CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Departamento de Computación

Un nuevo algoritmo de optimización basado en optimización mediante cúmulos de partículas utilizando tamaños de población muy pequeños

Tesis que presenta

Juan Carlos Fuentes Cabrera

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Director de la Tesis

Dr. Carlos A. Coello Coello

México, D.F. **Marzo 2008**

Agradecimientos

Agradezco a Dios por darme la oportunidad de terminar esta etapa en mi vida.

A papá, mamá y a mi hermana por su sacrificio, paciencia, comprensión e incondicional apoyo. Esta tesis está dedicada a ustedes.

A Karen por el cariño brindado durante todos estos años.

A mis maestras Rosita y Mari. Ustedes fueron las primeras en formar al hombre que ahora soy.

A los compañeros del CINVESTAV. En especial a los de la generación 2005 con quienes compartí desveladas, preocupaciones pero también buenos momentos.

Al Dr. Carlos A. Coello Coello por su apoyo durante el desarrollo de este trabajo. Gracias por compartir sus conocimientos, guiarme y ser un ejemplo a seguir.

Al Dr. Francisco José Rodríguez Henríquez y al Dr. Luis Gerardo de la Fraga, quienes con sus comentarios acertados a esta tesis, me permitieron obtener con ello un mejor trabajo.

Al CINVESTAV por brindarme la oportunidad de formar parte de esta gran institución.

Al CONACyT por su apoyo económico sin el cual no hubiera sido posible concluir este trabajo.

A los doctores y trabajadores del departamento de computación.

Este trabajo de tesis fue derivado del proyecto CONACyT titulado “Técnicas Avanzadas de Optimización Evolutiva Multiobjetivo” (Ref. 45683-Y), cuyo responsable es el Dr. Carlos A. Coello Coello.

Resumen

En la actualidad existen varios métodos deterministas para resolver problemas de optimización con ciertas características específicas. Sin embargo, estos métodos pueden resultar ineficientes e incluso inadecuados al enfrentar problemas con alta dimensionalidad, discontinuos y multimodales. Además existen problemas con espacios de búsqueda muy grandes, los cuales no pueden ser resueltos en tiempo polinomial sino exponencial. En estos casos es donde se justifica el uso de las técnicas llamadas *heurísticas*. En particular, en esta tesis estamos interesados en las heurísticas pertenecientes a la computación evolutiva llamadas *algoritmos evolutivos*.

Los algoritmos evolutivos se han utilizado exitosamente para resolver una amplia gama de problemas de optimización. Éstos operan sobre poblaciones, lo cual evita que fácilmente queden atrapados en óptimos locales y requieren poca información específica del problema, lo cual los hace herramientas de búsqueda más generales. La optimización mediante cúmulos de partículas (PSO) es una técnica evolutiva inspirada en el comportamiento social de las bandadas de aves.

En este trabajo presentamos dos algoritmos evolutivos basados en optimización mediante cúmulos de partículas. El primero fue desarrollado para resolver problemas de optimización mono-objetivo con restricciones. El segundo está diseñado para resolver problemas multiobjetivo sin restricciones. Ambos algoritmos utilizan un tamaño de población muy pequeño no mayor a cinco individuos.

Las dos propuestas realizadas son evaluadas utilizando funciones de prueba estándar. Los resultados del algoritmo mono-objetivo con restricciones presentado en esta tesis son comparados con respecto a tres técnicas representativas del estado del arte en el área. El número de evaluaciones de la funcion objetivo que nuestra propuesta requiere para alcanzar los óptimos globales es menor o igual que el número requerido por las técnicas con respecto a las cuales es comparada. Los resultados del algoritmo multiobjetivo sin restricciones presentado en este trabajo se comparan con respecto al *Non-dominated Sorting Genetic Algorithm II* (NSGA-II). Nuestra propuesta muestra poseer una mejor cobertura y distribución de las soluciones obtenidas, así como una rápida convergencia. En la mayoría de las funciones de prueba utilizadas, nuestro algoritmo requiere tan sólo de 3000 evaluaciones de la función objetivo.

Abstract

Currently, there are several deterministic methods for solving optimization problems with certain specific features. However, these methods may become inefficient and even inappropriate when facing high-dimensional, discontinuous and multimodal problems. Furthermore, there exist problems with a very large search space, which cannot be solved in polynomial time, but require exponential time. In these cases is in which the use of the so-called *heuristic* techniques is fully justified. Particularly, in this thesis we are interested in the heuristics within evolutionary computation, which are called *evolutionary algorithms*.

Evolutionary algorithms have been successfully used to solve a wide variety of optimization problems. They operate on a population of solutions, which avoids that they get easily trapped in local optima, and require little specific-domain information, which makes them more general search engines. Particle swarm optimization (PSO) is a type of evolutionary algorithm inspired on the social behavior of a flock of birds.

In this work, we present two evolutionary algorithms based on PSO. The first was developed to solve constrained single-objective optimization problems. The second was designed to solve unconstrained multi-objective optimization problems. Both approaches adopt a very small population size no larger than five individuals.

The two approaches developed as part of this work are validated using standard test functions. The results of the constrained single-objective optimizer presented in this thesis are compared with respect to three approaches representative of the state-of-the-art in the area. The number of evaluations of the objective function required by our proposed approach, in order to reach the global optima, is lower or equal to those required by the techniques with respect to which it was compared. The results of the unconstrained multi-objective optimization approach presented in this thesis are compared with respect to the *Nondominated Sorting Genetic Algorithm II* (NSGA-II). Our proposed approach is shown to provide a better coverage and spread of the solutions obtained, as well as a faster convergence. In most of the test functions adopted, our proposed approach requires only 3000 objective function evaluations.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Optimización *es el proceso de obtener el mejor resultado posible bajo ciertas condiciones dadas*. Es por ello que la optimización tiene una gran aplicación no sólo en los procesos de ingeniería sino en prácticamente cualquier rama de las las ciencias. En particular, en este trabajo estamos interesados en resolver problemas de optimización numérica.

Los métodos de optimización también son conocidos como técnicas de programación matemática. La rama de las matemáticas que se encarga del estudio de estas técnicas junto con los métodos para resolver problemas de toma de decisiones se conoce como *investigación de operaciones* [2].

Los problemas de la vida real pueden ser modelados como una o más funciones con ciertas variables de decisión. Optimización entonces puede ser definida como el proceso de obtener un valor mínimo o máximo de una función. Sin perder generalidad, por el principio de dualidad se puede considerar a la optimización como un proceso de minimización. *El principio de dualidad establece que si un punto x^* corresponde al valor mínimo de la función $f(x)$, el mismo punto corresponde al valor máximo del negativo de la función $-f(x)$* [2].

Si bien existen una gran variedad de métodos deterministas para resolver problemas de optimización, no siempre es posible aplicar dichos métodos porque requieren que se cumplan ciertas condiciones. Además existen problemas con espacios de búsqueda muy grandes, los cuales no pueden ser resueltos en tiempo polinomial sino exponencial. En estos casos es donde se justifica el uso de las técnicas llamadas *heurísticas*.

La palabra "heurística" se deriva del griego *heuriskein*, que significa "encontrar" o "descubrir". El significado de esta palabra ha variado históricamente. En [3] se define como heurística a una técnica que busca soluciones buenas (es decir, casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u optimalidad de las mismas. Incluso en algunos casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución factible en particular. En el presente trabajo estamos interesados en las heurísticas llamadas *algoritmos evolutivos*.

Los algoritmos evolutivos se han utilizado exitosamente para resolver una amplia gama de problemas de optimización. En los últimos años se han desarrollado una gran cantidad de nuevas técnicas evolutivas. Éstas no solamente están basadas en la evolución natural sino que algunas de ellas tratan de simular otros procesos o sistemas naturales. En particular, una técnica evolutiva reciente en el área de los algoritmos evolutivos llamada optimización mediante cúmulos de partículas (*Particle Swarm Optimization*, PSO por sus siglas en inglés) ha sido aplicada satisfactoriamente en varias áreas de investigación y en aplicaciones del mundo real. En esta tesis adoptamos esta técnica para desarrollar dos algoritmos evolutivos. Ambos algoritmos utilizan un tamaño de población muy pequeño no mayor a cinco individuos.

A continuación, a manera de introducción al tema de optimización, en las siguientes secciones se da una breve descripción del tema.

1.1. Optimización global

Cuando un problema de optimización involucra una sola función objetivo, a la tarea de encontrar la solución óptima se le denomina **optimización mono-objetivo** u **optimización global** [4].

El problema de optimización mono-objetivo puede ser definido matemáticamente como sigue:

Encontrar un vector

$$\vec{x} = x_1, x_2 ... x_n$$

que optimice la función

$$f(\vec{x})$$

sujeta a las restricciones:

$$g_i(\vec{x}) \leq 0, i = 1, 2, ..., m$$

$$h_j(\vec{x}) = 0, j = 1, 2, ..., p$$

donde:

$\vec{x}$ es un vector de dimensión n llamado *vector de decisión*,
$f(\vec{x})$ es llamada *función objetivo*,
$g_i(\vec{x})$ son las *restricciones de desigualdad* y,
$h_j(\vec{x})$ son las *restricciones de igualdad*.

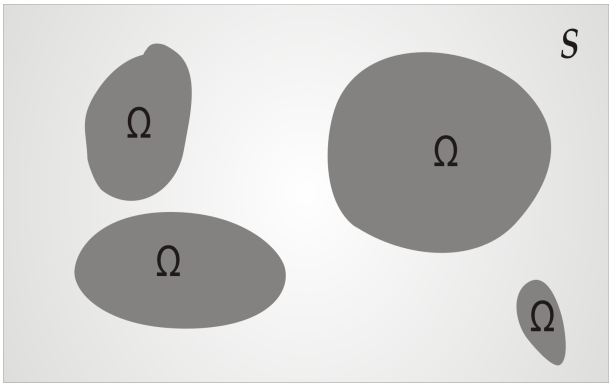

Figura 1.1: Ejemplo de regiones factibles disjuntas Ω dentro de un espacio de búsqueda S

El **vector de decisión** contiene a las **variables de decisión**. Estas variables irán modificando sus valores a lo largo de la resolución del problema dentro de un rango específico, el cual restringe a cada variable para tomar un valor dentro de un límite inferior y un límite superior. Los límites constituyen el espacio de las variables o simplemente el **espacio de decisión** S. En cualquier algoritmo de optimización, la búsqueda se ejecuta en dicho espacio.

La **función objetivo** es aquella función que se va a optimizar y está definida en términos de las variables de decisión.

Si una solución no satisface las restricciones (igualdad y desigualdad) o los límites (inferiores y superiores) de las variables se dice que la *solución no es factible*. En el caso contrario, si una solución satisface todas las restricciones al igual que también satisface los límites de las variables, se le denomina *solución factible*.

Al conjunto de todas las soluciones factibles se le conoce como **región factible**, y lo denotaremos por la letra griega Ω [4]. Al contrario de lo que se pudiera pensar, al dividirse el espacio de búsqueda encontrar soluciones óptimas se vuelve una tarea ardua. Incluso existen problemas en los cuales existen varias regiones factibles disjuntas formando islas en las cuales los métodos pueden quedar atrapados fácilmente dificultando la búsqueda de soluciones óptimas (ver figura 1.1).

Se dice que una restricción $g(\vec{x})$ es **activa** cuando $g(\vec{x}) = 0$ en el óptimo. Por definición todas las restricciones de igualdad $h(\vec{x})$ se consideran activas.

Aquellos problemas en los cuales no existen restricciones se denominan **problemas de optimización sin restricciones**. A los problemas con un número de restricciones de igualdad y/o desigualdad mayor a cero, se les denomina **problemas de optimización con restricciones**. Tanto las restricciones de igualdad como las de desigualdad pueden ser lineales o no lineales. Si la función objetivo o alguna de las restricciones es no lineal, estamos

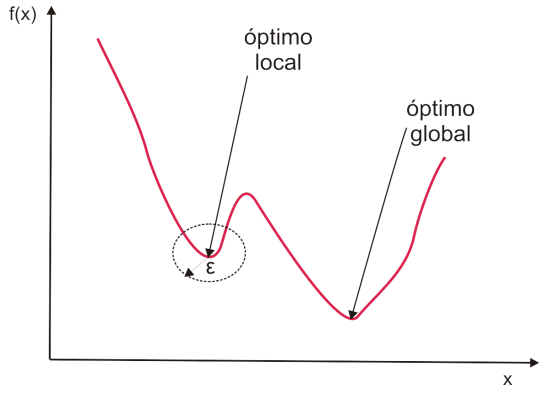

Figura 1.2: Ejemplo de una función de una variable con un mínimo local.

resolviendo **problemas de optimización no lineal**.

Cuando se optimiza una función, pueden existir una o varias soluciones factibles que son soluciones mínimas (o máximas) en la vecindad de algún vector de decisión las cuales reciben el nombre de **óptimo local**. Si una solución factible es mínima y en todo el espacio de búsqueda no existe otra que sea más pequeña (suponiendo minimización) ésta es llamada **óptimo global** (ver figura 1.2).

La soluciones denominadas *mínimo local* y *mínimo global* se pueden definir formalmente como:

Una función $f(\vec{x})$ *posee un mínimo local en* $\vec{x^l} \in \Omega$ *si y sólo si* $f(\vec{x^l}) \leq f(\vec{x})$ *para toda* $\vec{x}$ *a una distancia* ϵ *de* $\vec{x^l}$.

Es decir, existe una $\epsilon > 0$ tal que para toda $\vec{x}$ que satisface $\left|\vec{x} - \vec{x^l}\right| < \epsilon$, $f(\vec{x^l}) \leq f(\vec{x})$.

Dada una función $f : \Omega \subseteq \Re^n \rightarrow \Re, \Omega \neq \phi$, *para* $\vec{x} \in \Omega$, *el valor* $f^* \triangleq f(\vec{x^*}) > -\infty$ *es llamado mínimo global si y sólo si* $\forall \vec{x} \in \Omega : f(\vec{x^*}) \leq f(\vec{x})$.

De esta manera, $f(\vec{x^*})$ es la solución (o soluciones) mínima(s) global(les), f es la función objetivo, y el conjunto Ω es la región factible ($\Omega \subset S$).

1.1.1. Métodos clásicos de optimización

.

Existen varios métodos para resolver problemas de optimización con ciertas características específicas. Entre ellos, podemos distinguir técnicas para resolver problemas de optimización lineal, no lineal y problemas con restricciones.

La *programación lineal* es la encargada de estudiar métodos y técnicas para optimizar problemas en los cuales todas las funciones involucradas (función objetivo y restricciones) son lineales. Existen varios métodos para resolver este tipo de problemas, pero el más popular es el *método simplex*. Este método se basa en la observación de que si se encierra el conjunto factible de un problema lineal en un especie de *polihedro* entonces los valores máximos o mínimos de las funciones lineales deberán de estar en un vértice [5].

Para resolver problemas no lineales existen métodos de búsqueda directa como la *búsqueda de patrones* (Hooke y Jeeves), el *método simplex no lineal* (Nelder Mead) o el *método de direcciones conjugadas* (Powell) [6]. También existen métodos de búsqueda indirectos llamados métodos de gradiente como el *descenso empinado* (Cauchy) y el *gradiente conjugado* (Fletcher y Reeves) [2].

Dos métodos comúnmente utilizados para resolver problemas de optimización no lineal con restricciones son los *multiplicadores de Lagrange* y las *funciones de penalización*.Los multiplicadores de Lagrange proporcionan un conjunto de condiciones necesarias de optimalidad, convirtiendo el problema con restricciones en un problema equivalente sin restricciones (función Lagrangiana) agregando ciertos parámetros (multiplicadores de Lagrange) los cuales son tratados como variables pudiendo formar un sistema de ecuaciones [7]. Por su parte, la idea de utilizar funciones de penalización para resolver problemas de optimización con restricciones también es transformar dicho problema en uno sin restricciones. Las funciones de penalización se clasifican en [8]: *penalización interior* y *penalización exterior*. En la primera se requiere de soluciones factibles al inicio de la búsqueda. Este tipo de métodos tratan de guiar la búsqueda únicamente dentro de la zona factible, pues el factor de penalización extingue a las soluciones que no son factibles durante la búsqueda. Por su parte, la penalización exterior se permiten soluciones que no son factibles al inicio de la búsqueda y tratan de guiar la búsqueda hacia la zona factible. Lamentablemente, todas estas funciones requieren un ajuste fino del factor de penalización el cual depende del problema.

Sin embargo, como se comentó anteriormente, no siempre es posible aplicar dichos métodos. Por ejemplo, algunos métodos requieren que las funciones sean continuas o derivables. Otros necesitan que los sistemas de ecuaciones que se forman tengan solución y algunos más requieren de entrada una solución inicial factible.

1.2. Optimización multiobjetivo

Los problemas del mundo real generalmente involucran a más de una función objetivo. A la tarea de buscar una o más soluciones óptimas para este tipo de problemas se le conoce como **optimización multiobjetivo** [4]. La optimización multiobjetivo se encarga entonces de resolver problemas en los cuales se requiere optimizar simultáneamente un conjunto de funciones objetivo que normalmente se encuentran en conflicto unas con otras, es decir, mejorar una función implica deteriorar el desempeño de otra u otras. Tal como sucede en

el problema de optimización mono-objetivo, pueden existir también un número de restricciones que las soluciones deben satisfacer.

Podemos definir matemáticamente al problema de optimización multiobjetivo como:

Encontrar un vector

$$\vec{x}$$

que satisfaga las m restricciones de desigualdad:

$$g_i(\vec{x}) \leq 0; i = 1, ..., m$$

las p restricciones de igualdad:

$$h_i(\vec{x}) = 0; i = 1, ..., p$$

y optimice el vector de funciones objetivo:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]$$

Al igual que en la optimización mono-objetivo, las restricciones de igualdad y desigualdad determinan la *región factible* del problema y un punto $\vec{x}$ que se encuentre en dicha región define una solución factible.

Una de las diferencias más notables entre la optimización mono-objetivo y la optimización multiobjetivo consiste en que en la segunda *las funciones objetivo constituyen un espacio multidimensional* llamado **espacio de los objetivos**. Es decir, para cada solución en el espacio de decisión (variables) existe un punto en el espacio de los objetivos. Aunque los espacios están relacionados por un mapeo entre ellos, dicho mapeo es normalmente altamente no lineal y las propiedades de los dos espacios (decisión y objetivos) no son similares. Por ejemplo, dos soluciones cercanas en un espacio no necesariamente lo están en el otro.

Existen tres tipos de problemas multiobjetivo [9] según su planteamiento:

- Aquellos en que se requiere minimizar todas las funciones objetivo.
- Aquellos en que se requiere maximizar todas las funciones objetivo.
- Aquellos en que se requiere minimizar algunas funciones y maximizar otras.

Sin embargo, para simplificar el problema, también se puede aplicar el principio de dualidad para convertir los problemas de maximización en problemas de minimización, o

viceversa.

Si todas la funciones objetivo y las restricciones son lineales, el problema es llamado **problema multiobjetivo lineal**. Sin embargo, si una de las funciones objetivo o restricciones es no lineal, el problema es denominado **problema multiobjetivo no lineal**.

Al presentarse más de una función objetivo, dichas funciones pueden ser *conmesurables* (si están expresadas en las mismas unidades) o *no conmesurables* (cuando se expresan en unidades diferentes).

1.2.1. Optimalidad de Pareto

Al resolver problemas de optimizacion multiobjetivo, la noción de óptimo cambia. Mientras los problemas de optimización mono-objetivo tienen una solución óptima única, los problemas de optimización multiobjetivo normalmente presentan un conjunto de soluciones, las cuales al ser evaluadas, producen vectores cuyos componentes representan los mejores compromisos posibles en el espacio de las funciones objetivo [9]. En 1881, Francis Ysidro Edgerworth propuso una definición de óptimo que precisamente se refiere a la obtención de este tipo de compromisos. Poco tiempo después Vilfredo Pareto generalizó esta noción que hoy se conoce como "óptimo de Pareto" [10]. Formalmente, se puede definir un óptimo de Pareto se la manera siguiente:

*Un vector de variables de decisión $\overrightarrow{x*} \in \Omega$ (donde Ω es la zona factible) es un **óptimo de Pareto** si para cada $\overrightarrow{x} \in \Omega$ e $I = \{1, 2, ..., k\}$, $\forall_{i \in I}(f_i(\overrightarrow{x*}) \leq f_i(\overrightarrow{x}))$ y existe al menos una $j \in I$ tal que $f_j(\overrightarrow{x*}) < f_j(\vec{x})$.*

Es necesario señalar que existirán varios óptimos en un problema multiobjetivo sólo si los objetivos se encuentran en conflicto entre ellos; de lo contrario la cardinalidad del conjunto de óptimos es uno [4], aunque la ausencia de conflictos entre los objetivos ocurre muy escasamente en la práctica.

De acuerdo con la definición de óptimo de Pareto, se obtendrá entonces un conjunto de soluciones denominado **conjunto de óptimos de Pareto**. Al vector correspondiente a cada solución incluida en dicho conjunto se le denomina **no dominado**. Matemáticamente, se define entonces la relación de **Dominancia de Pareto** como:

*Un vector $\overrightarrow{u} = (u_1, ..., u_k)$ se dice que **domina** a otro $\overrightarrow{v} = (v_1, ..., v_k)$ (denotado por $\vec{u} \preceq \vec{v}$) si y sólo si u es parcialmente menor a v, es decir $\forall i \in \{1, 2, .., k\}, u_i \leq v_i \wedge \exists i \in \{1, 2, ...k\} : u_i < v_i$.*

Es decir, para que una solución domine a otra, no debe ser peor en ninguno de los objetivos y debe ser estrictamente mejor en al menos uno de ellos. Por tanto, dadas dos

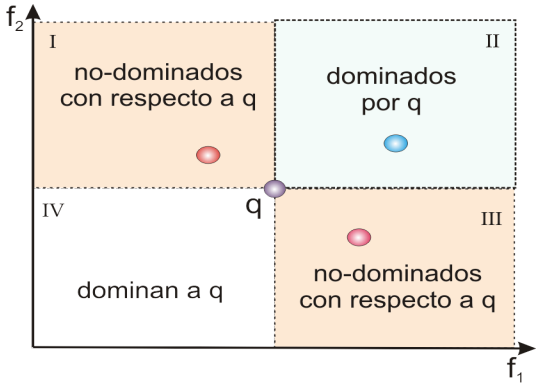

Figura 1.3: Relación de dominancia de Pareto

soluciones p y q existen tres posibilidades:

- q *domina* a p.
- p *domina* a q.
- p es *no dominada* con respecto a q.

En la figura 1.3 se puede apreciar que la solución q domina a cualquier solución situada en el cuadrante II pero las soluciones en los cuadrantes I y III son no dominadas con respecto a q. Cualquier solución en el cuadrante IV domina a q.

La relación de dominancia posee las siguientes propiedades [11]:

- **Reflexión** : *No es reflexiva* debido a que una solución no se puede dominar a ella misma.
- **Simetría**: *No es simétrica* dado que no es posible que una solución p que domine a otra solución q implique que q domine a p. En realidad si p domina a q, entonces p es no dominado por q.
- **Anti-simétrica**: Dado que la relación de dominancia no es simétrica, *no puede ser anti-simétrica*.
- **Transitiva**: La relación de dominancia *es transitiva*, es decir si p domina a q y q domina a r entonces p domina a r.

Cabe señalar que si una solución p no domina a q esto no implica que q domine a p. Como la relación de dominancia es transitiva pero no es reflexiva y no es antisimétrica, entonces no es una relación de orden parcial por lo tanto *la relación de dominancia es un orden parcial estricto*.

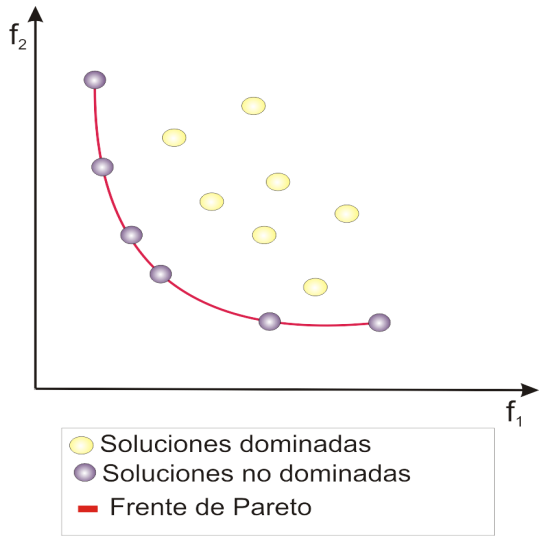

Figura 1.4: Ejemplo de soluciones no dominadas para dos funciones objetivo

El conjunto de óptimos de Pareto se puede definir como:

Dado un problema de optimización multiobjetivo $\vec{f}(x)$*, el* ***conjunto de óptimos de Pareto*** P^* *se define como* $P^* := x \in \Omega | \neg \exists x' \in \Omega \vec{f}(x') \preceq \vec{f}(x)$.

Es decir, dado un conjunto de soluciones, el conjunto de óptimos de Pareto lo constituyen todas las soluciones no dominadas.

Dado un problema de optimización multiobjetivo $\vec{f}(x)$ y un conjunto de óptimos de Pareto P^*, el **frente de Pareto** (FP^*) se define como:

$$FP^* := \{\vec{u} = \vec{f} = [f_1(x), ..., f_k(x)] | x \in P^*\}.$$

En general, no es posible encontrar una expresión analítica de la línea o superficie que contenga a dichos puntos, para un problema arbitrario. El procedimiento normal para generar el Frente de Pareto es obtener los puntos factibles Ω y su $f(\Omega)$ correspondiente. Cuando hay un número suficiente de ellos, es entonces posible determinar los puntos no dominados y producir el Frente de Pareto (ver figura 1.4).

Cuando se resuelven problemas de optimización multiobjetivo se persiguen principalmente dos metas:

1. Encontrar un conjunto de soluciones tan cercanas como sea posible al conjunto de óptimos de Pareto.

2. Encontrar un conjunto de soluciones tan diversas como sea posible.

Es decir, además de alcanzar el conjunto de óptimos de Pareto, las soluciones también deberán de estar bien distribuidas.

1.2.2. Métodos clásicos de optimización multiobjetivo

.

Cohon y Marks [12] clasifican las técnicas clásicas de optimización multiobjetivo en *a priori*, *a posteriori* y progresivas.

En las **técnicas a priori** las preferencias son dadas antes de que la técnica comience la búsqueda. Incluye a aquellos enfoques los cuales presuponen que ciertas tareas pueden ser ejecutadas por el tomador de decisiones antes de comenzar la búsqueda. Entre estos métodos se encuentran el lexicográfico, la programación por metas y el método min-max.

El *método lexicográfico* consiste en ordenar las funciones objetivo de acuerdo a su importancia. Después, se minimiza primero la función objetivo más importante y se continúa con el ordenamiento pre-establecido. En la *programación por metas* el tomador de decisiones asigna las metas que desea satisfacer por cada objetivo. Éstas se incorporan al problema como restricciones por lo que la función objetivo intenta minimizar las desviaciones de cada función objetivo con respecto a las metas establecidas para cada uno de ellos. El *método min-max* trata de minimizar las desviaciones relativas máximas con respecto a diferentes mínimos que son obtenidos optimizando cada objetivo por separado.

Las **técnicas a posteriori** realizan la búsqueda antes de realizar la toma de decisiones. Es decir, estas técnicas no requieren información preliminar sobre las preferencias del tomador de decisiones. Dos métodos muy populares son el método de restricciones ϵ y el método de las sumas ponderadas.

El *método de restricciones* ϵ consiste en fijar un objetivo y restrigir los objetivos restantes dentro de un valor ϵ especificado por el usuario. Este método funciona sin importar la forma del frente de Pareto, pero se requiere un análisis previo para obtener los rangos de variación del frente de Pareto, lo cual requiere varias optimizaciones mono-objetivo. Además que el número de evaluaciones de la función objetivo que el método requiere es elevado.

El *método de las sumas ponderadas* escala un conjunto de objetivos en un solo objetivo multiplicando cada uno de ellos por un peso proporcionado por el usuario. Este método garantiza encontrar soluciones en todo el conjunto de óptimos de Parteo sólo si el frente de Pareto es convexo. Además la asignación de los valores de los pesos no es trivial y este problema se agudiza confirme aumenta la cantidad de funciones objetivo.

Las **técnicas progresivas** incorporan la búsqueda y la toma de decisiones. Estas técnicas generalmente siguen tres pasos : 1) obtener una solución no dominada, 2) interactuar con el tomador de decisiones y modificar las preferencias con base en esa solución y 3) repetir los pasos anteriores hasta que el tomador de decisiones esté satisfecho o hasta que ya no sea posible obtener mejoras.

A pesar de que existen una gran variedad de métodos deterministas para resolver problemas de optimización multiobjetivo, estos métodos pueden resultar ineficientes e incluso inadecuados al enfrentar problemas con alta dimensionalidad, discontinuos, multimodales y/o NP-completos [9]. Por esta razón, se han desarrollado algunas alternativas como los métodos heurísticos tales como el Recocido Simulado [13], la Búsqueda Tabú [14] y la Computación Evolutiva [15] (en la cual estamos interesados en este trabajo y de la cual daremos detalles en el siguente capítulo). La Búsqueda Tabú al igual que los algoritmos evolutivos son realmente una *meta-heurística*, porque es un procedimiento heurístico que debe acoplarse a otra técnica que actúa como motor de búsqueda. Si bien algunas de estas técnicas eventualmente encuentran un óptimo, no se puede garantizar que encontrarán siempre la solución óptima.

1.3. Organización de la tesis

El resto del presente documento se describe brevemente a continuación.

Capítulo 2. Introducción a la Computación Evolutiva

En este capítulo se proporciona una pequeña introducción a la computación evolutiva, así como una breve descripción de los paradigmas que pertenecen a ésta. Se presentan algunas técnicas evolutivas multoibjetivo consideradas dentro del estado del arte en el área. Además, se introduce la técnica de optimización mediante cúmulos de partículas, que es la heurística evolutiva adoptada en esta tesis.

Capítulo 3. Propuesta de Optimización Mediante Cúmulos de Partículas en Espacios Restringidos

En este capítulo se describe nuestra propuesta para resolver problemas de optimización mono-objetivo con restricciones. También se muestran los resultados obtenidos al comparar nuestra propuesta con respecto a tres técnicas elegidas dentro del estado del arte en el área.

Capítulo 4. Propuesta de Optimización Multiobjetivo Mediante Cúmulos de Partículas

En este capítulo se describe nuestra propuesta para resolver problemas de optimización multiobjetivo. Se describen las medidas de desempeño utilizadas para medir el rendimien-

to de nuestra propuesta. También se muestran los resultados obtenidos al comparar nuestra propuesta con respecto al algoritmo elegido dentro del estado del arte en el área.

Capítulo 5. Conclusiones y Trabajo Futuro

Finalmente, en el capítulo 5 se presentan las conclusiones con base en los resultados obtenidos por los algoritmos desarrollados en esta tesis. Se mencionan también algunas ideas o líneas de investigación interesantes para continuar con los temas abordados en este documento.

Apéndice A. Funciones de Prueba con Restricciones

En este apéndice se proporcionan las funciones de prueba para problemas de optimización con restricciones utilizadas para comparar a nuestra propuesta con respecto a los algoritmos elegidos dentro del estado del arte en el área.

Apéndice B. Funciones de Prueba Multiobjetivo

En este apéndice se proporcionan las funciones de prueba para problemas de optimización con multiobjetivo utilizadas para comparar a nuestra propuesta con respecto al algoritmo elegido dentro del estado del arte en el área.

Apéndice C. Frentes de Pareto Obtenidos por Nuestro Micro-MOPSO

Por último, en este apéndice se presentan los frentes de Pareto obtenidos por nuestra propuesta de las funciones de prueba que se utilizaron para validar el algoritmo multiobjetivo propuesto.

Capítulo 2

Introducción a la Computación Evolutiva

El término **Computación Evoutiva** (CE) es usado para describir al campo de investigación que concierne a las técnicas heurísticas inspiradas en los principios de la teoría *Neo-Darwiniana* de la evolución natural. Dichas técnicas reciben el nombre de **Algoritmos Evolutivos** (AEs) [16].

2.1. Neo-Darwinismo

El paradigma conocido como *neo-darwinismo* está basado en la combinación de la teoría evolutiva de Darwin junto con el principio de selección de Weismann y la teoría genética de Mendel [1].

La teoría de la evolución de Charles Darwin ofrece una explicación de la diversidad biológica. Esta teoría se basa en dos piedras angulares: la primera es la **selección natural** en la cual dado un medio ambiente se favorece a aquellos individuos que compiten por los recursos limitados, es decir, se adaptan mejor a las condiciones del medio ambiente. Este fenómeno es conocido como la *ley del más apto*. La segunda piedra angular se refiere al hecho de que existen pequeños cambios físicos y de comportamiento, los cuales son hereditarios y afectan directamente el desarrollo del individuo en su medio ambiente. Cada individuo representa una combinación única de características fenotípicas (físicas) que son evaluadas por el medio ambiente. Si dichas características resultan favorables, éstas se propagan a través de sus descendientes; de lo contratio, son descartadas. Darwin concibió también que existen pequeños cambios aleatorios (mutaciones) que ocurren durante la reproducción, creando individuos con nuevas características. Estos nuevos individuos también son evaluados en el medio ambiente, es decir, los mejores sobreviven y se reproducen; por lo tanto, evolucionan [17].

El principio de selección de Weismann se basa en la teoría del germoplasma. Weismann

propuso que la información hereditaria se transmite a través de ciertas células llamadas *germinales*, mientras que otras llamadas *somáticas* no pueden transmitir nada.

La teoría genética de Mendel se basa en un conjunto de reglas o leyes relacionadas con la transmisión de características hereditarias de un organismo padre a sus descendientes.

A los individuos podemos verlos como entidades duales. Es decir, sus propiedades *fenotípicas* (rasgos específicos tales como características físicas y de comportamiento) son representadas en un nivel bajo por su *genotipo* (composición genética). Los genes son las unidades funcionales de la herencia. Los genes de un organismo están agrupados en cromosomas (los seres humanos tenemos 46) los cuales son los responsables de la transmisión de la información genética. Al lugar que ocupa un gen en un cromosoma se le llama *alelo*.

El Neo-Darwinismo establece que la historia de la mayoría de la vida en nuestro planeta puede ser explicada a través de un puñado de procesos estocásticos que actúan sobre y dentro de las poblaciones y especies [18]. Estos procesos son: *reproducción*, *mutación*, *competencia* y *selección*.

- La *reproducción* es una propiedad inherente a todas las especies la cual permite la formación de nuevos organismos. La reproducción se logra a través del paso de la información genética de los individuos a sus descendientes. La reproducción puede ser sexual o asexual.

- La *mutación* es una pequeña alteración o cambio en la información genética en el momento que es transmitida en la reproducción. La única unidad genética capaz de mutar es el gen. La mutación está garantizada en cualquier sistema que se reproduce y se encuenta en equilibrio.

- Como sabemos nuestro planeta consta de un espacio finito con recursos naturales limitados. Por lo tanto, se garantiza la *competencia* entre las especies por su supervivencia y crecimiento.

- La selección es el resultado de la competencia entre individuos y las especies por los recursos limitados. La selección sexual se refiere al proceso en el cual machos y hembras de una especie determinan (seleccionan) a su pareja para reproducirse.

De esta manera, la *evolución* surge como el resultado de la interacción de estos procesos estocásticos entre las poblaciones, generación tras generación. El neo-Darwinismo afirma que la selección natural es la fuerza evolutiva predominante, la cual condiciona las características fenotípicas de los organismos en la mayoría de las situaciones que toman parte en

Tabla 2.1: Planteamiento básico de un problema de optimización con la evolución natural

Problema de Optimización	Evolución natural
Problema	Medio ambiente
Candidato a Solución	Individuo
Calidad de la solución	Aptitud

la naturaleza.

La aptitud de un individuo es una medida indirecta ya que está en función del desarrollo de dicho individuo en comparación con los demás; es decir, su posibilidad de sobrevivir y reproducirse en un ambiente particular.

La evolución natural puede verse entonces como un problema de optimización donde la finalidad es adaptar a los mejores individuos a su ambiente (ver tabla 2.1).

2.2. Algoritmos evolutivos

Los Algoritmos Evolutivos (AEs) se han utilizado exitosamente para resolver una amplia gama de problemas de optimización [19]. El éxito de estas técnicas se debe a sus características dentro de las cuales se encuentran las siguientes: no necesitan conocimientos específicos sobre el problema a resolverse, operan sobre poblaciones, lo cual evita que fácilmente queden atrapadas en óptimos locales, requieren poca información específica del problema, lo cual los hace herramientas de búsqueda más generales.

Los principales componentes de un AE [17] son:

1. Representación (definición de los individuos)
2. Función de evaluación (función de aptitud)
3. Población
4. Mecanismo de selección de padres
5. Operadores de recombinación y mutación
6. Mecanismo de supervivencia

El primer paso para definir un AE es establecer un puente entre el contexto original del problema y el espacio de soluciones. Los objetos que forman posibles soluciones son referenciados por el fenotipo, mientras que su codificación es realizada por el genotipo. Cabe

mencionar que se debe de establecer un mapeo (relación) entre el fenotipo y el genotipo. Por ejemplo, dado un problema de optimización con números enteros, el conjunto de los enteros puede formar el conjunto de fenotipos. Sin embargo, se puede utilizar un código binario para la representación del genotipo (evidentemente, en este caso requeriríamos un mapeo adecuado entre estos dos espacios, el binario y el de los enteros).

La función de evaluación representa la tarea a resolver por la técnica evolutiva empleada. Típicamente, esta función está compuesta por una medida de calidad en el espacio del fenotipo. En CE, la función de evaluación es denominada función de aptitud. En los AEs, al resolver un problema de optimización, la función de evaluación (o aptitud) es generalmente la función objetivo o una simple transformación de ella.

La función de la población en un AE es mantener distintas soluciones posibles a un problema. Simultáneamente, los operadores de selección (selección de padres y mecanismo de supervivencia) se llevan a cabo a nivel de la población. La *diversidad* de una población es medida con base en el número de soluciones diferentes que están presentes en dicha población. El efecto de perder rápidamente diversidad en la población y quedar atrapado en un óptimo local es conocido como *convergencia prematura*.

El papel de la selección de padres es distinguir a los individuos con base en su calidad relativa (es decir, con respecto al resto de la población actual), a fin de permitir que los mejores generen descendientes. En CE, la selección es típicamente probabilística.

Los operadores de recombinación y mutación tienen como objetivo crear nuevos individuos a partir de los actuales. En CE, la mutación siempre es estocástica. La recombinación o cruza, mezcla información de los padres a nivel del genotipo para crear descendientes. La idea de la recombinación es simple: dados dos individuos con diferentes características (deseables), se busca producir descendientes que combinen las caracterísicas favorables de dichos padres.

El mecanismo de supervivencia distingue a los individuos de la población con base en su aptitud. Suele adoptarse un tamaño de población constante en los AEs, por lo que se requiere realizar una selección de los individuos (población actual y descendientes) que sobrevivirán a la siguiente generación.

2.3. Paradigmas de la computación evolutiva

La Computación Evolutiva engloba tres paradigmas principales [20] que colectivamente son conocidos como Algoritmos Evolutivos (AEs):

- Programación evolutiva.
- Estrategias evolutivas.

- Algoritmos genéticos.

2.3.1. Programación evolutiva

La **Programación Evolutiva** (PE) fue propuesta por Lawrence J. Fogel en los 1960s para resolver problemas de predicción de secuencias de símbolos haciendo uso de un operador de mutación para generar máquinas de estados finitos [1].

Fogel propuso la programación evolutiva para diseñar automáticamente máquinas de estados finitos de la siguiente manera (ver figura 2.1):

1. Una población de máquinas de estados finitos es inicializada aleatoriamente con memoria limitada de acuerdo al número máximo de estados.
2. Estas máquinas son expuestas a una secuencia de símbolos definidos como medio ambiente.
3. Los padres existentes son mutados para crear una máquina hijo.
4. Los padres y el hijo son evaluados de acuerdo a qué tan bien predicen el siguiente símbolo de la secuencia de símbolos conocida.
5. La mejor mitad de estas máquinas es seleccionada para conformar el conjunto padres en la siguiente iteración.
6. Si se requiere, la mejor máquina es usada para predecir el siguiente símbolo y el símbolo actual es agregado al conjunto de datos conocido.
7. El proceso se repite hasta cumplir una cierta condición de paro.

En la PE, la inteligencia se ve como un comportamiento adaptativo. Este tipo de AE enfatiza los nexos de comportamiento entre padres e hijos y es una abstracción de la evolución a nivel de las especies, por lo que no utiliza operadores de recombinación (especies distintas, no se recombinan) y hace uso de una selección probabilística. Un algoritmo básico de la PE se muestra en el algoritmo 1.

2.3.2. Estrategias evolutivas

Las **Estrategias Evolutivas** (EE) fueron desarrolladas en 1964 por Ingo Rechenberg y Hans-Paul Schwefel quienes eran estudiantes de doctorado en la Universidad Técnica de Berlín. En la versión original de esta técnica, la población consistía de un solo individuo y con éste se generaba un solo hijo (1+1)-EE, el cual se mantenía sólo si era mejor que el

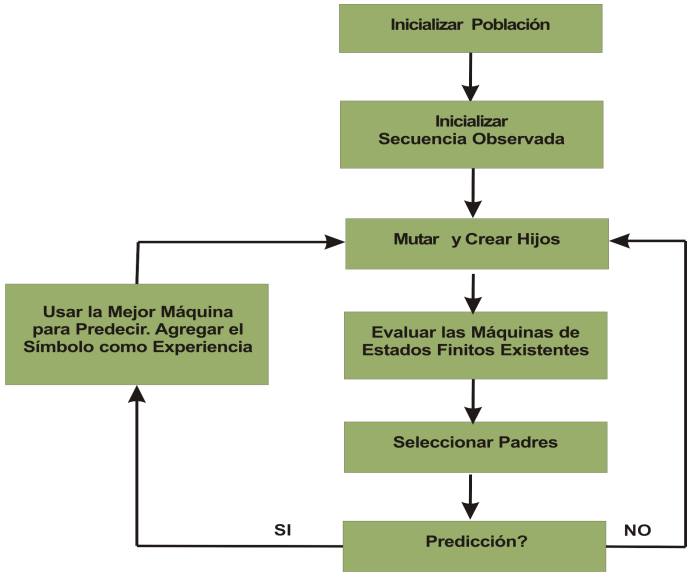

Figura 2.1: Diagrama de flujo de la programación evolutiva propuesta originalmente por Lawrence J. Fogel [1]

Algoritmo 1: Algoritmo básico de la programación evolutiva
Entrada: Población inicial, número de generaciones, porcentaje de mutación
Salida: Población evolucionada

```
begin
    Generar una población inicial
    Evaluar la población inicial
    repeat
        Aplicar un operador de mutación
        Evaluar población actual
        Realizar la selección
    until Cierta condición
end
```

padre [15].

Rechenberg desarrolló una regla teórica para ajustar la mutación tomando en cuenta la desviación estándar de forma tal que el algoritmo pudiera converger al óptimo global. Esta regla es conocida como "la regla de éxito $1/5$" y dice que la razón de mutaciones exitosas con respecto al total de las mutaciones realizadas debe ser exactamente $1/5$. Si es mayor, debe incrementarse la desviación estándar y si es menor, debe decrementarse.

Posteriormente, Rechenberg introdujo el concepto de población al proponer una EE en la cual hay μ padres y se genera un solo hijo $(\mu + 1)$-EE. A su vez, Schwefel propuso otra estrategia en la cual hay μ padres que generan λ hijos: $(\mu+\lambda)$-EE y (μ, λ)-EE. En el primer caso, los mejores μ individuos de la unión de los padres e hijos sobreviven a la siguiente generación mientras que en el segundo los mejores μ hijos son los que sobreviven y entonces $\lambda > \mu$.

Algunas características de las estrategias evolutivas [17] son:

- Suelen ser recomendables para optimización de parámetros continuos.
- Enfatizan el papel de la mutación sobre el de la recombinación.
- La mutación es implementada agregando ruido Gaussiano.
- Los parámetros de la mutación se auto-adaptan durante la ejecución del algoritmo.

Las estrategias evolutivas simulan el proceso evolutivo a nivel de los individuos, por lo que es posible la recombinación, aunque ésta es un operador secundario. En las EE se usa normalmente una selección determinística. Un algoritmo básico de las EE se muestra en el algoritmo 2.

Una de las principales contribuciones de las EE al campo de la CE es la **auto-adaptación**. En la auto-adaptación se evolucionan tanto las variables del problema como los parámetros requeridos por la técnica.

2.3.3. Algoritmos genéticos

Los **Algoritmos Genéticos** (AG) fueron desarrollados por John Holland a principios de los 1960s en la Universidad de Michigan [19]. La finalidad de su trabajo consistió en explicar el proceso adaptativo en los sistemas naturales y diseñar un sistema artificial el cual pudiese retener los mecanismos más importantes de los sistemas naturales.

Un algoritmo genético incluye tres tipos de operadores [21]: *selección*, *recombinación* y *mutación*. El operador principal es la cruza sexual, el operador secundario es la mutación y utiliza selección probabilística. Un AG consta de cinco componentes básicos:

Algoritmo 2: Algoritmo básico de las estrategias evolutivas
Entrada: Población inicial, número de generaciones
Salida: Población evolucionada

```
begin
    Generar una población inicial
    Evaluar la población inicial
    repeat
        Aplicar un operador de mutación
        Aplicar Recombinación
        Evaluar población actual
        Realizar Selección
    until Cierta condición
end
```

- Una representación de las soluciones potenciales del problema.
- Una forma de crear una población inicial de soluciones posibles (normalmente un proceso aleatorio).
- Una función de evaluación que juegue el papel del ambiente, clasificando las soluciones con base en su aptitud.
- Operadores genéticos que alteren la composición de los descendientes que se producirán para las próximas generaciones.
- Valores para los diferentes parámetros que utiliza el algoritmo.

El algoritmo 3 corresponde a un algoritmo genético básico.

Un algoritmo genético simple "SGA" utiliza una representación binaria, emplea una selección proporcional, implementa el operador de recombinación llamado cruza de un punto y hace uso de una mutación por intercambio de bits. En la representación binaria, cada cromosoma es representado por una cadena de ceros y unos (ver figura 2.2). La selección proporcional elige individuos de acuerdo a su contribución de aptitud con respecto al total de la población. La cruza de un punto funciona eligiendo un número aleatorio n dentro del rango $[0, l-1]$ (donde l es el tamaño de la cadena binaria) y luego partiendo ambas cadenas binarias de padres en ese punto para crear dos hijos intercambiando las subcadenas de los padres formadas por los últimos $l - n$ bits (ver figura 2.3). En la mutación por intercambio de bits, cada gen (bit) tiene una probabilidad de intercambiar su valor (si el valor actual es 0 cambia a 1 y viceversa) (ver figura 2.4).

El SGA se puede modelar matemáticamente utilizando cadenas de Markov [22]. Se ha demostrado que para converger al óptimo global se requiere mantener intacto al mejor individuo de cada generación. A este proceso se le conoce como **elitismo** [23].

Algoritmo 3: Algoritmo genético básico
Entrada: Población inicial, número de generaciones, porcentaje de mutación,
Salida: Población evolucionada

```
begin
    Generar una población inicial
    Evaluar la población inicial
    repeat
        Seleccionar padres
        Aplicar un operador de recombinación
        Aplicar un operador de mutación
        Evaluar población actual
        Realizar selección
    until Cierta condición
end
```

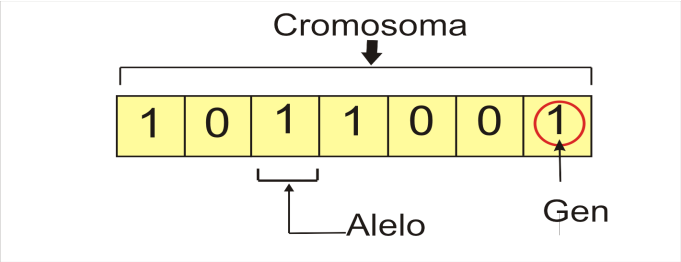

Figura 2.2: Ejemplo de la representación binaria de un cromosoma en un AG

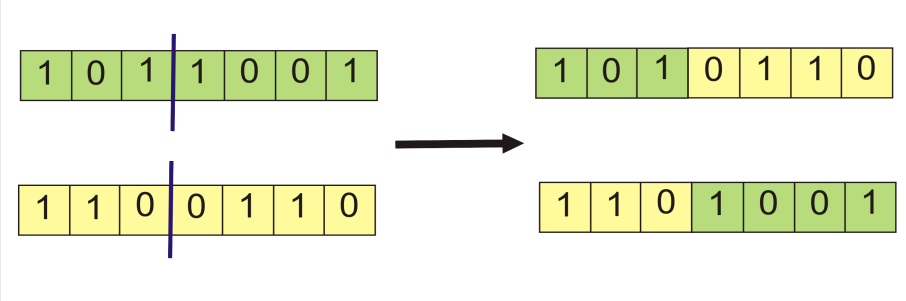

Figura 2.3: Ejemplo de cruza de un punto en un AG

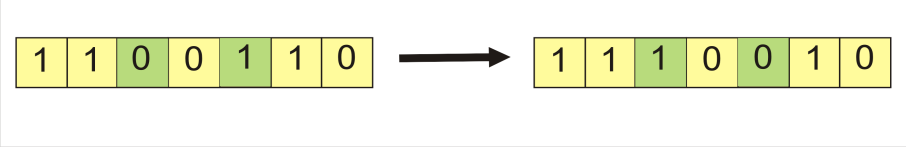

Figura 2.4: Mutación por intercambio de bits en un AG

2.3.4. Nuevas técnicas evolutivas alternativas

Una gran cantidad de nuevas técnicas se han diseñado en los últimos años. Éstas no solamente están basados en la evolución natural sino que algunas de ellas tratan de simular otros procesos o sistemas naturales. Como ejemplo podemos mencionar a la Evolución Diferencial (ED) [24], el Sistema Inmune Artificial (SIA) [25] y la Optimización mediante Cúmulos de Partículas [26] (PSO, por sus siglas en inglés). En la sección 2.5 se dan más detalles del funcionamiento del PSO, que es la técnica adoptada en esta tesis.

2.4. Algoritmos evolutivos multiobjetivos

La primera implementación de un algoritmo evolutivo multiobjetivo fue hecha por Schaffer en su tesis doctoral a mediados de los 1980s. Schaffer propuso el denominado *Vector Evaluated Genetic Algorithm* (VEGA) [27].

Dentro de los algoritmos evolutivos empleados para resolver problemas de optimización multiobjetivo podemos distinguir dos tipos de técnicas:

- Técnicas no basadas en la jerarquización de Pareto.
- Técnicas basadas en la jerarquización de Pareto.

Las *técnicas no basadas en la jerarquización de Pareto* no incorporan de manera directa el concepto de dominancia de Pareto. Uno de los algoritmos más populares es el VEGA. Como se mencionó anteriormente, este método fue propuesto por Schaffer en 1984 [27] y consiste básicamente en un algoritmo genético simple que usa un mecanismo de selección diferente. En cada generación, para un problema con k objetivos, se crean k subpoblaciones de tamaño M/k (M es el número de individuos o tamaño de población). Cada subpoblación utiliza una de las k funciones objetivo para la asignación de aptitud y se aplica selección proporcional. Después, las subpoblaciones son mezcladas con el fin de obtener una nueva población del mismo tamaño M. Por último el algoritmo genético aplica los operadores de recombinación y mutación. VEGA tiene varias desventajas. La principal es que se opone al concepto de dominancia de Pareto, pues un individuo que represente una buena solución compromiso será desechado si ésta no es la mejor para alguna de las funciones objetivo.

Como su nombre lo indica, las *técnicas basadas en jerarquización de Pareto* poseen un esquema de selección basado en el concepto de dominancia de Pareto. Históricamente, se pueden identificar dos generaciones. A continuación se dan algunas características y los algoritmos más representativos de cada una de ellas.

Los métodos que pertenen a la primera generación se caracterizan por el uso de jerarquización de Pareto (*Pareto Ranking*) y la compartición de aptitud (nichos). Entre los

algoritmos más representativos de la primera generación se encuentran: el *Multi-Objective Genetic Algorithm* (MOGA), el *Nondominated Sorting Genetic Algorithm* (NSGA) y el *Niched-Pareto Genetic Algorithm* (NPGA), los cuales se describen brevemente a continuación.

- **El Multi-Objective Genetic Algorithm** (MOGA) fue propuesto por Carlos Fonseca y Peter Fleming en 1993 [28]. En MOGA, la jerarquía de un individuo depende del número de individuos de la población que lo dominen. A todos los individuos no dominados se les asigna el valor de 1 en la clasificación, mientras que a los individuos que son dominados se les penaliza conforme a la densidad de población en la región correspondiente a la superficie de soluciones compromiso.

  En MOGA, la asignación de la aptitud se realiza como sigue:

  1. Ordenar la población de acuerdo a su jerarquía.
  2. Asignar una aptitud a los individuos interpolando desde el mejor individuo (el cual tiene jerarquía 1) hasta el peor (el cual tiene jerarquía $n \leq M$, donde M es el tamaño de la población) utilizando una función, por ejemplo, lineal.
  3. Promediar la aptitud de los individuos que poseen la misma jerarquía, de manera que éstos sean muestreados de la misma forma. De esta manera, se logra mantener constante la aptitud global de la población y con ello se mantiene una presión de selección adecuada.

- El **Nondominated Sorting Genetic Algorithm** (NSGA) fue propuesto por N. Srinivas y Kalyanmoy Deb en 1994 [29]. El NSGA se basa en el uso de varias capas de jerarquización de individuos. Antes de realizar la selección, la población es jerarquizada con base en su no dominancia: todos los individuos no dominados (primera capa) son jerarquizados en una categoría con un valor de aptitud falso (proporcional al tamaño de la población, con el fin de que estos individuos tengan la misma probabilidad para reproducirse). Para mantener diversidad en la población, se utiliza compartición de aptitud sobre los valores falsos asignados a los individuos jerarquizados. Posteriormente, este grupo jerarquizado es removido de la población (o ignorado) para obtener a los individuos no dominados de la población restante (segunda capa) a los que se les asigna una aptitud falsa menor que la aptitud asignada en la capa anterior. Después, este nuevo grupo de individuos es removido de la población para obtener la siguiente capa de individuos no dominados. Este proceso continúa hasta que todos los individuos de la población hayan sido clasificados. El NSGA utiliza el método del sobrante estocástico como mecanismo de selección. Como los individuos en el primer frente tiene el máximo valor de aptitud, se obtienen más copias de éstos en comparación con el resto de la población, permitiendo una mejor búsqueda en las regiones del frente de Pareto conocida hasta ese momento, provocando además

convergencia hacia estas regiones.

- El **Niched-Pareto Genetic Algorithm** (NPGA) fue propuesto por Jeffrey Horn et al. en 1993 [30]. El NPGA utiliza una selección mediante torneo binario con base en dominancia de Pareto. Dos individuos elegidos al azar son comparados contra un subconjunto de la población habiendo sólo dos resultados posibles. Si uno de ellos es dominado (por los individuos dentro del subconjunto de la población) y el otro individuo es no dominado, entonces el individuo no dominado gana el torneo. Los demás casos son empates, y en ellos se utiliza un esquema de compartición de aptitud para encontrar al ganador (es decir, el individuo que reside en la región menos densa del espacio de búsqueda es el ganador).

Los métodos pertenecientes a la segunda generación se caraterizan por ser elitistas. El elitismo puede ser introducido al utilizar una población secundaria (llamada también población externa), la cual siempre es actualizada y retiene a las soluciones no dominadas globales (o sea, con respecto a todas las generaciones transcurridas hasta el momento). Se hace notar que es posible utilizar una sola población en un algoritmo evolutivo multiobjetivo adoptando una selección $(\mu + \lambda)$ en la cual, como se mencionó en las estrategias evolutivas, los mejores μ individuos de la unión de los padres e hijos sobreviven a la siguiente generación, por lo que el elitismo está implícito. Dentro de los algoritmos elitistas más representativos encontramos: el *Strength Pareto Evolutionary Algorithm* (SPEA), la *Pareto Archived Evolution Strategy* (PAES), el *Pareto Envelope-based Selection Algorithm* (PESA), el *Micro-Genetic Algorithm* (μ-GA) y el *Non-dominated Sorting Genetic Algorithm II* (NSGA-II).

- El **Strength Pareto Evolutionary Algorithm** (SPEA) fue propuesto por Eckart Zitzler y Lothar Thiele en 1999 [31]. El SPEA utiliza un archivo externo que contiene soluciones no dominadas previamente encontradas (por lo cual el archivo externo recibe el nombre de conjunto externo no dominado). En cada generación, los individuos no dominados son copiados al conjunto externo no dominado. Para cada individuo en el conjunto externo se calcula un valor denominado *fortaleza* que es proporcional al número de soluciones a las que un individuo domina. La aptitud de un individuo se calcula con base en las fortalezas de todas las soluciones no dominadas que dominan al individuo. La asignación de aptitud del SPEA considera al mismo tiempo la aproximación al verdadero frente de Pareto y la distribución de las soluciones. En 2001, Zitzler et al. liberan SPEA2 [32], el cual tiene tres diferencias con respecto al SPEA:

  1. Incorpora una estrategia de grano fino para la asignación de la aptitud, la cual toma en cuenta el número de individuos que cada solución domina y el número de individuos que dominan a dicha solución.

2. Utiliza un estimador de densidad con base en los vecinos, el cual guía la búsqueda de manera más eficiente.
3. Posee una técnica de truncamiento del archivo la cual garantiza la preservación de soluciones de frontera.

- La **Pareto Archived Evolution Strategy** (PAES) fue propuesta por Joshua Knowles y David Corne en 1999 [33]. PAES consiste en una estrategia evolutiva (1+1) junto con un archivo histórico que guarda algunas de las soluciones no dominadas encontradas hasta el momento. Este archivo es utilizado como un conjunto de referencia por lo que cada individuo mutado es comparado contra éste. Para mantener diversidad el PAES utiliza un mecanismo que consiste en dividir el espacio objetivo de forma recursiva formando una especie de *rejilla adaptativa*. Es decir, cada solución es situada en una determinada celda dentro de una rejilla con base en los valores de sus funciones objetivo. Un mapa de esta rejilla se mantiene, indicando el número de soluciones que residen en cada celda. El único parámetro extra que necesita el PAES es el número de divisiones de la rejilla.

- El **Pareto Envelope-based Selection Algorithm** (PESA) fue propuesto por Corne et al. en el año 2000 [34]. El PESA consiste en dos poblaciones: una población pequeña interna y otra grande externa. Al igual que la PAES, el PESA adopta una hiper-rejilla en el espacio del fenotipo y la utiliza para mantener diversidad. El mecanismo de selección está basado en la medida de agrupamiento (densidad de población) y esta misma medida es utilizada para decidir qué soluciones ingresan al archivo externo, el cual contiene las soluciones no dominadas encontradas en el proceso.

- EL **Micro-Genetic Algorithm** (μ-GA) fue propuesto por Carlos A. Coello Coello y Gregorio Toscano en 2001 [35]. Un micro-algoritmo genético es un AG con una población muy pequeña (no más de cinco individuos) y un proceso de reinicialización. El μ-GA funciona de la siguiente manera: primero se genera una población con valores aleatorios. Dicha población aleatoria alimenta la memoria de la población la cual es dividida en dos partes: una porción reemplazable y otra porción no reemplazable. La porción no reemplazable de la memoria de la población nunca cambia durante todo el proceso y proporciona la diversidad que necesita el algoritmo. Por otro lado, la porción reemplazable cambia después de cada ciclo del μ-GA. Al inicio de cada ciclo, el μ-GA toma individuos de ambas porciones de población (con una cierta probabilidad) de manera que exista una población compuesta por individuos aleatorios (porción no reemplazable) e individuos evolucionados (porción reemplazable). Durante cada ciclo en el μ-GA se aplican los operadores genéticos convencionales. Una vez teminado un ciclo, se eligen dos soluciones no dominadas de la población final (suponiendo que existen dos o más soluciones no dominadas y en caso de que sólo exista una solución no dominada, ésta es seleccionada) y éstas son comparadas

con las soluciones contenidas en la memoria externa (inicialmente vacía). Si alguna de las dos soluciones (o ambas) siguen siendo no dominadas (después de la comparación) entonces se incluye(n) en la memoria externa. Las soluciones dominadas en la memoria externa son eliminadas de manera que dicha memoria se convierta en el archivo histórico de las soluciones no dominadas. La mayor desventaja del μ-GA es que requiere un número elevado de parámetros.

- El **Non-dominated Sorting Genetic Algorithm II** (NSGA-II) es la versión mejorada del NSGA y fue desarrollado por Deb et al. en el año 2000 [36]. El NSGA-II utiliza una población de individuos jerarquizados y ordenados de acuerdo a su nivel de no dominancia. Éste aplica operadores evolutivos para crear una población de hijos (de igual número que la población de padres) para después combinarlos y particionarlos en frentes. El NSGA-II emplea un estimador de densidad en la selección con base en la distancia en que se encuentra un individuo con respecto a sus vecinos (*crowding distance*). Dicho estimador de densidad es utilizado para mantener diversidad en el frente. A diferencia de las propuestas anteriones, el NSGA-II no utiliza un archivo externo sino que utiliza una selección ($\mu + \lambda$).

2.5. Optimización mediante cúmulos de partículas

En 1995, Russell C. Eberhart y James Kennedy propusieron una heurística llamada "optimización mediante cúmulos de partículas" (*Particle Swarm Optimization*-PSO). Esta técnica está inspirada en el comportamiento social de las bandadas de aves. La idea es simular los movimientos de una parvada de aves al intentar encontrar comida [26].

En PSO, las soluciones son denominadas *partículas*, las cuales vuelan o se mueven en el espacio de búsqueda lidereadas (guiadas) por la partícula del vecindario que ha encontrado la mejor solución hasta ese momento. Cada partícula almacena las coordenadas (posición) en el espacio del problema, las cuales son asociadas con la mejor solución (aptitud) llamado *pbest* (personal best). Las partículas pueden estar conectadas en diversas topologías creando vecindarios. El mejor valor obtenido por una partícula entre sus vecinos se denomina *lbest* (local best). A la mejor partícula de toda la población se le denomina *gbest* (global best). Al conjunto de todas las partículas se le denomina *cúmulo*.

La técnica del PSO al igual que los AEs utiliza una población de posibles soluciones aleatorias, las cuales irán evolucionando conforme transcurren las generaciones hasta encontrar una solución óptima (o una buena aproximación). La aptitud de una partícula es determinada por la función definida para el problema. A diferencia de los AEs, el algoritmo original del PSO no implementa operadores evolutivos como la cruza o la mutación. Además, éste permite que los individuos se beneficien de experiencias pasadas.

El algoritmo del PSO real actualiza la posición de cada partícula mediante una velocidad $\vec{v_i}$, la cual es un vector que se agrega a la posición actual de la partícula para desplazarla de un punto a otro tras cada iteración del algoritmo. La dirección del movimiento es una función con base en la posición actual de la partícula, su velocidad actual, así como su mejor posición previa y la mejor posición previa de la mejor partícula en el vecindario:

$$v_{id} = v_{id} + c_1 * rand_1 * (pb_{id} - x_{id}) + c_2 * rand_2 * (p_{lbd} - x_{id}) \tag{2.1}$$

$$x_{id} = x_{id} + v_{id} \tag{2.2}$$

donde:
c_1 y c_2 son constantes positivas,
$rand_1$ y $rand_2$ son dos números aleatorios en el rango $[0, 1]$,
$x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ representa la i-ésima partícula,
$pb_i = (p_{i1}, p_{i2}, ..., p_{iD})$ representa la mejor posición previa de la partícula (pbest),
lb representa el índice de la mejor partícula del vecindario(lbest) y
$V_i = (v_{i1}, v_{i2}, ..., v_{iD})$ representa la velocidad de la partícula i.

De la ecuación (2.1) utilizada para calcular la velocidad de una partícula se pueden observar dos aspectos importantes:

1. La parte "cognitiva" al retener y utilizar la mejor posición encontrada hasta el momento. Ésta representa la parte en la cual la partícula aprende a partir de su propia experiencia durante el vuelo.

2. La parte "social" al seleccionar un líder en el vecindario y guiar a la partícula hacia la mejor posición que éste ha encontrado hasta el momento. Es decir, existe una colaboración entre las partículas.

En el algoritmo 4 se muestra el algoritmo original del PSO. Éste requiere como entrada el número de partículas que constituyen el cúmulo y el número de iteraciones.

En PSO, se utilizan comúnmente dos topologías: la versión local y la versión global. En la versión local, como se mencionó anteriormente, las partículas están conectadas en diversas topologías (por ejemplo ver figuras 2.5, 2.6 y 2.7) creando vecindarios. Un vecindario es un subconjunto del cúmulo. Para cada partícula y en cada iteración, se elige un líder el cual debe de ser la mejor partícula obtenida dentro del vecindario hasta el momento; dicho líder es denominado *lbest*. Por lo tanto, la velocidad de cada partícula se actualiza de acuerdo a su *pbest* y al *lbest*.

Algoritmo 4: Pseudocódigo del PSO
Entrada: Número de partículas, número de generaciones
Salida: Mejor solución

begin
Inicializar una población de partículas con posiciones y velocidades aleatorias.
repeat
for $i = 1$ *to num_particulas* **do**
if $G(x_i) > G(pb_i)$ **then**
for $d = 1$ *to num_dimensiones* **do**
$pb_{id} = x_{id}$; // pb_{id} es el mejor hasta el momento
end
end
$g = i$; // Valor arbitrario
for j =*indices_de_los_vecinos* **do**
if $G(pb_j) > G(pb_g)$ **then**
$g = j$; //g es el índice del mejor individuo en el vecindario
end
end
for $d = 1$ *to num_dimensiones* **do**
$v_{id} = v_{id} + c_1 * rand_1 * (pb_{id} - x_{id}) + c_2 * rand_2 * (pb_{gd} - x_{id})$
$x_{id} = x_{id} + v_{id}$
end
end
until *num. máximo de generaciones*
Reportar la mejor solución.
end

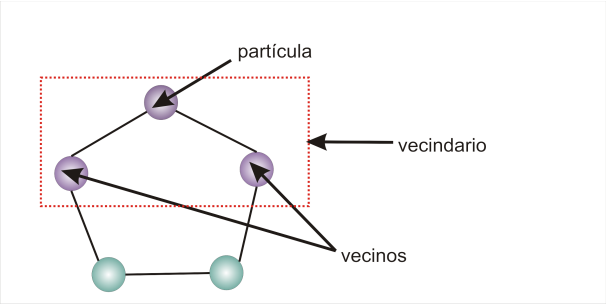

Figura 2.5: Ejemplo de una topología de anillo. Cada partícula está conectada con las dos partículas adyacentes.

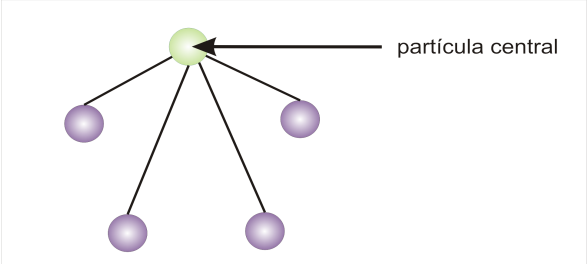

Figura 2.6: Ejemplo de una topología de estrella. Cada partícula está conectada sólo con la partícula central, la cual es el líder.

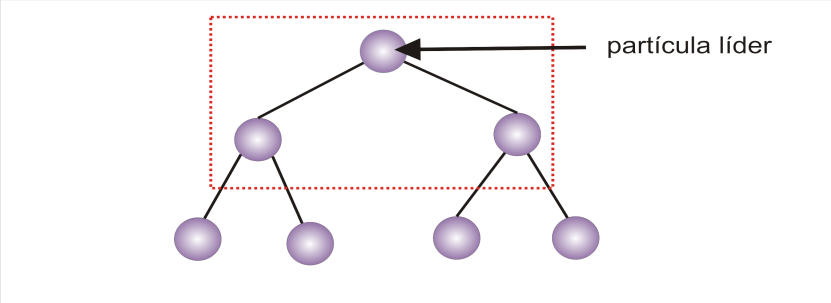

Figura 2.7: Ejemplo de una topología de árbol. Todas las partículas están colocadas formando un árbol. Cada partícula sigue a la partícula que se encuentra en el nivel inmediato superior.

En la versión global, cada partícula vuela a través del espacio de búsqueda con una velocidad que es ajustada de acuerdo a la mejor posición encontrada hasta el momento por todas las partículas. La versión global del PSO puede ser considerada una versión local del PSO siendo el vecindario de cada partícula todo el cúmulo (ver figura 2.8).

Muchos investigadores han trabajado en mejorar el rendimiento del PSO, abordando varias direcciones. Shi y Eberhart en [37] introducen un parámetro llamado inercia (w) en la ecuación de velocidad del algoritmo original del PSO (ver ecuación 2.3). La inercia es utilizada para balancear la búsqueda global y local. Una mayor inercia es más apropiada

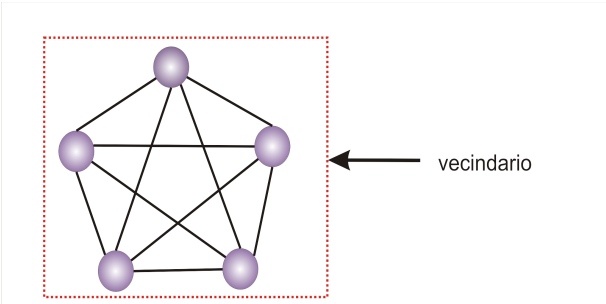

Figura 2.8: Ejemplo de una topología donde las partículas están todas conectadas con las demás. El vecindario que se forma es el cúmulo.

para la búsqueda global, a diferencia de una pequeña inercia que facilita la búsqueda local.

$$v_{id} = (w * v_{id}) + c_1 * rand_1 * (pb_{id} - x_{id}) + c_2 * rand_2 * (pb_{gd} - x_{id}) \qquad (2.3)$$

Por su parte, Kennedy y Mendes en 2002 realizaron un estudio del rendimiento del PSO con diferentes tipos de topologías, reportando que vecindarios pequeños pueden tener mejor rendimiento en problemas complejos, mientras que vecindarios grandes tienen un mejor rendimiento para problemas simples [38].

El algoritmo del PSO ha sido aplicado satisfactoriamente en varias áreas de investigación y en aplicaciones del mundo real. Además, existe evidencia empírica de la eficiencia y efectividad del PSO en una amplia gama de problemas [39].

Capítulo 3

Propuesta de Optimización Mediante Cúmulos de Partículas en Espacios Restringidos

Una gran variedad de Algoritmos Evolutivos (AEs) han sido usados para resolver diferentes tipos de problemas de optimización. Sin embargo, los AEs son técnicas de búsqueda sin restricciones y por lo tanto, requieren de un mecanismo adiccional que incorpore las restricciones de un problema en su función de aptitud [8]. Las funciones de penalización son las técnicas más comúnmente adoptadas para incorporar dichas restricciones en los AEs. No obstante, estas funciones de penalización tienen ciertas desventajas dentro de las cuales la más importante es que requieren un ajuste fino del factor de penalización (el cual depende del problema) ya que en ambos casos, penalizaciones bajas y altas pueden alterar el proceso de optimización [8].

En este capítulo se propone un algoritmo para resolver problemas de optimización con restricciones basado en el algoritmo de optimización mediante cúmulos de partículas. Nuestra propuesta utiliza un tamaño de población muy pequeño. El algoritmo propuesto incorpora un mecanismo para manejo de restricciones que permite seleccionar líderes de forma tal que las soluciones oscilen dentro y fuera de la zona factible.

3.1. Trabajo relacionado

Cuando se resuelven problemas de optimización con restricciones utilizando Algoritmos Evolutivos, se debe de tener sumo cuidado al incorporar restricciones a la función de aptitud. Resulta de vital importancia mantener diversidad en la población, así como hacer que las soluciones oscilen dentro y fuera de la zona factible [9, 40]. Muchos estudios muestran que a pesar de su popularidad las funciones de penalización (incluso haciendo uso de factores de penalización dinámicos) tienden a tener dificultades al tratar con espacios de búsqueda altamente restringidos y con problemas en los cuales las restricciones están ac-

tivas en el óptimo [8, 41, 42]. Motivados por este hecho, diversas técnicas de manejo de restricciones han sido propuestas para algoritmos evolutivos [43, 8].

Existe relativamente poco trabajo relacionado con la incorporación de restricciones en el algoritmo del PSO, a pesar del hecho de que la mayoría de las aplicaciones del mundo real tienen restricciones. En trabajos previos, algunos investigadores han asumido la posibilidad de poder generar en forma aleatoria soluciones factibles para alimentar la población del algoritmo del PSO [44, 45]. El problema con esta propuesta es que en algunos casos, el costo computacional puede ser sumamente elevado. Por ejemplo, en algunas de las funciones de prueba usadas en esta tesis aún generando un millón de soluciones aleatorias, no se logra producir una sola solución factible. Resulta evidente que este tipo de propuestas, al implicar estos elevados costos computacionales, están destinadas a ser prohibitivas en aplicaciones del mundo real.

Otras propuestas son aplicables sólo a ciertos tipos de restricciones, por ejemplo en [46] el algoritmo sólo puede ser resolver problemas con restricciones lineales. Algunas propuestas más, emplean mecanismos relativamente simples que seleccionan a un líder con base en la cercanía de una partícula a la región factible. Éstas adoptan también un operador de mutación con el objetivo de mantener diversidad durante la búsqueda (por ejemplo [47]). Sin embargo, queda mucho por hacer en lo relativo al uso del algoritmo del PSO para resolver problemas con restricciones.

3.2. Descripción de la propuesta

Nuestra propuesta está basada en la versión local del algoritmo del PSO, puesto que hay evidencia que este modelo es menos propenso a quedar atrapado en mínimos locales [39]. A pesar de la existencia de varias topologías [38], en este trabajo se opta por generar una topología cuyos vecindarios se forman aleatoriamente. Nuestra propuesta utiliza también un proceso de reinicialización para mantener diversidad en la población e incorpora un operador de mutación con la finalidad de aumentar la capacidad de búsqueda del algoritmo del PSO (ver Algoritmo 5).

Dado que nuestra propuesta utiliza tamaños de población muy pequeños (cinco partículas), decidimos llamarlo *Micro-PSO*, ya que es análoga al micro algoritmo genético (μ-GA) que ha sido usado durante varios años (ver por ejemplo [48, 49]). Los detalles del mecanismo adoptado para manejo de restricciones junto con el proceso de reinicialización y el operador de mutación incorporado son descritos a continuación.

Algoritmo 5: Pseudocódigo del Micro-PSO para espacios restringidos.
Entrada: Número de partículas, número de generaciones, número de vecinos, número de generaciones de reinicialización, número de partículas de reinicialización, porcentaje de mutación.
Salida: Mejor solución

```
begin
    for i = 1 to Número de partículas do
        Inicializar posición y velocidad de manera aleatoria;
        Inicializar los vecinos (aleatorios);
    end
    cont = 1;
    repeat
        if cont == Número de generaciones de reinicialización then
            Proceso de reinicialización;
            cont = 1;
        end
        Calcula el valor de aptitud G(x_i);
        for i = 1 to Número de partículas do
            if G(x_i) > G(xpb_i) then
                for d = 1 to Número de dimensiones do
                    xpb_id = x_id; // xpb_i es la mejor posición de la partícula;
                end
            end
            Selecciona el mejor vecino del vecindario lb_i;
            for d = 1 to Número de dimensiones do
                w = rand(); // Número aleatorio (0,1];
                v_id = w × v_id + c1 r1 (pbx_id − x_id) + c2 r2 (lb_id − x_id); x_id = x_id + v_id;
            end
        end
        cont =cont +1;
        Aplica operador de mutación;
    until Número máximo de generaciones ;
    Reportar la mejor solución encontrada.
end
```

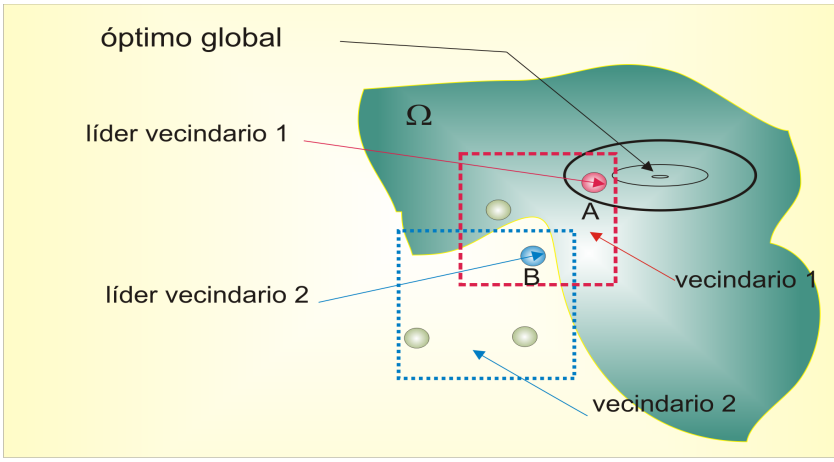

Figura 3.1: Caso hipotético de la selección de líderes utilizando el mecanismo de manejo de restricciones.

3.2.1. Mecanismo para manejo de restricciones

En este trabajo se adopta el mecanismo propuesto en [47] para seleccionar líderes. Este mecanismo está basado en dos elementos: factibilidad y el valor de aptitud de la partícula. Cuando dos partículas factibles son comparadas, la partícula que tenga el valor de aptitud más alto gana. Si una de las partículas no es factible y la otra es factible, entonces la partícula factible gana. Cuando dos partículas no factibles se comparan, la partícula que tenga el valor de aptitud más pequeño gana. La idea es seleccionar líderes los cuales aún cuando pueden ser no factibles, posean soluciones que yazcan cerca de la región factible.

Se utiliza la siguiente ecuación (3.1) para asignar aptitud a una solución:

$$Aptitud(\overrightarrow{x}) = \begin{cases} f_i(\overrightarrow{x}) & \text{si la solución es factible} \\ \sum_{j=1}^{n} g_j(\overrightarrow{x}) + \sum_{k=1}^{p} |h_k(\overrightarrow{x})| & \text{de otra forma} \end{cases} \tag{3.1}$$

En la figura 3.1 podemos observar un caso hipotético del funcionamiento del mecanismo de restricciones. Existe un óptimo global en una zona factible Ω. La partícula A es el líder del vecindario 1 (que consta de tres partículas), ya que es factible y es mejor solución que la otra factible. En el caso del vecindario 2, se observa que ninguna de las partículas que lo componen es factible, pero el mecanismo de restricciones permite seleccionar como líder a la partícula B. Esto se debe a que ésta es la partícula más cercana a la zona factible de las tres.

3.2.2. Proceso de reinicialización

El uso de tamaños de población pequeños acelera la pérdida de diversidad en cada iteración del algoritmo y, por consiguiente, su uso no es común en la literatura especializada. Sin embargo, en la literatura de los algoritmos genéticos, es sabido que teóricamente es posible el uso de tamaños de población muy pequeños (no más de cinco individuos) si se implementa un proceso de reinicialización apropiado [48, 49].

En este trabajo se incorpora un proceso de reinicialización tomando como base los procesos implementados en la literatura de los micro algoritmos genéticos [49, 50]. El mecanismo es el siguiente: después de un cierto número de iteraciones (generaciones de reemplazo), el cúmulo es ordenado con base en el valor de aptitud pero posicionando las soluciones factibles primero. Luego, se reemplazan las **rp** partículas (partículas de reemplazo) por partículas generadas aleatoriamente (posición y velocidad), pero a las **rp** partículas se les permite retener su mejor posición (solución) encontrada hasta el momento (*pbest*). La motivación para combinar partículas evolucionadas y partículas generadas aleatoriamente es evitar la convergencia prematura.

Para nuestro algoritmo, al sólo utilizar cinco partículas, es de capital importancia que las partículas de reemplazo (**rp**) retengan su mejor solución. Si bien éstas, en el proceso de reinicialización, generan nuevas soluciones aleatorias que pueden caer en zonas no exploradas dentro del espacio de búsqueda (de hecho eso es precisamente lo que se desea), si se reinicializara también su mejor posición (por una posición aleatoria), la partícula podría tardar demasiado en converger a una buena solución, volviendo a caer en las **rp** partículas en la siguiente generación de reemplazo.

3.2.3. Operador de mutación

Aunque el algoritmo original del PSO no tiene un operador de mutación, la incorporacion de dicho operador es una práctica relativamente común en la literatura especializada. La motivación principal de incorporar este operador es incrementar el rendimiento del PSO como optimizador, al aumentar la capacidad de exploración de esta heurística [51]. En este trabajo se implementó el operador de mutación desarrollado por Michalewicz para los algoritmos genéticos [52]. Vale la pena señalar que este operador de mutación ha sido usado previamente en el algoritmo del PSO, pero en el contexto de problemas de optimización multimodales sin restricciones [53]. Este operador varía el valor de una solución sumando o restando una magnitud delta (Δ), la cual se calcula dependiendo del número de la iteración actual. Este operador permite que al principio de la búsqueda se pueden producir cambios grandes, los cuales se vuelven cada vez más pequeños conforme se aproxima al final de la búsqueda.

En el micro-PSO, el operador de mutación se aplica en la posición de la partícula y en todas sus dimensiones:

$$x_{id} = \begin{cases} x_{id} + \Delta(t, U - x_{id}) & \text{si } R = 0 \\ x_{id} - \Delta(t, x_{id} - L) & \text{si } R = 1 \end{cases} \tag{3.2}$$

donde:
t es el ńumero de iteración actual,
U es valor del límite superior de la dimensión de la partícula,
L es el valor del límite inferior de la dimensión de la partícula,
R es un bit generado aleatoriamente (con un 50 % de probabilidad) y,
la función $\delta(t, y)$ regresa un valor dentro del rango $[0, y]$.

La función $\delta(t, y)$ se define como:

$$\Delta(t, y) = y \times (1 - r^{1-(\frac{t}{T})^b}) \tag{3.3}$$

donde:
r es un número generado aleatoriamente con una distribución uniforme dentro del rango $[0, 1]$,
T es el número máximo de iteraciones y, b es un parámetro que se puede ajustar el cual define el grado de no-uniformidad del operador.

En este trabajo, el valor del parámetro b es fijado en 5, tal como se sugiere en [52].

3.3. Experimentos y resultados

Para evaluar el rendimiento del micro-PSO para espacios restringidos, se utilizaron trece funciones de prueba descritas en [42] (y también descritas en el apéndice A, en la página 67). Estas funciones de prueba contienen características que las hacen difíciles de resolver utilizando algoritmos evolutivos. En la tabla 3.1 se proporciona un resumen de algunas de las características de las funciones de prueba. En la segunda columna podemos observar la dimensionalidad (número de variables) del problema. En la tercera columna se muestra el tipo de la función objetivo a optimizar. ρ nos proporciona una estimado de la dificultad para generar una solución factible de manera aleatoria. Para obtener ρ, se generan un millón de soluciones aleatorias uniformemente distribuidas en el espacio de búsqueda y se determinan de entre ellas las soluciones factibles, por lo que ρ es el porcentaje de soluciones factibles encontradas. En la tabla 3.1 también se muestran la cantidad de restricciones de desigualdad lineales (LI) y no lineales (NI), así como el número de restricciones de igualdad lineales (LE) y no lineales (NE) que poseen las funciones de prueba.

Se realizaron cincuenta ejecuciones independientes para cada función de prueba y se compararon los resultados obtenidos por nuestra propuesta con los de tres algoritmos representativos del estado del arte en el área: el ordenamiento estocástico (SR *Stochastic Ranking*) [42], la estrategia evolutiva multimiembro simple (SMES *Simple Multimembered Evolution Strategy*) [40] y el mecanismo de manejo de restricciones para PSO (CHM-PSO *Constraint-Handling Mechanism for PSO*) [47].

El ordenamiento estocástico (SR) utiliza una estrategia evolutiva multimiembro junto con una función de penalización estática y una selección basada en un proceso de ordenamiento estocástico. En el ordenamiento, el componente estocástico permite la asignación de prioridad a las soluciones no factibles en pocas ocasiones durante el proceso de selección. La idea es balancear la influencia de la función objetivo y la función de penalización. Esta técnica requiere un parámetro definido por el usuario llamado Pf el cual determina el balance [42].

La estrategia evolutiva multimiembro simple (SMES) está basada en un estrategia evolutiva del tipo $(\mu+\lambda)$. Ésta tiene tres mecanismos principales: un mecanismo de diversidad, un operador de recombinación y un tamaño de paso dinámico que define la suavidad de los movimientos ejecutados por la estrategia evolutiva [40].

El mecanismo de manejo de restricciones para PSO (CHM-PSO) utiliza un criterio simple basado en la cercanía de una partícula a la región factible para seleccionar al líder. Esta técnica implementa una versión global del algoritmo del PSO e incorpora una turbulencia, es decir un operador de mutación [47].

En todos los experimentos se utilizaron los siguientes parámetros:

- W = Número generado aleatoriamente por una distribución uniforme en el rango [0,1].
- C1 = C2 = 1.8.
- Tamaño de población = 5 partículas;
- Número de generaciones = 48,000;
- Número de generaciones de reemplazo (gr) = 100;
- Número de partículas de reemplazo (pr) = 2;
- Número de vecinos = 3;
- Porcentaje de mutación = 0.1;
- $\epsilon = 0.0001$, excepto para g04, g05, g06, g07, en las cuales utilizamos $\epsilon = 0.00001$.

Tabla 3.1: Características de las funciones de prueba utilizadas.

FP	Número variables	$f(\vec{x})$	ρ	LI	NI	LE	NE
g01	13	cuadrática	0.0003 %	9	0	0	0
g02	20	no lineal	99.9973 %	1	1	0	0
g03	10	no lineal	0.0026 %	0	0	0	1
g04	5	cuadrática	27.0079 %	0	6	0	0
g05	4	no lineal	0.0000 %	2	0	0	3
g06	2	no lineal	0.0057 %	0	2	0	0
g07	10	cuadrática	0.0000 %	3	5	0	0
g08	2	no lineal	0.8581 %	0	2	0	0
g09	7	no lineal	0.5199 %	0	4	0	0
g10	8	lineal	0.0020 %	3	3	0	0
g11	2	cuadrática	0.0973 %	0	0	0	1
g12	3	cuadrática	4.7697 %	0	9^3	0	0
g13	5	no lineal	0.0000 %	0	0	1	2

Las estadísticas de los resultados del micro-PSO para espacios restringidos están resumidas en la tabla 3.2. Nuestra propuesta es capaz de llegar al óptimo global en la mayoría de las funciones de prueba, con excepción de g10. En la tabla 3.3 se comparan el tamaño de población, el valor de tolerancia y el número de evaluaciones de la función objetivo de nuestra propuesta con respecto al SR, SMES, y CHM-PSO.

Al comparar nuestra propuesta con respecto al SR, se puede observar que la nuestra encontró una mejor solución para la función de prueba g02 y resultados muy similares en otros once problemas, excepto para g10 en la cual nuestra propuesta no encuentra un buen resultado (ver tabla 3.4). Sin embargo, se hace notar que el SR realiza 350,000 evaluaciones de la función objetivo y el Micro-PSO que se propone hace uso sólo de 240,000.

Es importante la comparación del micro-PSO contra el CHM-PSO ya que ambos están basados en la misma heurística. El Micro-PSO encuentra mejores soluciones para las funciones de prueba g02, g03, g04, g06, g07, g09 y g13 (ver tabla 3.5). Así mismo, el micro-PSO encuentra los mismos resultados o similares en otros cinco problemas (excepto para g10). Nótese de nuevo que el CHM-PSO realiza 340,000 evaluaciones de la función objetivo mientras que nuestra propuesta realiza 240,000, esto es, aproximadamente $29\,\%$ menos de evaluaciones. Cabe mencionar que el CHM-PSO es una de las mejores técnicas basadas en PSO para manejo de restricciones que se conocen a la fecha.

Cuando se compara nuestra propuesta contra el SMES, se observa que el Micro-PSO encuentra mejores soluciones para las funciones de prueba g02 y g09 y las mismas soluciones o similares en otros diez problemas, con excepción de g10 (ver tabla 3.6). Ambas

Tabla 3.2: Resultados obtenidos por el micro-PSO.

Resultados estadísticos del Micro-PSO						
FP	Óptimo	Mejor	Promedio	Media	Peor	Desv. Est.
g01	**-15.000**	**-15.0001**	-13.2734	-13.0001	-9.7012	1.41E+00
g02	**0.803619**	**0.803620**	0.777143	0.778481	0.711603	1.91E-02
g03	**1.000**	**1.0004**	0.9936	1.0004	0.6674	4.71E-02
g04	**-30665.539**	**-30665.5398**	**-30665.5397**	**-30665.5398**	-30665.5338	6.83E-04
g05	5126.4981	5126.6467	5495.2389	5261.7675	6272.7423	4.05E+02
g06	-6961.81388	-6961.8371	-6961.8370	-6961.8371	-6961.8355	2.61E-04
g07	24.3062	24.3278	24.6996	24.6455	25.2962	2.52E-01
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0.00E+00
g09	**680.630**	**680.6307**	680.6391	680.6378	680.6671	6.68E-03
g10	7049.250	7090.4524	7747.6298	7557.4314	10533.6658	5.52E+02
g11	0.750	0.7499	0.7673	0.7499	0.9925	6.00E-02
g12	**1.000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.00E+00
g13	0.05395	0.05941	0.81335	0.90953	2.44415	3.81E-01

Tabla 3.3: Comparación del tamaño de población, el valor de tolerancia y el número de evaluaciones de la función objetivo del Micro-PSO con respecto al SR, CHM-PSO y SMES.

Técnica	Tamaño de población	Tolerancia(ϵ)	Número de evaluaciones de la función objetivo
SR	200	0.001	350,000
CHMPSO	40	-	340,000
SMES	100	0.0004	240,000
Micro-PSO	5	0.0001	240,000

Tabla 3.4: Comparación del micro-PSO con respecto al ordenamiento estocástico (SR).

		Mejor solución		Solución promedio		Peor solución	
FP	Óptimo	Micro-PSO	SR	Micro-PSO	SR	Micro-PSO	SR
g01	**-15.0000**	**-15.0001**	**-15.000**	-13.2734	-15.000	-9.7012	-15.000
g02	**0.803619**	**0.803620**	**0.803515**	0.777143	0.781975	0.711603	0.726288
g03	**1.0000**	1.0004	**1.000**	0.9936	1.000	0.6674	1.000
g04	**-30665.539**	**-30665.539**	**-30665.539**	-30665.539	-30665.539	-30665.534	-30665.539
g05	**5126.4981**	5126.6467	**5126.497**	5495.2389	5128.881	6272.7423	5142.472
g06	**-6961.81388**	-6961.8371	**-6961.814**	-6961.8370	-6875.940	-6961.8355	-6350.262
g07	**24.3062**	24.3278	**24.307**	24.6996	24.374	25.2962	24.642
g08	**0.095825**	**0.095825**	**0.095825**	0.095825	0.095825	0.095825	0.095825
g09	**680.630**	**680.6307**	**680.630**	680.6391	680.656	680.6671	680.763
g10	**7049.250**	7090.4524	**7054.316**	7747.6298	7559.192	10533.6658	8835.655
g11	**0.750**	0.7499	**0.750**	0.7673	0.750	0.9925	0.750
g12	**1.000**	**1.0000**	**1.000**	1.0000	1.000	1.0000	1.000
g13	**0.05395**	0.05941	**0.053957**	0.81335	0.067543	2.44415	0.216915

Tabla 3.5: Comparación del Micro-PSO con respecto al mecanismo de manejo de restricciones para PSO (CHM-PSO).

		Mejor solución		Solución promedio		Peor solución	
FP	Óptimo	Micro-PSO	CHM-PSO	Micro-PSO	CHM-PSO	Micro-PSO	CHM-PSO
g01	**-15.0000**	**-15.0001**	**-15.000**	-13.2734	-15.000	-9.7012	-15.000
g02	**0.803619**	**0.803620**	0.803432	0.777143	0.790406	0.711603	0.755039
g03	**1.000**	**1.0004**	1.004720	0.9936	1.003814	0.6674	1.000249
g04	**-30665.539**	**-30665.539**	-30665.500	-30665.539	-30665.500	-30665.534	-30665.500
g05	**5126.4981**	5126.6467	**5126.6400**	5495.2389	5461.08133	6272.7423	6104.7500
g06	**-6961.8138**	-6961.837	**-6961.810**	-6961.837	-6961.810	-6961.835	-6961.810
g07	**24.3062**	**24.3278**	24.3511	24.6996	24.35577	25.2962	27.3168
g08	**0.095825**	**0.095825**	**0.095825**	0.095825	0.095825	0.095825	0.095825
g09	**680.6300**	**680.6307**	680.638	680.6391	680.85239	680.6671	681.553
g10	**7049.2500**	7090.4524	**7057.5900**	7747.6298	7560.04785	10533.6658	8104.3100
g11	**0.7500**	**0.7499**	**0.7499**	0.7673	0.7501	0.9925	0.75288
g12	**1.0000**	**1.0000**	**1.000**	1.0000	1.000	1.0000	1.000
g13	**0.05395**	**0.05941**	0.068665	0.81335	1.71642	2.44415	13.6695

técnicas realizan 240,000 evaluaciones de la función objetivo.

Tabla 3.6: Comparación del Micro-PSO con respecto al la estrategia evolutiva multimiembro simple (SMES).

		Mejor solución		Solución promedio		Peor solución	
TF	Optimal	Micro-PSO	SMES	Micro-PSO	SMES	Micro-PSO	SMES
g01	**-15.0000**	**-15.0001**	**-15.000**	-13.2734	-15.000	-9.7012	-15.000
g02	**0.803619**	**0.803620**	0.803601	0.777143	0.785238	0.711603	0.751322
g03	**1.0000**	1.0004	**1.000**	0.9936	1.000	0.6674	1.000
g04	**-30665.5390**	**-30665.5398**	**-30665.539**	-30665.5397	-30665.539	-30665.5338	-30665.539
g05	**5126.4980**	5126.6467	**5126.599**	5495.2389	5174.492	6272.7423	5304.167
g06	**-6961.8140**	-6961.8371	**-6961.814**	-6961.8370	-6961.284	-6961.8355	-6952.482
g07	**24.3062**	**24.3278**	**24.327**	24.6996	24.475	25.2962	24.843
g08	**0.095825**	**0.095825**	**0.095825**	0.095825	0.095825	0.095825	0.095825
g09	**680.6300**	**680.6307**	680.632	680.6391	680.643	680.6671	680.719
g10	**7049.2500**	7090.4524	**7051.903**	7747.6298	7253.047	10533.6658	7638.366
g11	**0.7500**	0.7499	**0.75**	0.7673	0.75	0.9925	0.75
g12	**1.0000**	**1.0000**	**1.000**	1.0000	1.000	1.0000	1.000
g13	**0.05395**	0.05941	**0.053986**	0.81335	0.166385	2.44415	0.468294

Capítulo 4

Propuesta de Optimización Multiobjetivo Mediante Cúmulos de Partículas

El algoritmo del PSO fue originalmente empleado para balancear pesos en redes neuronales y posteriormente fue utilizado en problemas de optimización mono-objetivo [39]. Al resolver problemas de optimización multiobjetivo con el algoritmo del PSO se debe modificar el esquema de selección de líderes, y agregar un mecanismo para preservar diversidad en el cúmulo.

Una forma de extender el algoritmo del PSO como optimizador multiobjetivo es incorporando un esquema de jerarquización de Pareto. Además, es deseable mantener un registro histórico de las mejores soluciones encontradas. En éste se van guardando las soluciones no dominadas generadas (elitismo) en cada iteración.

En este capítulo se propone un algoritmo para resolver problemas de optimización multiobjetivo basado en el algoritmo de optimización mediante cúmulos de partículas. Al igual que el micro-PSO para espacios restringidos descrito en el capítulo anterior, esta propuesta continúa utilizando tamaños de poblaciones muy pequeños.

4.1. Trabajo relacionado

En [54] se propone una taxonomía para clasificar a los Optimizadores Multiobjetivo basados en PSO (MOPSOs por sus siglas en inglés):

- Técnicas agregativas
- Orden lexicográfico

- Técnicas que utilizan sub-poblaciones
- Técnicas basadas en dominancia de Pareto
- Técnicas combinadas
- Otras propuestas

Como nuestra propuesta está basada en la dominancia de Pareto, en el estado del arte presentado a continuación nos limitaremos a describir los algoritmos que caen dentro de esta clasificación.

En un documento no publicado que data de 1999, Moore y Chapman presentaron el primer PSO para problemas multiobjetivo basado en la dominancia de Pareto [55]. En el documento se enfatiza la importancia de efectuar búsquedas individuales y grupales. En esta propuesta, el *pbest* de una partícula es una lista de los soluciones no dominadas que ésta ha ido encontrando durante el vuelo. Para seleccionar líderes, el *pbest* de una partícula es seleccionado aleatoriamente de la lista. Este algoritmo no incorpora esquema alguno para mantener diversidad.

Fieldsend y Singh [56] propusieron en 2002 un algoritmo que utiliza un archivo externo elitista. En el archivo se adopta una estructura de datos especial denominada "árbol dominado" para almacenar a las partículas no dominados obtenidas durante la búsqueda. Este archivo interactúa con la población primaria para seleccionar a los líderes. La selección de la mejor partícula del cúmulo *gbest* es con base en la estructura del árbol dominado. El algoritmo implementa también un operador de "turbulencia" que es básicamente un operador de mutación, el cual actúa sobre la velocidad de las partículas.

Coello Coello y Salazar [57] y posteriormente Coello Coello et al. [58] propusieron un PSO para Optimización Multiobjetivo (MOPSO por sus siglas en inglés) en 2002 y 2004, respectivamente. El algoritmo hace uso del concepto de dominacia de Pareto para determinar la dirección de vuelo de cada partícula. El MOPSO mantiene los vectores no dominados encontrados previamente en un repositorio global, el cual es utilizado por las partículas para guiar su vuelo. Este algoritmo incorpora también un operador especial de mutación, con el cual se trata de mejorar la capacidad de exploración del algoritmo, así como asegurar la exploración del rango completo de cada variable de decisión.

Li [59] en 2003 propuso un MOPSO basado en la versión global del PSO a la que incorpora los principales mecanismos del NSGA-II. En esta propuesta, se combinan las mejores posiciones personales de cada partícula del cúmulo y todas las nuevas posiciones calculadas formando un total de $2N$ soluciones (donde N es el tamaño del cúmulo). Éstas son comparadas y se seleccionan a las mejores partículas para conformar el cúmulo de la siguiente iteración (utilizando el ordenamiento no dominado del NSGA-II). Los líderes son seleccionados aleatoriamente del conjunto de las mejores partículas con base en un conteo

de nichos o una distancia de agrupamiento (*crowding distance*). La propuesta utiliza un operador de mutación pero únicamente es aplicado a la partícula con el menor valor del estimador de densidad.

Bartz-Beielstein [60] en 2003 propusieron un algoritmo que inicia con la idea de introducir elitismo (mediante el uso de un archivo externo) en el PSO. El este trabajo se analizan métodos para seleccionar líderes y borrar partículas del archivo a fin de aproximarse al verdadero frente de Pareto.

Reyes y Coello Coello [61] en 2005 utilizaron un factor de agrupamiento, la mutación y el concepto de la dominacia epsilon para mejorar el algoritmo del PSO en optimización multiobjetivo. El factor de agrupamiento del vecino más cercano es utilizado para seleccionar líderes mediante un torneo binario. El algoritmo usa un archivo externo para almacenar los líderes actuales y otro donde se almacenan las soluciones finales. El factor de agrupamiento es empleado para filtrar el archivo de líderes cuando éste llega a su límite. Por su parte el concepto de dominancia epsilon se utiliza para seleccionar a aquellas partículas que permanecerán en el archivo que guarda las soluciones finales. En la propuesta, el cúmulo es dividido en tres subconjuntos y a cada uno de ellos se le aplica un operador de mutación diferente.

A pesar de que hay un gran avance en el diseño de MOPSOs, existen todavía varias áreas abiertas de investigación en torno a este tema. Por ejemplo, los criterios para seleccionar líderes, el papel de los parámetros en el desempeño y el control de la diversidad en la convergencia hacia el frente de Pareto verdadero han sido poco estudiados hasta ahora.

4.2. Descripción de la propuesta

Nuestra propuesta está basada en la versión global del algoritmo del PSO. Ésta utiliza dos archivos externos: uno para guardar las soluciones no dominadas finales y otro para guardar soluciones que las partículas van encontrando durante el proceso de búsqueda. Al igual que el algoritmo propuesto por Li [59] se implementa el ordenamiento no dominado utilizado por el NSGA-II y se utiliza la distancia de agrupamiento (*crowding distance*) para seleccionar a los líderes. Nuestro algoritmo, a diferencia de los demás MOPSOs, primero elige una partícula líder y después se eligen a los vecinos que lo acompañarán en el vuelo. Nuestra propuesta utiliza también un proceso de reinicialización para mantener diversidad en la población e incorpora un operador de mutación con la finalidad de aumentar la capacidad de búsqueda del PSO (ver algoritmo 6 en la página 47).

Al igual que el Micro-PSO para espacios restringidos descrito en el capítulo anterior, esta propuesta multiobjetivo continúa utilizando tamaños de población muy pequeños de sólo cinco partículas, por lo que decidimos llamarlo Micro-MOPSO. Éste es análogo al micro algoritmo genético (μ-GA) [48, 49]. Los detalles del manejo de los archivos externos,

el mecanismo adoptado para seleccionar líderes, el proceso de reinicialización, así como el operador de mutación incorporado, son descritos a continuación.

4.2.1. Manejo de los archivos externos

Manejo del archivo final

Dado que nuestra propuesta sólo utiliza cinco partículas, se necesita un medio para almacenar y reportar las soluciones no dominadas que el algoritmo encuentra. El Micro-MOPSO utiliza un archivo externo llamado **archivo final** para este propósito. Al incorporar este archivo, estamos guardando las mejores soluciones en cada iteración (elitismo). Cabe mencionar que este archivo es utilizado para seleccionar a los líderes que guiarán la búsqueda (ver sección 4.2.2). El tamaño del archivo final (TAF) es un parámetro del Mico-MOPSO que debe definir el usuario.

En cada iteración, para cada una de las partículas del cúmulo, se calcula su nueva posición para después obtener las soluciones no dominadas del cúmulo actual. Estas soluciones tratan de ingresar al archivo final. Si alguna de éstas es dominada por cualquier solución que está dentro del archivo, ésta es rechazada. Por otro lado, las soluciones que resulten dominadas por las soluciones aceptadas para ingresar al archivo final, son expulsadas de éste.

Después de ingresar las soluciones no dominadas al archivo final, se verifica que éste no haya rebasado su tamaño permitido. En caso de que esto suceda, para cada solución (dentro del archivo final) se calcula la distancia de agrupamiento (ver siguiente sección). Posteriormente, se ordenan las soluciones con base en su valor de distancia de agrupamiento calculado y se retienen las TAF soluciones con mayor valor (las soluciones que tengan una mejor distribución).

A continuación se describe el cálculo de la distancia de agrupamiento.

Distancia de agrupamiento

La distancia de agrupamiento (DA) es una estimación de la densidad de soluciones que se encuentran alrededor de una solución particular. Para una solución i , se calcula la distancia promedio de dos soluciones en ambos lados de i ($i + 1$ e $i - 1$) a lo largo de cada objetivo. Esta cantidad es proporcional al perímetro del cuboide formado por las soluciones vecinas más cercanas utilizadas como vértices (ver figura 4.1 en la página 48). Esta cantidad es llamada distancia de agrupamiento (*crowding distance*) [36].

Los pasos para calcular la distancia de agrupamiento en una población son los siguientes:

Algoritmo 6: Pseudocódigo del Micro-MOPSO.
Entrada: Número de partículas (N), número de generaciones (G), número de generaciones de reinicialización (gr), número de partículas de reinicialización (pr), tamaño del archivo final (TAF), tamaño del archivo auxiliar (TAA), porcentaje de mutación.
Salida: Archivo final con el conjunto de soluciones no dominadas

```
begin
    Inicializa archivo final (vacío);
    Inicializa archivo auxiliar (vacío);
    for i = 1 to Número de partículas do
        Inicializar posición y velocidad de manera aleatoria;
    end
    Guarda cúmulo en archivo auxiliar;
    Obtener soluciones no dominadas;
    Guarda soluciones no dominadas en el archivo final;
    cont = 1;
    repeat
        Selecciona Líder;
        Selecciona vecindario;
        if cont == Número de generaciones de reinicialización then
            Proceso de reinicialización;
            cont = 1;
        end
        for i = 1 to Número de partículas do
            Actualiza velocidad;
            Calcula nueva posicion;
            if x_i domina a xpb_i then
                for d = 1 to Número de dimensiones do
                    xpb_id = x_id; // Actualiza el pbest de la partícula;
                end
            end
        end
        Aplica operador de mutación;
        Guarda cúmulo en archivo auxiliar;
        if Tamaño archivo auxiliar > permitido then
            Depurar Archivo auxiliar;
        end
        Obtener soluciones no dominadas;
        Guarda soluciones no dominadas en archivo final;
        if Tamaño archivo final > permitido then
            Depurar Archivo final;
        end
        cont =cont +1;
    until Número máximo de generaciones ;
    Reportar el archivo final con las soluciones no dominadas.
end
```

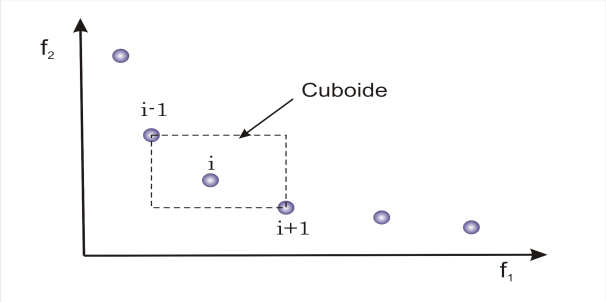

Figura 4.1: Ejemplo del cuboide formado por la partícula i para el cálculo de la distancia de agrupamiento.

1. Asignar a l el número de soluciones en la población F, esto es $l = |F|$
2. Para cada solución i inicializar $DA_i = 0$
3. Para cada función objetivo ($m = 1, 2, ..., M$)
4. Realizar un ordenamiento descendiente de f_m
5. Asignar un DA_i muy grande a las soluciones que están en los extremos
6. Para las soluciones restantes ($j = 2, 3, ..., l - 1$):

$$DA_j = DA_j + \frac{f_m^{j+1} - f_m^{j-1}}{f_m^{max} - f_m^{min}} \tag{4.1}$$

Cabe mencionar que para una solución i las soluciones vecinas $i + 1$ e $i - 1$ no necesariamente son vecinas para todos los objetivos. Los parámetros f_m^{max} y f_m^{min} de la ecuación 4.1 son los valores de los extremos de la m-ésima función objetivo. Si el valor de DA_i es pequeño indica que las soluciones se encuentran muy cercanas entre sí. Por ende, al utilizarse en un AE, se prefieren las soluciones con mayor DA puesto que indica que éstas están más dispersas.

Manejo del archivo auxiliar

Al utilizar tamaños de población muy pequeños en un MOEA, en cada iteración se acelera la pérdida de diversidad. Por lo tanto, el Micro-MOPSO incorpora dos mecanismos para preservar diversidad: un archivo auxiliar y un mecanismo de reinicialización.

La idea de utilizar el archivo auxiliar es mantener las soluciones que el algoritmo va encontrando durante el proceso de búsqueda. En nuestra propuesta este archivo es utilizado para seleccionar a los vecinos que acompañan a los líderes en el vuelo (ver sección 4.2.2).

El tamaño del archivo auxiliar (TAA) es un parámetro del Micro-MOPSO.

En este algoritmo que proponemos, después de calcular la nueva posición para cada partícula, las nuevas soluciones ingresan al archivo auxiliar. Posteriormente, se verifica que éste no haya rebasado su tamaño permitido. En caso de que esto suceda, se realiza una jerarquización con base en su dominancia (parecida a la del NSGA-II) y se retienen a las mejores TAA soluciones siempre y cuando éstas estén dentro de los primeros cinco frentes de Pareto. En este caso no importa que la cantidad de soluciones retenidas sea menor al TAA.

La jerarquización en el archivo auxiliar se realiza de la siguiente manera: primero se obtienen las soluciones no dominadas del archivo auxiliar (primer frente de Pareto). Si el número de estas soluciones es mayor a TAA, se calcula la distancia de agrupamiento de cada una de éstas, se ordenan y se retienen a las TAA con mayor valor. En caso contrario, estas soluciones del primer frente se retienen en el archivo, pero se excluyen para obtener las siguientes soluciones no dominadas (segundo frente de Pareto). Estas soluciones también se retienen siempre y cuando no excedan el tamaño del archivo. En caso de que esto suceda, se calcula la distancia de agrupamiento para cada una de ellas (segundo frente), se ordenan y se retienen a las soluciones con mayor valor para completar el tamaño del archivo auxiliar. Si el número de soluciones que están retenidas en el archivo es menor al permitido, se continúa con el proceso de obtener frentes de Pareto y retener a las mejores soluciones, hasta que se llene el archivo auxiliar o hasta que se retengan a las soluciones que forman el quinto frente de Pareto. Si sucede lo último, las soluciones que no se han retenido en el archivo auxiliar se expulsan (ver figura 4.2).

4.2.2. Selección de líderes y vecinos

Al extender el algoritmo del PSO para resolver problemas de optimización multiobjetivo, resulta fundamental diseñar un mecanismo adecuado para seleccionar a los líderes, pues son éstos los que guiarán a las partículas vecinas rumbo a las mejores soluciones compromiso. Como se comentó anteriormente, en nuestro Micro-MOPSO, a diferencia de las demás propuestas multiobjetivo basadas en PSO, primero se selecciona a la partícula líder y después se elige a las partículas vecinas que conformarán el cúmulo.

Selección de líderes

En nuestra propuesta, al utilizar sólo cinco partículas, resulta necesaria la incorporacion de un archivo exterior (en este caso utilizamos el archivo final explicado en la sección anterior) con el propósito de retener a las soluciones no dominadas encontradas durante el proceso de búsqueda. De esta manera, tenemos una mayor diversidad de partículas para elegir a los líderes.

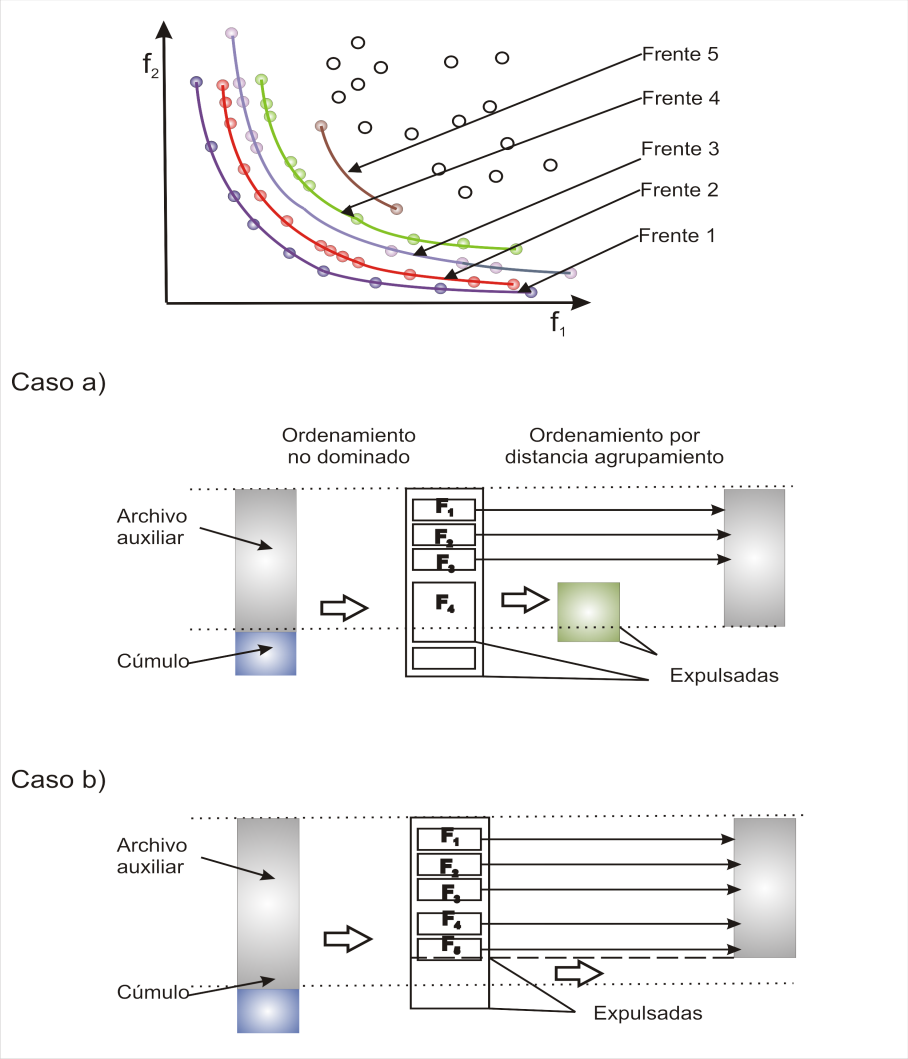

Figura 4.2: Ejemplo de la depuración del archivo auxiliar cuando éste supera su tamaño. Se van obteniendo y reteniendo los frentes a lo más hasta el frente 5. Caso a) Si al ir reteniendo estos frentes se excede el límite del archivo, se realiza un ordenamiento con base en la distancia de agrupamiento para retener a las soluciones que completen el tamaño del archivo auxiliar. Caso b) Si al retener el Frente 5, el archivo auxiliar no ha llegado a su límite, las soluciones restantes se expulsan de éste, no importando que el número de soluciones retenidas sea menor a su tamaño.

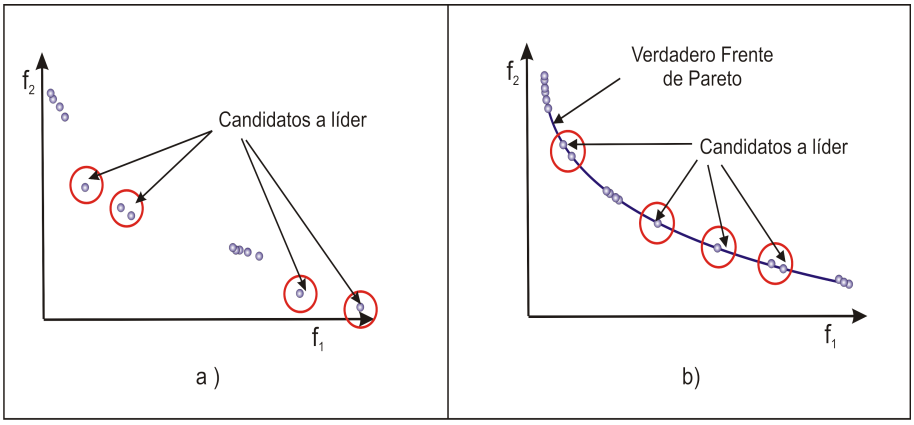

Figura 4.3: Caso hipotético de la selección de líderes. a) Las partículas en zonas menos densas son candidatas a líder. b) El mecanismo implementado ayuda a cubrir el frente de Pareto.

El algoritmo 7 muestra el proceso para seleccionar a una partícula líder ($gbest$) en una iteración determinada. La idea de ordenar a la población del archivo final con base en la distancia de agrupamiento es obtener una subpoblación (el porcentaje de subpoblación es definido por el parámetro **pps**) cuyas soluciones tengan la mejor distribución. Luego, se elige una partícula dentro de esta subpoblación como líder.

Algoritmo 7: Selección de líderes en el Micro-MOPSO.
Entrada: Archivo final AF, tamaño del archivo final (TAF), porcentaje de población para seleccionar al líder **pps**.
Salida: Partícula líder

```
begin
    for i = 1 to Número de partículas del archivo final do
        Calcula la distancia de agrupamiento DA_i;
    end
    Realizar un ordenamiento descendente (de acuerdo a la DA);
    Tomar el porcentaje de población (pps) con mayor DA;
    Elegir una partícula aleatoria;
    Asignar como líder a la partícula elegida
end
```

Al utilizar la distancia de agrupamiento como un criterio para seleccionar a las partículas líderes, nuestra propuesta da prioridad a aquellas soluciones no dominadas que se encuentran en zonas menos densas tratando de buscar más soluciones en dichas zonas. En el caso de que las soluciones no dominadas almacenadas en el archivo final ya se encuentren en el frente de Pareto verdadero, este mecanismo ayuda cubrir todo el frente de Pareto (ver figura 4.3).

Selección de vecinos

A diferencia de otros MOPSOs de la literatura, nuestra propuesta rompe un poco con el esquema original del PSO. El Micro-MOPSO no elige a la partícula líder dentro de un cúmulo dado, sino lo contrario, primero se elige al líder y después se elige a la población restante del cúmulo para realizar la búsqueda de mejores soluciones. Nuestro algoritmo utiliza el archivo auxiliar para seleccionar a dichos vecinos.

El algoritmo 8 muestra el proceso para seleccionar a la población restante del cúmulo en una iteración determinada. Se seleccionan como vecinos a las partículas con la menor distancia al *gbest*. La idea es simular una población suficientemente grande para poder dividirla en subcúmulos.

Algoritmo 8: Selección del vecindario Micro-MOPSO.
Entrada: Partícula líder *gbest*, archivo auxiliar, tamaño del cúmulo N.
Salida: Vecindario del *gbest*

```
begin
    for i = 1 to Número de partículas del archivo auxiliar do
        Calcula la distancia (DL) de la partícula i a la partícula líder gbest (en el
        espacio objetivo);
    end
    Realizar un ordenamiento descendente (de acuerdo a la DL);
    Elegir a las N-1 partículas más cercanas a la partícula gbest;
    Asignar a las N-1 partículas elegidas y a la partícula;
    gbest como el cúmulo para la siguiente iteración;
end
```

Al adoptar este mecanismo, se pretende también que al inicio el algoritmo tenga una buena exploración del espacio de búsqueda y conforme llegue al final, al acercarse al frente de Pareto verdadero, se tenga una buena explotación en las regiones encontradas (ver figura 4.4).

4.2.3. Proceso de reinicialización

Al igual que el Micro-PSO para espacios restringidos, nuestra propuesta multiobjetivo continúa utilizando un proceso de reinicialización. Éste es utilizado como complemento al archivo auxiliar con el objetivo de mantener diversidad en la población.

El proceso es prácticamente el mismo que el implementado en el Micro-PSO. La única diferencia es que en la versión multiobjetivo, este proceso se aplica después de elegir al líder y a los vecinos del cúmulo.

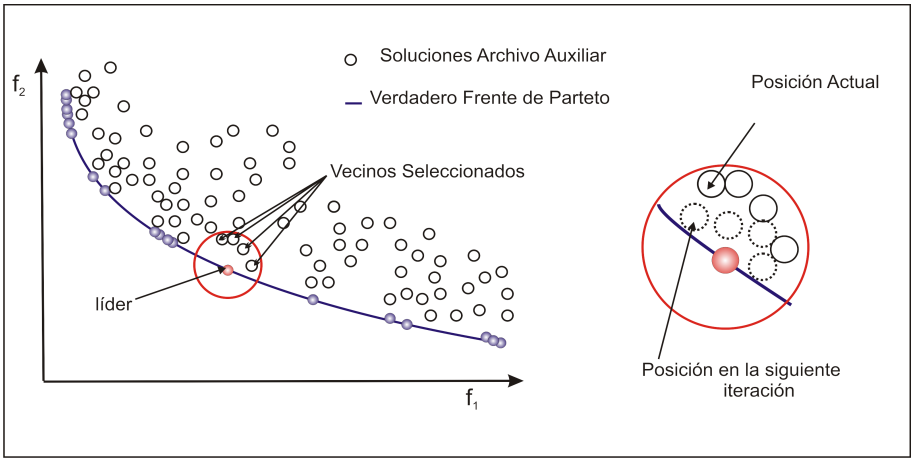

Figura 4.4: Caso hipotético de la selección de vecinos. El líder atrae a los vecinos seleccionados (las partículas más cercanas a él en distancia) hacia el frente de Pareto verdadero.

Después de un cierto número de iteraciones de reemplazo (**gr**), se obtienen las soluciones no dominadas del cúmulo. Luego, se reemplazan las **rp** partículas dominadas (partículas de reemplazo) por partículas generadas aleatoriamente (posición y velocidad). A estas **rp** partículas se les permite retener su *pbest*. En caso de que el número de soluciones dominadas sea menor a **rp**, se van tomando de manera aleatoria las soluciones no dominadas hasta completar las **rp** partículas requeridas.

4.2.4. Operador de mutación

El Micro-MOPSO adopta el mismo operador de mutación no uniforme incorporado en el Micro-PSO para espacios restringidos (ver sección 3.2.3 en la página 35). Este operador de mutación se aplica en la posición de la partícula y en todas sus dimensiones. En nuestra propuesta multiobjetivo, el valor del parámetro b (no uniformidad) es fijado en 5 como en el Micro-PSO para espacios restringidos.

4.3. Experimentos y resultados

4.3.1. Medidas de desempeño

Al utilizar técnicas evolutivas multiobjetivo, se deben de considerar tres aspectos para evaluar la calidad de las soluciones encontradas [62].

1. Minimizar la distancia del frente de Pareto producido por el algoritmo con respecto al verdadero frente de Pareto (cuando éste es conocido).

2. Maximizar la distribución de las soluciones encontradas. Es decir, lograr una distibución de soluciones no dominadas lo más uniforme que sea posible.

3. Maximizar el número de elementos del conjunto de óptimos de Pareto encontrados.

Con base en estos tres aspectos se adoptan tres medidas de desempeño para evaluar el rendimiento del Micro-MPSO y de esta forma poder compararlo contra el algoritmo del estado del arte en el área elegido, el NSGA-II.

Tasa de error

La tasa de error (TE) fue propuesta por Van Veldhuizen [63]. Esta medida de desempeño indica el porcentaje de soluciones del frente de Pareto obtenido por una técnica que no son miembros del frente de Pareto verdadero. Matemáticamente, se define como:

$$TE = \frac{\sum_{i=1}^{n} e_i}{n} \tag{4.2}$$

$$e_i = \begin{cases} 1 & \text{si el vector } i \text{ es miembro del frente de Pareto obtenido} \\ 0 & \text{de lo contrario} \end{cases} \tag{4.3}$$

Donde n es el número de vectores en el frente de Pareto obtenido. Si $TE = 0$, significa que todos los vectores generados por el algoritmo pertenecen al frente de Pareto verdadero. Una desventaja de esta medida de desempeño es que si ningún vector pertenece al frente de Pareto verdadero, ésta no permite distinguir qué tan alejados se encuentran los vectores de éste.

Distancia generacional invertida

El concepto de distancia generacional (DG) fue propuesto por Van Veldhuizen y Lamont [64]. Esta medida de desempeño indica qué tan lejos, en promedio, se encuentra el frente de Pareto obtenido por una técnica del frente de Pareto verdadero. La desventaja de ésta es que se requiere conocer el frente de Pareto verdadero. Matemáticamente, se define como:

$$DG = \frac{(\sum_{i=1}^{n} d_p^i)^{1/p}}{n} \tag{4.4}$$

Donde n es el número de vectores no dominados del conjunto de óptimos de Pareto encontrados por el algoritmo, para $p = 2$, d_i es la distancia Euclidiana (en el espacio de los objetivos) entre la i-ésima solución del frente de Pareto encontrado por el algoritmo y la

solución más cercana al frente de Pareto verdadero. Cuando $DG = 0$, todas las soluciones no dominadas encontradas por el algoritmo están en el frente de Pareto verdadero.

La Distancia Generacional Invertida (DGI) indica el promedio de la distancia del frente de Pareto verdadero con respecto al frente de Pareto obtenido por una técnica determinada. Si comparamos dos algoritmos, el mejor será aquel que tenga menor DGI. Se hace notar que se obtiene una DGI pequeña sólo si todos los vectores no dominados del frente de Pareto obtenido están cercanos al frente de Pareto verdadero, y cubren éste en toda su extensión (es decir, las aproximaciones aglomeradas en una región pequeña del frente de Pareto son penalizadas).

Espaciamiento

El espaciamiento (E) es una medida de desempeño que describe la dispersión de las soluciones no dominadas en el frente de Pareto obtenido por una técnica. Ésta fue propuesta por Schott [65] y mide la varianza de la distancia de cada solución no dominada del conjunto de óptimos de Pareto encontradas por el algoritmo con respecto a su vecino más cercano. Nótese que esta medida de desempeño no requiere que se conozca el frente de Pareto verdadero. Sin embargo, parte de la premisa de que el algoritmo ha convergido al frente de Pareto verdadero, pues de lo contrario carecería de sentido obtener una buena dispersión. La desventaja de esta medida de desempeño es que puede dar valores erróneos en algunos casos. Por ejemplo, si sólo se obtienen dos soluciones, E arroja una dispersión ideal. E se define matemáticamente como:

$$E = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d} - d_i)^2} \tag{4.5}$$

$$d_i = min_{i,j \neq i}(\sum_{m=1}^{M} |f_i^m - f_j^m|) \tag{4.6}$$

Donde: n es el número de elementos del conjunto de óptimos de Pareto obtenido por el algoritmo, $j = 1, ..., n$, $\overline{d}$ es la media de todas las d_i y M es el número de objetivos. Cuando todas las soluciones encontradas están uniformemente distribuidas, el valor de E es cero.

Cobertura

La Cobertura ($C(A, B)$) fue propuesta por Zitzler et al. [62]. Ésta compara dos conjuntos de vectores no dominados A y B, calculando la fracción de cada uno que es "cubierta" (dominada) por el otro. Matemáticamente:

$$C(A,B) = \frac{|\{b \in B; \exists a \in A : a \preceq B|\}}{|B|} \tag{4.7}$$

Si todos los puntos en B son dominados por los elementos de A, $C(A,B) = 1$. En el caso opuesto, cuando ningún elemento de A domina a algún elemento de B, $C(A,B) = 0$. Como se mencionó anteriormente, la relación de dominancia de Pareto no es simétrica, por lo tanto se necesita calcular C(A,B) y C(B,A) para comparar los dos conjuntos de vectores no dominados obtenidos. La desventaja de esta medida de desempeño es que no verifica la uniformidad de las soluciones.

4.3.2. Experimentos

Para evaluar el rendimiento del Micro-MOPSO, se utilizaron cinco funciones de prueba llamadas ZDT (Zitzler-Deb-Thiele), descritas en [62], la función de Kursawe [66] y la funcion de Viennet [67] . Todas las funciones de prueba están descritas en el apéndice B. Estas funciones de prueba contienen características que las hacen difíciles de resolver utilizando algoritmos evolutivos.

Se realizaron treinta ejecuciones independientes para cada función de prueba. Se utilizan las medidas de desempeño descritas en la sección 4.3.1 y se comparan los resultados obtenidos por nuestra propuesta con un algoritmo representativo del estado del arte en el área: el NSGA-II [68].

El número de evaluciones de la función objetivo se fijó en 3000. En el Apéndice C se muestran los frentes de Pareto obtenidos por nuestro Micro-MOPSO de las funciones de prueba utilizadas.

En todos los experimentos se utilizaron los siguientes parámetros:

- W = Número generado aleatoriamente por una distribución uniforme en el rango [0,1]
- C1 = C2 = 1.8
- Tamaño de población = 5 partículas
- Número de generaciones = 600
- Tamaño de archivo final (**TAF**)= 100
- Tamaño de archivo auxiliar (**TAA**)= 200
- Porcentaje de población selección líder (**pps**) = 0.20
- Número de generaciones de reemplazo (**gr**) = 100

Tabla 4.1: Comparación de resultados del Micro-MOPSO con respecto al NSGA-II, en ZDT1.

Métrica	Promedio Micro-MOPSO	Promedio NSGA-II	Des.Est. Micro-MOPSO	Des.Est. NSGA-II
TE	**0.53**	1.00	0.4796	0.0
DGI	**0.000307**	0.015145	0.000229	0.002259
E	**0.005017**	0.031030	0.000462	0.009431
C	**1.0**	0.0	0.0	0.0

- Número de partículas de reemplazo (**rp**) = 2
- Porcentaje de mutación = 0.1

4.3.3. Resultados

Para cada una de las funciones de prueba se muestra una tabla comparativa con las medidas de desempeño empleadas. También se ofrece una figura, en la cual se puede evaluar el rendimiento del Micro-MOPSO de manera visual y se puede hacer una comparación con respecto al rendimiento del NSGA-II.

ZDT1

En la tabla 4.1 se puede observar que el Micro-MOPSO es mejor que el NSGA-II en todas las medidas de desempeño utilizadas para la función de prueba ZDT1. La TE nos indica que algunas soluciones obtenidas por nuestro algoritmo están sobre el frente de Pareto verdadero, mientras que ninguna solución obtenida por el NSGA-II se encuentra en éste. Nótese que la DGI obtenida por nuestra propuesta es bastante más pequeña que la obtenida por el NSGA-II. Es de resaltar que, en la cobertura (C), las soluciones encontradas por el Micro-MOPSO dominan por completo a las soluciones encontradas por el NSGA-II. Por su parte, nuestro algoritmo también logra una mejor distribución de las soluciones que el NSGA-II.

En la figura 4.5 se puede observar que algunas soluciones del Micro-MOPSO alcanzan el frente de Pareto verdadero con una buena dsitribución. En cambio, las soluciones del NSGA-II están alejadas del frente de Pareto verdadero.

ZDT2

En la tabla 4.2 se puede observar que nuevamente el Micro-MOPSO es mejor que el NSGA-II en las cuatro medidas de desempeño utilizadas para la función de prueba ZDT2.

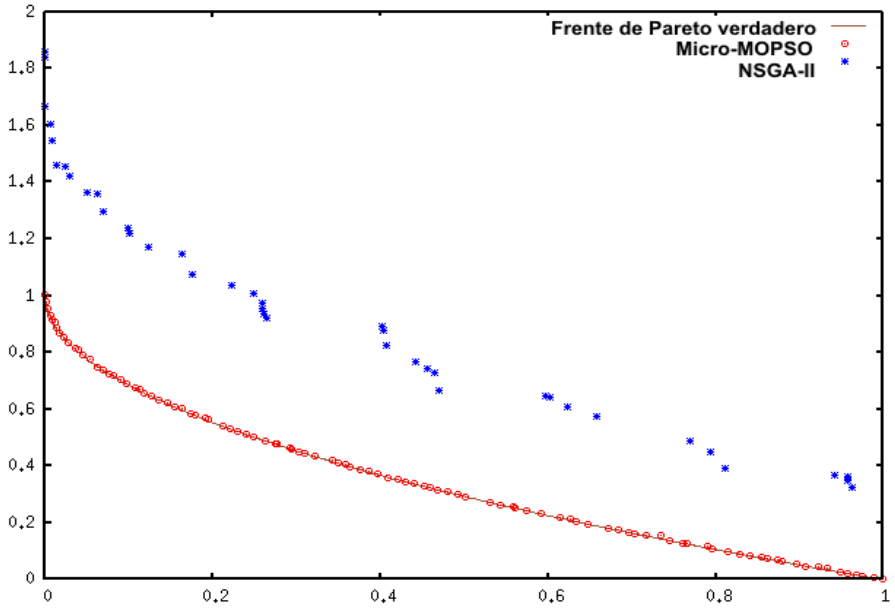

Figura 4.5: Comparación de los frentes de Pareto encontrados por el NSGA-II y el Micro-MOPSO para la función ZDT1.

Tabla 4.2: Comparación de resultados del Micro-MOPSO con respecto al NSGA-II, en ZDT2.

Métrica	Promedio Micro-MOPSO	Promedio NSGA-II	Des.Est. Micro-MOPSO	Des.Est. NSGA-II
TE	**0.496**	1.00	0.3342	0.0
DGI	**0.000908**	0.033991	0.003608	0.005991
E	**0.004888**	0.048027	0.000540	0.018218
C	**1.0**	0.0	0.0	0.0

La TE nuevamente nos indica que, en promedio, cerca de la mitad de las soluciones obtenidas por nuestro algoritmo están sobre el frente de Pareto verdadero. Por su parte, ninguna solución obtenida por el NSGA-II se encuentra en éste. La DGI obtenida por el Micro-MOPSO es bastante más pequeña que la obtenida por el NSGA-II. Al igual que sucede en la función de prueba ZDT1, en la cobertura (C) las soluciones encontradas por nuestra propuesta dominan por completo a las soluciones encontradas por el NSGA-II. Con respecto al espaciamiento, nuestro algoritmo logra una mejor distribución de las soluciones.

En la figura 4.6 se puede observar nuevamente que algunas soluciones del Micro-MOPSO alcanzan el frente de Pareto verdadero con una buena distribución. En cambio, las soluciones del NSGA-II están alejadas del frente de Pareto verdadero.

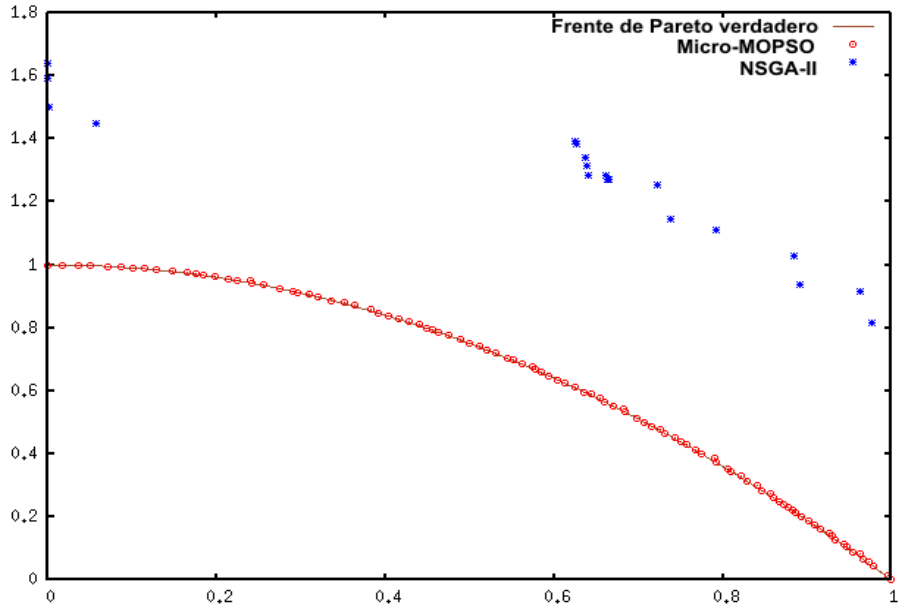

Figura 4.6: Comparación de los frentes de Pareto encontrados por el NSGA-II y el Micro-MOPSO para la función ZDT2.

Tabla 4.3: Comparación de resultados del Micro-MOPSO con respecto al NSGA-II, en ZDT3.

Métrica	Promedio Micro-MOPSO	Promedio NSGA-II	Des.Est. Micro-MOPSO	Des.Est. NSGA-II
TE	**0.935**	1.00	0.24491	0.0
DGI	**0.002888**	0.021682	0.004336	0.002653
E	**0.008511**	0.034824	0.009113	0.009288
C	**1.0**	0.005666	0.0	0.031037

ZDT3

En la tabla 4.3 se puede observar que nuestro algoritmo es mejor que el NSGA-II en todas las medidas de desempeño utilizadas para la función de prueba ZDT3. La TE nos indica que muy pocas soluciones obtenidas por nuestro algoritmo están sobre el frente de Pareto verdadero. Sin embargo, ninguna solución obtenida por el NSGA-II se encuentra en dicho frente. La DGI obtenida por el Micro-MOPSO es más pequeña que la obtenida por el NSGA-II. Al igual que sucede en las dos funciones de prueba anteriores (ZDT1 y ZDT2), en la cobertura (C) las soluciones encontradas por el Micro-MOPSO dominan por completo a las soluciones encontradas por el NSGA-II. Nuevamente, nuestro algoritmo logra una mejor distribución de las soluciones.

En la figura 4.7 se puede observar que aunque el frente es discontinuo, algunas soluciones de nuestra propuesta alcanzan el frente de Pareto verdadero con una buena distribución, mientras que las soluciones del NSGA-II están alejadas del frente de Pareto verdadero.

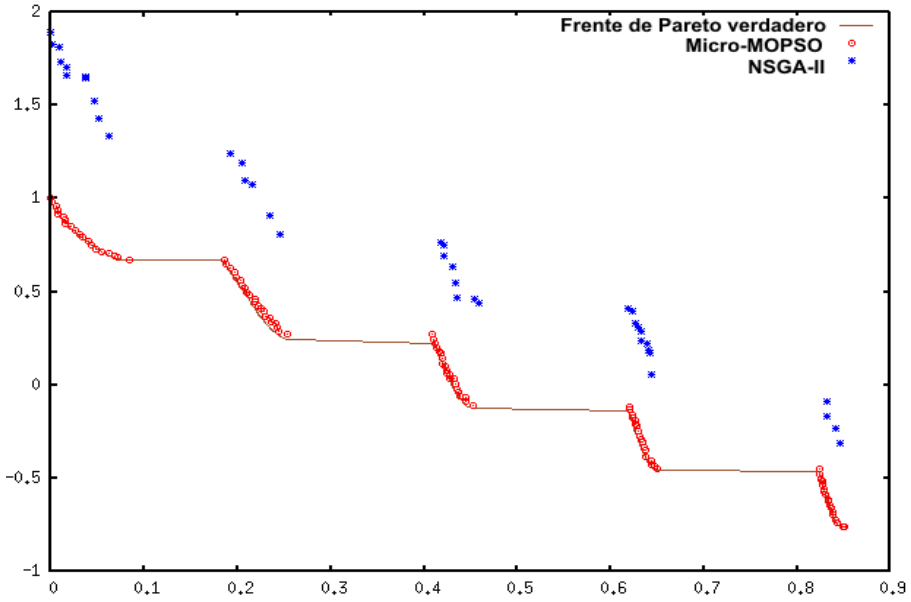

Figura 4.7: Comparación de los frentes de Pareto encontrados por el NSGA-II y el Micro-MOPSO para la función ZDT3.

Tabla 4.4: Comparación de resultados del Micro-MOPSO con respecto al NSGA-II, en ZDT4.

Métrica	Promedio Micro-MOPSO	Promedio NSGA-II	Des.Est. Micro-MOPSO	Des.Est. NSGA-II
TE	1	1.00	0.0	0.0
DGI	1.585645	**0.594165**	0.790579	0.178251
E	**0.000144**	5.321164	0.000478	3.837878
C	0.112333	**0.516**	0.192867	0.285591

ZDT4

En la tabla 4.4 se puede observar que el NSGA-II es mejor que el Micro-MOPSO en dos de las cuatro medidas de desempeño utilizadas para la función de prueba ZDT4. La TE nos indica que ninguna solución obtenida por ambos algoritmos está sobre el frente de Pareto verdadero La DGI obtenida por el Micro-MOPSO es mayor que la obtenida por el NSGA-II. En la cobertura (C), las soluciones encontradas por NSGA-II en promedio cubren a más soluciones de nuestra propuesta. En el espaciamiento, si bien la medida favorece a nuestro algoritmo, esto no necesariamente indica que éste tuvo un buen desempeño. Esto se debe a que el Micro-MOPSO sólo encuentra soluciones en una región muy pequeña (como se puede observar en la figura 4.8) y puede ocurrir que en esta región las soluciones se encuentren bien distribuidas.

En la figura 4.8 se puede observar que ambos algoritmos están alejados del frente de Pareto verdadero. También se puede observar que las soluciones de nuestra propuesta están aglutinadas en una región muy pequeña.

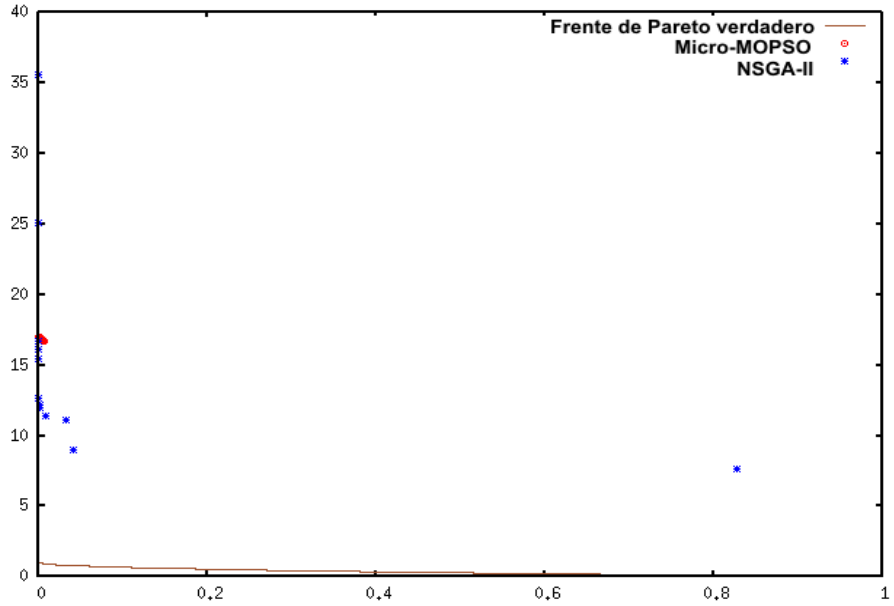

Figura 4.8: Comparación de los frentes de Pareto encontrados por el NSGA-II y el Micro-MOPSO para la función ZDT4.

Tabla 4.5: Comparación de resultados del Micro-MOPSO con respecto al NSGA-II, en ZDT6.

Métrica	Promedio Micro-MOPSO	Promedio NSGA-II	Des.Est. Micro-MOPSO	Des.Est. NSGA-II
TE	**0.133333**	1.00	0.00182	0.0
DGI	**0.000065**	0.057039	0.000006	0.004604
E	**0.033692**	0.124517	0.149965	0.085331
C	**0.865333**	0.318666	0.241214	0.152399

ZDT6

En la tabla 4.5 se puede observar que el Micro-MOPSO es mejor que el NSGA-II en las cuatro medidas de desempeño utilizadas para la función de prueba ZDT6. La TE nos indica que, en promedio, la mayoría de las soluciones obtenidas por nuestro algoritmo están sobre el frente de Pareto verdadero. En contraste, ninguna solución obtenida por el NSGA-II se encuentra en dicho frente. Se hace notar que la DGI obtenida por nuestra propuesta es bastante más pequeña que la obtenida por el NSGA-II. Las soluciones encontradas por el Micro-MOPSO dominan casi por completo a las soluciones encontradas por el NSGA-II. Con respecto al espaciamiento, los algoritmos están casi empatados, aunque el NSGA-II tiene una menor desviación estándar para esta métrica.

En la figura 4.9 se puede observar que algunas soluciones del Micro-MOPSO alcanzan el frente de Pareto verdadero con una buena dsitribución. En cambio, las soluciones del

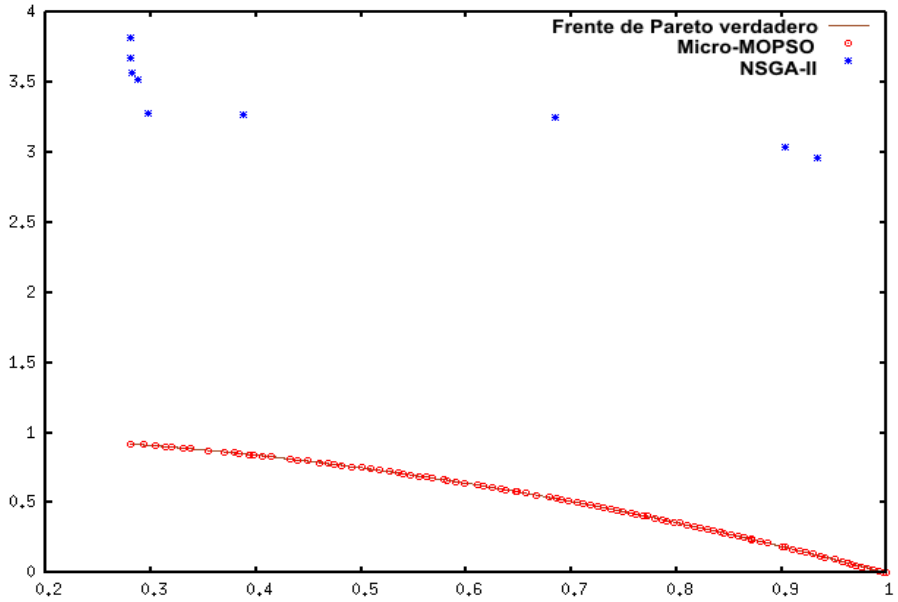

Figura 4.9: Comparación de los frentes de Pareto encontrados por el NSGA-II y el Micro-MOPSO para la función ZDT6.

Tabla 4.6: Comparación de resultados del Micro-MOPSO con respecto al NSGA-II, en Kursawe.

Métrica	Promedio Micro-MOPSO	Promedio NSGA-II	Des.Est. Micro-MOPSO	Des.Est. NSGA-II
TE	0.954	**0.837**	0.39281	0.09352
DGI	**0.003396**	0.004571	0.000717	0.000569
E	**0.082417**	0.100954	0.028810	0.025294
C	**0.854666**	0.603333	0.054502	0.096680

NSGA-II están alejadas del frente de Pareto verdadero.

Kursawe

Para la función de prueba de Kursawe, en la tabla 4.6 se puede observar que el Micro-MOPSO tiene resultados muy parecidos al NSGA-II en las cuatro medidas de desempeño. En comparación con las otras funciones de prueba, la DGI obtenida por nuestra propuesta ya no es mucho más pequeña que la calculada por el NSGA-II. En esta función, si bien las soluciones encontradas por el Micro-MOPSO dominan en promedio a un 85 %o de las soluciones encontradas por el NSGA-II, éstas dominan en promedio a las del Micro-MOPSO en un 60 %. Por lo que respecta al espaciamiento, nuestro algoritmo obtuvo un mejor valor que el NSGA-II.

En la figura 4.10 se puede observar que algunas soluciones de ambos algoritmos han alcazado el frente de Pareto verdadero.

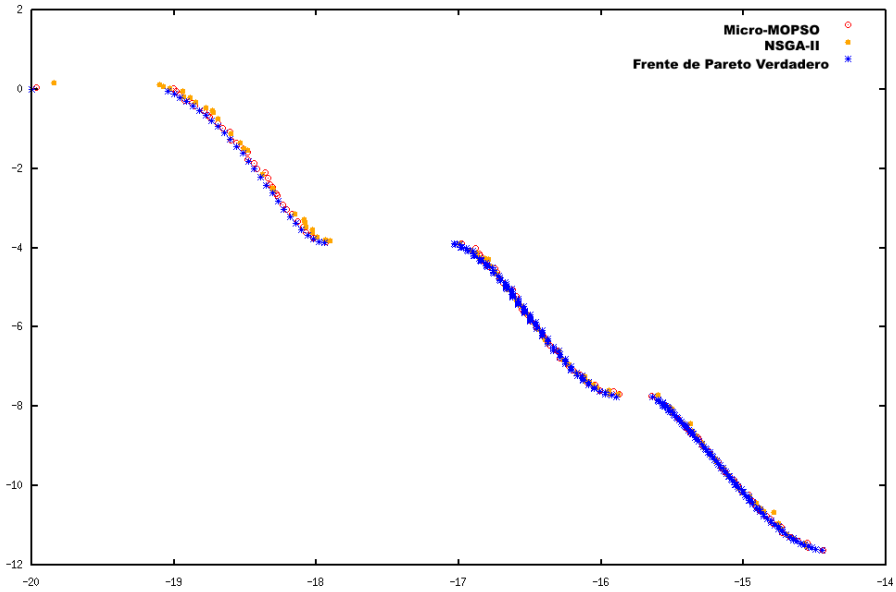

Figura 4.10: Comparación de los frentes de Pareto encontrados por el NSGA-II y el Micro-MOPSO para la función Kursawe.

Tabla 4.7: Comparación de resultados del Micro-MOPSO con respecto al NSGA-II, en Viennet.

Métrica	Promedio Micro-MOPSO	Promedio NSGA-II	Des.Est. Micro-MOPSO	Des.Est. NSGA-II
TE	**0.592**	0.600	0.4430	0.36232
DGI	**0.000686**	0.002348	0.000578	0.00453
E	**0.040249**	0.061371	0.020662	0.010512
C	0.905388	**0.927**	0.035982	0.075929

Viennet

Para la función de prueba de Viennet, en la tabla 4.7 se puede observar que el Micro-MOPSO tiene resultados muy parecidos al NSGA-II en las cuatro medidas de desempeño. Si bien nuestra propuesta tiene una DGI menor y una mejor dispersión, las soluciones obtenidas por el NSGA-II, en promedio, dominan a más soluciones obtenidas por el Micro-MOPSO.

En la figura 4.11 se puede observar que algunas soluciones de ambos algoritmos han alcanzado el frente de Pareto verdadero.

Con base en los resultados obtenidos y de acuerdo a las medidas de desempeño empleadas, nuestro algoritmo multiobjetivo resultó ser mejor que el NSGA-II en la mayoría de las funciones de prueba utilizadas. Nuestra propuesta logra generar todo el frente de Pareto con

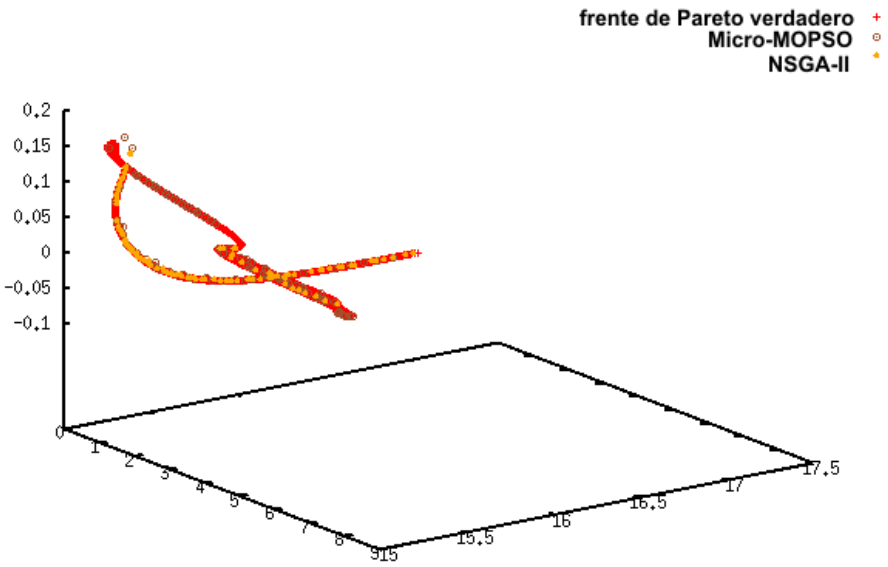

Figura 4.11: Comparación de los frentes de Pareto encontrados por el NSGA-II y el Micro-MOPSO para la función Viennet.

una buena dispersión incluso en funciones cuyos frentes son discontinuos (ZDT3 y Kursawe). El Micro-MOPSO tiene una convergencia acelerada. Nuestra propuesta requiere de tan sólo 1500 evaluaciones de la función objetivo para llegar al frente de Pareto verdadero de la función ZDT1 (ver figura C.2 en la pág. 78). A su vez, con 1000 evaluaciones nuestro algoritmo alcanza también el frente de Pareto verdadero de la función ZDT6 (ver figura C.6 en la pág. 80) . Para problemas con más de dos funciones objetivo, el Micro-PSO muestra que también logra generar soluciones dentro del frente de Pareto verdadero. Sin embargo, para la función de prueba ZDT4 nuestro algoritmo muestra un desempeño deficiente. Aún elevando el número de evaluaciones a 10000, el Micro-PSO no converge al frente de Pareto verdadero.

Capítulo 5

Conclusiones y Trabajo Futuro

En esta tesis hemos desarrollado un algoritmo basado en PSO utilizando un tamaño de población muy pequeño de sólo cinco partículas para resolver problemas de optimización con restricciones. Tal como sucede en el caso de los micro algoritmos genéticos, nuestra propuesta tiene éxito sólo si cuenta con un proceso de reinicialización adecuado para mantener diversidad en la población. Para nuestra propuesta es de vital importancia que el mecanismo para manejo de restricciones permita seleccionar líderes de forma tal que las soluciones oscilen dentro y fuera de la zona factible.

En cuanto a los resultados obtenidos, el número de evaluaciones de la función objetivo que nuestra propuesta requiere para alcanzar los óptimos globales, es menor o igual que el número requerido por las técnicas en el estado del arte en el área con respecto a las cuales se comparó (SR, SMES y CHM-PSO). Por ejemplo, nuestro algoritmo requiere de aproximadamente $31\,\%$ y $29\,\%$ menos de evaluaciones de la función objetivo si se compara respectivamente con el SR y con el CHM-PSO. Por su parte, nuestra propuesta y la SMES utilizan ambos el mismo número de evaluaciones 240,000 (ver tabla 3.3 en la página 39). Los resultados de las tablas 3.4, 3.5 y 3.6 (en las páginas 39, 40 y 41 respectivamente) muestran que nuestra propuesta es competitiva y por lo tanto puede ser una alternativa viable para resolver problemas de optimización con restricciones.

También se desarrolló un algoritmo basado en PSO utilizando un tamaño de población muy pequeño de sólo cinco partículas para resolver problemas de optimización multiobjetivo. Tal como sucede en la propuesta mono-objetivo para manejo de restricciones, resulta fundamental mantener diversidad en la población. Para nuestra propuesta multiobjetivo es indispensable adoptar dos mecanismos para mantener diversidad: el archivo externo y el proceso de reinicialización. La selección de líderes con base en la distancia de agrupamiento permitió obtener una mejor distribución de las soluciones encontradas por nuestro algoritmo multiobjetivo. Si bien se rompió con el esquema original del PSO al seleccionar primero al líder y posteriormente a los vecinos, esto permitió a nuestra propuesta una rápida convergencia hacia el frente de Pareto verdadero. Por ejemplo, nuestro Micro-PSO multiobjetivo requiere de tan sólo 1,500 y 1,000 evaluaciones de la función objetivo para llegar a los frentes de Pareto verdaderos de las funciones de prueba ZDT1 y ZDT6 respec-

tivamente (ver figuras C.2 y C.6 en la páginas 78 y 80).

Con base en los resultados obtenidos, nuestro algoritmo multiobjetivo resultó ser competitivo en la mayoría de las funciones de prueba utilizadas (ver tabla 4.1 en la pág. 57, tabla 4.2 en la pág. 58, tabla 4.3 en la pág. 59, tabla 4.4 en la pág. 60, tabla 4.5 en la pág. 61, tabla 4.6 en la pág. 62 y tabla 4.7 en la pág. 62). Además de tener una rápida convergencia, nuestra propuesta logra generar todo el frente de Pareto con una buena dispersión incluso en funciones cuyos frentes son discontinuos (ZDT3 figura 4.7 en la pág. 60 y Kursawe figura 4.10 en la pág. 63). El Micro-PSO muestra que puede utilizarse para problemas de más de dos funciones objetivo. Para la función de prueba Viennet (3 funciones objetivo) nuestra propuesta logra generar soluciones dentro del frente de Pareto verdadero (ver figura 4.10 en la pág. 63). Los resultados indican que nuestra propuesta multiobjetivo es competitiva, y por lo tanto, puede ser una alternativa viable para resolver problemas de optimización multiobjetivo sin restricciones.

5.1. Trabajo Futuro

Como posible trabajo futuro, consideramos interesante comparar nuestra propuesta con un conjunto extendido de funciones de prueba para optimización evolutiva con restricciones [69]. También, creemos necesario estudiar la sensibilidad de los parámetros que requiere nuestra propuesta, a fin de encontrar un conjunto de estos parámetros de forma tal que nuestro algoritmo tenga menor variabilidad, es decir, reducir la desviación estándar de los resultados obtenidos (ver tabla 3.2 en la pág. 39). Se podría idear un método para determinar en qué momento aplicar el mecanismo de reinicialización y cuándo dejar de hacerlo. Además, sería interesante diseñar nuevos mecanismos de reinicialización que podrían mejorar la convegencia de nuestro algoritmo (es decir, reducir el número de evaluaciones de la función objetivo requeridas para llegar a los óptimos globales) así como aumentar la calidad de las soluciones encontradas. Por último, resultaría muy interesante incorporar un mecanismo de auto-adaptación a nuestra propuesta.

Para el micro-PSO multiobjetivo, consideramos interesante incorporar el manejo de restricciones utilizada en la propuesta mono-objetivo. Al igual que en la propuesta mono-objetivo, creemos necesario estudiar la sensibilidad de los parámetros que requiere nuestro algoritmo multiobjetivo, a fin de encontrar un conjunto de estos parámetros que hagan que el algoritmo tenga un desempeño más robusto. Podrían diseñarse nuevos mecanismos para la selección de líderes y vecinos de manera tal que aceleraran la convergencia del algoritmo. Se podría idear otro método para depurar el archivo auxiliar, e incluso diseñar nuevos métodos para preservar diversidad. Sería interesante también, realizar una versión paralela del algoritmo, así como la incorporación de un mecanismo de auto-adaptación al mismo.

Apéndice A

Funciones de Prueba con Restricciones

A.1. Función g01

Minimizar: $f(\overrightarrow{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$
sujeta a :

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

Donde $0 \leq x_i \leq 1\,(i = 1, ..., 9)$, $0 \leq x_i \leq 100\,(i = 10, 11, 12)$ y $0 \leq x_{13} \leq 1$. El óptimo global está en $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, donde $f(x^*) = -15$. Las restricciones g_1, g_2, g_3, g_4, g_5 y g_6 están activas.

A.2. Función g02

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

sujeta a:

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

Donde $n = 20$ and $0 \leq x_i \leq 10\,(i = 1, ..., n)$. El óptimo global es desconocido. La

mejor solución reportada está en [42] $f(x^*) = 0.803619$ en nuestra propuesta encontramos la solución $f(x^*) = 0.803620$. La restricción g_1 está cerca de ser activa ($g_1 = -10^{-8}$).

A.3. Función g03

Maximixar: $f(\overrightarrow{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$
sujeta a :

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

Donde $n = 10$ and $0 \leq x_i \leq 1\,(i = 1, ..., n)$. El óptimo global está en $x_i^* = \frac{1}{\sqrt{n}}(i = 1, ..., n)$ donde $f(x^*) = 1$.

A.4. Función g04

Minimizar: $f(\overrightarrow{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$
sujeta a :
$g_1(\overrightarrow{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$
$g_2(\overrightarrow{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$
$g_3(\overrightarrow{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$
$g_4(\overrightarrow{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$
$g_5(\overrightarrow{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$
$g_5(\overrightarrow{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$

Donde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45 (i = 3, 4, 5)$. La solución óptima está en $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ donde $f(x^*) = -30665.539$. Las restricciones g_1 y g_6 están activas.

A.5. Función g05

Minimizar: $f(\overrightarrow{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + \frac{0.000002}{3}x_2^3$
sujeta a :

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

Donde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 55$ y $-0.55 \leq x_4 \leq 55$. La mejor solución conocida está en $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ donde $f(x^*) = 5126.4981$.

A.6. Función g06

Minimizar: $f(\overrightarrow{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$
sujeta a:

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

Donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$. La solución óptima está en $x^* = (14.095, 0.84296)$ donde $f(x^*) = -6961.81388$. Ambas restricciones están activas.

A.7. Función g07

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

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

Donde $-10 \leq x_i \leq 10\,(i = 1, ..., 10)$. El óptimo global está en $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$, donde $f(x^*) = 24.3062091$. Las restricciones g_1, g_2, g_3, g_4, g_5 y g_6 están activas.

A.8. Función g08

Maximizar: $f(\overrightarrow{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$
sujeta a:

$$g_1(\overrightarrow{x}) = x_1^2 - x_2 + 1 \leq 0$$

$$g_2(\overrightarrow{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

Donde $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$. La solución óptima está en $x^* = (1.2279713, 4.2453733)$, donde $f(x^*) = 0.095825$.

A.9. Función g09

Minimizar: $f(\overrightarrow{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$
sujeta a:

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

Donde $-10 \leq x_i \leq 10(i = 1, ..., 7)$. La solución óptima está en $x^* = (2.330499, 1.951372, -0.4775414, 4.365726\ , -0.62448070, 1.038131, 1.594227)$, donde $f(x^*) = 680.6300573$. Las restricciones g_1 y g_4 están activas.

A.10. Función g10

Minimizar: $f(\overrightarrow{x}) = x_1 + x_2 + x_3$
sujeta a:

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

Donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000(i = 2, 3)$, $10 \leq x_i \leq 1000(i = 4, ..., 8)$. La solución óptima está en $x^* = (579.19, 1360.13, 5109.92, 182.0174, 295.5985\ , 217.9799, 286.40, 395.5979)$, donde $f(x^*) = 7049.25$. Las restricciones g_1, g_2 y g_3 están activas.

A.11. Función g11

Minimizar: $f(\overrightarrow{x}) = x_1^2 + (x_2 - 1)^2$
sujeta a:

$$h_1(\overrightarrow{x}) = x_2 - x_1^2 = 0$$

Donde $-1 \leq x_1 \leq 1$ y $-1 \leq x_2 \leq 1$. La solución óptima está en $x^* = (\pm\frac{1}{\sqrt{2}}, \frac{1}{2})$, donde $f(x^*) = 0.75$.

A.12. Función g12

Maximizar: $f(\overrightarrow{x}) = \frac{100-(x_1-5)^2-(x_2-5)^2-(x_3-5)^2}{100}$
sujeta a:

$$g_1(\overrightarrow{x}) = (x_i - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

Donde $0 \leq x_i \leq 10(i = 1, 2, 3)$ y $p, q, r = 1, 2, ...9$. La región factible del espacio de búsqueda consiste en 9^3 esferas disjuntas. Un punto (x_1, x_2, x_3) es factible si y sólo si existe p, q, r tales que cumplan la restricción. El óptimo global está en $x^* = (5, 5, 5)$, donde $f(x^*) = 1$.

A.13. Función g13

Minimizar: $f(\overrightarrow{x}) = e^{x_1 x_2 x_3 x_4 x_5}$
sujeta a:

$$h_1(\overrightarrow{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$h_2(\overrightarrow{x}) = x_2 x_3 - 5x_4 x_5 = 0$$
$$h_2(\overrightarrow{x}) = x_1^3 + x_2^3 + 1 = 0$$

Donde $-2.3 \leq x_i \leq 2.3(i = 1, 2)$ y $-3.2 \leq x_i \leq 3.2(i = 3, 4, 5)$. La solución óptima está en $x^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$, donde $f(x^*) = 0.0539498$.

Apéndice B

Funciones de Prueba Multiobjetivo

B.1. Función ZDT1

Minimizar: $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})]$

donde:

$$
\begin{aligned}
&f_1(\vec{x}) = x_1 \\
&\quad f_2(\vec{x}) = g(\vec{x}) \times h(f_1(\vec{x}), g(\vec{x})) \\
&\quad g(\vec{x}) = 1 + \tfrac{9}{(n-1)} \sum_{i=2}^{n} x_i \\
&\quad h(f_1(\vec{x}), g(\vec{x})) = 1 - \sqrt{\tfrac{f_1(\vec{x})}{g(\vec{x})}}
\end{aligned}
$$

para $n = 30$ y $x_i \in [0, 1]$. Tiene un frente de Pareto convexo y conectado.

B.2. Función ZDT2

Minimizar: $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})]$

donde:

$$
\begin{aligned}
&f_1(\vec{x}) = x_1 \\
&\quad f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x})) \\
&\quad g(\vec{x}) = 1 + \tfrac{9}{(n-1)} \sum_{i=2}^{n} x_i \\
&\quad h(f_1(\vec{x}), g(\vec{x})) = 1 - \left(\tfrac{f_1(\vec{x})}{g(\vec{x})}\right)^2
\end{aligned}
$$

para $n = 30$ y $x_i \in [0, 1]$. Tiene un frente de Pareto no convexo y conectado.

B.3. Función ZDT3

Minimizar: $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})]$
donde:

$$
\begin{aligned}
f_1(\vec{x}) &= x_1 \\
f_2(\vec{x}) &= g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x})) \\
g(\vec{x}) &= 1 + \tfrac{9}{(n-1)} \sum_{i=2}^{n} x_i \\
h(f_1(\vec{x}), g(\vec{x})) &= 1 - \sqrt{\tfrac{f_1(\vec{x})}{g(\vec{x})}} - \left(\tfrac{f_1(\vec{x})}{g(\vec{x})}\right) \sin(10\pi f_1(\vec{x}))
\end{aligned}
$$

para $n = 30$ y $x_i \in [0, 1]$. Tiene un frente de Pareto discontinuo.

B.4. Función ZDT4

Minimizar: $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})]$
donde:

$$
\begin{aligned}
f_1(\vec{x}) &= x_1 \\
f_2(\vec{x}) &= g(\vec{x}) \times h(f_1(\vec{x}), g(\vec{x})) \\
g(\vec{x}) &= 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i)) \\
h(f_1(\vec{x}), g(\vec{x})) &= 1 - \sqrt{\tfrac{f_1(\vec{x})}{g(\vec{x})}}
\end{aligned}
$$

para $n = 10$, $x_1 \in [0, 1]$ y $x_2, \ldots, x_n \in [-5, 5]$. Tiene un frente de Pareto convexo, conectado y existen 21^9 frentes de Pareto locales.

B.5. Función ZDT6

Minimizar: $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})]$
donde:

$$
\begin{aligned}
f_1(\vec{x}) &= 1 - \exp(-4x_1) \cdot \sin^6(6\pi x_1) \\
f_2(\vec{x}) &= g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x})) \\
g(\vec{x}) &= 1 + 9 \left(\frac{\sum_{i=2}^{n} x_i}{(n-1)} \right)^{0.25} \\
h(f_1(\vec{x}), g(\vec{x})) &= 1 - \left(\tfrac{f_1(\vec{x})}{g(\vec{x})}\right)^2
\end{aligned}
$$

para $n = 10$ y $x_i \in [0, 1]$. Tiene un frente de Pareto convexo y continuo.

B.6. Función Kursawe

Minimizar: $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_2(\vec{x})]$
donde:

$$f_1(\vec{x}) = \sum_{i=1}^{n-1}(-10e^{(-0.2)\times\sqrt{x_i^2+x_{i+1}^2}})$$
$$f_2(\vec{x}) = \sum_{i=1}^{n}(|x_i|^a + 5 \times sen(x_i)^b)$$

a=0.8, b=3.0 para $n = 3$ y $x_i \in [-5, 5]$.

B.7. Función Viennet

Minimizar: $\vec{f}(\vec{x}) = [f_1(x, y), f_2(x, y), f_2(x, y)]$
donde:

$$f_1(x, y) = 0.5 \times (x^2 + y^2) + sen(x^2 + y^2)$$
$$f_2(x, y) = \frac{(3x-2y+4)^2}{8} + \frac{(x-y+1)^2}{27} + 15$$
$$f_3(x, y) = \frac{1}{(x^2+y^2+1)} - 1.1 \times e^{-x^2-y^2}$$
$$g(\vec{x}) = \sum_{i=3}^{n}(x_i - 0.5)^2$$

para $x, y \in [-30, 30]$.

Apéndice C

Frentes de Pareto obtenidos por nuestro Micro-MOPSO

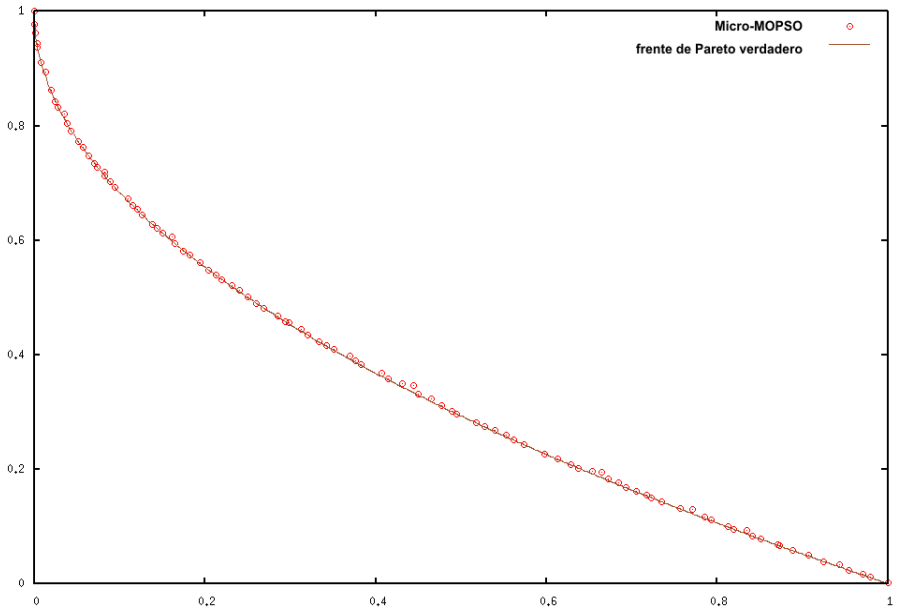

Figura C.1: Frente de Pareto obtenido por el Micro-MOPSO para la función ZDT1 (realizando 3000 evaluaciones de la función objetivo).

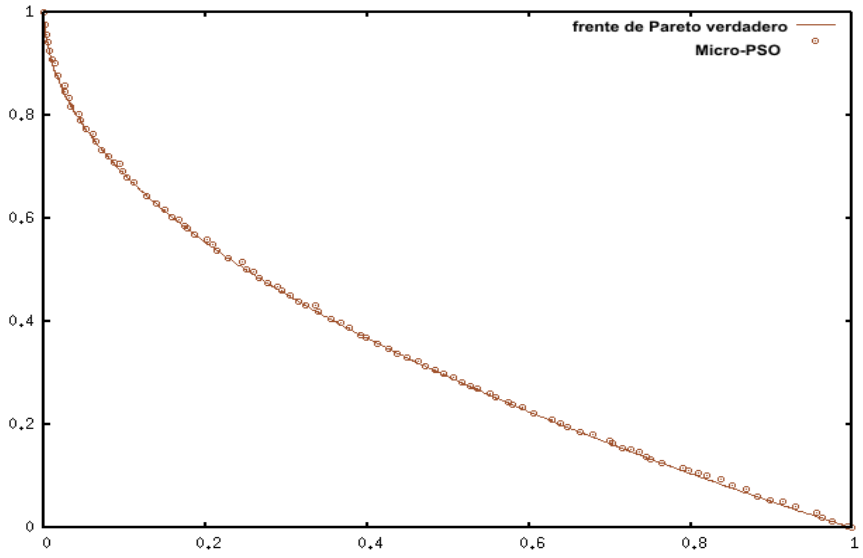

Figura C.2: Frente de Pareto obtenido por el Micro-MOPSO para la función ZDT1 (realizando 1500 evaluaciones de la función objetivo).

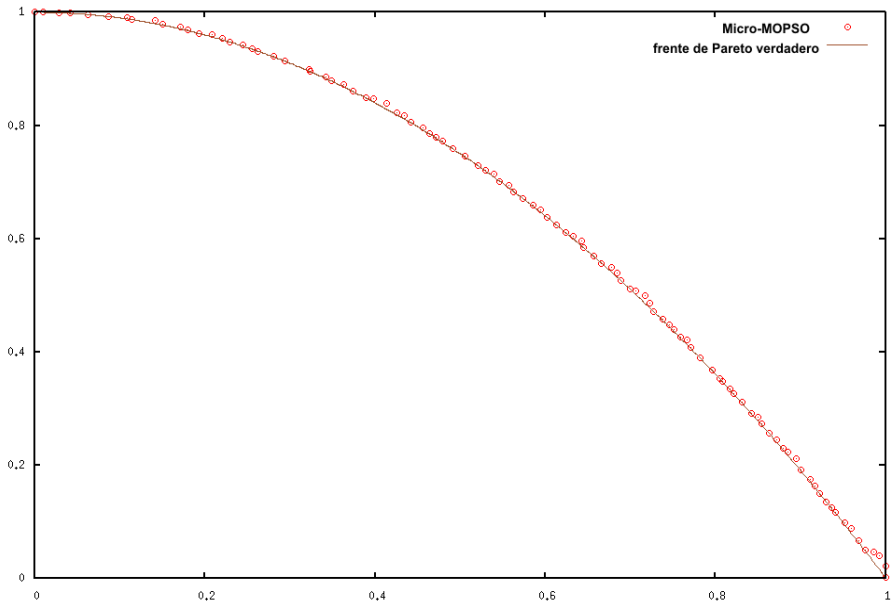

Figura C.3: Frente de Pareto obtenido por el Micro-MOPSO para la función ZDT2 (realizando 3000 evaluaciones de la función objetivo).

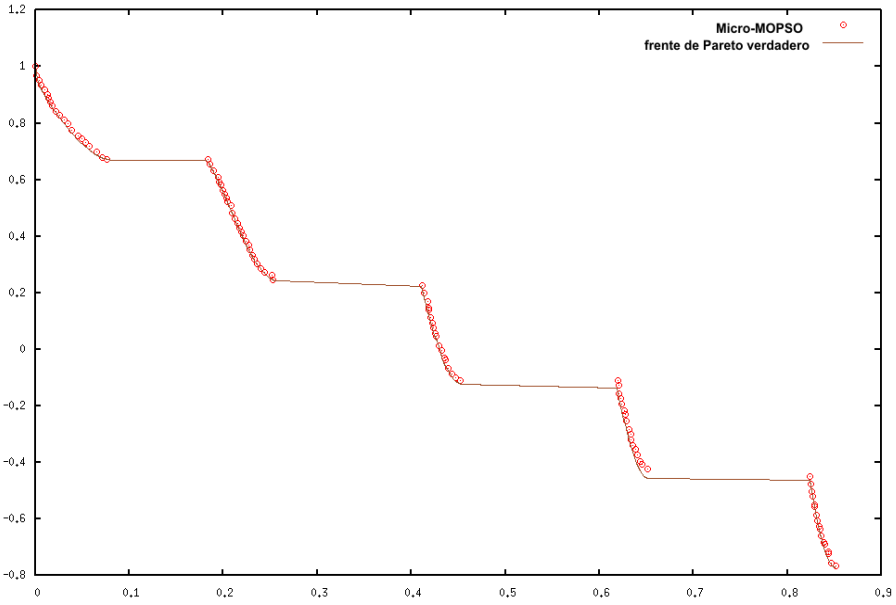

Figura C.4: Frente de Pareto obtenido por el Micro-MOPSO para la función ZDT3 (realizando 3000 evaluaciones de la función objetivo).

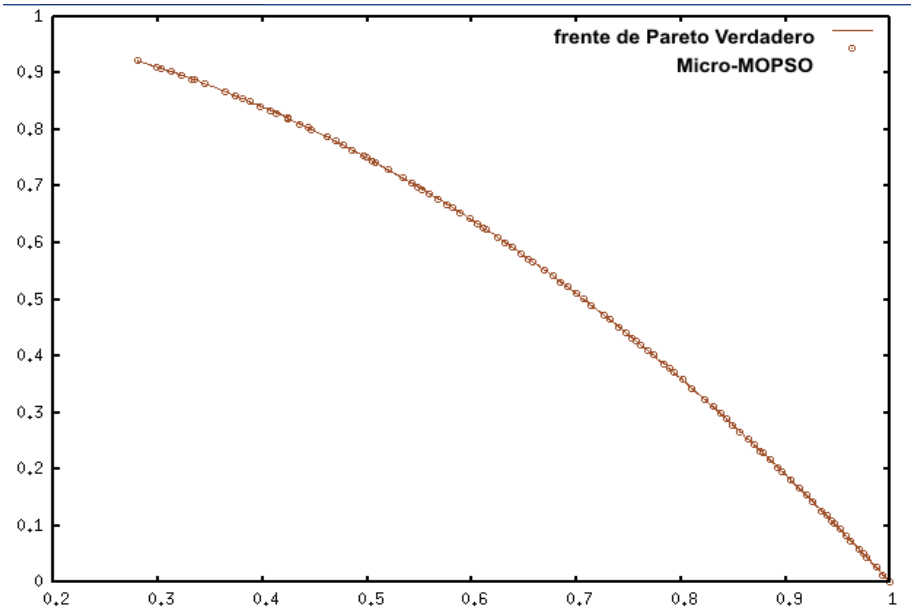

Figura C.5: Frente de Pareto obtenido por el Micro-MOPSO para la función ZDT6 (realizando 3000 evaluaciones de la función objetivo).

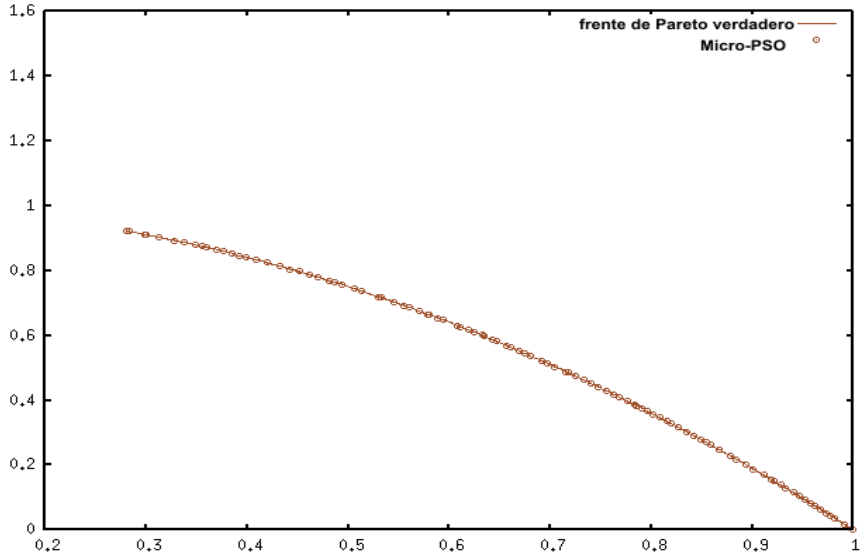

Figura C.6: Frente de Pareto obtenido por el Micro-MOPSO para la función ZDT6 (realizando 1000 evaluaciones de la función objetivo).

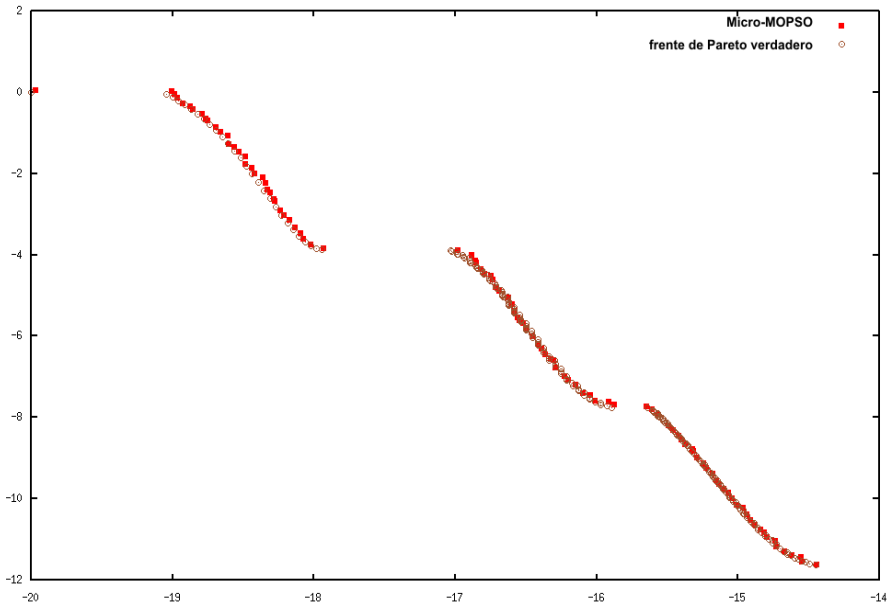

Figura C.7: Frente de Pareto obtenido por el Micro-MOPSO para la función Kursawe (realizando 3000 evaluaciones de la función objetivo).

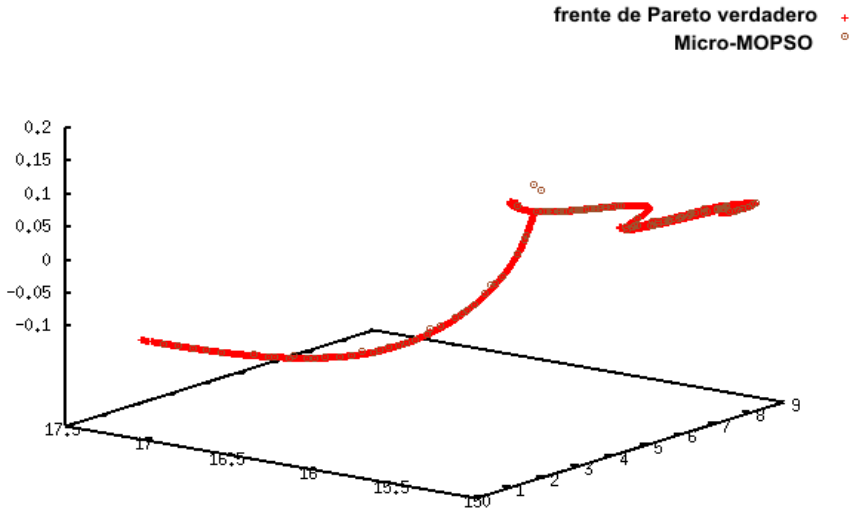

Figura C.8: Frente de Pareto obtenido por el Micro-MOPSO para la función Viennet (realizando 3000 evaluaciones de la función objetivo).

Bibliografía

[1] David B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press Mar, New York,USA, 2000.

[2] S.S. Rao. *Optimization theor and applications*. John Wiley and Sons, Ltd, New York,USA, 2001.

[3] Colin B. Reeves. In *Modern Heuristic Techniques for Combinatorial Problems*, Great Britain, 1993. John Wiley and Sons.

[4] Deb Kalyanmoy. *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons, Ltd, England, 2001.

[5] Pablo Pedregal. *Introduction to optimization*. Springer, USA, 2003.

[6] L.C.W.Dixon. *Nonlinear optimisation*. The English Universities Press, Glasgow,Great Britain, 1973.

[7] Dimitri P. Bertsekelas. *Constraint optimization and Lagrange multipliers methods*. Academic Press Inc., New York,USA, 1982.

[8] Carlos A. Coello Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11–12):1245–1287, January 2002.

[9] Carlos A. Coello Coello David A. Van Veldhuizen and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, USA, second edition, 2007.

[10] Hillermeier Claus. *Nonlinear multiobjective optimization: a generalized homotopy approach*. Birkh¨user Verlag, Basel-Boston-Berlin, Basel, Switzerland, 2001.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, second edition, 2001.

[12] J.L. Cohon and D.H. Marks. A review and evaluation of multiobjective programming techniques. *Water Resouces Research*, 11:208–220, 1975.

[13] C. Gellatt S. Kirkpatrick and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[14] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Massachusetts,USA, 1999.

[15] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York,USA, 1996.

[16] David B. Fogel. An introduction to evolutionary computation and some applications. In *Evolutionary Algorithms in Engineering and Computer science*, pages 23–41, England, 1999. John Wiley and Sons.

[17] E. Eiben and Jim E. Smith. *Introduction to evolutionary computing*. Springer, Germany, 2003.

[18] A. Hoffman. *Arguments on Evolution: A Paleontologist's Perspective*. Oxford University Press, New York,USA, 1989.

[19] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, USA, 1989.

[20] K. De Jong. Evolutionary computation: Recent developments and open issues. In *Evolutionary Algorithms in Engineering and Computer science*, pages 23–41, England, 1999. John Wiley and Sons.

[21] Melanie Mitchell. *An introduction to genetic algorithms*. MIT Press, USA, 1996.

[22] Michael D. Vose. *The Simple Genetic Algorithm Foundations and Theory*. The MIT Press, USA, 1999.

[23] G¨unter Rudolph. Convergence analysis of canonical genetci algorithms. In *IEEE Transaction on Neural Netwoks*, volume 5, pages 96–101, January 1994.

[24] Kenneth V. Price. An introduction to differential evolution. pages 79–108, 1999.

[25] Leandro N. d. Castro. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, London, 2002.

[26] R. Eberhart and J. Kennedy. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.

[27] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and theirApplications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[28] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, CA, 1993. Morgan Kaufman.

[29] N. Srinivas and K. Deb. Multiple objective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(2):221–248, 1994.

[30] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[31] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[32] Eckart Zitzler, Marco Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[33] Joshua Knowles and David Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 98–105, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[34] David W. Corne, Joshua D. Knowles, and Martin J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[35] Carlos A. Coello Coello and Gregorio Toscano Pulido. A Micro-Genetic Algorithm for Multiobjective Optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[36] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[37] Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE Congress on Evolutionary Computation*, pages 69–73. IEEE Press, 1998.

[38] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Congress on Evolutionary Computation,(CEC 2002)*, pages 1761–1676, Honolulu,Hawaii USA, 2002.

[39] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kauffmann Publishers, San Francisco, California, 2001.

[40] Efrén Mezura-Montes and Carlos A. Coello Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *Transactions on Evolutionary Computation*, 9(1):1–17, February 2005.

[41] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[42] T.P. Runanarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):248–249, September 2000.

[43] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[44] Xiaohui Hu and Russell Eberhart. Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, volume 5. Orlando, USA, IIIS, July 2002.

[45] Xiaohui Hu, Russell C. Eberhart, and Yuhui Shi. Engineering Optimization with Particle Swarm. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 53–57. Indianapolis, Indiana, USA, IEEE Service Center, April 2003.

[46] Ulrich Paquet and Andries p Engelbrecht. A New Particle Swarm Optimiser for Linearly Constrained Optimization. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, volume 1, pages 227–233, Piscataway, New Jersey, December 2003. Canberra, Australia, IEEE Service Center.

[47] Gregorio Toscano Pulido and Carlos A. Coello Coello. A constraint-handling mechanism for particle swarm optimization. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1396–1403, Portland, Oregon, USA, June 2004. IEEE Press.

[48] Kalmanje Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. *SPIE Proceedings: Intelligent Control and Adaptive Systems*, 1196:289–296, 1989.

[49] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective optimization using a micro-genetic algorithm. In L. Spector, E.D. Goodman, A. Wu, W. Langdon,

H.M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk M., Garzon, and E. Burke, editors, *Genetic and Evolutionary Computation Conference, GECCO,2001*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[50] Gregorio Toscano Pulido and Carlos A. Coello Coello. The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 252–266, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[51] Paul S. Andrews. An investigation into mutation operators for particle swarm optimization. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation, (CEC'2006)*, pages 3789–3796, Vancouver, Canada, July 2006.

[52] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[53] S.C. Esquivel and C.A. Coello Coello. On the use of particle swarm optimization with multimodal functions. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'2003)*, pages 1130–1136, Canberra, Australia, 2003. IEEE Press.

[54] Margarita Reyes-Sierra and Carlos A. Coello Coello. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[55] J. Moore and R. Chapman. Application of particle swarm to multiobjective optimization, 1999.

[56] Jonathan E. Fieldsend and Sameer Singh. A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, pages 37–44, Birmingham, UK, September 2002.

[57] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.

[58] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.

[59] Xiaodong Li. A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, pages 37–48. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.

[60] Thomas Bartz-Beielstein, Philipp Limbourg, Konstantinos E. Parsopoulos, Michael N. Vrahatis, Jörn Mehnen, and Karlheinz Schmitt. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1780–1787, Canberra, Australia, December 2003. IEEE Press.

[61] Margarita Reyes Sierra and Carlos A. Coello Coello. Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 505–519, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[62] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[63] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Test Suites. In Janice Carroll, Hisham Haddad, Dave Oppenheim, Barrett Bryant, and Gary B. Lamont, editors, *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, San Antonio, Texas, 1999. ACM.

[64] David A. Van Veldhuizen and Gary B. Lamont. Evolutionary computation and convergence to a pareto front. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22-25 1998. Stanford University Bookstore.

[65] Jason R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[66] Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science Vol. 496*, pages 193–197, Berlin, Germany, October 1991. Springer-Verlag.

[67] Rémy Viennet, Christian Fontiex, and Ivan Marc. Multicriteria Optimization Using a Genetic Algorithm for Determining a Pareto Set. *International Journal of Systems Science*, 27(2):255–260, 1996.

[68] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[69] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the CEC 2006 spe-

cial session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore, 2006.