P_t especificado. Durante la mayor parte del proceso evolutivo se cumple que $P_f > P_t$, por lo que el tamaño de la zona

factible disminuye gradualmente durante la ejecución del algoritmo, demandando una mayor precisión en la búsqueda del óptimo.

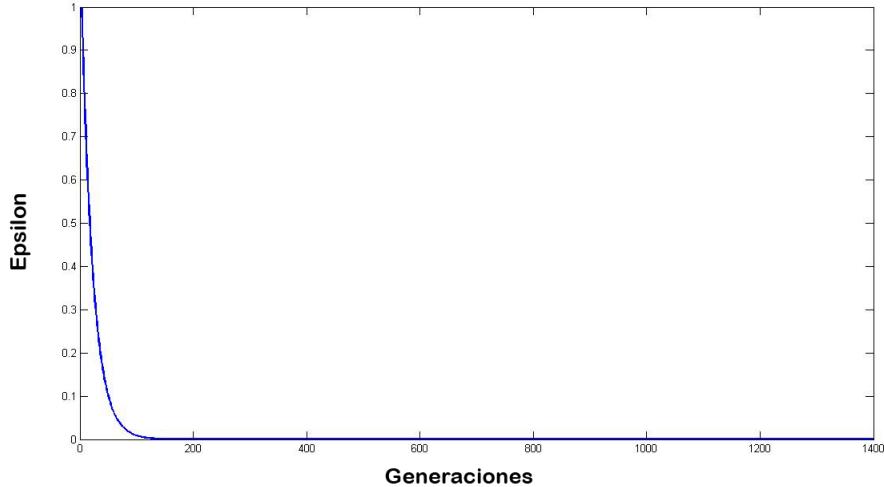


Figura 6.7: Parámetro ϵ para el problema g03

El porcentaje de soluciones factibles está por encima de P_t en las primeras generaciones del algoritmo, una vez que la población tiende a converger al óptimo, este porcentaje comienza a oscilar alrededor de P_t hasta que la varianza se reinicia, ya que a partir de ese momento la búsqueda se hace más precisa y el número de soluciones dentro de la zona factible aumenta.

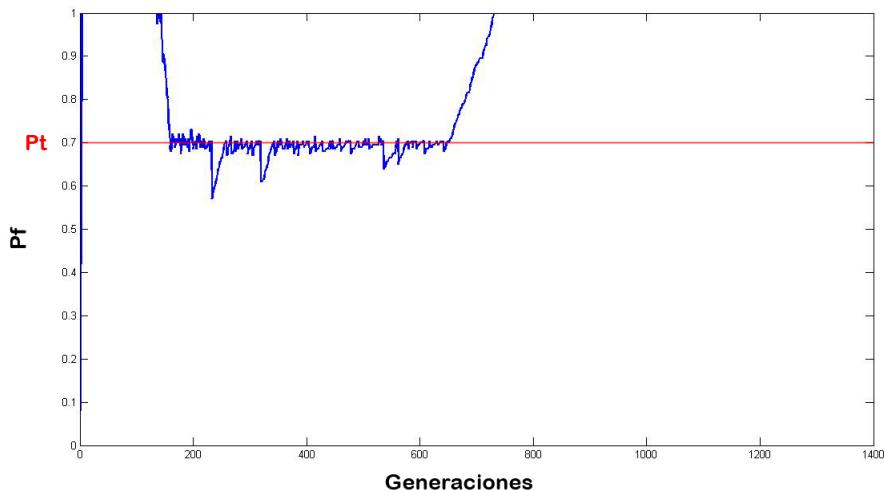


Figura 6.8: Porcentaje de soluciones factibles para el problema g03

6.4. Resultados de la función g04

Óptimo	-30665.539
Mejor	-30665.53867
Peor	-30664.78996
Media	-30665.48916
Mediana	-30665.53704
Desviación Estándar	0.154527293
Evaluaciones de Función Promedio	117053.3333

Cuadro 6.5: Resultados de la función g04

En la mayoría de las pruebas realizadas en el problema g04 la población convergió rápidamente a la solución, en promedio se necesitaron alrededor de 300 generaciones para poder encontrar el mejor resultado. La figura 6.9 muestra la bondad del mejor individuo en una corrida del algoritmo EMEDA.

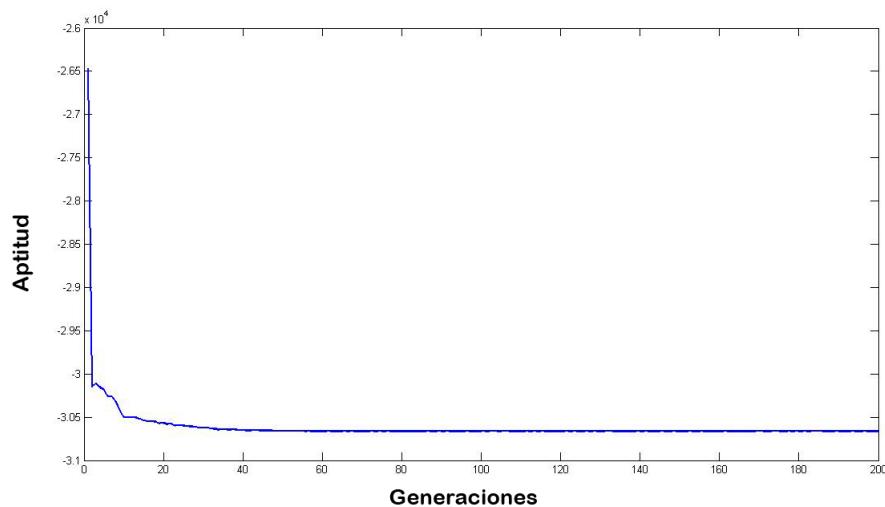
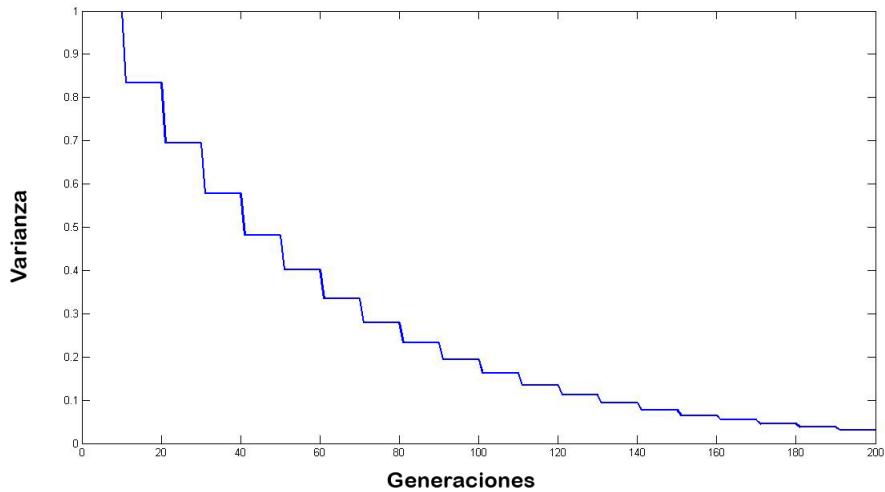


Figura 6.9: Aptitud del mejor individuo para el problema g04

En consecuencia a que la población mejora continuamente conforme avanzan las generaciones, el valor de σ disminuye con el objetivo de realizar mutaciones cada vez más cercanas al mejor individuo y llevar a cabo una mejor exploración.

Figura 6.10: Parámetro σ para el problema g04

6.5. Resultados de la función g05

Óptimo	5126.4981
Mejor	5126.572786
Peor	5446.364251
Media	5179.547757
Mediana	5143.061301
Desviación Estándar	70.81324364
Evaluaciones de Función Promedio	600000

Cuadro 6.6: Resultados de la función g05

Durante las primeras iteraciones del algoritmo se presentan cambios bruscos en la aptitud del mejor individuo de la población, esto debido a que la tolerancia en las restricciones de igualdad es muy alta, permitiendo que malos individuos no sean penalizados. A medida que la solución converge al óptimo, la zona factible se reduce provocando que individuos catalogados como aptos dejen de serlo.

El ajuste del parámetro ϵ provoca que se realice una mejor exploración del espacio de búsqueda, aunque se corre el riesgo de estacionarse en una región no factible.

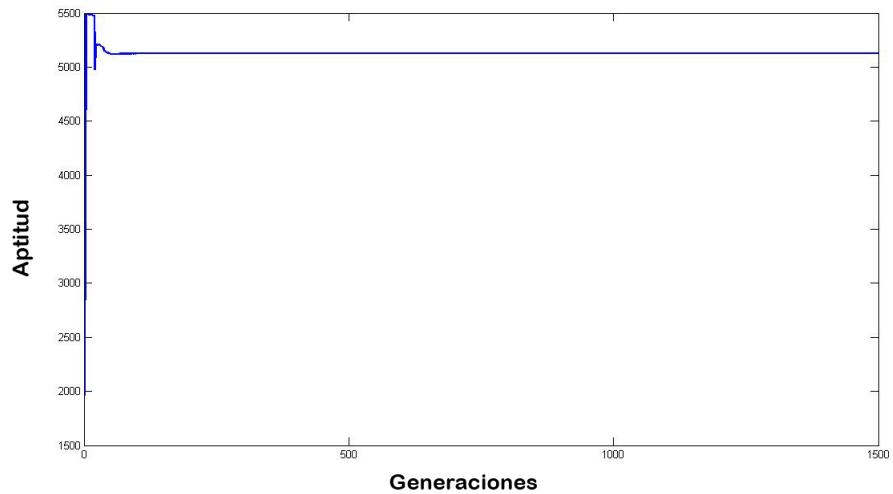


Figura 6.11: Aptitud del mejor individuo para el problema g05

La población mejora continuamente durante el proceso evolutivo debido al ajuste de la zona factible, esto ocasiona que la varianza disminuya hasta llegar al mínimo valor permitido σ_l como se muestra en la figura 6.12.

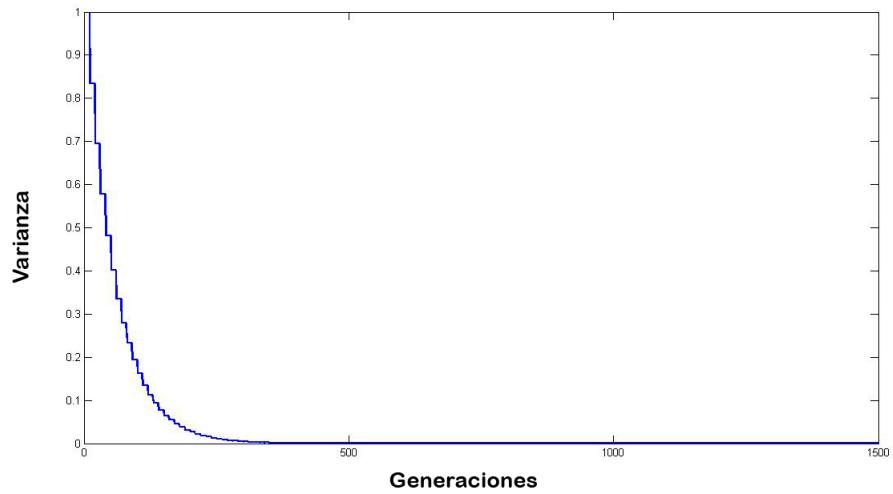
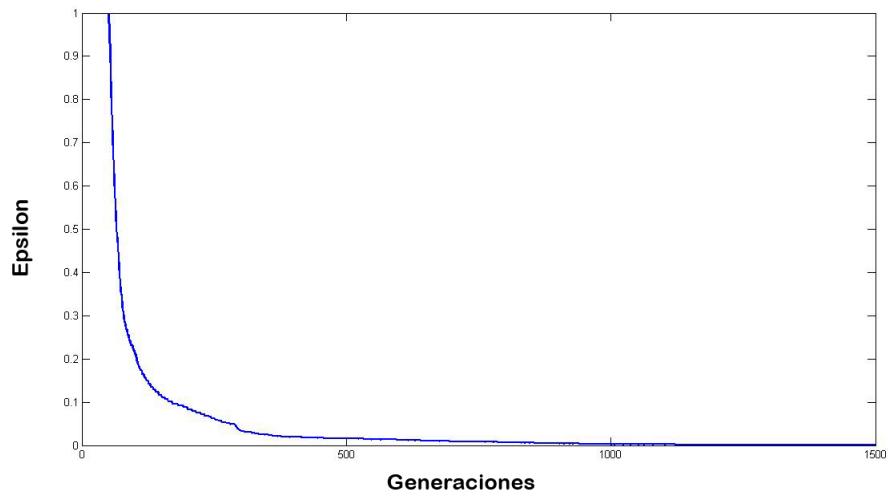


Figura 6.12: Parámetro σ para el problema g05

La zona factible se ajusta continuamente durante la ejecución del algoritmo, debido a que el porcentaje de soluciones factibles se encuentra por encima del valor P_t en varias ocasiones durante el proceso evolutivo, exigiendo una mayor precisión en la búsqueda del óptimo.

Figura 6.13: Parámetro ϵ para el problema g05

El porcentaje de individuos factibles P_f oscila bruscamente debido a que al disminuir el valor de ϵ , individuos catalogados como aptos en generaciones anteriores son penalizados, disminuyendo en gran medida el número de soluciones factibles.

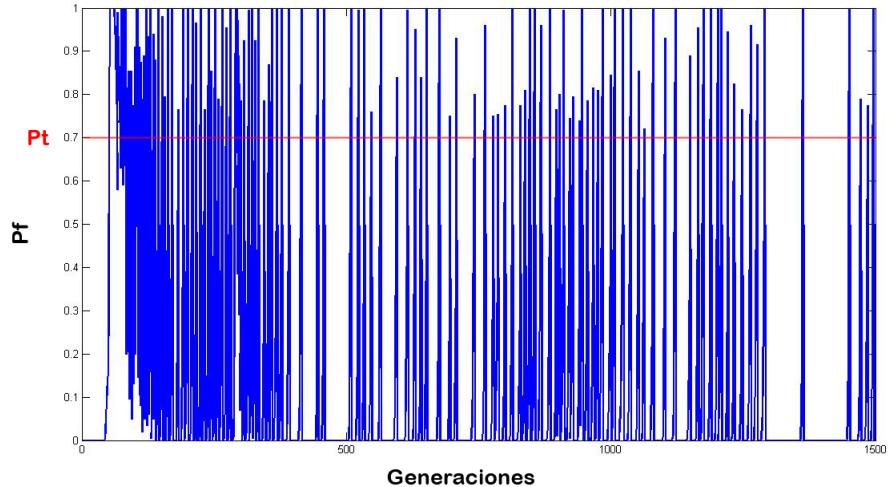


Figura 6.14: Porcentaje de soluciones factibles para el problema g05

6.6. Resultados de la función g06

Óptimo	-6961.81388
Mejor	-6961.813876
Peor	-6961.6444
Media	-6961.798601
Mediana	-6961.813344
Desviación Estándar	0.037521559
Evaluaciones de Función Promedio	175173.3333

Cuadro 6.7: Resultados de la función g06

La figura 6.15 muestra el comportamiento del valor de aptitud del mejor individuo durante la ejecución del algoritmo EMEDA, donde se aprecia como la población converge en un número pequeño de iteraciones al óptimo global. Por otro lado, la figura 6.16 muestra como el parámetro σ reduce su valor conforme la población se acerca a la solución.

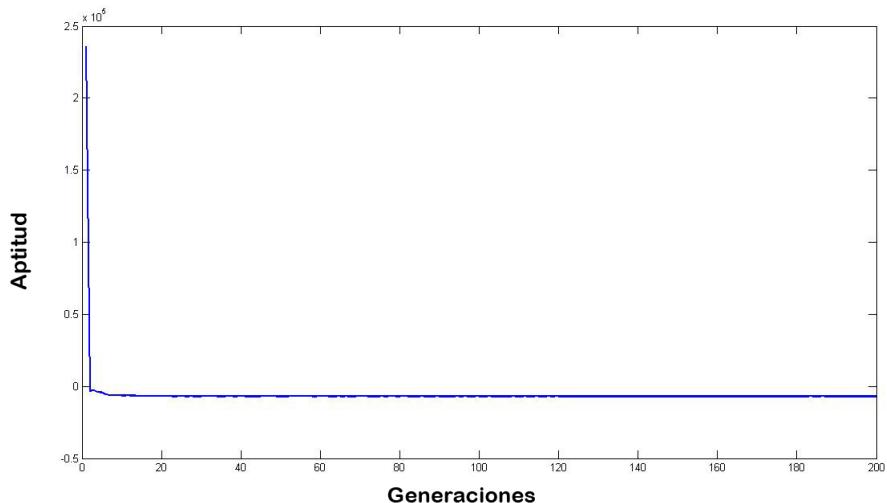
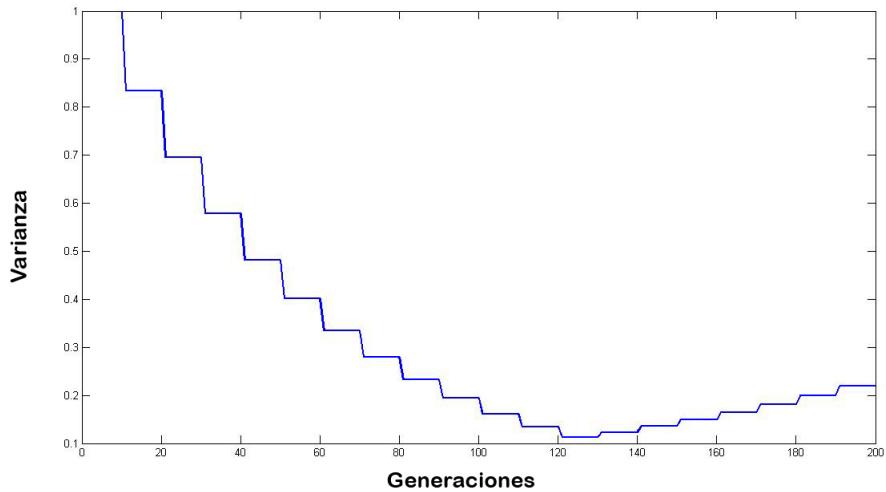


Figura 6.15: Aptitud del mejor individuo para el problema g06

Figura 6.16: Parámetro σ para el problema g06

6.7. Resultados de la función g07

Óptimo	24.3062091
Mejor	24.334821
Peor	25.607953
Media	24.57801667
Mediana	24.4436305
Desviación Estándar	0.311701117
Evaluaciones de Función Promedio	286706.6667

Cuadro 6.8: Resultados de la función g07

La aptitud del mejor individuo cambia constantemente durante la ejecución del algoritmo. En las primeras generaciones la población converge rápidamente hacia el óptimo global, para después avanzar lentamente durante lo que resta del proceso evolutivo.

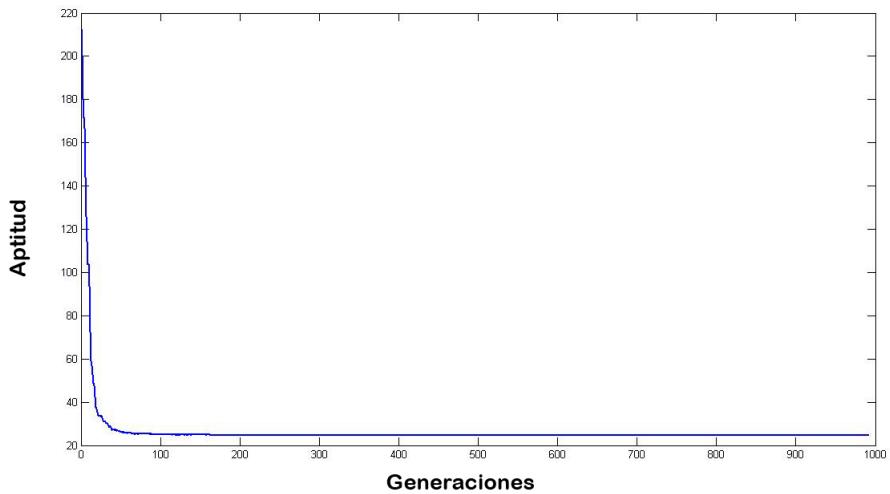


Figura 6.17: Aptitud del mejor individuo para el problema g07

Dado que la aptitud del mejor individuo cambia continuamente, el factor σ necesario para generar las mutaciones del mejor individuo tiende a disminuir conforme transcurren las generaciones, provocando una exploración más precisa del espacio de búsqueda.

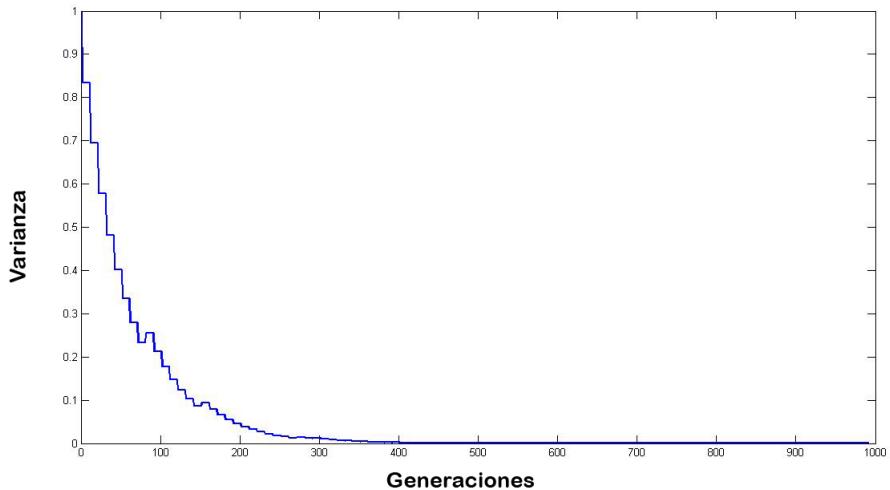


Figura 6.18: Parámetro σ para el problema g07

6.8. Resultados de la función g08

Óptimo	0.095825
Mejor	0.095825
Peor	0.095825
Media	0.095825
Mediana	0.095825
Desviación Estándar	4.23451E-17
Evaluaciones de Función Promedio	4080

Cuadro 6.9: Resultados de la función g08

Una sola corrida del algoritmo EMEDA toma alrededor de 10 generaciones encontrar la solución al problema g08. La figura 6.19 muestra el valor de aptitud del mejor individuo al ejecutar el algoritmo.

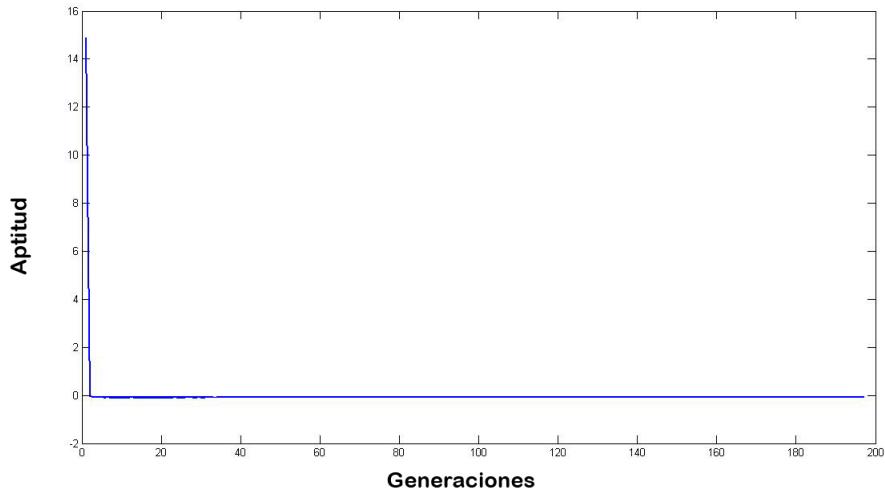
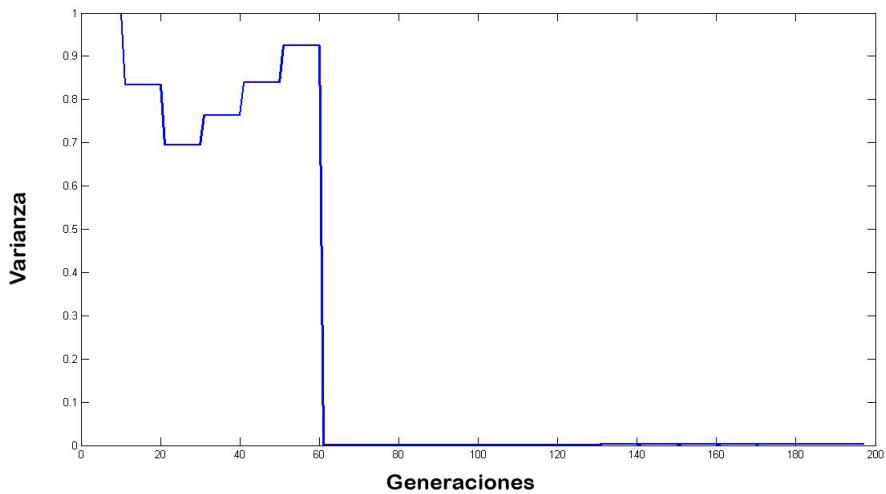


Figura 6.19: Aptitud del mejor individuo para el problema g08

Dado que la población converge al óptimo global en pocas iteraciones la varianza disminuye, después, al no detectar una mejora en la población, la varianza se incrementa hasta rebasar el máximo permitido, ocasionando que ésta se reinicie desde un valor pequeño.

Figura 6.20: Parámetro σ para el problema g08

6.9. Resultados de la función g09

Óptimo	680.6300573
Mejor	680.631954
Peor	680.794962
Media	680.6649346
Mediana	680.6582465
Desviación Estándar	0.031698573
Evaluaciones de Función Promedio	300746.6667

Cuadro 6.10: Resultados de la función g09

La figura 6.21 muestra el valor de aptitud del mejor individuo al ejecutar el algoritmo EMEDA.

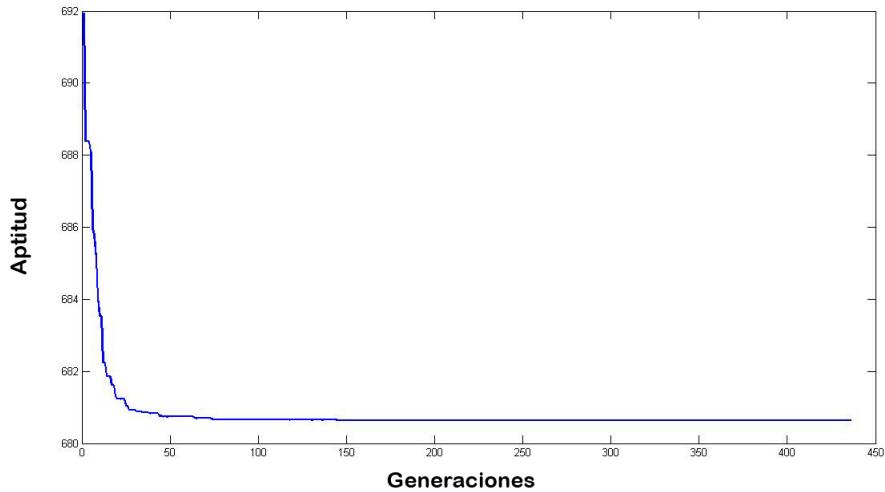
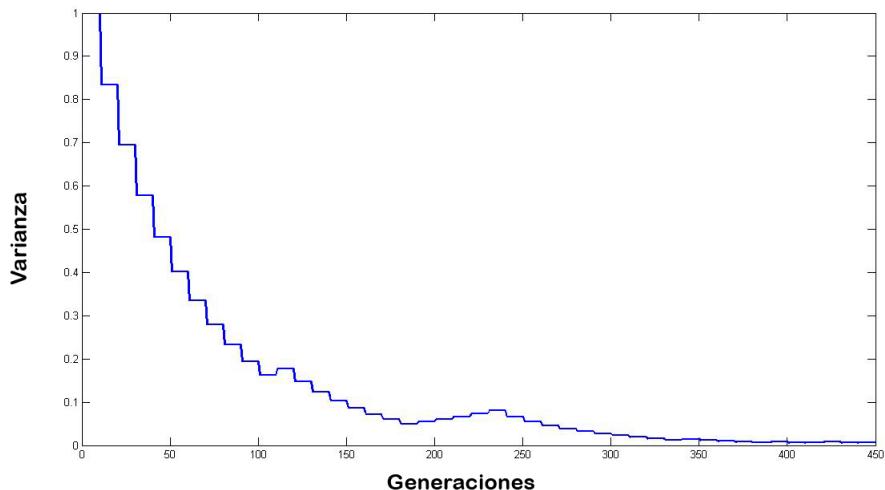


Figura 6.21: Aptitud del mejor individuo para el problema g09

El parámetro σ disminuye a medida que la población converge al óptimo global, en caso de que no se presente una mejora en la solución, el valor de la varianza aumenta para extender el rango de búsqueda del algoritmo. En la siguiente figura se presenta el comportamiento del parámetro σ en una corrida del algoritmo propuesto.

Figura 6.22: Parámetro σ para el problema g09

6.10. Resultados de la función g10

Para el problema g10 se consideró como restricciones de igualdad a las tres primeras restricciones, a partir del conocimiento de que en el óptimo dichas restricciones son activas, esto con la finalidad de ayudar al algoritmo a realizar una mejor exploración del espacio de búsqueda, ya que los resultados obtenidos quedaban alejados de la solución.

Óptimo	7049.3307
Mejor	7068.698869
Peor	8380.459924
Media	7575.471101
Mediana	7400.275838
Desviación Estándar	464.6210243
Evaluaciones de Función Promedio	600000

Cuadro 6.11: Resultados de la función g10

Durante las primeras generaciones, la aptitud de los individuos oscila bruscamente debido al ajuste del parámetro ϵ que ocasiona que la zona factible reduzca su tamaño, provocando que el número de individuos aptos se vea disminuido.

El ajuste de la tolerancia en las restricciones de igualdad, ayuda al algoritmo a llevar a cabo una mejor exploración, aunque lo hace susceptible de estacionarse en un óptimo fuera de la región factible.

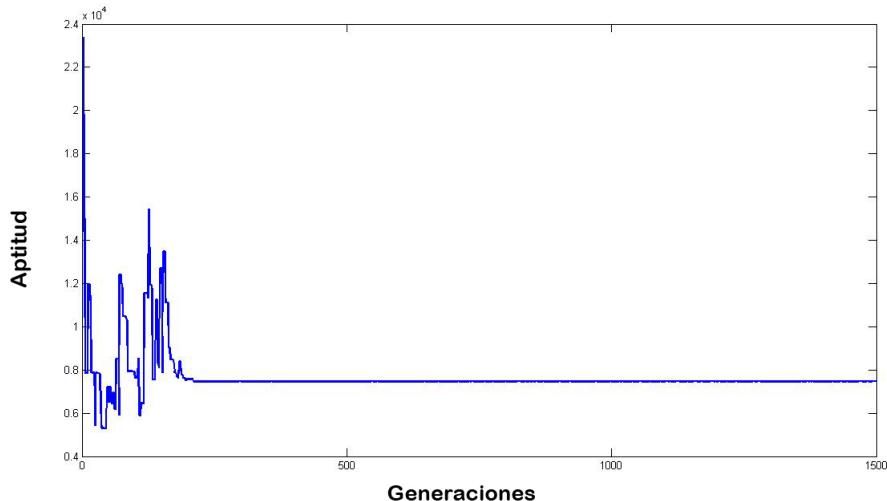


Figura 6.23: Aptitud del mejor individuo para el problema g10

La oscilación en el valor de aptitud de la población indica que varias soluciones se encuentran fuera de la región factible, lo cual puede beneficiar al algoritmo, pues se lleva a cabo una mejor exploración del espacio de búsqueda, por otro lado, si después de varias generaciones no se encuentra un óptimo dentro de la zona factible, la población comienza a estacionarse fuera de dicha zona.

La aptitud de la población permanece en constante movimiento durante la ejecución del algoritmo debido al ajuste de la región factible, esto ocasiona que la varianza disminuya hasta alcanzar el mínimo permitido, dando origen a mutaciones cada vez más próximas al mejor individuo.

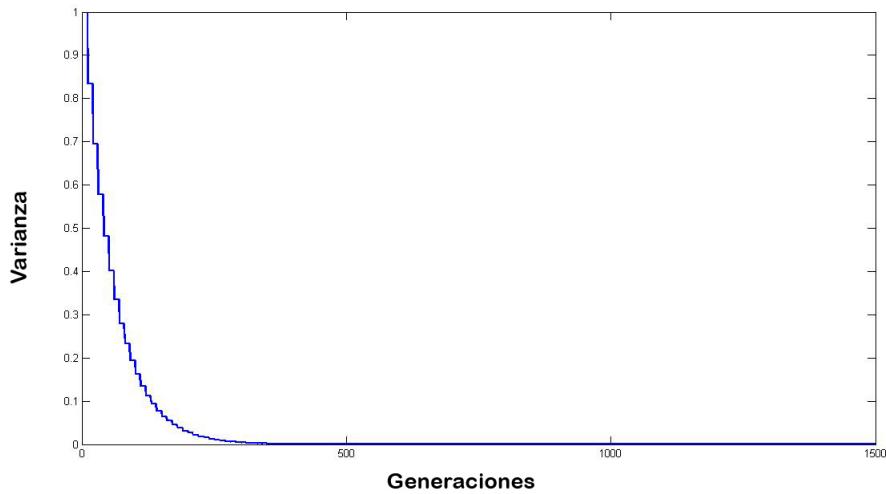


Figura 6.24: Parámetro σ para el problema g10

Aún cuando la aptitud del mejor individuo cambia de manera constante, la población tiende a estacionarse en una zona no factible; esto causa que las mutaciones realicen una exploración cada vez más precisa alrededor de dicha región, en vez de expandir el rango de búsqueda para encontrar una mejor solución.

Conforme la población converge a un óptimo fuera de la zona factible, el porcentaje de soluciones factibles disminuye, lo cual ocasiona que la tolerancia para catalogar a un individuo como apto permanezca constante, permitiendo una mayor flexibilidad al momento de la penalización. Lo anterior provoca que el rango de exploración del espacio de búsqueda se centre en una zona no factible, haciendo muy difícil para el algoritmo encontrar el óptimo global.

El parámetro ϵ desciende en forma escalonada debido a que el porcentaje de soluciones factibles P_f permanece por debajo del valor P_t durante cierto número

de generaciones, una vez que sobrepasan el valor de P_t , la zona factible reduce su valor, ocasionando que el valor de P_f vuelva a disminuir.

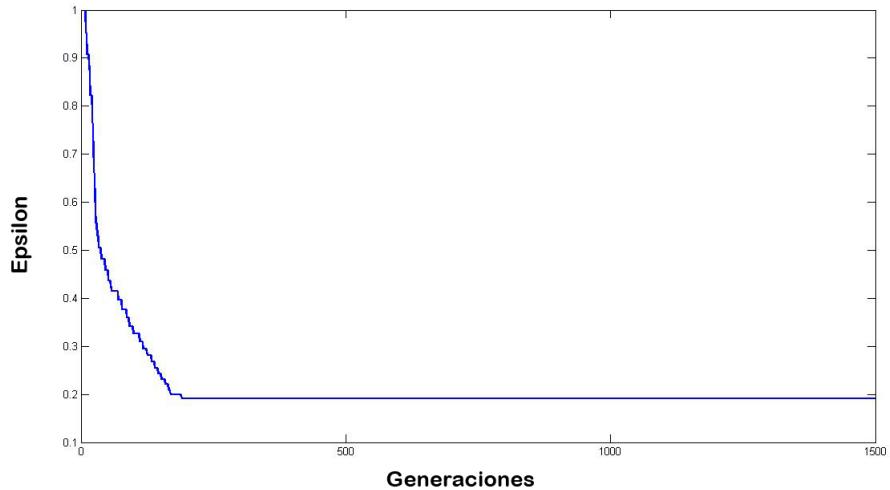


Figura 6.25: Parámetro ϵ para el problema g10

El ajuste de la zona factible ayuda a realizar una mejor exploración del espacio de búsqueda, permitiendo que ciertos individuos no sean penalizados aún cuando no cumplan con todas las restricciones, pero esto a su vez puede llegar a influir negativamente en el funcionamiento del algoritmo, ya que puede ocasionar que la población camine en dirección a una zona no factible, reduciendo en gran medida las posibilidades de encontrar el óptimo global.

El porcentaje de individuos factibles oscila fuertemente durante las primeras generaciones, debido a que la mayoría de los individuos aptos son penalizados una vez que se reduce la zona factible. Cuando el algoritmo comienza a converger al óptimo, el valor de P_f en varias ocasiones se mantiene por debajo del valor P_t especificado, esto a causa de que los individuos de la población permanecen ligeramente penalizados, por lo que resulta difícil para el algoritmo encontrar una solución dentro de la región factible.

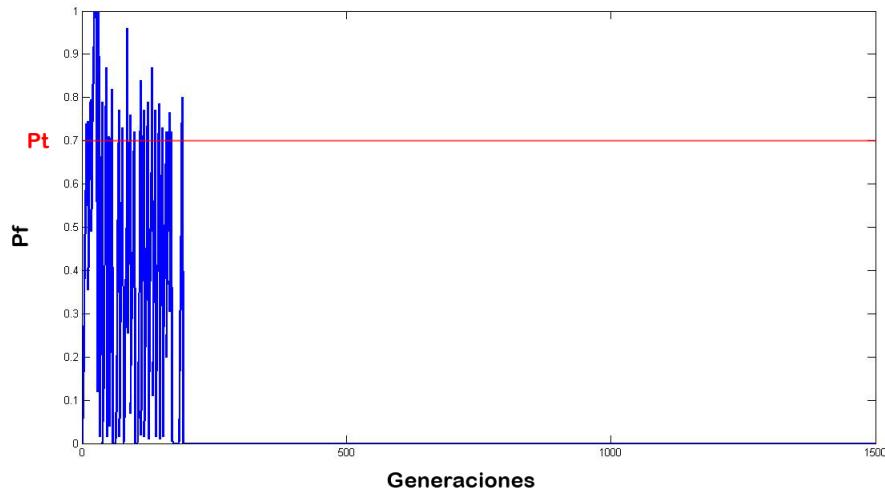


Figura 6.26: Porcentaje de soluciones factibles para el problema g11

6.11. Resultados de la función g11

Óptimo	0.75
Mejor	0.75
Peor	0.751089
Media	0.750184033
Mediana	0.7500715
Desviación Estándar	0.000272217
Evaluaciones de Función Promedio	106440

Cuadro 6.12: Resultados de la función g11

En la función g11, la aptitud de la población se dirige hacia el óptimo global conforme se reduce la zona factible, exigiendo poco a poco una mayor precisión al llevar a cabo la exploración del espacio de búsqueda. La figura 6.27 muestra la aptitud del mejor individuo durante el proceso evolutivo al ejecutar el algoritmo EMEDA.

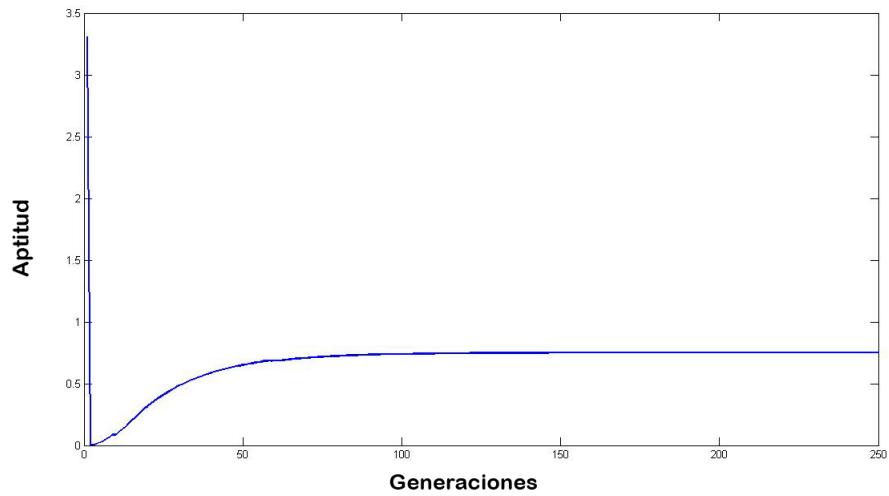


Figura 6.27: Aptitud del mejor individuo para el problema g11

La figura 6.28 muestra como el parámetro σ desciende durante el proceso evolutivo, indicando que la población mejora continuamente conforme transcurren las generaciones, provocando que las mutaciones se efectúen cada vez más cerca del mejor individuo.

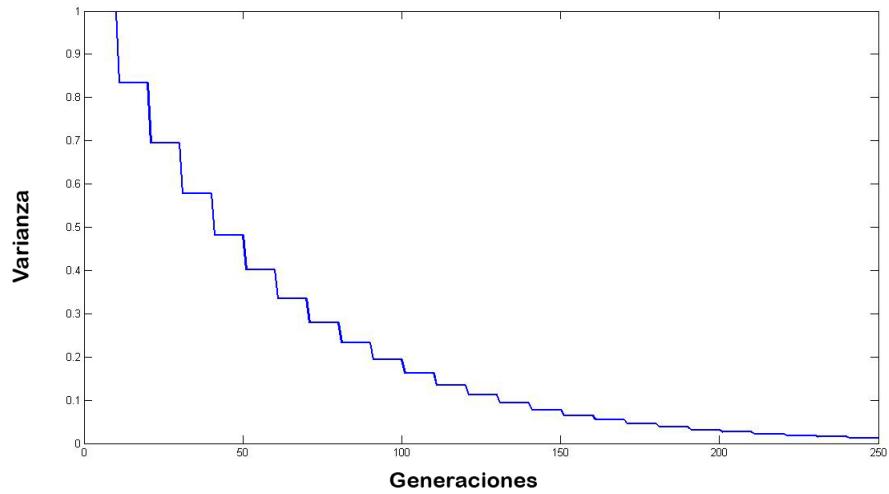
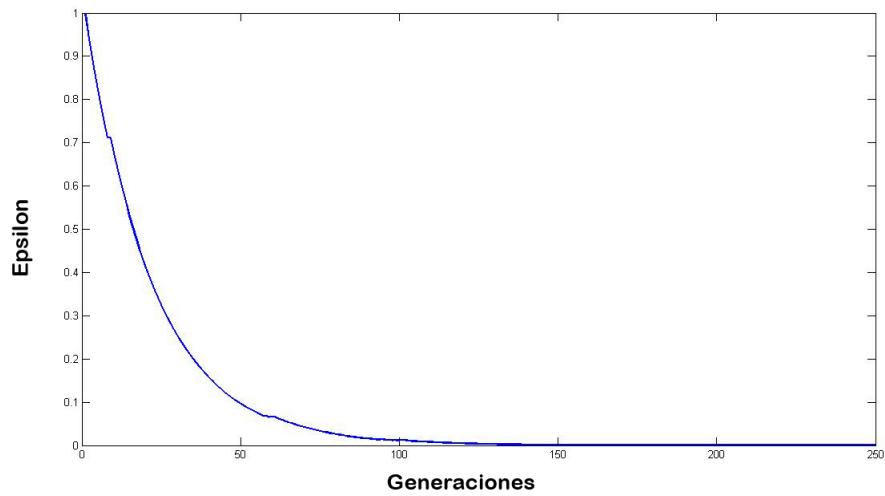


Figura 6.28: Parámetro σ para el problema g11

El valor de ϵ disminuye gradualmente durante la ejecución del algoritmo, exigiendo una mayor precisión en la búsqueda del óptimo.

Figura 6.29: Parámetro ϵ para el problema g11

El porcentaje de soluciones factibles se encuentra por encima de P_t la mayor parte del tiempo, sólo en ciertos lapsos este valor decae, pero inmediatamente vuelve a subir, como se muestra en la figura 6.30

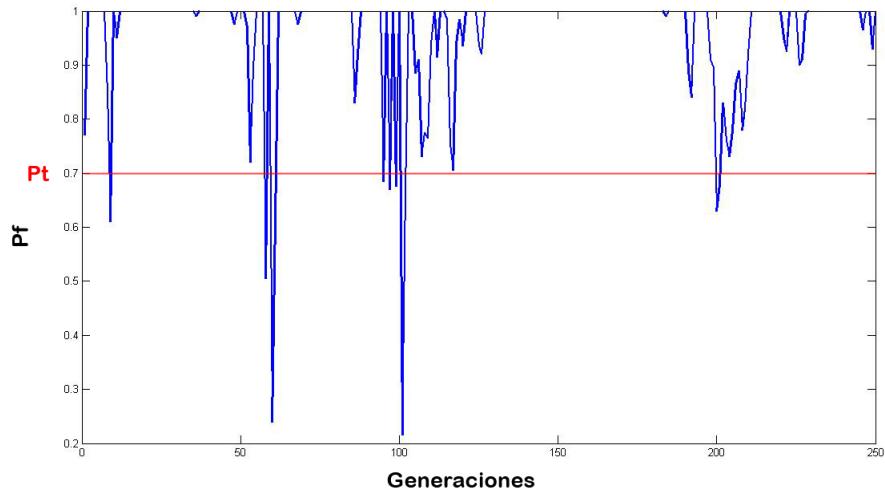


Figura 6.30: Porcentaje de soluciones factibles para el problema g11

6.12. Resultados de la función g12

Óptimo	1
Mejor	1
Peor	1
Media	1
Mediana	1
Desviación Estándar	0
Evaluaciones de Función Promedio	6653.333333

Cuadro 6.13: Resultados de la función g12

La función g12 es relativamente fácil de resolver, pues solo le lleva al algoritmo alrededor de 16 iteraciones encontrar la solución. Rápidamente la población se acerca al óptimo global, pero no es, sino hasta cuando disminuye la varianza para crear las mutaciones del mejor individuo, que el algoritmo encuentra el máximo global.

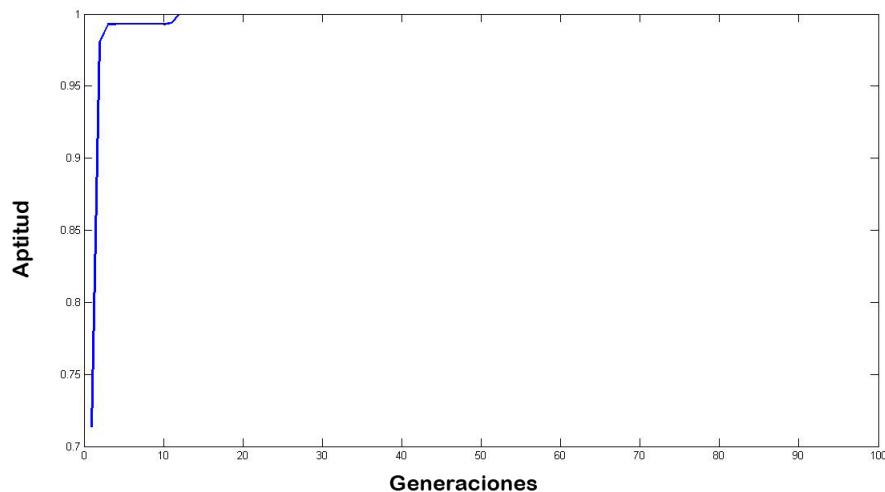
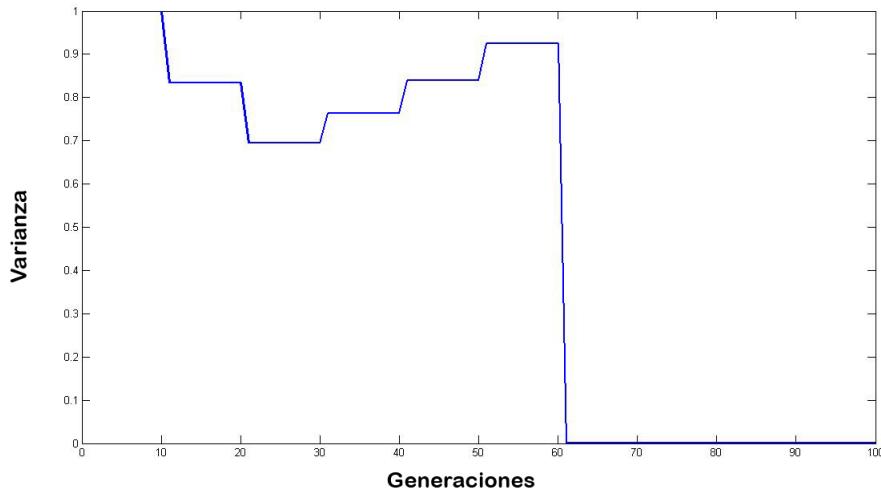


Figura 6.31: Aptitud del mejor individuo para el problema g12

El parámetro σ solo se reduce un par de ocasiones antes de que vuelva a ascender al no detectar una mejora en la población, ya que la población converge al óptimo global aceleradamente. Una vez que se rebasa el máximo valor permitido, la varianza se reinicia tomando un valor muy pequeño, esto con la intención de comenzar una nueva búsqueda en las cercanías del mejor individuo.

Figura 6.32: Parámetro σ para el problema g12

6.13. Resultados de la función g13

Junto con la función g10, la función g13 es el problema que mayor complejidad presenta al algoritmo EMEDA. El hecho de hacer del parámetro ϵ un parámetro adaptable, ayuda en gran manera al algoritmo a acercarse al óptimo global, aunque esto no sucede en todos los casos.

Óptimo	0.0539488
Mejor	0.055475
Peor	0.650706
Media	0.241015467
Mediana	0.153651
Desviación Estándar	0.186561798
Evaluaciones de Función Promedio	67533.33333

Cuadro 6.14: Resultados de la función g13

Durante las primeras generaciones se define el curso que tomará la población en lo que resta del proceso evolutivo, si el algoritmo tiende a converger al óptimo desde el inicio, existe una gran probabilidad de obtener una buena solución, en caso contrario, es posible que se estacione en un óptimo local.

Al momento de que el algoritmo comienza a converger a un óptimo, la aptitud de la población cambia lentamente conforme transcurren las generaciones, centrando la búsqueda alrededor del mejor individuo.

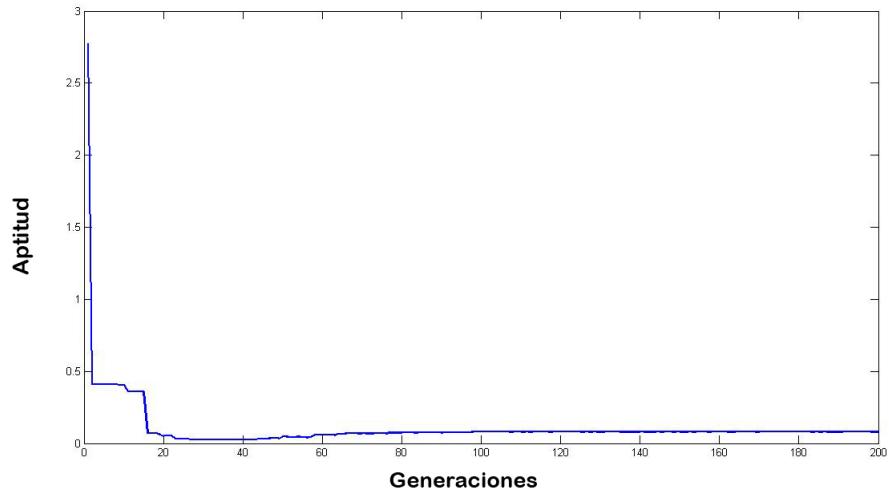


Figura 6.33: Aptitud del mejor individuo para el problema g13

El valor de la varianza disminuye en las primeras iteraciones, que es el momento en que el algoritmo se acerca al óptimo, reduciendo de esta manera el rango de búsqueda; durante las generaciones subsecuentes, la varianza comienza a aumentar al no detectar una mejora en la población, ocasionando que las mutaciones se alejen cada vez más del mejor individuo.

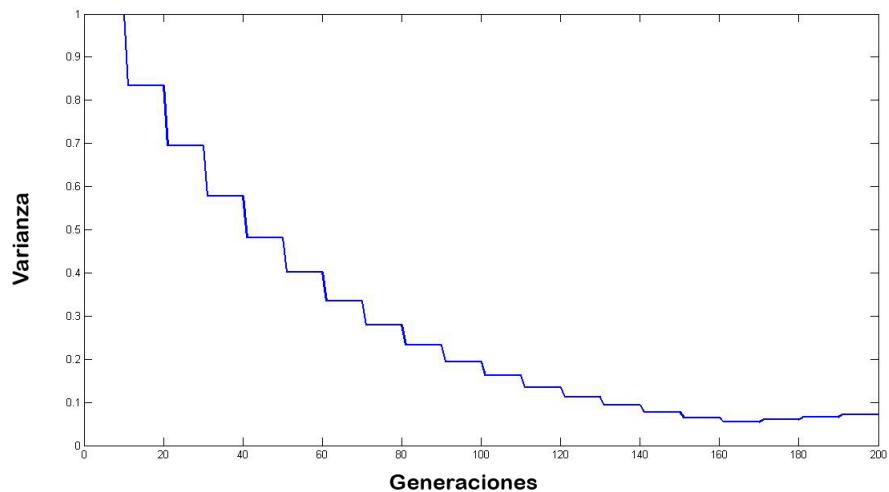


Figura 6.34: Parámetro σ para el problema g13

El tamaño de la zona factible permanece sin cambio al inicio del proceso evolutivo, debido a que el porcentaje de soluciones factibles P_f es muy bajo. Una vez que el valor de ϵ comienza a disminuir, lo hace de forma escalonada, indicando

que el valor de P_f cambia frecuentemente. En las últimas generaciones la región factible disminuye gradualmente, demandando mayor exactitud en las soluciones.

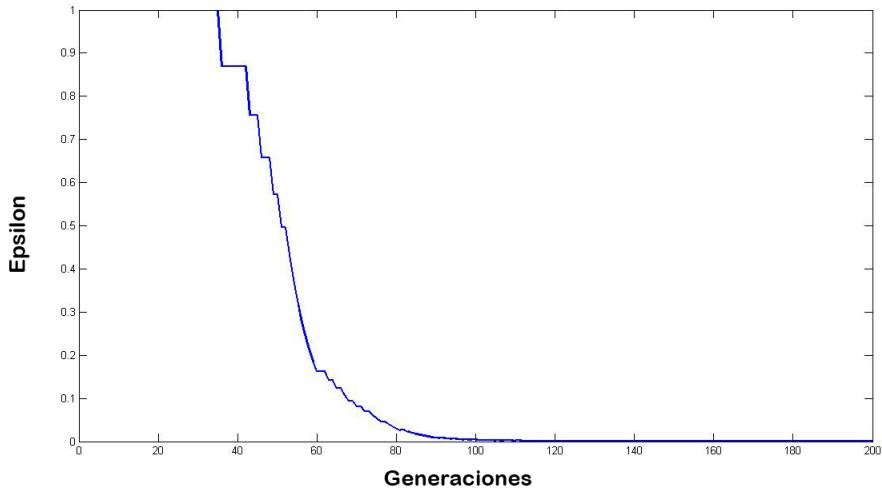


Figura 6.35: Parámetro ϵ para el problema g13

Al ejecutar el algoritmo, el porcentaje de soluciones factibles se mantiene por debajo de P_t durante varias generaciones, provocando que la zona factible permanezca sin cambio alguno. Una vez que el parámetro ϵ disminuye, el valor de P_f comienza a oscilar bruscamente alrededor de P_t , hasta que la población converge al óptimo y la zona factible se reduce de tal manera que el número de individuos no penalizados aumenta.

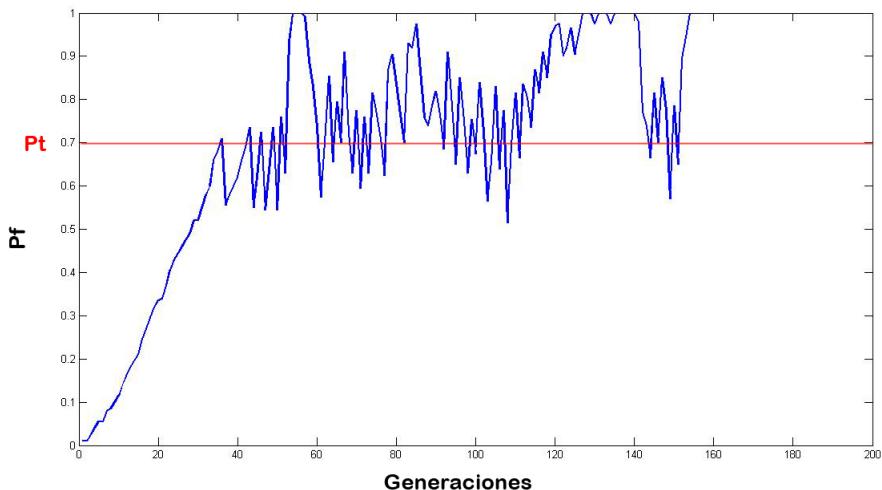


Figura 6.36: Porcentaje de soluciones factibles para el problema g13

6.14. EMEDA vs. Stochastic Ranking

En el cuadro 6.15 se muestran los resultados obtenidos para las 13 funciones de benchmark utilizando el algoritmo EMEDA y la técnica denominada “*Stochastic Ranking*” (SR), ésta última propuesta por Runarsson y Yao (2004).

$f(x)$	Óptimo	Stochastic Ranking			EMEDA		
		Mejor	Media	Mediana	Mejor	Media	Mediana
g_{01}	-15	-15	-15	-15	-15	-15	-15
g_{02}	0.803619	0.803619	0.782715	0.793082	0.803613	0.7730288	0.785468
g_{03}	1	1.001	1.001	1.001	1	0.999	0.999
g_{04}	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.489	-30665.537
g_{05}	5126.498	5126.497	5126.497	5126.497	5126.573	5179.548	5143.061
g_{06}	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.799	-6961.813
g_{07}	24.306	24.306	24.306	24.306	24.335	24.578	24.444
g_{08}	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g_{09}	680.630	680.630	680.630	680.630	680.632	680.665	680.658
g_{10}	7049.330	7049.248	7049.250	7049.248	7068.699	7575.471	7400.276
g_{11}	0.750	0.750	0.750	0.750	0.750	0.750	0.750
g_{12}	1	1	1	1	1	1	1
g_{13}	0.053948	0.53942	0.066770	0.053942	0.055475	0.241015	0.153651

Cuadro 6.15: Stochastic Ranking vs. EMEDA

Los resultados obtenidos mediante el algoritmo EMEDA para las funciones g_{01} , g_{03} , g_{04} , g_{06} , g_{07} , g_{08} , g_{09} , g_{11} , y g_{12} se aproximan en gran manera a los reportados por Runarsson y Yao; en todos ellos la solución resultó muy cerca del óptimo global.

En la función g_{02} , los mejores resultados obtenidos por ambos algoritmos son muy similares y se encuentran cerca del óptimo, sin embargo los valores para la media y la mediana se encuentran ligeramente alejados de la solución en ambos casos, pues mientras el SR reporta una media y una mediana de 0.782715 y 0.793082 respectivamente, el EMEDA presenta valores de 0.7730288 y 0.785468 para dichas medidas.

Para la función g_{05} , la técnica “*Stochastic Ranking*”, obtiene mejores resultados que el algoritmo propuesto, ya que presenta una media de 5128.881 y una mediana de 5127.372, lo cual está muy cercano al óptimo. Por otro lado, el algoritmo EMEDA muestra una media de 5179.548 y una mediana de 5143.061, presentando una mayor varianza en los resultados, aún cuando la mejor solución de ambos métodos se aproxima al óptimo global.

Las funciones g_{10} y g_{13} son difíciles de resolver para el algoritmo EMEDA debido a las restricciones de igualdad que presentan, de tal forma que en varias ocasiones la solución no logra entrar a la zona factible. Los resultados obtenidos por el SR en cuanto al mejor individuo, la media y la mediana son mejores a los

encontrados por el algoritmo propuesto para ambas funciones, aún así el mejor resultado obtenido por los dos algoritmos para la función g13 se acerca en gran medida a la solución .

Capítulo 7

Conclusiones

Durante este trabajo de investigación se estudiaron diversas técnicas de la computación evolutiva como los son los algoritmos genéticos, los cuales funcionan en base a la teoría de la evolución, haciendo uso de operadores de selección, crusa y mutación. Aunque estos algoritmos han demostrado ser una herramienta poderosa para la resolución de problemas de optimización, existe un cierto tipo de funciones llamadas *funciones deceptivas*, diseñadas para engañar al algoritmo genético, por lo que resultan difíciles de resolver. Esto dio inicio a la necesidad de desarrollar nuevas técnicas capaces de lidiar con este tipo de problemas, por lo que se crearon los algoritmos de estimación de distribución (EDAs).

Los EDAs son algoritmos en los que los operadores de reproducción y mutación son reemplazados por modelos de probabilidad, en donde la interacción entre variables debe ser tomada en consideración. Para obtener una estimación de distribución que encamine al algoritmo a la consecución del óptimo global, se toma una muestra de los mejores individuos de la población actual, los cuales son introducidos al modelo de probabilidad empleado, resultando una nueva población con individuos más aptos que en la generación anterior.

A través de los años se han propuesto diversos algoritmos que manejan de manera distinta la interacción entre las variables, existen desde modelos básicos como el UMDA y el PBIL que no presentan dependencia alguna entre variables, hasta algoritmos como el MIMIC y el BMDA que presentan dependencias bivariadas, proporcionando una mejor solución que en el caso univariado. También existen algoritmos que consideran dependencias múltiples entre las variables, esto con la intención de realizar una mejor estimación de la distribución de la población. Es importante mencionar que entre más interacciones existan entre las variables, mayor será el costo computacional del algoritmo.

La resolución de problemas de optimización con restricciones mediante algoritmos de estimación de distribución es un área que no se ha explorado a fondo, razón por la cual se llevo a cabo un estudio de las diversas estrategias que existen para manipular restricciones en espacios de búsqueda multidimensionales. La idea principal de estos métodos consiste en penalizar a aquellos individuos fuera de la región factible de tal manera que la probabilidad de que influyan en la próxima generación se vea disminuida.

El principal problema que se presenta al momento de penalizar a los individuos de una población consiste en elegir los factores de penalización adecuados, de manera que la exploración del espacio de búsqueda dirija a la solución a la zona factible. Al momento de la implementación, no es recomendable descartar a todos aquellos individuos fuera de la zona factible, ni tampoco es bueno estimar la distribución sólo en base a ellos; en muchas ocasiones para encontrar el óptimo global, lo ideal consiste en tener un balance entre soluciones factibles y no factibles.

En el capítulo 5 se presentó el algoritmo EMEDA (Expectation Maximization EDA), el cual tiene como objetivo resolver problemas de optimización con restricciones utilizando un modelo de probabilidad basado en la suma de distribuciones normales. Dicho algoritmo se compone de cuatro partes fundamentales:

1. Algoritmo EM
2. Una técnica de evaluación
3. Un método de selección
4. Un método de control de diversidad

El algoritmo EM, es un proceso iterativo que se encarga de estimar a partir de una serie de puntos, una función de probabilidad basada en la suma de distribuciones normales capaz de generar dichos puntos. Lo anterior se realiza con la intención de crear mejores individuos que puedan ser parte de la siguiente generación.

Cada método de penalización tiene sus ventajas y sus desventajas, para el caso del algoritmo propuesto, se utilizó una penalización estática, es decir, los factores de penalización permanecen sin cambio durante el proceso evolutivo, esto ayuda a que los individuos fuera de la región factible sean fuertemente penalizados, e influyan en menor grado en las futuras generaciones.

Para mantener un balance entre soluciones factibles y no factibles, al momento de trabajar con restricciones de igualdad, se implemento un método que permite modificar el tamaño de la zona factible dependiendo del porcentaje de individuos dentro de dicha zona. Esto ocasiona que se lleve a cabo una mejor exploración del espacio de búsqueda, y orientar a la población hacia el óptimo global.

Para evitar el problema de la convergencia prematura, se realizan una serie de mutaciones del mejor individuo mediante una distribución normal, cuya varianza se ajusta durante la ejecución del algoritmo, según la convergencia de la población.

Por último, se llevaron a cabo diversas pruebas del algoritmo EMEDA utilizando las funciones de benchmark, en donde aquellas funciones que presentan restricciones de igualdad mostraron un mayor grado de dificultad que las demás funciones. Las funciones g10 y g13 fueron las que más problemas presentaron al algoritmo, mientras que para las demás funciones se obtuvieron buenos resultados.

El ajuste de la varianza para llevar a cabo las mutaciones del mejor individuo mejoró en gran medida el desempeño del algoritmo, logrando realizar una mejor exploración del espacio, y encaminar a la población hacia el óptimo global. Por otro lado, el ajuste de la zona factible, ayudó al algoritmo a desempeñar una búsqueda más a fondo de la solución para aquellas funciones con restricciones de igualdad, ya que funciones como la g03 y la g11 resultan difíciles de resolver si se mantiene fija la región factible.

Dado que la resolución de problemas de optimización con restricciones mediante algoritmos de estimación de distribución es un campo de investigación bastante amplio, aún queda mucho trabajo por realizar en esta línea. En cuanto a la propuesta de este trabajo de investigación, se pueden llevar a cabo las siguientes tareas:

1. Considerar una estimación de la distribución mediante una suma de distribuciones normales multivariadas, ocasionando que exista mayor interacción entre las variables.
2. Implementar un método autoadaptable para calcular el número de distribuciones normales a utilizar en el modelo de estimación de distribución.
3. Aplicar diferentes métodos de penalización, con la intención de realizar una mejor exploración del espacio de búsqueda.

Apéndice A

Funciones de Benchmark

g01

Minimizar:

$$f(x) = 5 \sum_{i=1}^4 x_i - 5 \sum_{i=1}^4 x_i^2 - \sum_{i=5}^{13} x_i,$$

sujeto a:

$$\begin{aligned} g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\ g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\ g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\ g_4(x) &= -8x_1 + x_{10} \leq 0 \\ g_5(x) &= -8x_2 + x_{11} \leq 0 \\ g_6(x) &= -8x_3 + x_{12} \leq 0 \\ g_7(x) &= -2x_4 - x_5 + x_{10} \leq 0 \\ g_8(x) &= -2x_6 - x_7 + x_{11} \leq 0 \\ g_9(x) &= -2x_8 - x_9 + x_{12} \leq 0 \end{aligned}$$

donde los límites del espacio de búsqueda son $0 \leq x_i \leq 1$ ($i = 1, \dots, 9$), $0 \leq x_i \leq 100$ ($i = 10, 11, 12$) y $0 \leq x_{13} \leq 1$. El óptimo global se encuentra en $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, las restricciones (g_1, g_2, g_3, g_7, g_8 y g_9) son activas y $f(x^*) = -15$.

g02

Maximizar:

$$f(x) = \left| \frac{\sum_{i=1}^n \cos^4(x_i) - 2 \prod_{i=1}^n \cos^2(x_i)}{\sqrt{\sum_{i=1}^n i x_i^2}} \right|$$

sujeto a:

$$\begin{aligned} g_1(x) &= 0.75 - \prod_{i=1}^n x_i \leq 0 \\ g_2(x) &= \sum_{i=1}^n x_i - 7.5n \leq 0 \end{aligned}$$

donde $n = 20$ y $0 \leq x_i \leq 10$ ($i = 1, \dots, n$). El máximo global es desconocido, el mejor resultado publicado es $f(x^*) = 0.803619$, la restricción g_1 está cerca de ser activa ($g_1 = -10^{-8}$).

g03

Maximizar:

$$f(x) = (\sqrt{n})^n \prod_{i=1}^n x_i$$

sujeto a:

$$h_1(x) = \sum_{i=1}^n x_i^2 - 1 = 0$$

donde $n = 10$ y $0 \leq x_i \leq 1$ ($i = 1, \dots, n$). El máximo global está en $x_i^* = 1/\sqrt{n}$ ($i = 1, \dots, n$) y $f(x^*) = 1$.

g04

Minimizar:

$$f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

sujeto a:

$$\begin{aligned}
g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\
g_2(x) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\
g_3(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \\
g_4(x) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \\
g_5(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \\
g_6(x) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0
\end{aligned}$$

donde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ y $27 \leq x_i \leq 45$ ($i = 3, 4, 5$). El óptimo global es $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ y $f(x^*) = -30665.539$. Las restricciones g_1 y g_6 son activas.

g05

Minimizar:

$$f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

sujeto a:

$$\begin{aligned}
g_1(x) &= -x_4 + x_3 - 0.55 \leq 0 \\
g_2(x) &= -x_3 + x_4 - 0.55 \leq 0 \\
h_3(x) &= 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\
h_4(x) &= 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\
h_5(x) &= 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0
\end{aligned}$$

donde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ y $-0.55 \leq x_4 \leq 0.55$. La mejor solución conocida es $x^* = (679.9453.1026, 067, 0.1188764, -0.3962336)$ tal que $f(x^*) = 5126.4981$.

g06

Minimizar:

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

sujeto a:

$$\begin{aligned}
g_1(x) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\
g_2(x) &= (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0
\end{aligned}$$

donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$. El mínimo global se encuentra en $x^* = (14.095, 0.84296)$ y $f(x^*) = -6961.81388$. Ambas restricciones son activas.

g07

Minimizar:

$$f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + \\ (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

sujeto a:

$$\begin{aligned} g_1(x) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\ g_2(x) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\ g_3(x) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\ g_4(x) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\ g_5(x) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\ g_6(x) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\ g_7(x) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\ g_8(x) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0 \end{aligned}$$

donde $-10 \leq x_i \leq 10$ ($i = 1, \dots, 10$). La solución óptima es $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ tal que $f(x^*) = 24.3062091$. Las restricciones (g_1, g_2, g_3, g_4, g_5 y g_6) son activas.

g08

Maximizar:

$$f(x) = \frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

sujeto a:

$$\begin{aligned} g_1(x) &= x_1^2 - x_2 + 1 \leq 0 \\ g_2(x) &= 1 - x_1 + (x_2 - 4)^2 \leq 0 \end{aligned}$$

donde $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$. El óptimo global se localiza en $x^* = (1.2279713, 4.2453733)$ tal que $f(x^*) = 0.095825$.

g09

Minimizar:

$$f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

sujeto a:

$$\begin{aligned} g_1(x) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\ g_2(x) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\ g_3(x) &= -196 + 23x_1 + x_2^3 + 6x_6^2 - 8x_7 \leq 0 \\ g_4(x) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \end{aligned}$$

donde $-10 \leq x_i \leq 10$ ($i = 1, \dots, 7$). El óptimo se encuentra en $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ y $f(x^*) = 680,6300573$. Las restricciones g_1 y g_4 son activas.

g10

Minimizar:

$$f(x) = x_1 + x_2 + x_3$$

sujeto a:

$$\begin{aligned} g_1(x) &= -1 + 0.0025(x_4 + x_6) \leq 0 \\ g_2(x) &= -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\ g_3(x) &= -1 + 0.01(x_8 - x_5) \leq 0 \\ g_4(x) &= -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0 \\ g_5(x) &= -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \\ g_6(x) &= -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0 \end{aligned}$$

donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ ($i = 2, 3$) y $10 \leq x_i \leq 1000$ ($i = 4, \dots, 8$). La solución óptima se localiza en $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, tal que $f(x^*) = 7049.3307$. Las restricciones g_1 , g_2 y g_3 son activas.

g11

Minimizar:

$$f(x) = x_1^2 + (x_2 - 1)^2$$

sujeto a:

$$h(x) = x_2 - x_1^2 = 0$$

donde $-1 \leq x_1 \leq 1$ y $-1 \leq x_2 \leq 1$. El mínimo global es $x^* = (\pm 1/\sqrt{2}, 1/2)$ y $f(x^*) = 0.75$.

g12

Maximizar:

$$f(x) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$

sujeto a:

$$g(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

donde $0 \leq x_i \leq 10$ ($i = 1, 2, 3$) y $p, q, r = 1, 2, \dots, 9$. La región factible del espacio de búsqueda consiste en 9^3 esferas disjuntas. Un punto (x_1, x_2, x_3) es factible sí y sólo si existen p, q, r , tal que la desigualdad se satisface. El óptimo global se localiza en $x^* = (5, 5, 5)$ de manera que $f(x^*) = 1$.

g13

Minimizar:

$$f(x) = e^{x_1 x_2 x_3 x_4 x_5}$$

sujeto a:

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$

$$h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0$$

$$h_3(x) = x_1^3 + x_2^3 + 1 = 0$$

donde $-2.3 \leq x_i \leq 2.3$ ($i = 1, 2$) y $-3.2 \leq x_i \leq 3.2$ ($i = 3, 4, 5$). El óptimo global se encuentra en $x^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ donde $f(x^*) = 0.0539498$.

Apéndice B

Algunas Variantes Exploradas

Durante el desarrollo de este trabajo de investigación se implementaron diversas estrategias para estimar la distribución de la población a partir de una muestra de los mejores individuos. La primera de ellas se basa en el concepto de una sola distribución normal multivariada, para esto es necesario encontrar en cada iteración, el vector de medias $\mu = (\mu_1, \dots, \mu_n)$, y la matriz de covarianzas Σ , tal que:

$$f(x; \mu, \Sigma) = \delta(x; \mu, \Sigma),$$

donde

$$\delta(x; \mu, \Sigma) = (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Dado que la matriz de covarianzas Σ es simétrica positiva definida, se puede utilizar la factorización de Cholesky para descomponerla en una matriz triangular inferior L , de manera que:

$$\Sigma = LL^T$$

Una vez factorizada la matriz de covarianzas, se procede a generar la nueva población de individuos utilizando la siguiente expresión:

$$X = \mu + LZ,$$

donde $Z = (Z_1, \dots, Z_m)$ es un conjunto de individuos provenientes de una distribución normal estándar.

Este método no mostró buenos resultados, pues el asumir que la población proviene de una sola distribución normal multivariada, ocasiona una serie de imprecisiones, en especial cuando la población se encuentra distribuida sobre todo el

espacio de búsqueda, o bien cuando ésta puede ser dividida en diferentes grupos o clusters.

El segundo método implementado consiste en estimar la distribución de la población mediante la mezcla de varias distribuciones normales multivariadas, de manera que en cada iteración los parámetros μ_i y Σ_i , correspondientes a la media y la matriz de covarianzas de la i -ésima distribución deben ser calculados, tal que:

$$f(x; w, \mu, \Sigma) = \sum_{i=1}^g w_i \delta(x; \mu_i, \Sigma_i),$$

donde

$$\delta(x; \mu_i, \Sigma_i) = (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} e^{[-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)]}$$

A diferencia del algoritmo EMEDA, donde se lleva a cabo la estimación de cada una de las distribuciones marginales, esta técnica consiste en estimar la distribución de la población mediante la suma de diversas distribuciones normales multivariadas, de manera que la interacción entre las variables sea tomada en consideración.

Para estimar la función de probabilidad mediante la mezcla de distribuciones normales, se desarrollaron dos técnicas diferentes. La primera consiste en implementar el algoritmo EM (EXpectation Maximization), explicado en el capítulo 5, para calcular los parámetros necesarios de cada una de las funciones Gaussianas. La segunda técnica se encarga de dividir a la población en k diferentes grupos utilizando el algoritmo “k-medias”, de manera que cada individuo sea asignado al grupo más cercano, esto con la intención de minimizar la función objetivo $f(x)$, definida como:

$$f(x) = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$

donde μ_i es la media de las observaciones pertenecientes a i -ésimo grupo.

Una vez identificados los grupos $S = S_1, \dots, S_k$, se procede a calcular cada uno de los parámetros necesarios para llevar a cabo la estimación de la distribución, tal que:

$$w_i = \frac{m_i}{N}$$

$$\mu_i^k = \frac{\sum_{j=1}^{m_i} x_j^k}{m_i}$$

$$\Sigma_i^{a,b} = \frac{1}{m_i} \sum_{j=1}^{m_i} (x_j^a - \mu_i^a)(x_j^b - \mu_i^b)$$

Ya que se encontraron los parámetros w_i , μ_i y Σ_i , es posible generar un nuevo conjunto de individuos a través de la función de probabilidad estimada, de manera que formen parte de la siguiente generación.

Aunque el modelo anterior considera dependencias entre las variables, los resultados obtenidos no fueron los deseados, ya que el algoritmo presenta una alta velocidad de convergencia, lo cual ocasiona que la población se estacione en un óptimo local.

Entre los métodos implementados para controlar la diversidad de la población, se llevó a cabo un proceso encargado de generar cierto número de mutaciones del mejor individuo provenientes de una distribución normal; dicho proceso consiste en una función que permite al valor de la varianza σ^2 , cambiar de manera exponencial y además oscilatoria, tal y como se muestra en la figura B.1, logrando que el rango de búsqueda disminuya conforme avanza el proceso evolutivo.

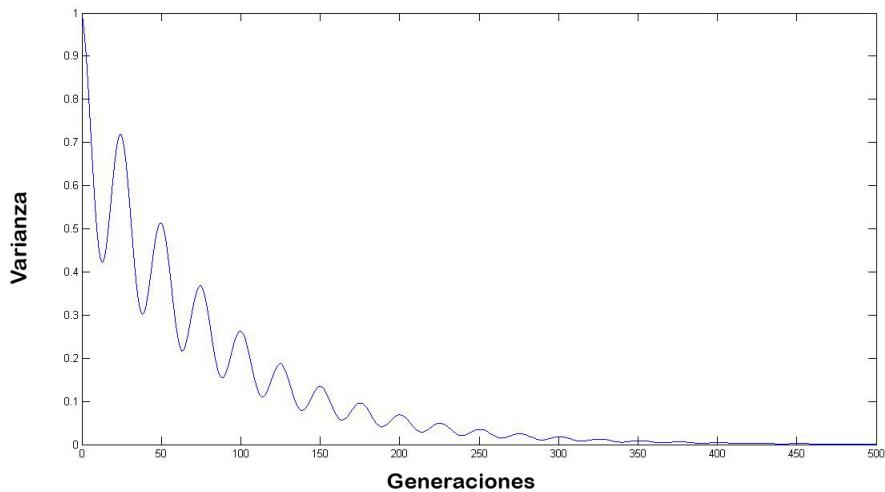


Figura B.1: Parámetro σ^2 vs. Generaciones

Esta técnica ayuda en gran medida al algoritmo a realizar una mejor explotación del espacio de búsqueda, aunque después de varias pruebas mostró ser menos eficiente que el método de ajuste de varianza adaptable implementado en el algoritmo EMEDA.

En cuanto al método de selección necesario para elegir la muestra de individuos de los cuales se construye el modelo de probabilidad, se probaron diversas alternativas, entre las que destacan la selección por truncamiento y la selección proporcional, pero debido a la fuerte presión de selección que ejercen sobre la población, se decidió utilizar una selección por torneo, y de esta manera llevar a cabo una mejor exploración del espacio.

Bibliografía

- [1] Chow, C. y Liu, C. (1968) *Approximating Discrete Probability Distributions with Dependence Trees*. IEEE Transactions on Information Theory, Vol. IT-4, No. 3.
- [2] Holland, J. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.
- [3] Bethke, A.D. (1981) *Genetic Algorithms as Function Optimizers*. PhD thesis, Department of Computer and Communication Sciences. University of Michigan.
- [4] Michalewicz, Z. (1992) *Genetic Algorithms + Data Structures = Evolution Programs*. Springer.
- [5] Bean, J. y Hadj-Alouane, A. (1992) *A Dual Genetic Algorithm for Bounded Integer Programs*. Technical Report TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan.
- [6] Powell, D. y Skolnick, M. (1993) *Using Genetic Algorithms in Engineering Design Optimization with non-linear Constraints*. University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers.
- [7] Schoenauer, M. y Xanthakis, S. (1993) *Constrained GA Optimization*. University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers.
- [8] Baluja, S. (1994) *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. Technical Report CMU-CS-94-163, Carnagie Mellon University, Pittsburgh, PA.
- [9] Hancock, P. (1994) *An Empirical Comparison of Selection Methods in Evolutionary Algorithms*. Department of Psychology, Univeristy of Stirling.

- [10] Homaifar, A., Lai, S. y Qi, X. (1994) *Constrained Optimization Via Genetic Algorithms Simulation*.
- [11] Joines, H. y Houck, C. (1994) *On the Use of non-stationary Penalty Functions to Solve nonlinear Constrained Optimization Problems with GAs*. David Fogel, editor, Proceedings of the first IEEE Conference on Evolutionary Computation. IEEE Press.
- [12] Paredis, J. (1994) *Co-evolutionary Constraint Satisfaction*. Springer Verlag.
- [13] Michalewicz, Z. y Attia, N. (1994) *Evolutionary Optimization of Constrained Problems*. Proceedings of the 3rd Annual Conference on Evolutionary Programming. World Scientific.
- [14] Baluja, S. y Caruana, R. (1995) *Removing the Genetics from the Standard Genetic Algorithm*. Proceedings of the Twelfth International Conference on Machine Learning.
- [15] Díaz-Francés, E. (1995) *The EM Algorithm. An Application to Finite Mixture Distributions*. Centro de Investigación en Matemáticas, A.C.
- [16] Le Richie, R., Knopf-Lenoir, C. y Haftka, R. (1995) *A Segregated Genetic Algorithm for Constrained Structural Optimization*. University of Pittsburgh, Morgan Kaufmann Publishers.
- [17] Michalewicz, Z. y Nazhiyath, G. (1995) *Genocop III: A co-evolutionary Algorithm for Numerical Optimization with nonlinear Constraints*. David B. Fogel, editor, Proceedings of the Second IEEE International Conference on Evolutionary Computation. IEEE Press.
- [18] Hoffmeister, F. y Sprave, J. (1996) *Problem-independent Handling of Constraints by use of Metric Penalty Functions*. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96). The MIT Press
- [19] De Bonet, J., Isbell, C. y Viola, P. (1997) *MIMIC: Finding Optima Estimating Probability Densities*. Advances in Neural Information Processing Systems 1997 MIT Press.
- [20] McLachlan, G. y Krishnan, T. (1997) *The EM Algorithm and Extensions*. John Wiley & Sons.
- [21] Bean, J. y Hadj-Alouane, A. (1997) *A Genetic Algorithm for the Multiple-choice Integer Program*. Operations Research.

- [22] Camponogara, E. y Talukdar, S. (1997) *A Genetic Algorithm for Constrained and Multiobjective Optimization*. Jarmo T. Alander, editor, 3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA). University of Vaasa.
- [23] Kuri, A. y Villegas, C. (1998) *A Universal Eclectic Genetic Algorithm for Constrained Optimization*. Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98.
- [24] Kazarlis, S. y Petridis, V. (1998) *Varying Fitness Functions in Genetic Algorithms: Studying the Rate of Increase of the Dynamic Penalty Terms*. Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V). Lecture Notes in Computer Science Vol. 1498.
- [25] Pelikan, M. y Mühlenbein, H. (1999) *The Bivariate Marginal Distribution Algorithm*.
- [26] McLachlan, G. y Peel, D. (1999) *Computing Issues for the EM Algorithm in Mixture Models*.
- [27] Nocedal, J. y Wright, S. (1999) *Numerical Optimization*. Springer.
- [28] Larrañaga, P., Etxeberria, R., Lozano, J. A. y Peña, J. M. (1999) *Optimization by Learning and Simulation of Bayesian and Gaussian Networks*. Technical Report EHU-KZAA-IK-4/99. Intelligent System Group, Dept. of Computer Science and Artificial Intelligence. University of the Basque Country.
- [29] Pelikan, M., Goldberg, D. y Cantú-Paz, E. (1999) *BOA: The Bayesian Optimization Algorithm*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), I, 525-532.
- [30] Jiménez, F. y Verdegay, J. (1999) *Evolutionary Techniques for Constrained Optimization Problems*. 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99). Verlag Mainz. ISBN 3-89653-808-X.
- [31] Runarsson, T. y Yao, X. (2000) *Stochastic Ranking for Constrained Evolutionary Optimization*. IEEE Transactions on System, MAN, and Cybernetics: Part C, Vol. X, No. XX.
- [32] Alvarez, L.F. (2000) *Design Optimization Based on Genetic Programming*. Univeristy of Bradford, UK.
- [33] Hamida, S. y Schoenauer, M. (2000) *An Adaptive Algorithm for Constrained Optimization Problems*. Springer-Verlag. Lecture Notes in Computer Science Vol. 1917.

- [34] Wu, B. y Yu, X. (2001) *Fuzzy Penalty Function Approach for Constrained Function Optimization with Evolutionary Algorithms*. Fudan University Press.
- [35] Larrañaga, P. y Lozano, J. A. (2002) *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers.
- [36] Larrañaga P., Lozano J. y Mühlenbein, H. (2003) *Estimation of Distribution Algorithms Applied To Combinatorial Optimization Problems*.
- [37] MacKay, D. (2003) *Information Theory, Interface, and Learning Algorithms*. Cambridge University Press.
- [38] Soto, M. (2003) *Un Estudio sobre los Algoritmos Evolutivos con Estimación de Distribuciones basados en Políárboles y su costo de Evaluación*. Tesis Doctoral. Instituto de Cibernética, Matemática y Física de la Habana.
- [39] Mezura, E. (2004) *Alternative Techniques to Handle Constraints in Evolutionary Optimization*. Tesis Doctoral. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional.
- [40] Runarsson, T. y Yao, X. (2004) *Search Biases in Constrained Evolutionary Optimization*. IEEE Transactions on Evolutionary Computation.
- [41] Lu, Q. y Yao, X. (2005) *Clustering and Learning Gaussian Distribution for Continuous Optimization*. IEEE Transactions on Systems, MAN, and Cybernetics - Part C: Application and Reviews, Vol. 35, No. 2.
- [42] Feoktistov, V. (2006) *Differential Evolution In Search of Solutions*. Springer.
- [43] McLachlan, G. (2007) *Model-Based Clustering*. Department of Mathematics and the Institute of Molecular Bioscience, University of Queensland.
- [44] Hernández A., Buckles, B. y Coello C. *Estrategias Evolutivas: La Versión Alemana del Algoritmo Genético*.

Índice de figuras

2.1.	Representación de un cromosoma	7
2.2.	Proporción del valor de aptitud de una población	10
2.3.	Probabilidad de selección por rango y por ruleta de una misma población	12
2.4.	Proceso de cruce de dos individuos	14
2.5.	Proceso de cruce uniforme	14
2.6.	Proceso de mutación de un individuo	15
2.7.	Selección (μ, λ)	21
2.8.	Selección $(\mu + \lambda)$	21
3.1.	Funcionamiento de un EDA básico. 1) Selección de una muestra S de M individuos. 2) Crear el modelo de probabilidad en base a la muestra S . 3) Generar la nueva población en base a la simulación del modelo de probabilidad.	26
3.2.	Estructura del algoritmo MIMIC	31
3.3.	Árbol de Dependencias BMDA	35
3.4.	Árbol de Dependencias de Chow y Liu	37
3.5.	Estructura del algoritmo PADA	40
3.6.	Estructura del algoritmo BOA	41
4.1.	Espacio de búsqueda con restricciones	49
5.1.	Paso 1. Proponer centroides iniciales	68
5.2.	Paso 2. Asociar cada observación al centroide correspondiente	68
5.3.	Paso 3. Re-calcular las posiciones de los centroides	69
5.4.	Algoritmo EM, utilizando 3 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales	71
5.5.	Algoritmo EM, utilizando 2 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales	72
5.6.	Algoritmo EM, utilizando 4 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales	73

5.7. Algoritmo EM, utilizando 7 gaussianas para estimar la distribución de los datos provenientes de 3 distribuciones normales	74
5.8. Espacio de Búsqueda con Restricciones de igualdad	75
5.9. Control de Diversidad de la Población	82
5.10. Mutaciones del mejor individuo	84
5.11. Ajuste de varianza	85
6.1. Aptitud del mejor individuo para el problema g01	93
6.2. Parámetro σ para el problema g01	93
6.3. Aptitud del mejor individuo para el problema g02	94
6.4. Parámetro σ para el problema g02	95
6.5. Aptitud del mejor individuo para el problema g03	96
6.6. Parámetro σ para el problema g03	96
6.7. Parámetro ϵ para el problema g03	97
6.8. Porcentaje de soluciones factibles para el problema g03	97
6.9. Aptitud del mejor individuo para el problema g04	98
6.10. Parámetro σ para el problema g04	99
6.11. Aptitud del mejor individuo para el problema g05	100
6.12. Parámetro σ para el problema g05	100
6.13. Parámetro ϵ para el problema g05	101
6.14. Porcentaje de soluciones factibles para el problema g05	101
6.15. Aptitud del mejor individuo para el problema g06	102
6.16. Parámetro σ para el problema g06	103
6.17. Aptitud del mejor individuo para el problema g07	104
6.18. Parámetro σ para el problema g07	104
6.19. Aptitud del mejor individuo para el problema g08	105
6.20. Parámetro σ para el problema g08	106
6.21. Aptitud del mejor individuo para el problema g09	107
6.22. Parámetro σ para el problema g09	107
6.23. Aptitud del mejor individuo para el problema g10	108
6.24. Parámetro σ para el problema g10	109
6.25. Parámetro ϵ para el problema g10	110
6.26. Porcentaje de soluciones factibles para el problema g10	111
6.27. Aptitud del mejor individuo para el problema g11	112
6.28. Parámetro σ para el problema g11	112
6.29. Parámetro ϵ para el problema g11	113
6.30. Porcentaje de soluciones factibles para el problema g11	113
6.31. Aptitud del mejor individuo para el problema g12	114
6.32. Parámetro σ para el problema g12	115
6.33. Aptitud del mejor individuo para el problema g13	116
6.34. Parámetro σ para el problema g13	116
6.35. Parámetro ϵ para el problema g13	117

6.36. Porcentaje de soluciones factibles para el problema g13	117
B.1. Parámetro σ^2 vs. Generaciones	133

Índice de cuadros

4.1. Principales Propiedades de las Funciones de benchmark	49
6.1. Parámetros del algoritmo EMEDA	91
6.2. Resultados de la función g01	92
6.3. Resultados de la función g02	94
6.4. Resultados de la función g03	95
6.5. Resultados de la función g04	98
6.6. Resultados de la función g05	99
6.7. Resultados de la función g06	102
6.8. Resultados de la función g07	103
6.9. Resultados de la función g08	105
6.10. Resultados de la función g09	106
6.11. Resultados de la función g10	108
6.12. Resultados de la función g11	111
6.13. Resultados de la función g12	114
6.14. Resultados de la función g13	115
6.15. Stochastic Ranking vs. EMEDA	118

Lista de algoritmos

1.	Algoritmo Genético Básico	19
2.	EDA básico	27
3.	UMDA	29
4.	PBIL	30
5.	MIMIC	32
6.	BMDA	34
7.	Árbol de Dependencias de Chow y Liu	37
8.	EMNA	45
9.	Stochastic Ranking	52
10.	Algoritmo EMEDA	60
11.	Algoritmo PDF	61
12.	Algoritmo EM	66
13.	Algoritmo K_medias	69
14.	Algoritmo Evaluar	77
15.	Selección_por_Torneo	80
16.	Selección_por_Truncamiento	81
17.	Control_de_Diversidad	83
18.	Mutaciones_del_Mejor_Individuo	86