Una Técnica para Manejo de Restricciones aplicable a Optimización Evolutiva Multiobjetivo

Tesis que para obtener el grado de Maestro
en Ciencias presenta:

Giovanni Lizárraga Lizárraga
Maestría en Ciencias de la Computación y Matemáticas Industriales
Centro de Investigación en Matemáticas

Asesor:
Dr. Salvador Botello Rionda
Centro de Investigación en Matemáticas

Asesor:
Dr. Arturo Hernández Aguirre
Centro de Investigación en Matemáticas

13 de Agosto del 2002

Resumen

En este trabajo se presenta una nueva técnica de optimización de funciones, basada en el paradigma de manejo de restricciones a través de algoritmos multiobjetivo que recuperan el llamado "frente de Pareto". Aunque el método aquí presentado puede funcionar bien con cualquier algoritmo multiobjetivo, para efectos de este trabajo se decidió emplear la Pareto Archived Evolution Strategy (PAES), debido a que presenta ciertas facilidades que se comentarán a lo largo de la presente investigación.

Una de las desventajas de emplear métodos multiobjetivo que recuperan el frente de Pareto para manejo de restricciones, es que estos algoritmos están diseñados para dispersar la población lo más ampliamente posible a lo largo del frente, por lo que no realizan una exploración puntual sobre la zona donde se encuentra el óptimo. La aportación del presente trabajo es una modificación que permite buscar con más detalle en la zona óptima y factible.

Los puntos esenciales de la modificación son, primero, un procedimiento de reducción (o "cortes") del espacio de búsqueda en torno a la zona óptima factible. Estos "cortes" están determinados por los mejores individuos encontrados al momento de "cortar". Otro punto importante es el uso de elitismo para preservar la mejor solución generada. Esto último debido a que, aunque la mayoría de los métodos multiobjetivo modernos tienen mecanismos de elitismo, estos están orientados a preservar un conjunto no dominado lo más cercano al frente de Pareto y lo mejor disperso posible, pero no conservan al mejor individuo factible generado hasta el momento.

La técnica fue probada con un conjunto de funciones con diferentes características para evaluar su desempeño. Además, se realizaron experimentos

con problemas de aplicación real, como es el diseño de armaduras de acero sujetas a cargas axiales.

Finalmente se presenta una técnica sencilla para la obtención del frente de Pareto de problemas multiobjetivo sujetos a restricciones, para los cuales también se realizaron varias pruebas.

Agradecimientos

A mis padres, Alejandro y Evangelina Lizárraga por hacerme la persona que soy y haberme mostrado el camino correcto a través del ejemplo.

A mis hermanos por su apoyo a todo lo largo de mis estudios en Guanajuato.

A mis abuelos, que en paz descansen, y a mis abuelas, por ser un gran ejemplo a seguir para mí.

A mis asesores, Salvador Botello y Arturo Hernández por su guía, su forma de trabajar, y por no dejarme rendirme en los momentos difíciles.

A todos mis maestros de posgrado, por todo lo que me enseñaron a lo largo de mis estudios.

A CIMAT por todo lo que me ha dado.

A mis compañeros de CIMAT, por todas las experiencias que viví junto a ellos. Muchas de las cosas más importantes que aprendí en mis estancia en CIMAT, se dieron fuera de las aulas.

A CONACYT, cuyo apoyo hizo posible mis estudios de posgrado.

Índice General

Índice de Figuras

Índice de Tablas

Introducción

Los algoritmos genéticos y estrategias evolutivas son técnicas para optimización de funciones que permiten obtener resultados razonablemente buenos, debido a lo cual se ha realizado un alto grado de investigación en esta área. Sin embargo, la concepción original de estos no considera el manejo de restricciones, las cuales se presentan muy comúnmente en los problemas reales, por lo que se ha estado trabajando en nuevas técnicas que permitan manejar restricciones con algoritmos evolutivos (algoritmos genéticos, programación evolutiva y estrategias evolutivas).

Otra área que se ha estado trabajando considerablemente es el manejo de múltiples funciones objetivo, (muy presentes también en problemas reales), debido a que los algoritmos evolutivos fueron diseñados para optimizar una sola función objetivo. El resultado ha sido una serie de técnicas que permiten obtener un conjunto de soluciones óptimas en el sentido de que ninguna de ellas es mejor que la otra. Dentro de estas técnicas existe un grupo que se basa en el llamado "criterio de optimalidad de Pareto", las cuales han dado buenos resultados.

Uno de los criterios que se ha utilizado para el manejo de restricciones es usar algoritmos multiobjetivo, considerando a las restricciones como nuevas funciones objetivo. De esta forma, se obtiene un conjunto de soluciones dispersas a lo largo de la zona factible y no factible, de las cuales se toma a la mejor de la zona factible. El problema con este enfoque es que da una aproximación muy burda al óptimo factible, debido a que los algoritmos multiobjetivo, especialmente los basados en el "criterio de optimalidad de Pareto", dispersan la población sin realizar una búqueda mas centrada en un punto específico.

Pensando en lo anterior, en el presente trabajo se presenta una propuesta para optimizar problemas en los que las restricciones se manejen con técnicas multiobjetivo, con la diferencia de que se realiza una búsqueda más centrada en torno al área donde se encuentra el óptimo global factible. También se trabaja el problema poco tratado en la literatura de la optimización de funciones multiobjetivo sujetas a restricciones.

La presente tesis se encuentra organizada en cinco capítulos. El capítulo 1 contiene una descripción general de lo que es la optimización, incluyendo sus conceptos básicos, tipos en que se divide y los métodos de trabajo más usados. En el capítulo 2 se presenta la computación evolutiva, que es un método muy usado en optimización. En el capítulo 3 se da una revisión del estado del arte de las técnicas multiobjetivo, así como de su uso para el manejo de restricciones. En el capítulo 4 se describe el método propuesto en este trabajo para la optimización de funciones sujetas a restricciones. En el capítulo 5 se describen los problemas de prueba y se presentan los resultados del desempeño del algoritmo propuesto. También se realizan comparaciones con otros métodos. Finalmente, se concluye la presente tesis con las conclusiones y el trabajo futuro a seguir.

Capítulo 1

Optimización

Optimizar recursos es una necesidad esencial para la vida moderna. Los ingenieros buscan diseños que cumplan con los requerimientos de funcionalidad y que además reduzcan los costos. Los empresarios buscan formas de mejorar la producción y disminuir los costos. Los transportistas buscan la manera de recorrer los puntos de descarga en el menor tiempo posible.

Pero la optimización no es exclusiva de los humanos, la naturaleza también optimiza [1]. Los sistemas físicos tienden a un estado de mínima energía. Las moléculas en un sistema químico aislado reaccionan unas con otras mientras el total de la energía potencial de sus electrones es minimizado. Los rayos de luz siguen caminos que toman el menor tiempo de viaje.

La optimización, también llamada programación matemática, es una herramienta muy importante para la toma de decisiones en problemas muy diversos. Comprende una serie de técnicas enfocadas a encontrar la mejor solución para un problema en particular.

Se puede definir a la optimización como el proceso de buscar en el dominio de un problema hasta encontrar una solución óptima, entendiéndose por solución óptima aquella que maximiza o minimiza la(s) función(es) objetivo y que cumple con una serie de restricciones impuestas en el problema [25]. No siempre es posible encontrar la solución óptima, para muchos casos sólo se obtienen aproximaciones.

1.1 Proceso de Optimización

Los pasos generales para resolver un problema de optimización en la práctica se dan a continuación [2]:

1.1.1 Definición del problema.

A diferencia de los ejemplos de los libros, los problemas en el mundo real son descritos inicialmente de forma vaga e imprecisa. Por lo tanto, el primer paso es dar una definición clara del problema, donde se incluye el objetivo a alcanzar, las variables involucradas, las restricciones a que está sujeto el problema, etc.

1.1.2 Formulación del modelo matemático.

Una vez definido el problema, es necesario plantearlo de forma que pueda manejarse matemáticamente. Un modelo matemático es una idealización o representación a base de fórmulas y símbolos que describen la esencia del problema. Si existen n valores a ser elegidos como solución para el problema, estos son llamados variables de decisión o de búsqueda, que para efectos de este trabajo se representarán como un vector de la forma $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$. Se debe definir también una medida de desempeño apropiada, que depende de las variables de decisión y se expresa normalmente como $f(\mathbf{x})$ la cual es llamada función objetivo.

En caso de que existan restricciones, se deben expresar en forma de desigualdades. Para efectos de esta tesis todas las restricciones se plantearán de la forma $g_i \leq 0$ donde $i \in \{1, .., m\}$, siendo m el número de restricciones. En caso de que la restricción sea una igualdad, ésta se convertirá en desigualdad al restar al lado izquierdo de la restricción una variable de holgura δ. En caso de que la desigualdad sea de la forma $\geq$, se convertirá a la forma $\leq$ multiplicandola por -1. En caso de que el lado derecho de la desigualdad sea una constante diferente de 0, se restará a ambos lados de la desigualdad el valor de dicha constante.

El modelo matemático para un problema será finalmente:

$$\begin{array}{l} \text{minimizar } f(\boldsymbol{x}) \\ \text{sujeto a:} \\ g_i(\boldsymbol{x}) \leq 0 \\ \text{donde:} \\ \boldsymbol{x} \in \mathcal{R}^n;\ n = \text{número de variables de decisión} \\ i \in \{1, ..., m\};\ m = \text{número de restricciones} \end{array} \tag{1.1}$$

En caso de que el problema sea maximizar $f(\boldsymbol{x})$, simplemente se redefinirá la función objetivo como $f(\boldsymbol{x})_{nueva} = -f(\boldsymbol{x})_{anterior}$.

Todas aquellas constantes y coeficientes que incluya el modelo, son llamados parámetros del modelo. La elección adecuada de estos parámetros puede tener una enorme influencia en los resultados. Si la solución varía considerablemente respecto a los parámetros, se presenta un segundo problema a solucionar que es la selección adecuada de éstos. Mientras menos varíe el resultado, más robusto es el modelo. Por esto es importante medir el desempeño del modelo cuando cambian los parámetros. A este proceso se le llama análisis de sensibilidad.

Un punto importante en la elaboración de un modelo, es mantener un equilibrio entre la fidelidad de la descripción del problema y la factibilidad de solución. Si el modelo es muy simple, los resultados obtenidos pueden no corresponder con la realidad del problema. Si es muy complejo, su solución puede ser muy difícil o imposible de obtener.

Otro punto a considerar es la correcta construcción de la función objetivo. Debe estar basada en una medida cuantitativa del objetivo identificado cuando el problema fué definido. Si existen diversos objetivos, estos normalmente son combinados o aglomerados en una medida compuesta.

1.1.3 Derivando soluciones a partir del modelo

Después de la formulación del modelo matemático, la siguiente etapa es generar un procedimiento para solucionar el modelo. Existen diversos procedi-

mientos para resolver problemas de optimización, los cuales en la práctica, se implementan en una computadora.

Para efecto de esta tesis, los problemas tratados se encontrarán ya bien definidos y el modelo matemático bien formulado. El algoritmo de solución en sí tendrá la función del modelo a probar, con sus propios parámetros.

1.1.4 Probando el modelo

Una vez creado un modelo, debe revisarse su efectividad por medio de una serie de pruebas. Un modelo recién creado puede tener una gran cantidad de errores que se deben identificar y corregir. Estos errores pueden ser condiciones del problema no contempladas, relaciones entre los parámetros o las variables que no se tomaron en cuenta, etc. Este proceso de probar y corregir debe repetirse hasta que el modelo de consistentemente buenos resultados para un número significativo de pruebas. A este proceso se le llama validación del modelo [2].

Para la validación, es común usar un conjunto de prueba ya establecido, cuya solución se conoce. De esta forma es fácil medir el desempeño, además de permitir comparar el modelo con otros anteriores. El conjunto de prueba debe ser lo más variado posible, de manera que represente una amplia gama de situaciones que se podrían presentar a futuro y permita conocer cuál es el comportamiento del modelo en general.

1.2 Tipos de optimización

En optimización, los modelos que definen el problema pueden presentar diferentes características, según las cuales es posible que convenga más usar un método de optimización en lugar de otro. Algunos métodos consumen más tiempo que otros, pero obtienen soluciones más precisas. En otros casos es posibles que algunos métodos ni siquiera sean capaces de encontrar una solución. Por lo anterior, es importante conocer el tipo de problema que se tiene y que cuáles son las ventajas y desventajas de los métodos que existen para resolverlo.

A continuación se muestran brevemente algunos de los criterios en los que se clasifican los problemas que se pueden presentar en optimización [1]. Sólo se pretende dar un pequeño panorama de la diversidad que existe en optimización y no se intenta ser exhaustivo.

1.2.1 Problemas de variables discretas y continuas

En algunos problemas las variables pueden tomar valores enteros solamente, como pueden ser los problemas de transporte. Por ejemplo: n fábricas producen cierta cantidad de tractores cada una, que deben ser tranportados a z destinos donde cada destino necesita una cantidad particular de tractores. Existe un costo asociado a tranportar de cada fábrica a cada destino. ¿Cuántos tractores de cada fábrica deben transportarse a cada destino? Claramente el resultado de este problema deben ser números enteros pues no se pueden transportar, por ejemplo, 3.5 tractores de una fábrica a un destino.

También se pueden considerar dentro de esta categoría, problemas en los cuales aunque las variables pueden tomar valores fraccionarios, estos son discontinuos y discretos (o sea que deben tomarse de un conjunto predeterminado de valores), por ejemplo encontrar el diámetro óptimo para un sistema de tuberías, los cuales tienen valores que son múltiplos de $\frac{1}{4}$". Otro ejemplo es minimizar el peso de una estructura de acero donde las variables son números enteros asociados a un catálogo de perfiles de acero.

El término de optimización discreta se refiere en general a problemas en los cuales el conjunto de posibles soluciones es finito mientras que en la optimización continua el conjunto de posibles soluciones es infinito. La optimización discreta puede ser muy complicada, pues dos puntos adyacentes pueden tenen un comportamiento muy distinto entre sí, mientras que en la optimización continua normalmente se puede deducir información de los alrededores basándose en la información de un punto [1].

1.2.2 Optimización con restricciones y sin restricciones

Una de las principales divisiones que se puede hacer entre los problemas de optimización son aquellos que tienen restricciones y aquellos que no las tienen.

Los problemas de optimización sin restricciones provienen de muchas aplicaciones prácticas. Si existen restricciones naturales en las variables, como rangos de validez, es posible descartar esas restricciones y asumir que no afectan a la solución óptima, o simplemente diseñar el método de exploración y búsqueda para que, de forma natural, nunca se sobrepasen los rangos permitidos. Muchos problemas sin restricciones pueden surgir a partir de otros con restricciones reformulando la función objetivo para que tome en cuenta las restricciones como penalizaciones.

Los problemas de optimización con restricciones surgen de modelos que incluyen restricciones de forma explícita, las cuales pueden tomar la forma de simples rangos para los valores de las variables, relaciones lineales, hasta fórmulas de igualdad y desigualdad que representan complicadas relaciones no lineales entre las variables.

1.2.3 Optimización lineal y no lineal

Cuando tanto la función objetivo como las restricciones son lineales se dice que el problema es de programación lineal. Existen una gran cantidad de problemas prácticos de áreas como programación de metas y economía que corresponden a este modelo, por lo que este tipo de optimización ha sido profundamente investigado. En este caso la zona factible está formada por un conjunto de hiperplanos interconectados.

Cuando por lo menos una de las restricciones o la función objetivo son no lineales entonces se trata de un problema no lineal, los cuales surgen comúnmente en física, ingeniería y ciencias en general.

1.2.4 Optimización de un solo objetivo y multiobjetivo

En la formulación 1.1 se consideró que sólo se deseaba optimizar una función, pero en la vida real es común que sea más de una la función que se desea optimizar. Existen varios enfoques para resolver este tipo de problemas, en algunos se aprovecha cualquier información disponible sobre la importancia de los objetivos, como son los métodos de aglomeración, donde las funciones son combinadas en un valor escalar único dando cierto peso a cada una. Otro enfoque útil cuando no se tiene información o preferencia sobre las funciones objetivo es el criterio de optimalidad de Pareto, el cual se tratará en el capítulo 3.

1.3 Métodos de solución

Existen diversas formas de resolver los problemas de optimización; una de ellas son los métodos analíticos directos, en los que las derivadas de las funciones se igualan a cero y se trata de encontrar los mejores puntos que resuelven la ecuación resultante. Estos métodos tienen un uso muy limitado, entre otras cosas porque sólo pueden trabajar con funciones continuas que o cumplen con ciertos criterios de diferenciabilidad.

Más generales son los métodos iterativos en los que a partir de uno o varios puntos iniciales se realiza una exploración tratando de acercarse cada vez más al óptimo. Algunos algoritmos usan información de iteraciones anteriores, mientras que otros usan la información local de un punto.

Un buen algoritmo debe tener las siguientes propiedades [1]:

- Ser robusto. Debe tener un buen comportamiento en una amplia variedad de problemas de su clase, para toda inicialización razonable.
- Eficiencia. No debe requerir excesivo tiempo de cálculo y espacio.
- Concordancia. Debe se capaz de identificar la solución con precisión, sin ser demasiado sensible a errores de los datos o de redondeo de la computadora.

Algunas de estas propiedades pueden estar en conflicto unas con otras, como por ejemplo rapidez con requerimientos de memoria o funcionar para un gran número de casos (robusto) pero no ser muy preciso en sus resultados (concordancia).

A continuación se mencionan algunas técnicas usadas en optimización:

Método simplex. Es una técnica desarrollada en los cuarentas para resolver problemas de optimización lineal, aunque en ocasiones se utiliza para algunos problemas no lineales dada su efectividad y la potencia de los paquetes comerciales que se han desarrollado basándose en este método.

Métodos de gradiente. Se utilizan en programación no lineal. Se guían por medio del gradiente de la función objetivo para realizar su búsqueda y existe una gran variedad de ellos.

Computación evolutiva. Desarrollada a partir de los sesentas, incluye una serie de técnicas que tratan de simular procesos naturales de la biología, como son la reproducción, selección natural y evolución. Son aplicables a una gran variedad de problemas, fáciles de implementar y muy eficientes aunque suelen tener poca presición en los resultados. Una propiedad importante de los métodos de la computación evolutiva es su capacidad de evitar los mínimos locales. Un mínimo local es un punto $\boldsymbol{x}^*$ si cumple con las restricciones y para cierta cantidad δ, $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ para toda $\boldsymbol{x}$ que cumpla las restricciones tal que $|\boldsymbol{x}^* - \boldsymbol{x}| < \delta$. Esta capacidad de evadir mínimos locales se debe a los mecanismos que tienen incorporados para realizar la exploración. Se hablará más acerca de la computación evolutiva en el siguiente capítulo.

Capítulo 2

Computación Evolutiva

La computación evolutiva comprende una serie de técnicas surgidas a partir de los sesentas, que tratan de imitar los procesos naturales de la evolución y selección natural. Las principales áreas en que se divide son: los Algoritmos Genéticos (AG), propuestos por John Holland [16]; las Estrategias Evolutivas (EE), en cuyo desarrollo inicial estuvieron involucrados Ingo Rechenberg, Hans-Paul Schwefel y Paul Bienert [18]; y la Programación Evolutiva (PE), propuesta por Fogel [34].

Las diferencias entre los distintos paradigmas que conforman la computación evolutiva provienen principalmente del enfoque y motivación original que tuvo cada uno de ellos, aunque actualmente estas divisiones son un tanto artificiales [15].

En este capítulo se da una descripción de los procesos naturales en los que se basa la computación evolutiva y una explicación de cómo se implementa.

2.1 Conceptos biológicos

2.1.1 Genética

Todo ser viviente está formado por células. Toda célula contiene un conjunto de cadenas de ADN (ácido desoxirribonucleico) llamados cromosomas. Estas cadenas forman un código que permite reconstruir por completo a un organismo viviente con todas sus características. La unidad básica de este código son los genes, formados por bloques de ADN. Cada gene determina una característica del individuo (por ejemplo el color de los ojos, la forma del pelo, etc) y es la unidad básica de la herencia. Al conjunto de todos los cromosomas de un organismo se le llama genoma. A las características visibles de los individuos, como tamaño, color, etc. se le llama fenotipo y son resultado de las diferentes configuraciones de los genes. A la codificación de estas características dentro del genoma se le llama genotipo. Es decir, la información contenida en los códigos genéticos (genotipo), dará como resultado un conjunto de rasgos visibles (fenotipo) en el organismo resultante una vez que éste se desarrolle.

Dependiendo del tipo de organismos los cromosomas pueden tener dos tipos de estructura: diploide y haploide. Los organismos cuyos cromosomas vienen en pares son llamados diploides mientras que aquellos cuyos cromosomas vienen en un solo arreglo son llamados haploides. En el caso de los organismos diploides, durante la reproducción sexual, los cromosomas de los hijos se forman tomando uno de los genes de la cadena diploide de cada uno de los padres para cada gene. Para los organismos haploides, simplemente se toma un gen de uno u otro padre para cada gene. Durante la reproducción pueden darse errores al copiar la información genética de padres a hijos dando por resultado hijos con características que pueden ser completamente diferentes a las de los padres. A este fenómeno se le llama mutación.

2.1.2 Evolución

Charles Darwin [35] propuso la teoría de la evolución de las especies, según la cual, los individuos cambian sus características de manera que tengan mayor

capacidad para sobrevivir en su medio ambiente. Algunos de los elementos mas importantes de este proceso son: selección natural, reproducción, mutación e influencia del medio ambiente. La selección natural es un proceso mediante el cual los individuos mejor adaptados al medio ambiente son los que sobreviven y llegan a la edad adulta. A la adaptación de un individuo al medio ambiente se le llama aptitud. Mientras mayor sea la aptitud de un individuo mayor probabilidad tiene de sobrevivir y tener descendencia. La reproducción es el medio mediante el cual un individuo genera decendencia, y puede ser sexual (dos individuos) o asexual (un solo individuo). Tanto de la reproducción como de la mutación ya se habló en la sección 2.1.1. El medio ambiente ejerce una gran influencia en la aptitud de los organismos. Un organismo con alta aptitud en un desierto puede tener una aptitud muy baja en un bosque. Es decir, la aptitud es relativa al medio ambiente.

Durante el proceso evolutivo pueden darse ciertos fenómenos genéticos [15] tales como:

Epístasis: el efecto de genes individuales y sus interacciones no son lineales respecto a la aptitud del organismo.

Epigénesis: el ambiente controla si un gene es expresado o no.

Pleitropía: un gene afecta a más de una característica fenotípica.

Poligenia: un número de genes afectan una misma característica fenotípica.

Estas fenómenos pueden afectar la evolución de un individuo y hacen la interacción entre el fenotipo y el genotipo mas compleja.

2.2 Tipos de computación evolutiva

De una forma general todos los diferentes tipos de computación evolutiva contienen los siguientes pasos [3], los cuales están inspirados en los conceptos vistos en la sección anterior:

1. Generar una población aleatoria inicial de un tamaño mayor a cero donde cada individuo representa una solución potencial al problema.

2. Seleccionar a los individuos más aptos de acuerdo al resultado de evaluarlos con una función de aptitud.

3. Aplicar operadores de reproducción (cruza, mutación, etc.) para generar nuevos individuos.

4. Iterar a partir del paso 2 hasta alcanzar las condiciones de paro.

En las siguientes subsecciones se da una explicación de dos de las principales ramas de la computación evolutiva y una descripción de cómo se implementan sus algoritmos correspondientes, (los algoritmos genéticos y las estrategias evolutivas). Estas descripciones no pretenden abarcar todos los tipos de operadores con que cuentan éstas, sino dar solamente un panorama de las implementaciones más comunes.

2.2.1 Algoritmos genéticos

Los algoritmos genéticos fueron propuestos por Holland en 1975 [16] dando un fundamento teórico de estos donde hace hincapié en el uso de representación binaria, operadores de cruza y el teorema de los esquemas. La mutación es un operador secundario que tiene la función de evitar que la búsqueda quede atrapada en mínimos locales y mantener diversidad en la población.

La implementación de un algoritmo genético se realiza como sigue:

Primero se debe determinar el tamaño de la cadena binaria que representa a las variables de búsqueda. Si se tiene que el espacio de búsqueda tiene un límite superior e inferior $\overline{x}_i$ y $\underline{x}_i$ respectivamente donde $i \in (1, \ldots, n)$, donde n es el número de variables de búsqueda, y se desea que las soluciones tengan un precisión de p cifras decimales, entonces el espacio de búsqueda debe ser dividido en $(\overline{x}_i - \underline{x}_i) \cdot 10^p$ secciones. Si llamamos l_i al mínimo entero que cumple con la relación $(\overline{x}_i - \underline{x}_i) \cdot 10^p \leq 2^{l_i} - 1$, entonces representar a cada variable de búsqueda x_i como una cadena de números binarios de longitud l_i dará una precisión igual o mayor a la deseada. De manera que la representación fenotípica de x_i se calcula con la siguiente fórmula:

$$x_i = \underline{x}_i + Binario\_a\_Decimal(genotipoX_i) \cdot \frac{\overline{x}_i - \underline{x}_i}{2^{l_i} - 1} \tag{2.1}$$

donde $genotipoX_i$ es la cadena binaria que representa a la variable x_i dentro del individuo y $Binario\_a\_Decimal()$ es una función que convierte una cadena binaria en su equivalente decimal. El tamaño total de la cadena de bits o genoma que representa al individuo será $l = \sum_{i=1}^{n} l_i$ donde los primeros l_1 bits corresponden a la variable x_1, los siguientes l_2 bits representan a la variable x_2, y así sucesivamente.

Existe un problema relacionado a la representación binaria conocido como "risco de Hamming" que se presenta debido a que dos números consecutivos pueden ser muy diferentes entre sí en su representación binaria. Por ejemplo, el número siete en binario es 0111 mientras que el ocho es 1000, si la solución estuviera ubicada en ocho, y la población se estuviera acercando por la izquierda, una vez estando en siete será muy difícil pasar a ocho.

Para resolver el problema expuesto arriba, es muy común utilizar los códigos de Gray. Este es un tipo de codificación binaria en el cual dos números consecutivos difieren entre sí solamente en un bit, como se muestra en la tabla 2.1.

Binario	Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

Tabla 2.1: Comparación de codificación en binario y Gray

La inicialización de la población es un proceso muy sencillo, solamente se generan de manera aleatoria los genomas como cadenas de bits de ceros y unos.

El proceso de selección puede realizarse de diversas formas, como son ruleta, jerarquización, torneo binario, etc. La selección por ruleta se describe a continuación:

1. Calcular la suma $sum = \sum_{i=1}^{N}$ de la aptitud de los individuos, donde N es el tamaño de la población.

2. Generar un número al azar p entre 0 y sum.

3. Ir sumando la aptitud de cada individuo en un acumulado a hasta que $a \geq p$ y pasar a la siguiente etapa al último individuo cuya aptitud fué sumada.

4. Repetir el paso 2 y 3 hasta haber obtenido N individuos.

Este tipo de selección puede causar problemas de convergencia prematura si la diferencia de aptitudes entre los mejores y peores individuos es grande. La selección por jerarquías resuelve este problema asignándole una aptitud de uno al peor individuo, dos al segundo peor, y así sucesivamente, con lo que existe una mejor proporción de las aptitud de los individuos. La selección por torneo binario consiste en tomar a dos individuos al azar y pasar a la siguiente etapa al individuo que tenga mayor aptitud.

En la etapa de cruza se toma a dos individuos al azar y con cierta probabilidad P_c se cruzan para generar dos nuevos individuos. De lo contrario, los dos individuos pasan a la siguiente etapa sin ser modificados. El valor de P_c generalmente es alto, entre 0.60 y 1.00.

La cruza entre los individuos puede realizarse de diversas formas. Algunas de las más populares son cruza de un punto, cruza de dos puntos y cruza uniforme. En la cruza de un punto se elige un punto de cruza, que no es más que una posición dentro de la cadena de bits, de forma aleatoria. Todos los bits del hijo 1 hasta antes del punto de cruza son tomados del padre 1, mientras que los bits a partir del punto de cruza son tomados del segundo padre. Para el hijo 2 se toman los bits de forma inversa (véase tabla 2.2).

La cruza de dos puntos es similar, sólo que en vez de generar un punto al azar se generan dos, y las secciones de código se toman de forma alternada de uno y otro padre. Finalmente, en la cruza uniforme, para cada bit del hijo 1, con probabilidad de $\frac{1}{2}$ se toma el bit correspondiente del primer padre o del segundo. El hijo 2 toma el valor del bit del otro padre.

El operador de mutación trabaja de la siguiente forma. Para cada bit de cada individuo de la población se genera un número al azar entre cero y

Padre 1	10|0110
Padre 2	00|1001
Hijo 1	10|1001
Hijo 2	00|0110

Tabla 2.2: Ejemplo de cruza de un punto. El punto de cruza se simboliza con |.

uno. Si este número es menor a cierta probabilidad P_m se cambia el bit. De lo contrario se deja igual. Véase tabla 2.3. El valor de P_m es generalmente pequeño, entre 0.01 y 0.5. Si P_m es demasiado grande el algoritmo genético se comporta como una búsqueda aleatoria.

Individuo original	00**1**1110101
Individuo mutado	00**0**1110101

Tabla 2.3: Ejemplo de mutación. El bit mutado en negritas.

2.2.2 Estrategias evolutivas

Las estrategias evolutivas nacieron en los sesentas en Alemania. A diferencia de los algoritmos genéticos, las estrategias evolutivas usan representación real. Inicialmente se utilizaron para resolver problemas muy sencillos y unimodales, por lo que se utilizaron poblaciones pequeñas y una alta presión de selección en la que los peores individuos tenían probabilidad cero de sobrevivir. La primera implementación de las estrategias evolutivas fue la (1 + 1)-EE en la que la población sólo contaba con un individuo, el único operador que existía para generar nuevos individuos era la mutación, basada en funciones gaussianas, donde una variable era modificada de la siguiente forma $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + N(0,\boldsymbol{\sigma})$ donde N(0, $\boldsymbol{\sigma}$) regresa un valor al azar con distribución normal de media cero y varianza $\boldsymbol{\sigma}$.

El algoritmo básico de la estrategia (1 + 1) es como sigue:

1. Generar un padre inicial al azar.

2. Generar un hijo a partir de una mutación del padre.

3. Si el hijo es mejor que el padre, sustituir al padre por el hijo. De lo contrario desechar al hijo.

4. Hasta alcanzar un número de iteraciones dado regresar al paso 2.

Poco después se introdujo el concepto de población, apareciendo las estrategias $(\mu + 1)$-EE, (μ, λ)-EE y $(\mu+\lambda)$-EE las cuales incluyen un operador de cruza entre los padres, aunque la mutación se mantiene como el operador principal de exploración. En la estrategia $(\mu + 1)$-EE, μ padres generan un hijo y éste sustituye al peor de los padres. En (μ, λ)-EE, μ padres generan λ hijos, después se seleccionan los μ mejores hijos como los padres de la siguiente generación. La estrategia $(\mu + \lambda)$-EE es muy similar a la anterior, con la diferencia de que los μ padres de la siguiente generación son seleccionados del conjunto fomado por la unión de padres e hijos de la generación actual.

Con la llegada de las estrategias (μ, λ)-EE y $(\mu+\lambda)$-EE apareció una de las características más distintivas de las estrategias evolutivas, la autoadaptación de los parámetros de control de la búsqueda, o sea las desviaciones estándar de las mutaciones $\boldsymbol{\sigma}$ [17]. Las $\boldsymbol{\sigma}$ pasaron a formar parte de la información de cada individuo y quedaron sujetas a las operaciones de cruza y mutación, de manera que los individuos que mejor adapten sus parámetros para realizar mejor la búsqueda tendrán mejor aptitud y serán seleccionados para las sigientes generaciones. Se tiene una σ_i para cada variable de búsqueda x_i de cada individuo.

Anteriormente se mantenía un valor fijo para $\boldsymbol{\sigma}$ durante toda la búsqueda con la desventaja de que la convergencia era muy lenta. Después se introdujo la regla de éxito $\frac{1}{5}$ para tratar de acelerar la convergencia. Esta regla establece que si la relación de mutaciones exitosas respecto al número total de mutaciones es mayor a $\frac{1}{5}$ entonces se aumenta el tamaño de $\boldsymbol{\sigma}$. Por otro lado, si esta relación es menor a $\frac{1}{5}$ entonces se diminuye la magnitud de $\boldsymbol{\sigma}$. Si la relación es igual a $\frac{1}{5}$ se mantiene igual. El problema con la regla de éxito $\frac{1}{5}$ es que en ciertos problemas llevar a una convergencia prematura [33].

El algoritmo básico para las estrategias $(\mu + \lambda)$ y (μ, λ) es como sigue:

1. Generar población al azar.

2. Cruzar la población de μ individuos para generar λ hijos.

3. Mutar cada uno de los hijos.

4. Seleccionar μ individuos para la nueva generación. Para la estrategia $(\mu + \lambda)$-EE se escogen de la unión de las dos poblaciones. Para la estrategia $(\mu+\lambda)$-EE se escogen de la población de hijos exclusivamente.

5. Hasta terminar el número de iteraciones preestablecido, volver al paso 2.

La representación en las EE es fenotípica y consta, generalmente, de números reales pues normalmente las EE son usadas para minimizar funciones con variables continuas.

La población inicial se genera de forma aleatoria dentro del espacio de búsqueda con la siguiente fórmula:

$$x_i = (\overline{x}_i - \underline{x}_i) \cdot random() + \underline{x}_i \tag{2.2}$$

donde $\overline{x}_i$ y $\underline{x}_i$ son los límites superior e inferior, respectivamente, del espacio de búsqueda para la variable x_i. La función $random()$ genera un número al azar con distribución uniforme entre cero y uno. La magnitud de las $\boldsymbol{\sigma}$ se inicializa según [18]:

$$\sigma_i = \frac{(\overline{x}_i - \underline{x}_i)}{\sqrt{n}}, \quad i \in \{1, \ldots n\} \tag{2.3}$$

donde n es el número de variables de búsqueda. Estos valores también pueden usarse como límite superior para el valor de σ_i [12].

El proceso de selección, como se mencionó anteriormente, es determinístico y con alta presión de selección. Los mejores individuos se seleccionan como padres de la siguiente generación, los peores individuos mueren.

Existen diversas maneras de hacer la cruza en las EE. A continuación se mencionan algunas de ellas [17]:

Discreta: cada uno de los valores del hijo se toma de manera aleatoria de uno u otro padre (existe un tipo de cruza llamado panmítica en que un padre es cruzado con varios padres de la población para generar un hijo). Como ejemplo, véase la tabla 2.4. Estos tipos de cruza pueden aplicarse tanto a las variables de búsqueda $\boldsymbol{x}$ como a las variables que guían la búsqueda ($\boldsymbol{\sigma}$) pudiendo incluso hacerse diferente para cada una de ellas.

Padres	
Padre 1	5.2, 6.2, 1.2
Padre 2	3.1, 2.2, 5.2
Posibles hijos	
Hijo 1	5.2, 6.2, 1.2
Hijo 2	5.2, 6.2, 5.2
Hijo 3	5.2, 2.2, 1.2
Hijo 4	5.2, 2.2, 5.2
Hijo 5	3.1, 6.2, 1.2
Hijo 6	3.1, 6.2, 5.2
Hijo 7	3.1, 2.2, 1.2
Hijo 8	3.1, 2.2, 5.2

Tabla 2.4: Ejemplo de cruza discreta

Intermedia: Los valores que toma el hijo son el promedio de los valores de los padres. Como ejemplo, véase la tabla 2.5.

Padres	
Padre 1	5, 6, 1
Padre 2	3, 2, 4
Posibles hijos	
Hijo	4, 4, 2.5

Tabla 2.5: Ejemplo de cruza intermedia

Generalizada intermedia: el hijo adquiere un valor intermedio entre los valores del padre uno y el padre dos basados en la siguiente fórmula:

$$valor_{hijo} = (valor_{padre2} - valor_{padre1}) \cdot \eta + valor_{padre1} \tag{2.4}$$

donde η es un número real generado al azar con distribución uniforme entre cero y uno.

La mutación consiste en alterar las variables tanto de control como objetivo ($\boldsymbol{x}$, $\boldsymbol{\sigma}$) a través de números aleatorios. Son las mutaciones las que realmente realizan la búsqueda. Primero se mutan las variables de control y después las variables objetivo. La regla para mutar la $\boldsymbol{\sigma}$ es la siguiente:

$$\sigma_i^{t+1} = \sigma_i^t \exp(\tau' N(0,1) + \tau N_i(0,1)) \tag{2.5}$$

donde $N(0,1)_i$ son números aleatorios de distribución normal con media cero y varianza 1, que se generan para cada σ_i, mientras que $N(0,1)$ es otro número aleatorio que permanece fijo durante la mutación. τ y τ' son constantes equivalentes al factor de aprendizaje. Existen diversos valores propuestos en la literatura. Para este trabajo se considerarán los siguientes: $\tau = \frac{\varphi}{\sqrt{2\sqrt{n}}}$, $\tau' = \frac{\varphi}{\sqrt{2n}}$ donde n es el número de variables objetivo y φ es el llamado índice de aprendizaje y normalmente (aunque no necesariamente) se fija a 1.

Después de mutar a las variables de control se aplica la siguiente regla para mutar a las variables objetivo: $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + N(0,\boldsymbol{\sigma})$ donde $N(0,\boldsymbol{\sigma})$ es una función que regresa un valor de distribución gaussiana con media cero y varianza $\boldsymbol{\sigma}$.

2.3 Otros conceptos de computación evolutiva

2.3.1 Nichos

En ocasiones es posible que se desen obtener varias soluciones a un mismo problema, como en el caso de las funciones multimodales donde pueden existir varios mínimos. Dado que los algoritmos genéticos tienden a converger a un solo, punto se han diseñado técnicas para dispersar la población. Una de las más conocidas es la técnica de nichos (sharing). La idea es que en una región determinada, los recursos que ésta posee deben ser repartidos entre

los individuos que la habitan. Por lo tanto, mientras más individuos hay, menos recursos le corresponden a cada uno y la posibilidad de supervivencia disminuye. Esto se refleja en la función objetivo cuyo valor es disminuido dependiendo del número de individuos que estén en las cercanías. La aptitud final de un individuo será:

$$f_{final,i} = \frac{f_i}{\sum_{j=1}^{N} sh(d(i,j))} \tag{2.6}$$

donde N es el tamaño de la población. La función $d(i,j)$ devuelve la distancia entre los individuos j e i. La función $sh(d)$ es la función de nicho que tiene la forma:

$$\begin{cases} 1 - \left(\frac{d}{\sigma_{nicho}}\right)^{\alpha} & \text{si } d < \sigma_{nicho} \\ 0 & \text{de otra forma} \end{cases} \tag{2.7}$$

donde σ_{nicho} es el radio del nicho, un parámetro definido por el usuario que es la distancia máxima que debe haber entre un individuo a otro para que sus aptitudes comiencen a ser afectadas. El parámetro α normalmente se fija a 1. La función $d(i,j)$ puede aplicarse en el genotipo, el fenotipo y como se verá en el siguiente capítulo, en el espacio de las funciones objetivo cuando éstas son más de una.

2.3.2 Elitismo

El elitismo consiste en no permitir que se pierdan los mejores individuos encontrados hasta el momento. El mecanismo puede ser simplemente copiar al(los) mejor(es) individuos actuales a la población siguiente. Las estrategias evolutivas $(1+1)$, $(\mu+1)$ y $(\mu+\lambda)$ son elitistas por sí mismas.

Capítulo 3

Optimización Multiobjetivo

La mayoría de los problemas en la vida real suelen tener más de un solo objetivo a optimizar, los cuales normalemente están en conflicto unos con otros. A veces se cuenta con información sobre estos objetivos, como son el orden de importancia o los valores finales a los que se desea llegar; a veces no se cuenta con ninguna información previa.

A través de los años, se han desarrollado una serie de métodos especialmente diseñados para optimizar problemas con varias funciones objetivo, muchos de ellos basados en computación evolutiva.

En este capítulo se se introducen los conceptos básicos de la optimización multiobjetivo y se hace una breve descripción de los principales métodos, basádos en computación evolutiva, que se han propuesto en los últimos años para resolver este problema. También se trata brevemente la optimización sujeta a restricciones a través de métodos multiobjetivo.

3.1 Definición del problema

La optimización multiobjetivo consiste en encontrar un vector de soluciones que represente el mejor compromiso posible entre todas las funciones objetivo

a optimizarse. El modelo matemático es el siguiente:

$$\begin{aligned}
&\text{minimizar } \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\} \\
&\text{sujeto a:} \\
&g_i(\boldsymbol{x}) \leq 0 \\
&\text{donde:} \\
&\boldsymbol{x} \in \mathcal{R}^n;\ n = \text{número de variables de decisión} \\
&i \in \{1, ..., m\};\ m = \text{número de restricciones} \\
&k = \text{número de funciones objetivo}
\end{aligned} \tag{3.1}$$

Se le llama a $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ vector de variables de decisión. Se desea determinar del conjunto $\mathcal{S}$ de todos los vectores que satisfacen las restricciones, el conjunto de valores x_1^*, x_2^*, ...,x_n^* que generan los valores óptimos para todas las funciones objetivo.

3.2 Criterio de optimalidad de Pareto

En la mayoría de los problemas no existe un punto que minimice todas la funciones al mismo tiempo, por lo que se usa otro criterio para decidir que soluciones son las adecuadas. A este se le llama criterio de optimalidad de Pareto, en honor a su creador Vilfredo Pareto [20]. Para entender este criterio primero se define el concepto de dominancia de Pareto:

$$\begin{aligned}
&\boldsymbol{x}, \boldsymbol{y} \in S \\
&\boldsymbol{x} \prec \boldsymbol{y} \quad si\ \forall i: \\
&f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}); \quad i \in (1, \ldots, k) \\
&y \\
&\exists r: f_r(\boldsymbol{x}) < f_r(\boldsymbol{y})
\end{aligned} \tag{3.2}$$

el símbolo $\prec$ se utiliza para representar la relación de dominancia de manera que $\boldsymbol{x} \prec \boldsymbol{y}$ significa $\boldsymbol{x}$ domina a $\boldsymbol{y}$.

Se dice que un vector $\boldsymbol{x}^*$ es un óptimo de Pareto si no existe ningún vector $\boldsymbol{y} \in S$ tal que $\boldsymbol{y} \prec \boldsymbol{x}$. Una característica de los óptimos de Pareto es que no

es posible mejorar una de sus funciones objetivo sin empeorar alguna(s) de las otras. El conjunto de todos vectores $\boldsymbol{x}^* \in \mathcal{S}$ que son óptimos de Pareto se les llama conjunto de Pareto.

Se le llama frente de Pareto al conjunto de puntos $\boldsymbol{f}$ tal que $\boldsymbol{f} = \{f_1(\boldsymbol{x}^*), f_2(\boldsymbol{x}^*), \ldots f_k(\boldsymbol{x}^*)\}$.

3.3 Criterios para el manejo de múltiples objetivos

A continuación se da una breve descripción de algunos algoritmos clásicos para optimizar problemas multiobjetivo.

3.3.1 Suma ponderada

En este método las funciones objetivo son sumadas para formar una sola con la siguiente fórmula:

$$F = \sum_{j=1}^{k} w_j f_j(\mathrm{x}) \tag{3.3}$$

donde w_j es el peso usado para la función objetivo j y k es el número de funciones a minimizar. Comúnmente la suma de los pesos w_j es igual a uno. El procedimiento consiste en escoger un juego de pesos w y optimizar hasta converger a un punto, después escoger otro juego de pesos y volver a correr el programa para obtener otro punto, y repetir esto tantas veces como puntos se deseen esperando que estos formen parte del conjunto de óptimos de Pareto. Este método es eficiente si se tiene conocimiento del tipo de solución que se desea y se escoge un vector de pesos adecuado. Sin embargo, si lo que se desea es encontrar el frente de Pareto, se necesita un número considerable de corridas para aproximarlo. También tiene la desventaja de ser incapaz de recuperar correctamente el frente de Pareto cuando éste no es convexo [26].

3.3.2 Método de la perturbación ϵ

Para evitar la dificultad antes mencionada, el problema es replanteado como de optimización de un solo objetivo usando las funciones objetivo como restricciones excepto por una que permanece como función a optimizarse. El planteamiento es como sigue:

$$\begin{aligned} &\textit{Minimizar} \quad f_r \\ &\textit{sujeto a:} \\ &f_j(\mathrm{x}) \leq \epsilon_j, \quad \forall j \neq r \end{aligned} \tag{3.4}$$

Para encontrar diversas soluciones del conjunto de óptimos de Pareto, se usan diferentes valores de ϵ_j. Para que el método funcione es necesario poseer cierto conocimiento sobre los valores más adecuados de ϵ.

3.3.3 Programación por metas

Este método se aplica cuando el diseñador tiene metas claras que desea alcanzar usando éstas para crear una nueva función objetivo que consiste en minimizar las desviaciones absolutas de las metas respecto a los valores de las funciones objetivo. El problema se puede plantear así:

$$\textit{minimizar} \sum_{i=1}^{k} |f_i(\boldsymbol{x}) - T_i|, \quad \boldsymbol{x} \in \mathcal{S} \tag{3.5}$$

donde T_i es el valor de la meta propuesta para la i-ésima función objetivo y S es el espacio de soluciones factibles.

Este método es rápido, pero depende mucho de los valores propuestos para las metas.

3.3.4 Desventajas

En general estos métodos tienen diversas desventajas [26] como son:

1. Es necesario aplicar varias veces el algoritmo para encontrar diversas soluciones de Pareto.
2. Es necesario cierto conocimiento *a priori* del problema.
3. Son sensibles a la forma del frente de Pareto.
4. La dispersión de las soluciones en el frente dependen de la eficiencia del método de optimización.

3.4 Algoritmos basados en el criterio de optimalidad de Pareto

Los algoritmos genéticos son muy adecuados para encontrar frentes de Pareto, ya que al manjar poblaciones de puntos pueden dar grupos de soluciones que en conjunto aproximen al frente. Por lo anterior, el diseño de algoritmos genéticos multiobjetivo es un área que ha tenido un gran impulso en los últimos años. A continuación se presenta una breve explicación de algunos algoritmos genéticos que utilizan el criterio de optimalidad de Pareto para solucionar problemas multiobjetivo.

3.4.1 MOGA

Propuesto por Fonseca y Fleming [27] el *Multiobjective Optimization Genetic Algorithm* asigna la aptitud a los individuos con base en jerarquías. Consideremos a un individuo x_i en una generación t que es dominado por $p_i^{(t)}$ individuos de la población actual. Su jerarquía estará dada por:

$$\textit{jerarquía}(x_i, t) = 1 + p_i^{(t)} \tag{3.6}$$

Todos los individuos no dominados tienen gerarquía uno.

El proceso de asignación de la aptitud se da a continuación:

1. Ordenar la población por categorías.
2. Asignar la aptitud a los individuos interpolando desde el mejor al peor de manera usual de acuerdo a una función comúnmente (aunque no necesariamente) lineal.
3. Promediar la aptitud de los individuos de la misma gerarquía.

Para mantener la diversidad en la población se utilizan nichos.

3.4.2 NPGA

Propuesto por Horn et al. [7] el *Niched Pareto Genetic Algorithm* difiere de un algoritmo genético común en el operador de selección. En éste se utiliza torneo binario, sólo que en vez de comparar a los individuos por dominancia entre sí se compara contra cierto porcentaje de la población seleccionado al azar. Se da la recomendación de utilizar un 10% del tamaño de la población para realizar la comparación, ya que para porcentajes mayores se dan problemas de convergencia prematura y para porcentajes menores puede no haber convergencia. El objetivo de la comparación es determinar si los individuos son dominados o no respecto a la población contra la que se les compara. Si uno de los individuos es dominado y el otro no, se pasa al no dominado a la siguiente etapa. En caso de que ambos individuos sean dominados o no dominados, se utiliza un desempate por medio de nichos aplicado en el espacio de las variables objetivo. Para esto se realiza un conteo de cuantos individuos están dentro de cierto radio alrededor del contendiente. Aquel que con menos individuos dentro del radio es elegido para la siguiente etapa.

3.4.3 SPEA

Propuesto por Zitzler y Thiele [6], el *Strength Pareto Evolutionary Algorithm* (SPEA) tiene como principales características las siguientes:

- Preserva las soluciones no dominadas encontradas hasta el momento, el llamado "conjunto no dominado externo".
- Usa el concepto de dominancia de Pareto para asignar un valor de aptitud escalar a los individuos.
- Preserva la diversidad dentro del frente por medio de un algoritmo de *clustering*.
- La aptitud de los individuos es determinada exclusivamente por las soluciones almacenadas en el conjunto no dominado externo. Las relaciones de dominancia entre los miembros de la población es irrelevante.
- Todas las soluciones del conjunto no dominado externo participan en la selección.

El mecanismo para asignar la aptitud de los individuos es el siguiente. Primero se pasa a todos lo miembros no dominados de la nueva generación al conjunto externo. Luego a cada elemento del conjunto externo se le asigna un valor s (*strength*) que es directamente proporcional al número de individuos a los cuales domina. Después se asigna a cada elemento de la población, tanto padre como hijo, su propio valor s que es igual a la suma de los valores s de los miembros del conjunto externo que lo dominan. El proceso de selección se realiza por torneo binario. Mientras más pequeño es el valor s más probabilidad tiene un individuo de ser seleccionado. La cruza se realiza tomando en cuenta tanto elementos de la población como del conjunto externo. El tamaño del conjunto externo es fijo, por lo tanto si el número de individuos no dominados llega a superar el tamaño máximo del archivo se utiliza un algoritmo de *clustering* para eliminar individuos de las zonas más densamente pobladas. El método adoptado se llama *average linkage method* [21].

3.4.4 NSGA

Propuesto por Srinivas and Deb [11], *Non-dominated Sorting Genetic Algorithm* hace una clasificación de la población según dominancia. El ciclo principal del algoritmo es igual al del algoritmo genético clásico excepto que

antes de entrar a la etapa de selección se hace un preproceso. Primero, se localiza al conjunto no dominado de la población actual y se le asigna una aptitud ficticia igual para todos. Después, esta aptitud se modifica a través del criterio de nichos. A continuación se separa este conjunto y se aplica el mismo proceso con el resto de la población, cuidando que la aptitud ficticia del nuevo conjunto no dominado sea menor a la aptitud del peor individuo del conjunto no dominado anterior. Con este proceso se da preferencia en la selección a los individuos no dominados con respecto a toda la población, de forma que la población se aproxime iterativamente al frente de Pareto. A su vez, el criterio de nichos dispersa a la población evitando que converja a un solo punto.

3.4.5 NSGA-II

Propuesto por Deb. et al. [5] el NSGA-II busca compensar las deficiencias más criticadas del NSGA que son:

- Su alto costo computacional al hacer la clasificación por dominancia que para el peor de los casos es de $O(kN^3)$ comparaciones entre individuos, donde k es el número de funciones objetivo y N el tamaño de la población, lo cual representa un costo computacional elevado, sobre todo para poblaciones grandes.
- Carencia de un sistema de elitismo.
- Necesidad de un radio de nicho. Los resultados del algoritmo pueden ser muy sensibles a este parámetro.

En el NSGA-II el número de comparaciones se reduce a $O(mN^2)$ gracias a un ingenioso sistema en que se compara por dominancia a todos los individuos de la población y se lleva un registro para cada individuo de cuántos individuos lo dominaban y a cuáles individuos éste domina. De esta forma no son necesarias comparaciones adicionales cada vez que se identifica un frente de Pareto. La desventaja de este sistema es que requiere de memoria adicional para almacenar la información extra. Para dispersar uniformemente a la población a lo largo de el frente de Pareto, se diseñó el siguiente sistema:

para cada una de las funciones objetivo se ordena la población de menor a mayor. Para cada individuo se lleva un acumulado al que se suma la distancia normalizada entre sus vecinos a cada lado. Este acumulado se inicializa a cero al comienzo de cada generación. Dado que mientras menos congestionada se encuentre la zona donde está un individuo, mayor será la distancia entre sus vecinos a cada lado, este acumulado sirve como métrica de que tan sólo se encuentra cada individuo. A los individuos de los extremos se les da un acumulado grande para no perderlos.

La selección se realiza por medio de torneo binario. Al momento de comparar dos individuos si uno de ellos pertenece a un mejor frente que el otro, se selecciona al del mejor frente, con lo que se busca que la población avance hacia el frente de Pareto. De estar ambos en el mismo frente se seleccionaba a aquel que tenga un mayor acumulado de las distancias, esto para mantener una distribución más uniforme.

Por último, se agregó elitismo haciendo uso del criterio $(\mu + \lambda)$ en el que la selección se realiza de la unión de la población padre con la población hijo.

3.4.6 PAES

Propuesta por Knowles y Corne [8], la *Pareto Archived Evolution Strategy* es básicamente una estrategia evolutiva $(1 + 1)$ en la que las soluciones no dominadas encontradas hasta el momento se conservan. A este conjunto de soluciones preservadas se les llama archivo externo (aquí se le llamará archivo). Para mantener la diversidad, PAES introduce un novedoso sistema de malla adaptativa en el cual el espacio de las funciones objetivo es subdividido recursivamente. Cada individuo generado cae en una posición dentro de la malla, de manera que se puede llevar un control de cuantos individuos hay en cada posición, por lo cual en todo momento se puede saber cuales son las posiciones más pobladas y cuáles las menos pobladas. Dado que el archivo tiene un tamaño máximo predefinido, al llenarse éste se preservan las soluciones que se encuentren en posiciones menos pobladas de la malla. En el siguiente capítulo se da una descripción más detallada de este algoritmo.

3.4.7 MicroAG

Propuesto por Coello y Toscano [23], el *Micro Genetic Algorithm* realiza una serie de búsquedas con poblaciones pequeñas y pocas generaciones cada una. Se mantienen dos tipos de memoria. En una de éstas se guardan las mejores soluciones no dominadas encontradas hasta el momento. Esta memoria inicialmente se encuentra vacía. A esta se le conoce como memoria externa. El otro tipo de memoria es llamada memoria de la población que se inicializa al azar y está dividida en dos partes, la parte reemplazable y la no reemplazable. La parte no reemplazable no cambia en toda la corrida y tiene la finalidad de mantener la diversidad. Durante cada ciclo del MicroAG se toman algunos individuos tanto de la memoria reemplazable como de la no reemplazable para incializar a la población del ciclo y se corre un algorimo genético convencional preservando a los individuos no dominados. Al terminar el ciclo se toman dos miembros no dominados de la población final y se comparan con el contenido de la memoria externa. Los elementos dominados de la memoria externa son eliminados. Si los miembros tomados de la población final son no dominados se agregan a la memoria externa. Estos dos elementos también son compardos con dos elementos de la memoria reemplazable descartando a los dominados y tomando a los dominantes. La idea es que la memoria reemplazable adquiera con el tiempo individuos no dominados globales. Para preservar la diversidad dentro de la memoria externa, se utiliza un mecanismo de malla similar al usado en PAES [8].

3.5 Manejo de restricciones por medio de métodos multiobjetivo

La optimización de funciones sujetas a restricciones puede transformarse en un problema multiobjetivo considerando a las restricciones en nuevas funciones objetivo. De manera que tenemos el nuevo problema definido de la siguiente forma: Encontrar $\boldsymbol{x}$ tal que $f_i(\boldsymbol{x}) < 0; i = (1, \ldots, m)$ y $f_0(\boldsymbol{x}) \leq f_0(\boldsymbol{y})$ donde $f_0(\boldsymbol{x})$ es la función objetivo original, $f_i(\boldsymbol{x})$ son las restricciones, $\boldsymbol{y}$ es cualquier vector factible diferente de $\boldsymbol{x}$ y m es el número de restricciones. Con esto se obtienen un conjunto de soluciones que aproximan un frente de Pareto el cual debe tener porciones factibles y no factibles.

A continuación se presentan algunas técnicas que utilizan este principio para optimizar funciones sujetas a restricciones. Aunque existen en la literatura muchas otras técnicas [19], las aquí mostradas dan una idea de lo que se ha estado haciendo en este campo.

3.5.1 Búsqueda lineal entre frentes de Pareto

Camponogara y Talukdar [24] proponen un algoritmo en el cual se determinan direcciones de búsqueda basadas en las posiciones de dos elementos de la población, uno de ellos dominado y el otro no dominado. Esto con la finalidad de avanzar en la dirección del frente de Pareto. Primero, transforma el problema de minimización sujeto a restricciones en un problema de optimización multiobjetivo tomando como una de las funciones objetivo f_1 a la función a minimizar, y la otra función objetivo como $f_2 = \sum_{i=1}^{m} max[0, g_i(x)]^{\beta}$ donde β es igual a uno o dos. Se agrupa a la población por frentes, los cuales se numeran de uno a r, donde r es el número de frentes. El frente número uno es el conjunto no dominado de la población actual; el dos es el frente no dominado que se obtiene al no tomar en cuenta a los elementos del primer frente; el frente número tres es el que se obtiene al descartar los elementos de los dos frentes anteriores, y así sucesivamente. En caso de que sólo exista un frente, se elimina a la mitad de los individuos de la población y se sustituyen por individuos generados al azar y se reclasifica a la población por frentes. Después se toman al azar dos elementos de la población $\boldsymbol{x}_i$ y $\boldsymbol{x}_j$ donde los subíndices indican el frente al que pertenecen y además $i < j$. Se toma una dirección de exploración $d = \frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)}{|\boldsymbol{x}_i - \boldsymbol{x}_j|}$ y se generan varios hijos a partir de un individuo en la dirección d. Después de generados los hijos se eliminan aquellos individuos de los frentes con el índice más alto y se reinicia todo el ciclo. Al final se elige al individuo que menos viole las restricciones y que tenga menor valor para la función objetivo.

3.5.2 COMOGA

Propuesto por Surry et al. [22]. El algoritmo denominado *Constrained Optimization by Multi-Objective Genetic Algorithm* utiliza selección por medio de torneo binario e introduce la idea de elegir a los miembros de la siguiente

población basándose o en la función objetivo o en las restricciones dependiendo de cierta probabilidad p_{cost}. Cuando la comparación se hace respecto a la función objetivo, simplemente se escoge la solución con menor valor. Cuando la comparación se hace respecto a las restricciones, ésta se basa en el criterio de optimalidad de Pareto, considerando a las restricciones como funciones objetivo. Dado que diferentes valores de p_{cost} pueden llevar a diferentes resultados, su valor se ajusta de forma dinámica a lo largo del proceso. Se establece una proporción τ que representa la razón de individuos factibles que se desean en la población. Si en un momento dado la proporción de individuos factibles es menor a τ se disminuye p_{cost} con la fórmula: $p_{cost} = (1 - \epsilon)p_{cost}$. Si por otro lado, la proporción de individuos factibles es mayor a τ, p_{cost} se ajusta con la siguiente fórmula: $p_{cost} = 1-(1-p_{cost})(1-\epsilon)$.

El algoritmo principal de COMOGA se muestra a continuación.

1. Calcular la violación de restricciones en todas las soluciones.
2. Dar un valor o clasificación a los miembros de la población basado en las restricciones violadas
3. Evaluar la función objetivo.
4. Seleccionar una proporción p_{cost} de padres basándose en el costo, y el resto basándose en la clasificación.
5. Aplicar cruza y mutación.
6. Ajustar p_{cost} basándose en el parámetro τ.

3.5.3 NPGA para manejo de restricciones

Mezura propone el uso del NPGA para manejar restricciones [3]. En su propuesta, modifica el operador de selección usado en NPGA [7] de manera que al hacer la comparación entre dos individuos durante la selección por torneo binario se toma el primero de los siguientes criterios que se cumpla: si ambos individuos son factibles se toma al de mejor valor de aptitud; si uno es factible y el otro no, se comparan por factibilidad; si ambos son no factibles, se compara por dominancia. A diferencia del algoritmo original, no

se utilizan nichos para preservar la diversidad, en vez de eso se maneja un parámetro adicional llamado *selection rank* (R_s) el cual es una probablidad de que la selección se realice tal como se describie anteriormente o se tome a uno de los dos individuos al azar. Esto mantiene la diversidad y disminuye la alta presión de selección que introduce el torneo binario.

Capítulo 4

Descripción del Algoritmo

4.1 PAES

En este capítulo se describe el método de optimización de uno o varios objetivos sujeto(s) a restricciones, I-PAES. Pero antes se describe también la estrategia evolutiva para optimización multiobjetivo PAES, en la cual se apoya el método I-PAES. Después se mencionan detalles de cómo se implementaron los operadores de la estrategia evolutiva y finalmente como se implementa el manejo de variables enteras.

4.1.1 Descripción

PAES es un método de optimización multiobjetivo propuesto por Joshua Knowles y David Corne en 1999 [8]. Básicamente es una estrategia evolutiva (1 + 1), donde las mejores soluciones encontradas hasta el momento son conservadas. A estas soluciones se le llama archivo externo (para acortar, en este trabajo se le llamará archivo), y su número se fija en un tamaño máximo *tam_max*. La aptitud de las soluciones está basada en el criterio de optimalidad de Pareto, por lo que el archivo contiene al conjunto de indiviuos no dominados encontrado hasta el momento. Para preservar la diversidad, se introduce un sistema de malla, en el cual el espacio de las funciones objetivo es

subdividido de forma recursiva formando una malla k-dimensional, de manera de que cada individuo se encuentra ubicado en alguna de las subdivisiones (o particiones como se les llamará aquí) de la malla. Una vez lleno el archivo, al generarse un nuevo individuo no dominado, su ingreso al archivo dependerá de qué tan poblada esté la partición en la que se encuentra respecto a las particiones de los elementos del archivo.

El algoritmo general de PAES se muestra a continuación. La función $agrega(x)$ inserta al individuo x en su partición correspondiente dentro de la malla y lo agrega al archivo. La función $quita(x)$ hace justo lo inverso, saca a un individuo del archivo y de su partición correspondiente en la malla. La función $generaIndividuoAlAzar()$ genera un inidividuo de manera aleatoria dentro del espacio de búsqueda .El individuo c es la solución actual a partir de la cual se están generando los nuevos individuos, que también llamaremos soluciones candidato. El individuo h es una mutación generada a partir de c. A un individuo cualquiera del archivo se le representa con a. Se introduce la siguiente notación : $x_1 \square x_2$ significa que x_1 está más solo que x_2 o en otras palabras, que x_1 se encuentra en una partición de la malla menos poblada que x_2.

ALGORITMO PRINCIPAL DE PAES

tam_max = tamaño máximo del archivo
c = solución actual
h = solución candidato
a_h *= elemento del archivo que domina a la solución candidato h*
a_d *= elemento del archivo que es dominado por la solución candidato h*
ocupacion = número de individuos contenidos en archivo
individuosGenerados = número de individuos generados hasta el momento
Inicio
individuosGenerados = 0
ocupacion = 1
c = generaIndividuoAlAzar( ) ; agrega(c) (4.1)
h = muta(c) (4.2)
individuosGenerados += 1
si (c $\prec$ *m)*

ir a 4.4
de lo contrario si ($m \prec c$ *)*
quita(c); agrega(h) ; c = h
de lo contrario si ($\exists\ a_h \in archivo : a_h \prec h$ *)*
ir a 4.4
de lo contrario si ($\exists\ a_d \in archivo : h \prec a_d$ *)*
agrega(h) ; $\forall\ a_d$ { *quita(* a_d *) ; ocupacion − = 1* }
de lo contrario
test(h, c, archivo) (4.3)
si (individuosGenerados $<$ *generadosMax)* (4.4)
ir a 4.2
Fin

La función *test* se detalla a continuación:

FUNCIÓN test(h, c, archivo)
Inicio
si (ocupacion $<$ *tam_max)*
agrega(h)
si (h $\square$ *c)*
c = h
de lo contrario si ($\exists\ a_p \in archivo : h \square a_p$ *)* (4.5)
quita(a_p *) ; agrega(h)*
si (h $\square$ *c)*
c = h
de lo contrario (4.6)
si (h $\square$ *c)* (4.7)
quita(c) ; agrega(h) ; c = h (4.8)
Fin

4.1.2 Observaciones

Existen algunos detalles de PAES que se mencionan a continuación. En *test* se muestra que, si el candidato se encuentra más solo que algún elemento del archivo , paso (4.5), de no ser así, se verifica si el candidato está más solo que la solución actual, paso (4.7), y de ser así se sustituye a la solución actual por el candidato. Sin embargo, de 4.1 se aprecia que la solución actual está incluida dentro del archivo, por lo tanto ¿Cómo es posible que el candidato esté más solo que la solución actual si no está más solo que los individuos de las particiones más pobladas de la malla?

Otro punto es que la solución actual sólo es sustituida por una solución candidato generada por ella misma, pero nunca se verifica si en el archivo existe otra solución que mejore la diversidad. Como en la estrategia evolutiva, los parámetros de búsqueda se ajustan de forma que la exploración se vuelve cada vez mas local, existe el riesgo de que la búsqueda quede atrapada en una región del espacio en perjuicio de la diversidad.

Por último, la implementación de la malla que se usa normalmente no funciona para números altos de funciones objetivo. Primero veamos como es está implementación. Cada partición de la malla está identificada por un número. La generación de este número es a partir de la forma en que se contruye la malla. Esta se genera por medio de una bisección recursiva del espacio de cada función objetivo. Como resultado tenemos que el espacio de las funciones objetivo está subdividido en 2^{div*k} particiones, donde div es el número de bisecciones y k es el número de funciones objetivo. El número que identifica cada partición se obtiene al asignar 0 o 1 en cada bisección de cada función objetivo dependiendo de si la posición del individuo está a la izquierda o a la derecha de la división, con lo que al final se genera un número binario de tamaño $div*k$. El número de individuos en cada partición de la malla se registra en un arreglo de enteros de tamaño 2^{div*k}, donde cada posición del arreglo representa una posición en la malla.

No es difícil ver que para k más grandes de 10 o 15 esto causará problemas. Supongamos que tenemos 20 funciones objetivo con tres bisecciones, el número de particiones será $2^{3*20} = 1152921504606846976$. En las computadoras personales estándar que se usan actualmente, este número no puede ser representado como entero. Como flotante o double tampoco puede ser

representado con todas sus cifras significativas. También se dan problemas con la capacidad de memoria, pues se necesitaría un arreglo de enteros de 1152921504606846976 elementos. Si un entero se representa en la máquina con 16 bits se necesitarán 2 * 1152921504606846976 bytes de memoria lo cual sobrepasa considerablemente la capacidad de cualquier computadora actual. Estas no son limitaciones del método en sí, sino de la implementación que se hace de éste.

4.1.3 Modificaciones

Debido a los puntos mencionados anteriormente se eliminan los pasos 4.6, 4.7 y 4.8. Además, a cada cierto número de individuos generados g, se escogerá una nueva solución actual entre los elementos del archivo. Por lo tanto, entre los pasos 4.3 y 4.4 se agregan los siguientes pasos 4.9 y 4.10.

masSolo = individuo de la(s) partición(es) menos poblada(s)

si ({ individuosGenerados modulo g } == 0) (4.9)

c = masSolo (4.10)

En lo que se refiere a la malla, ésta se implementó de la siguiente manera. Cada posición de la malla se representa como un arreglo de k enteros, donde los elementos del arreglo representan la posición dentro de las bisecciones para cada función objetivo y k es el número de funciones objetivo (función objetivo original mas restricciones). Por ejemplo, si tenemos 20 funciones objetivo con 5 bisecciones, la posición de los individuos se representará con un arreglo de 20 enteros donde cada entero da una posición de las 2^5 posibles. El número de individuos en cada partición se registra por medio de un arreglo de estructuras, las cuales contienen la partición que representan dentro de la malla y el número de individuos en esa partición. Como cada individuo puede estar en una sola partición a la vez, necesitaremos un arreglo de *max_size* estructuras. Esto reduce considerablemente los problemas de la implementación clásica de la malla mencionados en la sección 4.1.2.

4.2 I-PAES

4.2.1 Descripción general

I-PAES es un método de optimización de funciones de un solo objetivo sujeta a restricciones. Sigue el criterio de manejo de restricciones a través de algoritmos multiobjetivos, donde las restricciones son agregadas como nuevas funciones objetivo. Al aplicar un algoritmo multiobjetivo que siga el criterio de optimalidad de Pareto, la solución final será una población que trata de aproximar el llamado "Frente de Pareto", que es una superficie k-dimensional formada por puntos que son no-dominados entre ellos. Esta superficie debe abarcar una región de puntos factibles y no factibles, eligiéndose al final el mejor punto de la región factible.

I-PAES, como su nombre lo indica, utiliza como método multiobjetivo el PAES con las modificaciones mencionadas en la sección 4.1.3. La aportación que realiza I-PAES es que el espacio de búsqueda es reducido (o cortado), gradualmente en torno a la zona óptima factible, de manera que se obtienen soluciones más precisas. El problema con otros algoritmos similares es que al no reducir el espacio de búsqueda se obtienen soluciones muy burdas debido a que los algoritmos multiobjetivo tienden a dispersar la población. I-PAES es muy similar a PAES, su algoritmo principal se muestra más adelante:

En 4.12, r representa cierta cantidad de individuos generados desde el último corte, es decir, se llamará a la función *reducirEspacio* cada cierto número r de individuos generados. Para que entren en acción los mecanismos de dispersión del PAES, r debe ser por lo menos igual o mayor al tamaño *max_size* del archivo, como por ejemplo 2**max_size*.

ALGORITMO PRINCIPAL DE I-PAES

tam_max = tamaño máximo del archivo
c = solución actual
h = solución candidato
a_h *= elemento del archivo que domina a la solución candidato h*

a_d *= elemento del archivo que es dominado por la solución candidato h*
ocupacion = número de individuos contenidos en archivo
individuosGenerados = número de individuos generados hasta el momento
Inicio
individuosGenerados = 0
ocupacion = 1
c = generaIndividuoAlAzar( ) ; agrega(c)
h = muta(c) (4.11)
individuosGenerados += 1
si (c $\preceq$ *h)*
  ir a 4.14
de lo contrario si (h $\preceq$ *c)*
  quita(c); agrega(h) ; c = h
de lo contrario si ($\exists$ $a_h \in$ *archivo :* $a_h \preceq h$ *)*
  ir a 4.14
de lo contrario si ($\exists$ $a_d \in$ *archivo :* $h \prec a_d$ *)*
  agrega(h) ; $\forall$ a_d { *quita(* a_d *) ; indiviuduosEnArchivo − = 1* }
de lo contrario
  test(h, c, archivo)
si ({ individuosGenerados módulo g } == 0)
  c = masSolo
si ({ inidividuosGenerados módulo r } == 0) (4.12)
  reducirEspacio(archivo) (4.13)
si (individuosGenerados < generadosMax) (4.14)
  ir a 4.11
Fin

4.2.2 Operador de cortes

La función *reducirEspacio* es la encargada de disminuir gradualmente el espacio de búsqueda. Los pasos principales de esta se muestran a continuación:

FUNCIÓN reducirEspacio(archivo)
$\underline{x}_{pob}$ = *vector con el menor valor de x en cada dimensión del espacio de búsqueda de toda la población*
$\overline{x}_{pob}$ = *vector con el mayor valor de x en cada dimensión del espacio de búsqueda de toda la población*
Inicio
seleccion(archivo) (4.15)
obtenerMinMax(archivo, $\underline{x}_{pob}$, $\overline{x}_{pob}$ *)* (4.16)
cortar($\underline{x}_{pob}$, $\overline{x}_{pob}$ *)* (4.17)
ajustarParámetros(archivo) (4.18)
Fin

En el paso 4.15 se toma cierto porcentaje p de la población, como podría ser un 15% siendo estos lo mejores individuos. Los mejores individuos serán aquellos que tengan un valor más bajo para la función objetivo en el caso de que todos sean factibles. Si el porcentaje de individuos factibles es menor a p, se escogerán también aquellos que menos violen las restricciones, esto último para cada una de las restricciones. A continuación se muestra un pseudocódigo de *seleccion*:

FUNCIÓN seleccion(archivo)
i = índice de la restricción que se está evaluando
m = número de restricciones
violacion = bandera que indica si aún existen restricciones violadas
x = individuo contenido en el archivo
$\overline{x}$ = *último individuo del archivo ordenado respecto a una restricción o función objetivo*
restric(x, i) = valor de la restricción i para el individuo x

numFinal = número total de individuos a elegir (15% del archivo)
ocupacion = número de elementos en el archivo
ordenaR(archivo, i) = función que realiza un ordenamiento de menor a mayor de los elementos del archivo respecto a la restricción i. Cuando i = 0, el ordenamiento se hace respecto a la función objetivo
Inicio
violacion = no (4.19)
i = 1 (4.20)
ordenaR(archivo, i) (4.21)
si (restric($\overline{x}$, *i) > 0)*
  eliminar $\overline{x}$
  ocupacion − = 1
  violacion = si
si (ocupacion <= numFinal)
  ir a 4.25
de lo contrario si (i = m)
  si (violacion = si)
    regresar a 4.19
  de lo contrario
    ordenaR(archivo, 0) (4.22)
    tomar lo primeros numFinal individuos (4.23)
    ocupacion = numFinal (4.24)
    ir a 4.25
de lo contrario
  i = i + 1
  ir a 4.21
Fin (4.25)

La función *obtenerMinMax(archivo,* $\underline{x}_{pob}$, $\overline{x}_{pob}$ *)* (4.16) obtiene el menor y mayor valores de las variables de búsqueda. En otras palabras, buscar $\overline{x}_{pob}$

y $\underline{x}_{pob}$ que son, respectivamente, el mayor y el menor x_i entre los individuos seleccionados donde $i \in \{1, \ldots, n\}$; n = número de variables de búsqueda.

La función cortar($\underline{x}_{pob}$, $\overline{x}_{pob}$) (4.17) establece los nuevos límites dentro de los cuales se moverán las variables de búsqueda. Existen varios puntos importantes a considerar. Por ejemplo, la velocidad con que se reduce el espacio de búsqueda debe controlarse para evitar perder demasiado espacio y quedar atrapados en mínimos locales. Por otro lado, si la reducción es dura, es decir, una vez establecido un nuevo límite no hay forma de sobrepasarlo, no se podrá recuperar la zona óptima si ésta llega a quedar fuera en alguno de los cortes.

Para evitar las situaciones anteriores, se tomaron las siguientes medidas. Primero no se permite que el hipervolumen de búsqueda se reduzca mas allá de cierta proporción α del espacio anterior. Es decir, el nuevo hipervolumen de búsqueda debe ser mayor o igual a α multiplicada por el hipervolumen anterior. Segundo, para cada una de las variables de búsqueda, una vez obtenido el mayor y menor valor de $\boldsymbol{x}$, $\overline{\boldsymbol{x}}_{pob}$ y $\underline{\boldsymbol{x}}_{pob}$ respectivamente se agregará una holgura o delta de manera que exista la posibilidad de recuparar un poco del espacio eliminado. Finalmente, si se llega a sobrepasar el rango de búsqueda original del problema, se ajusta el nuevo rango al rango original. A continuación se muestra el algoritmo de *cortar*.

FUNCIÓN cortar($\underline{x}_{pob}$, $\overline{x}_{pob}$)
n = número de variables de búsqueda
$\overline{x}_i$ = límite superior del espacio de búsqueda para la variable i
$\underline{x}_i$ = límite inferior del espacio de búsqueda para la variable i
$\overline{x}_{pob,i}$ = mayor valor existente en la población para la variable i
$\underline{x}_{pob,i}$ = menor valor existente en la población para la variable i
Inicio
$\forall i : i \in \{ 1, \ldots, n \}$
$holgura_i = 0.05 * (\overline{x}_{pob,i} - \underline{x}_{pob,i})$
$ancho\_pob_i = \overline{x}_{pob,i} - \underline{x}_{pob,i}$
$ancho_i^t = \overline{x}_i^t - \underline{x}_i^t$

$$deltaMin_i = \frac{\beta * ancho_i^t - ancho\_pob_i}{2} \tag{4.26}$$
$delta_i = max(holgura_i, deltaMin_i)$
$$\overline{x}_i^{t+1} = \overline{x}_{pob,i} + delta_i \tag{4.27}$$
$$\underline{x}_i^{t+1} = \underline{x}_{pob,i} - delta_i \tag{4.28}$$
$si (\overline{x}_i^{t+1} > \overline{x}_{original,i})$ (4.29)
$\underline{x}_i^{t+1} -= \overline{x}_i^{t+1} - \overline{x}_{original,i}$
$\overline{x}_i^{t+1} = \overline{x}_{original,i}$
$si (\underline{x}_i^{t+1} < x_{original,i})$
$\overline{x}_i^{t+1} += \underline{x}_{original,i} - \underline{x}_i^{t+1}$
$\underline{x}_i^{t+1} = \underline{x}_{original,i}$
$si (\overline{x}^{t+1} > \overline{x}_{original,i})$
$\overline{x}_i^{t+1} = \overline{x}_{original,i}$ (4.30)
Fin

La variable β del paso 4.26 es la proporción máxima que debe reducirse el ancho del espacio en cada una de las variables de búsqueda para evitar que el espacio total se reduzca mas allá de la proporción α. La deducción de β a partir de α se muestra en el apéndice A. La deducción de la fórmula 4.26 se muestra en el apéndice B. La parte 4.27 y 4.28 tienen como objetivo, fijar el nuevo espacio de búsqueda de manera que la población quede centrada en el. El código en 4.29 y 4.30 tiene como objetivo no sobrepasar los límites originales del espacio de búsqueda el problema. En caso de pasarse alguno de los límites, estos son ajustados al valor original, y el exceso se pasa al otro extremo. Una vez actualizado el espacio de búsqueda, los $\boldsymbol{\sigma}$ de toda la población se reinician de acuerdo al rango de los nuevos intervalos según la fórmula siguiente:

$$\sigma_i = (\overline{x}_i - \underline{x}_i)/\sqrt{n} \quad j \in (1, \ldots, n) \tag{4.31}$$

4.2.3 Elitismo

También se agrega al I-PAES elitismo para preservar al mejor individuo factible generado. Este individuo se puede perder debido a que los métodos multiobjetivo que manejan el frente de Pareto, normalmente usan procedimientos para dispersar la población en el frente, eliminanado a individuos de las regiones más congestionadas. El elitismo fuè implementado de la siguiente manera; cada vez que un nuevo individuo es creado, se verifica si es un individuo factible. De ser así se compara su valor de la función objetivo con el mejor encontrado. Si su valor es mejor, entonces el nuevo individuo se marca o se señala como en nuevo mejor individuo encontrado, el cual no deberá ser eliminado hasta encontrar uno mejor.

4.3 Parámetros de la estrategia evolutiva

En la estrategia evolutiva con la que se implementó el I-PAES, cada individuo tiene un conjunto de vectores $(\boldsymbol{x}, \boldsymbol{\sigma}, \boldsymbol{F})$, donde $\boldsymbol{x}$ son las variables de búsqueda en un espacio de n dimensiones, $\boldsymbol{\sigma}$ son las variables de control también en un espacio de n dimensiones y $\boldsymbol{F}$ son los valores de la funciones a optimizar en un espacio de k dimensiones, donde k es el número de funciones objetivo. Los valores de $\boldsymbol{x}$ para el padre inicial son generados acorde a un distribución de probabilidad uniforme sobre el espacio de búsqueda para cada una de las n dimensiones de éste. Los valores iniciales de $\boldsymbol{\sigma}$ se generan de la siguiente forma. Primero se calcula el máximo valor que pueden tener con la fórmula:

$$\overline{\sigma}_j = (\overline{x}_j - \underline{x}_j)/\sqrt{n} \quad j \in \{1, \ldots, n\} \tag{4.32}$$

Después se inicializan los parámetros de control con la fórmula siguiente $\sigma_j = N(0, \overline{\sigma}_j)$, donde $N(0, K)$ es una distribución normal de media 0 y desviación K. Se debe tener en cuenta que $\sigma_j <= \overline{\sigma}_j$.

La mutación de los parámetros de control se realiza de acuerdo a la siguiente fórmula:

$$\sigma_j^{t+1} = \sigma_j^t \exp(\tau' N(0,1) + \tau N_j(0,1))$$

donde el subíndice j en $N_j(0,1)$ indica que se recalcula para cada valor de j. Los índices de aprendizaje τ y τ' son iguales a $\frac{1}{\sqrt{2\sqrt{n}}}$ y $\frac{1}{\sqrt{2n}}$ respectivamente.

Después de haber mutado las variables de control, las variables de búsqueda se modifican como se muestra a continuación:

$$x_j^{t+1} = x_j^t + \sigma_j^{t+1} N_j(0,1)$$

En caso de que las variables mutadas caigan fuera de los límites del espacio, se repetirá la mutación. Si después de 10 mutaciones no se logra que el individuo quede dentro de los límites se ajustarán sus valores para que queden dentro.

4.4 Caso especial: problemas con variables enteras

En los casos en que las variables de búsqueda son números enteros, se hicieron las siguientes modificaciones:

4.4.1 I-PAES

La reducción del espacio de búsqueda se realizará contrayendo en η los límites de cada dimensión, es decir:

$$\overline{x}_i = \overline{x}_i - \eta$$

$$\underline{x}_i = \underline{x}_i + \eta$$

donde $i \in (1, \ldots, n)$ y η es el número de enteros que se disminye el espacio de búsqueda en cada corte. No se reducirá más allá del valor mínimo y máximo de la población seleccionada, es decir si $\underline{\boldsymbol{x}}_{pob}$ y $\overline{\boldsymbol{x}}_{pob}$ son vectores con los valores mínimo y máximo de la población para cada dimensión de búsqueda, entonces:

$$\overline{x}_i = (\overline{x}_i - \eta) < \overline{x}_{pob,i} ? \overline{x}_{pob,i} : \overline{x}_i - \eta$$

$$\underline{x}_i = (\underline{x}_i + \eta) > \underline{x}_{pob,i} ? \underline{x}_{pob,i} : \underline{x}_i + \eta$$

La *holgura* se sigue calculando igual $holgura = 0.05 * (\overline{x}_i - \underline{x}_i)$ con la diferencia de que su valor se redondea al entero más cercano y no puede ser menor a uno.

4.4.2 Parámetros de la estrategia evolutiva

El $\overline{\sigma}$ se calcula igual que antes, pero redondeándolo al entero más cercano y nunca puede ser menor a 1. Las σ del individuo inicial se hacen igual a $\overline{\sigma}$. Las variables de búsqueda del individuo inicial se generan con una distribución uniforme entera en el intérvalo de búsqueda. La mutación de las variables de búsqueda se realiza de la siguiente forma:

$$x_i^{t+1} = x_i^t + signo(sigma_i) * rand(|sigma_i|)$$

donde $rand()$ es una función que genera un número entero al azar entre 0 y su argumento. La función *signo* regresa -1 si su argumento es negativo y 1 si su argumento es positivo. La mutación de las sigmas se hace de la siguiente manera:

$$(random() < 0.45) ? |sigma| = |sigma| + \theta : |sigma| = |sigma| - \theta$$

$$(random() < 0.5) ? sigma = |sigma| : sigma = -|sigma|$$

donde θ es la una cantidad que el usuario fija y representa un cambio en el tamaño de sigma. La función *random* genera un número real entre 0 y 1. La razón por la que se utiliza 0.45 en la comparación es porque se desea que las sigmas tiendan a disminuir de magnitud con el paso de las iteraciones, por eso se le da preferencia a la reducción.

Capítulo 5

Pruebas y Resultados

En este capítulo se muestran los problemas de prueba, así como los resultados de estos al ser evaluados con I-PAES y una comparación con los resultados obtenidos por otros métodos, de los cuales se hace una breve descripción. Las corridas para I-PAES se hicieron con los siguientes parámetros: tamaño de archivo = 200; porcentaje de población seleccionada en cada corte = 15%; holgura = 0.05; cortes cada 2 veces el tamaño del archivo; índice de aprendizaje de la mutación = 10; número de evaluaciones de la función objetivo = 350,000, $\alpha = 0.90$.

Los resultados referidos como NPGA-RR corresponden a NPGA para manejo de restricciones con representación real, el cual usó 500000 evaluaciones de función. Los referidos como NPGA-RB corresponden a NPGA para manejo de restricciones con representación binaria, el cual usaron 80000 evaluaciones de función. Ambos fueron tomados de [3] y se dio una breve descripción del algoritmo en el capítulo 3. Los resultados referidos como SR (stochastic ranking) fueron tomados de [12] con $P_f = 0.45$, donde se usan 350000 evaluaciones de función. Se da una breve descipción del algoritmo anterior la siguiente sección.

5.1 Descripción de Stochastic Ranking

Stochastic Ranking es un método de optimización de funciones sujetas a restricciones. Es una (μ,λ)-EE donde la selección se realiza ordenando por burbujeo a los hijos y tomando los μ mejores para la siguiente generación. Al realizar la comparación entre dos hijos contiguos durante el ordenamiento por burbujeo, ésta se hace basándose, con cierta probabilidad Pf, en la función objetivo o en un acumulado de las restricciones ϕ el cual es igual a $\phi = \sum_{i=1}^{m} max\{0, g_i\}$ donde m es el número de restricciones. El pseudocódigo para el ordenamiento por burbujeo es el siguiente:

Ordenamiento Estocástico por Burbujeo
I_j = individuo j de la población λ.
INICIO
de i = 1 a N hacer
  de j = 1 a $\lambda - 1$ hacer
    u = random(0, 1)
    if ($\phi(I_j) = \phi(I_{j+1}) = 0$) ó (u ¡ Pf) entonces
      si ($f(I_j)$ ¿ $f(I_{j+1})$) entonces
        swap(I_j,I_{j+1})
    de lo contrario
      si ($\phi(I_j)$ ¿ $\phi(I_{j+1})$) entonces
        swap(I_j, I_{j+1})
  si no se realizo ningún swap terminar
FIN

La función *random*(0, 1) devuelve un número real entre cero y uno mientras que la función *swap* intercambia de posición de los dos individuos que recibe como argumento. La idea es permitir a individuos no factibles sobrevivir para mantener la diversidad en la exploración dando oportunidad de revisar zonas no factibles con más detalle y crear puentes entre zonas factibles desconectadas.

5.2 Problemas de prueba

Los problemas de prueba usados en este trabajo los podemos clasificar en cuatro grupos. Los problemas g01, ..., g13 son problemas artificiales creados específicamente para probar algoritmos de optimización sujetos a restricciones [12]. Los problemas i1, ..., i3 son problemas de ingeniería de aplicación real tomados de [3]. Los problemas k1, ..., k3 son problemas de diseño de armaduras reticulares. Los problemas m1 y m2 tratan de la optimización de funciones multiobjetivo sujetas a restricciones.

5.2.1 Problema g01

Minimizar:

$$F(\boldsymbol{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{5.1}$$

sujeto a:

$$\begin{aligned}
g_1(\boldsymbol{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(\boldsymbol{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(\boldsymbol{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(\boldsymbol{x}) &= -8x_1 + x_{10} \leq 0 \\
g_5(\boldsymbol{x}) &= -8x_2 + x_{11} \leq 0 \\
g_6(\boldsymbol{x}) &= -8x_3 + x_{12} \leq 0 \\
g_7(\boldsymbol{x}) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(\boldsymbol{x}) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(\boldsymbol{x}) &= -2x_8 - x_9 + x_{12} \leq 0
\end{aligned} \tag{5.2}$$

donde los límites del espacio de búsqueda son $0 \leq x_i \leq 1$ $(i = 1, \ldots, 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$ y $0 \leq x_{13} \leq 1$. El mínimo global esta en $\boldsymbol{x}^* =$ {1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1 } donde seis restricciones están activas (g_1, g_2, g_3, g_7, g_8 y g_9). Se considera que restricción está activa si su valor está en el límite para ser violada, o sea en cero. $F(\boldsymbol{x}^*) = -15$.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{1, 0.999999939809, 0.999997901977, 1, 0.999981406123, 1, 1, 0.999999242667, 0.999981194574, 2.99987534752, 2.99995011286, 2.99993014684, 0.999982112914\}$ con $F(\boldsymbol{x}) = -14.99968877$. Como puede verse en tabla 5.1, I-PAES tuvo menos precisión que SR al converger al óptimo. También quedo atrapado en un mínimo local donde $F(\boldsymbol{x}^*) = -13$ en $\frac{1}{5}$ de las corridas.

Problema g01		
	I-PAES	SR
Mejor	-14.9997	-15
Peor	-12.446	-15
Promedio	-14.494	-15
Desviación	9.3E-01	0
Mediana	-14.997	-15
Sol. Factibles	30	30

Tabla 5.1: Comparación de resultados del problema g01

5.2.2 Problema g02

Descripción

Maximizar:

$$F(\boldsymbol{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{5.3}$$

sujeta a:

$$g_1(\boldsymbol{x}) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$
$$g_2(\boldsymbol{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0 \tag{5.4}$$

donde $n = 20$ y $0 \leq x_i \leq 10$ $(i = 1, \ldots, n)$. El óptimo global no se conoce. La mejor solución reportada en la literatura es $\boldsymbol{x}^* = \{3.16237443645701, 3.12819975856112, 3.09481384891456, 3.06140284777302, 3.02793443337239, 2.99385691314995, 2.95870651588255, 2.92182183591092, 0.49455118612682, 0.48849305858571, 0.48250798063845, 0.47695629293225, 0.47108462715587, 0.46594074852233, 0.46157984137635, 0.45721400967989, 0.45237696886802, 0.44805875597713, 0.44435772435707, 0.44019839654132\}$ con $F(\boldsymbol{x}^*)[3] = 0.803619$. La restricción g_1 está cerca de ser activa $(g_1 = -10^{-8})$.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{3.14860401788, 3.10915903011, 3.08909341555, 3.05835689132, 3.04000196011, 3.00100530894, 2.94955289769, 2.94207158769, 0.49907406319, 0.486231653274, 0.49055938302, 0.492879188045, 0.481722447567, 0.471623533316, 0.452037376504, 0.442565813637, 0.451211591495, 0.437863945589, 0.444359423833, 0.437834075871\}$ con $F(\boldsymbol{x}) = -0.803375563$. Como se aprecia en la tabla 5.2 el mejor resultado obtenido por SR superó al mejor de I-PAES. Sin embargo, I-PAES tuvo mejores resultados en todos los demás parámetros. La varianza es significativamente más pequeña en I-PAES lo que significa que tiene una mayor consistencia en los resultados que arroja. Respecto a Mezura, I-PAES obtuvo resultados significativamente superiores.

Problema g02				
	I-PAES	NPGA-RR	NPGA-RB	SR
Mejor	-0.803376	-0.577987	0.787933	-0.803515
Peor	-0.768291	-0.214291	0.689747	-0.726288
Promedio	-0.793281	-0.337094	0.751301	-0.781975
Desviación	9.0E-03	9.5E-02	6.3E-4	2.0E-02
Mediana	-0.793342	-0.302032	0.750559	-0.7858
Sol. Factibles	30	30	30	30

Tabla 5.2: Comparación de resultados del problema g02

5.2.3 Problema g03

Maximizar:

$$F(\boldsymbol{x}) = (\sqrt{n})^n \sum_{i=1}^{4} x_i \tag{5.5}$$

sujeto a:

$$h_1(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \tag{5.6}$$

$$\tag{5.7}$$

donde $n = 10$ y $0 \leq x_i \leq 1 (i = 1, \ldots, n)$. El máximo global está en $\boldsymbol{x}_i^* = \frac{1}{\sqrt{n}}$ $(i = 1, \ldots, n)$ donde $\mathrm{F}(\boldsymbol{x}^*) = 1$. Este problema contiene restricciones de igualdad por lo que se uso un $\delta = 0.0001$ para crear una zona factible transformando el problema de $h_k = 0$ a $g_k \leq |h_K| - \delta$. Por esta razón los valores obtenidos fueron un poco mejor que el óptimo global, pero redondeados al tercer decimal esta diferencia desaparece.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{0.316965968, 0.315664596, 0.314608242, 0.315958975, 0.315915392, 0.317873891, 0.316867036, 0.314518512, 0.314381436, 0.319636209\}$ con $F(\boldsymbol{x}) = -1.000421429$. Como puede verse en la tabla 5.3 tanto I-PAES como SR tuvieron un excelente desempeño.

5.2.4 Problema g04

Descripción

Minimizar:

$$F(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{5.8}$$

Problema g03		
	I-PAES	SR
Mejor	-1.000	-1.000
Peor	-1.000	-1.000
Promedio	-1.000	-1.000
Desviación	9.7E-05	0.000
Mediana	-1.000	-1.000
Sol. Factibles	30	30

Tabla 5.3: Comparación de resultados del problema g03

sujeta a:

$$\begin{aligned}
g_1(\boldsymbol{x}) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\
g_2(\boldsymbol{x}) &= -85.334407 - 0.0056858_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\
g_3(\boldsymbol{x}) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 000 \\
g_4(\boldsymbol{x}) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \\
g_5(\boldsymbol{x}) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \\
g_6(\boldsymbol{x}) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0
\end{aligned} \tag{5.9}$$

donde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$, (i =3, 4, 5). El óptimo global se encuentra en $\boldsymbol{x}^*$ =(78, 33, 29.995256025682, 45, 36.775812905788) donde $F(\boldsymbol{x}*) = -30665.539$. Las restricciones g_1 y g_6 están activas.

Resultados

La mejor solución encontrada por el algoritmo fué: $\boldsymbol{x} = \{78, 33.00000002, 29.99525605, 45, 36.77581285\}$ con $F(\boldsymbol{x}) = -30665.53867$. Como se aprecia en la tabla 5.4, I-PAES tuvo un desempeño muy similar a SR.

Problema g04				
	I-PAES	NPGA-RR	NPGA-RB	SR
Mejor	-30665.539	-30557.850	-30659.997	-30665.539
Peor	-30665.539	-29519.307	-30551.493	-30665.539
Promedio	-30665.539	-30101.770	-30630.670	-30665.539
Desviación	0	2.5E+02	2.5E+01	2.0E-05
Mediana	-30665.539	-30072.869	-30637.013	-30665.539
Sol. Factibles	30	30	30	30

Tabla 5.4: Comparación de resultados del problema g04

5.2.5 Problema g05

Minimizar:

$$F(\boldsymbol{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + \frac{0.000002}{3}x_2^3 \tag{5.10}$$

sujeto a:

$$\begin{aligned} g_1(\boldsymbol{x}) &= -x_4 + x_3 - 0.55 \leq 0 \\ g_2(\boldsymbol{x}) &= -x_3 + x_4 - 0.55 \leq 0 \\ g_3(\boldsymbol{x}) &= 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\ g_4(\boldsymbol{x}) &= 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\ g_5(\boldsymbol{x}) &= 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0 \end{aligned} \tag{5.11}$$

donde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ y $-0.55 \leq x_4 \leq 0.55$. La mejor solución conocida es $\boldsymbol{x}^* = \{679.9453, 1026.067, 0.1188764, -0.3962336\}$ donde $F(\boldsymbol{x}^*) = 5126.4981$.

Resultados

No se encontraron soluciones factibles para este problema. Ver tabla 5.5. Al estudiar más detenidamente la evolución de las corridas, se observó que la

población no convergió a un punto sino que se mantentuvo dispersa. Debido a esto las variables de control se mantuvieron con tamaños grandes, y como el espacio factible es muy pequeõ no se logró encontrarlo.

Problema g05		
	I-PAES	SR
Mejor	x	5126.497
Peor	x	5142.472
Promedio	x	5128.881
Desviación	x	3.5
Mediana	x	5127.372
Sol. Factibles	0	30

Tabla 5.5: Comparación de resultados del problema g05

5.2.6 Problema g06

Minimizar:

$$F(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{5.12}$$

sujeto a:

$$\begin{aligned} g_1(\boldsymbol{x}) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\ g_2(\boldsymbol{x}) &= (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \end{aligned} \tag{5.13}$$

donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$. El óptimo esta en $\boldsymbol{x}^* = \{14.095, 0.84296\}$ donde $F(\boldsymbol{x}^*) = -6961.81388$. Ambas restricciones están activas.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{14.0950000092$ $0.842960808844\}$ con $F(\boldsymbol{x}) = -6961.813854$. En este problema, como puede

apreciarse en la tabla 5.6, SR fue más consistente en encontrar el óptimo lográndolo, según [12], 20 veces de 30 corridas lo cual también se refleja en la mediana. Sin embargo, I-PAES obtuvo un mejor promedio, mejor valor en peor encontrada y una menor varianza, lo que significa que es más consistente en los resultados que obtiene.

Problema g06		
	I-PAES	SR
Mejor	-6961.814	-6961.814
Peor	-6961.810	-6350.262
Promedio	-6961.813	-6875.940
Desviación	8.5E-05	1.6E+02
Mediana	-6961.814	-6961.814
Sol. Factibles	30	30

Tabla 5.6: Comparación de resultados del problema g06

5.2.7 Problema g07

Descripción

Minimizar:

$$F(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2.0(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \quad (5.14)$$

sujeto a:

$$g_1(\boldsymbol{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$
$$g_2(\boldsymbol{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$
$$g_3(\boldsymbol{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$
$$g_4(\boldsymbol{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(\boldsymbol{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$
$$g_6(\boldsymbol{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$
$$g_7(\boldsymbol{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$
$$g_8(\boldsymbol{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \quad (5.15)$$

donde $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$. El óptimo global se encuentra en $\boldsymbol{x}^*$ =(2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375926) y $F(\boldsymbol{x}*) = 24.3062091$. Las restricciones g_1, g_2, g_3, g_4, g_5 y g_6 están activas.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{2.16996489702, 2.36701436984, 8.76882720318, 5.07418756668, 0.943992761955, 1.32027308617, 1.31870032997, 9.82673763033, 8.26988778617, 8.36187863755\}$ con $F(\boldsymbol{x}) = 24.33817628$. En la tabla 5.7 se aprecia que los resultados de SR superaron a I-PAES en todos los aspectos.

Problema g07				
	I-PAES	NPGA-RR	NPGA-RB	SR
Mejor	24.338	24.382	25.473	24.307
Peor	24.995	32.126	33.764	24.642
Promedio	24.527	26.312	27.621	24.374
Desviación	1.7E-01	1.9E+00	1.8E+00	6.6E-02
Mediana	24.467	25.701	26.708	24.357
Sol. Factibles	30	30	30	30

Tabla 5.7: Comparación de resultados del problema g07

5.2.8 Problema g08

Minimizar:

$$F(\boldsymbol{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)} \tag{5.16}$$

sujeto a:

$$\begin{aligned} g_1(\boldsymbol{x}) &= x_1^2 - x_2 + 1 \leq 0 \\ g_2(\boldsymbol{x}) &= 1 - x_1 + (x_2-4)^2 \leq 0 \end{aligned} \tag{5.17}$$

donde $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$. El óptimo está en $\boldsymbol{x}^* = \{1.2279713, 4.2453733\}$ donde $F(\boldsymbol{x}^*) = 0.095825$.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{1.227971353, 4.245373368\}$ con $F(\boldsymbol{x}) = -0.095825041$. En este problema ambos algoritmos tuvieron el mismo desempeño. Los resultados se muestran en la tabla 5.8

Problema g08		
	I-PAES	SR
Mejor	-0.095825	-0.095825
Peor	-0.095825	-0.095825
Promedio	-0.095825	-0.095825
Desviación	0.0	2.6E-17
Mediana	-0.095825	-0.095825
Sol. Factibles	30	30

Tabla 5.8: Comparación de resultados del problema g08

5.2.9 Problema g09

Minimizar:

$$F(\boldsymbol{x}) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7 \tag{5.18}$$

sujeto a:

$$\begin{aligned} g_1(\boldsymbol{x}) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\ g_2(\boldsymbol{x}) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\ g_3(\boldsymbol{x}) &= -196 + 23x_1 + x_2^2 + 6x_6^2 + 8x_7 \leq 0 \\ g_4(\boldsymbol{x}) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \end{aligned} \tag{5.19}$$

donde $-10 \leq x_i \leq 10 (i = 1, \ldots, 7)$. El óptimo está en $\boldsymbol{x}^* = \{$ 2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870 1.038131, 1.594227 $\}$ donde $F(\boldsymbol{x}^*) =$ 680.6300573. Las retricciones g_1 y g_4 están activas.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{2.326603718, 1.957743917, -0.468352679, 4.349668424, -0.621354832, 1.047882344, 1.590801921\}$ con $F(\boldsymbol{x}) = 680.6304707$. Ambos algoritmos llegaron al óptimo tomando en cuenta hasta el tercer decimal. I-PAES superó a SR en todos los demás parámetros. Ver tabla 5.9.

5.2.10 Problema g10

Descripción

Minimizar:

$$F(\boldsymbol{x}) = x_1 + x_2 + x_3 \tag{5.20}$$

Problema g09		
	I-PAES	SR
Mejor	680.630	680.630
Peor	680.634	680.763
Promedio	680.631	680.656
Desviación	8.1E-04	3.4E-02
Mediana	680.631	680.641
Sol. Factibles	30	30

Tabla 5.9: Comparación de resultados del problema g09

sujeto a:

$$
\begin{aligned}
g_1(\boldsymbol{x}) &= -1 + 0.0025(x_4 + x_6) \leq 0 \\
g_2(\boldsymbol{x}) &= -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\
g_3(\boldsymbol{x}) &= -1 + 0.01(x_8 - x_5) \leq 0 \\
g_4(\boldsymbol{x}) &= -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0 \\
g_5(\boldsymbol{x}) &= -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0 \\
g_6(\boldsymbol{x}) &= -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0
\end{aligned} \tag{5.21}
$$

donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ $(i = 2, 3)$, $10 \leq x_i \leq 1000$ $(i = 4, \ldots, 8)$. El óptimo global se encuentra en $\boldsymbol{x}^*$ =(579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979) donde $F(\boldsymbol{x}*) =$ 7049.3307. Las restricciones g_1, g_2 y g_3 están activas.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{105.6345328,$ 1179.227593, 6070.09281, 122.497943, 257.1979828, 277.4889774, 265.2967614, 357.197398 $\}$ con $F(\boldsymbol{x}) = 7062.019117$. Como se aprecia en los resultados de este problema (ver tabla 5.10), tanto Mezura como SR superaron a I-PAES la mejor solución encontrada. Sin embargo, I-PAES supera a ambos algoritmos en todos los demás parámetros excepto la mediana, cuyo valor es superado por SR.

Problema g10				
	I-PAES	NPGA-RR	NPGA-RB	SR
Mejor	7062.019	X	7057.377	7054.316
Peor	7588.054	X	13664.340	8835.655
Promedio	7342.944	X	9051.7188	7559.192
Desviación	1.4E+02	X	1.8E+03	5.3E02
Mediana	7448.014	X	9132.765	7372.613
Sol. Factibles	30	0	15	30

Tabla 5.10: Comparación de resultados del problema g10

5.2.11 Problema g11

Minimizar:

$$F(\boldsymbol{x}) = x_1^2 + (x_2 - 1)^2 \tag{5.22}$$

sujeto a:

$$h_1(\boldsymbol{x}) = x_2 - x_1^2 = 0 \tag{5.23}$$

donde $-1 \leq x_i \leq 1 (i = 1, 2)$. El óptimo está en $\boldsymbol{x}^* = \{\pm\frac{1}{\sqrt{2}}, \frac{1}{2}\}$ donde $F(\boldsymbol{x}^*) = 0.75$. Como se aprecia, este problema cuenta con una igualdad, por lo que se tomaron las mismas medidas que en g03.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{2.326603718,$ 1.957743917, -0.468352679, 4.349668424, -0.621354832, 1.047882344, 1.590801921$\}$ con $F(\boldsymbol{x}) = 0.749913273$. Como puede apreciarse en la tabla 5.11 ambos algoritmos tuvieron muy buen desempeño para este problema. El peor valor encontrado por SR superó al peor encontrado por I-PAES.

Problema g11		
	I-PAES	SR
Mejor	0.750	0.750
Peor	0.751	0.750
Promedio	0.750	0.750
Desviación	2.6E-04	8.0E-05
Mediana	0.750	0.750
Sol. Factibles	30	30

Tabla 5.11: Comparación de resultados del problema g11

5.2.12 Problema g12

Descripción

Minimizar:

$$F(\boldsymbol{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100} \tag{5.24}$$

sujeto a:

$$g_1(\boldsymbol{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0 \tag{5.25}$$

donde $0 \leq x_i \leq 10$, (i =1, 2, 3), y p, q, $r = (1, \ldots, 9)$). La zona factible del espacio de búsqueda consiste de 9^3 esferas disjuntas. Un punto x_1, x_2, x_3 es factible si y sólo si existen p, q y r que satisfacen las restricciones descritas anteriormente (ecuacion 5.25). El óptimo global se localiza en $\boldsymbol{x}^* = (5, 5, 5)$ donde $F(\boldsymbol{x}*) = 1$.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{5.000000039,$ $5.00000005,$ $5.000000015\ \}$ con $F(\boldsymbol{x}) = -1$. Los tres algoritmos revisados

tuvieron un desempeño excelente para este problema encontrando consistentemente el óptimo global.

Problema g12				
	I-PAES	NPGA-RR	NPGA-RB	Stochastic Ranking
Mejor	-1	-1	-1	-1
Peor	-1	-1	-1	-1
Promedio	-1	-1	-1	-1
Desviación	0	0	0	-1
Mediana	-1	-1	-1	-1
Sol. Factibles	30	30	30	30

Tabla 5.12: Comparación de resultados del problema g12

5.2.13 Problema g13

Minimizar:

$$F(\boldsymbol{x}) = e^{x_1 x_2 x_3 x_4 x_5} \tag{5.26}$$

sujeto a:

$$\begin{aligned} h_1(\boldsymbol{x}) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ h_2(\boldsymbol{x}) &= x_2 x_3 - 5 x_4 x_5 = 0 \\ h_3(\boldsymbol{x}) &= x_1^3 + x_2^3 + 1 \end{aligned} \tag{5.27}$$

donde $-2.3 \leq x_i \leq 2.3 (i = 1, 2)$ y $-3.2 \leq x_i \leq 3.2 (i = 3, 4, 5)$. El óptimo está en $\boldsymbol{x}^* = \{-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645\}$ donde $F(\boldsymbol{x}^*) = 0.0539498$.

Resultados

No se encontraron soluciones factibles para estre problema. Ver tabla 5.13. Al revisar con detenimiento el desarrollo de las corridas se observó la misma

situación comentada en el problema g05.

Problema g13		
	I-PAES	SR
Mejor	X	0.053957
Peor	X	0.216915
Promedio	X	0.067543
Desviación	X	3.1E-02
Mediana	X	0.057006
Sol. Factibles	0	30

Tabla 5.13: Comparación de resultados del problema g13

5.2.14 Problema i1

Descripción

Una viga soldada es diseñada para minimizar su costo sujeto a restricciones en su esfuerzo cortante (τ), esfuerzo de pandeo en la viga (σ), carga en la barra (P_c), y la deflexión en la barra (δ), y restricciones de movimiento. Hay cuatro variables de diseño como se muestra en la figura 5.1: $h(x_1)$, $l(x_2)$, $t(x_3)$ y $b(x_4)$. El problema puede plantearse de la siguiente manera:

Minimizar:

$$F(\boldsymbol{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{5.28}$$

sujeto a:

$$\begin{aligned}
&g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{max} \leq 0\\
&g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{max} \leq 0\\
&g_3(\boldsymbol{x}) = x_1 - x_4 \leq 0\\
&g_4(\boldsymbol{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0\\
&g_5(\boldsymbol{x}) = 0.125 - x_1 \leq 0\\
&g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{max} \leq 0\\
&g_7(\boldsymbol{x}) = P - P_c(\boldsymbol{x}) \leq 0
\end{aligned} \tag{5.29}$$

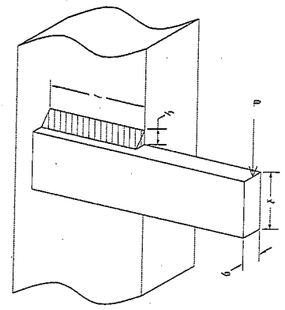

Figura 5.1: Viga soldada del problema i1

donde:

$$\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{5.30}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2} \tag{5.31}$$

$$\tau'' = \frac{MR}{J} \tag{5.32}$$

$$M = P\left(L + \frac{x_2}{2}\right) \tag{5.33}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{5.34}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_2}{2}\right)^2\right]\right\} \tag{5.35}$$

$$\sigma(\boldsymbol{x}) = \frac{6PL}{x_4x_3^2} \tag{5.36}$$

$$\delta(\boldsymbol{x}) = \frac{4PL^3}{Ex_3^3x_4} \tag{5.37}$$

$$P_c(\boldsymbol{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{5.38}$$

$P = 6000lb, L = 14in, E = 30x10^6psi, G = 12x10^6psi$

$\tau_{max} = 13,600psi, \sigma_{max} = 30,000psi, \delta_{mas} = 0.25in$

donde $0.1 \leq x_1 \leq 2.0$, $0.1 \leq x_2 \leq 10.0$, $0.1 \leq x_3 \leq 10.0$ y $0.1 \leq x_4 \leq 2.0$

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{0.205729639744, 3.47048866678, 9.03662391011, 0.205729639806\}$ con $F(\boldsymbol{x}) = 1.724852309$. Como se aprecia en la tabla 5.14, I-PAES obtuvo resultados significativamente superiores a Mezura en todos los parámetros.

Problema i1			
	I-PAES	NPGA-RR	NPGA-RB
Mejor	1.724852	1.724896	1.728226
Peor	1.724852	3.899063	1.993408
Promedio	1.724852	2.260263	1.792654
Desviación	2.2E-08	5.6E-01	7.4E-02
Mediana	1.724852	2.103550	1.765414
Sol. Factibles	30	30	30

Tabla 5.14: Comparación de resultados del problema i1

5.2.15 Problema i2

Descripción

Un recipiente cilíndrico es cubierto por ambos extremos por unas tapas hemisféricas como se muestra en la figura 5.2. El objetivo es minimizar el costo total, incluyendo el costo del material, forma y soldadura. Hay cuatro variables de diseño: T_s (ancho de la estructura), T_h (ancho de la cabeza), R (radio interno) y L (longitud de la sección cilíndrica del recipiente, sin incluir la cabeza). T_s y T_h son enteros múltiplos de 0.0625 pulgadas, los cuales son los anchos disponibles de las placas de acero, R y L son continuas. El

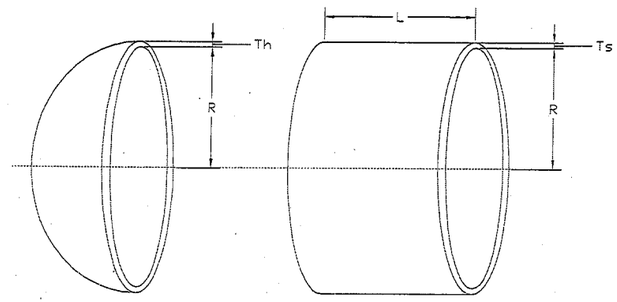

Figura 5.2: Recipiente cilíndrico de problema i2

problema se define como sigue:

Minimizar:

$$F(\boldsymbol{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{5.39}$$

sujeta a:

$$\begin{aligned} g_1(\boldsymbol{x}) &= -x_1 + 0.0193x_3 \leq 0 \\ g_2(\boldsymbol{x}) &= -x_2 + 0.00954x_3 \leq 0 \\ g_3(\boldsymbol{x}) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0 \\ g_4(\boldsymbol{x}) &= x_4 - 240 \leq 0 \end{aligned} \tag{5.40}$$

donde $1 \leq x_1 \leq 99$, $1 \leq x_2 \leq 99$, $10 \leq x_3 \leq 200$ y $10 \leq x_4 \leq 200$.

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{0.8125, 0.4375, 42.098445494, 176.636597126\}$ con $F(\boldsymbol{x}) = 6059.71434795$. En este problema también se obtienen mejores resultados que Mezura en todos lo parámetros estadísticos. Los resultados en la tabla 5.15

Problema i2			
	I-PAES	NPGA-RR	NPGA-RB
Mejor	6059.714	6092.012	6059.946
Peor	6410.087	7348.003	6469.322
Promedio	6098.761	6582.353	6177.253
Desviación	1.0E+02	3.67E+02	1.3E+02
Mediana	6060.062	6488.932	6116.738
Sol. Factibles	30	30	30

Tabla 5.15: Comparación de resultados del problema i2

5.2.16 Problema i3

Descripción

Consiste en minimizar el peso de un resorte de tensión–compresión sujeto a restricciones de deflexión mínima, esfuerzo cortante, frecuencia de vibración, límites de los diámetros externos y de variables de diseño (ver figura 5.3). Las variables de diseño son: D (el diámetro medio del resorte), d (diámetro del material) y N (el número de bobinas activas). El problema se expresa formalmente como sigue:

Minimizar:

$$F(\boldsymbol{x}) = (N+2)Dd^2 \tag{5.41}$$

sujeta a:

$$g_1(\boldsymbol{x}) = 1 - \frac{D^3 N}{71785d^4} \leq 0$$

$$g_2(\boldsymbol{x}) = \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{510d^2} - 1 \leq 0$$

$$g_3(\boldsymbol{x}) = 1 - \frac{140.45d}{D^2 N} \leq 0$$

$$g_4(\boldsymbol{x}) = \frac{D+d}{1.5} - 1 \leq 0 \tag{5.42}$$

donde $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$ y $2 \leq x_3 \leq 15$.

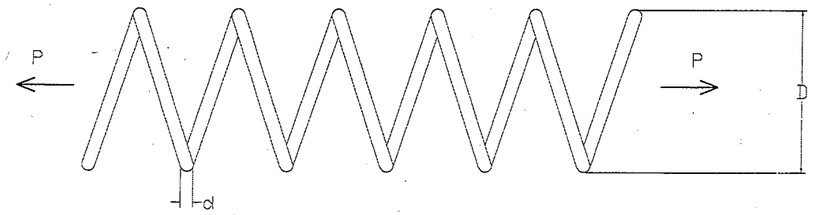

Figura 5.3: Resorte de tensión–compresión del problema i3

Resultados

La mejor solución encontrada por el algoritmo fue: $\boldsymbol{x} = \{0.8125\ 0.4375\ 42.0950690307\ 176.678472065\ \}$ con $F(\boldsymbol{x}) = 0.012665344$. Como se aprecia en la tabla 5.16, I-PAES supera en todos los parámetros a Mezura.

Problema i3			
	I-PAES	NPGA-RR	NPGA-RB
Mejor	0.012665	0.012710	0.012681
Peor	0.012691	0.256994	0.012973
Promedio	0.012673	0.023642	0.012742
Desviación	7.4E-06	4.8E-02	5.9E-05
Mediana	0.012670	0.013390	0.012727
Sol. Factibles	30	30	30

Tabla 5.16: Comparación de resultados del problema i3

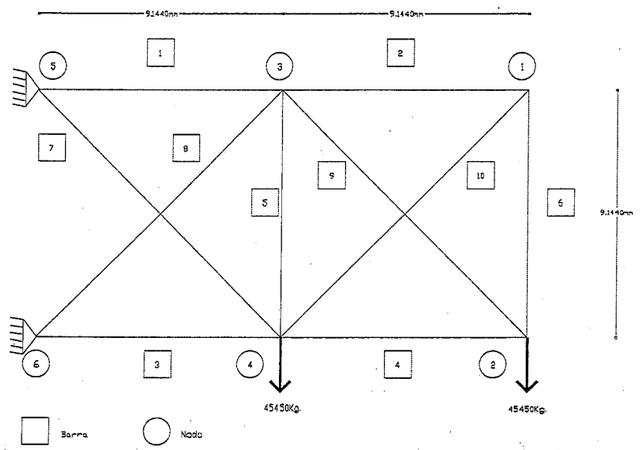

Figura 5.4: Armadura de acero del problema k1

5.2.17 Problema k1

Descripción

Considere la armadura mostrada en la figura 5.4. El problema consiste en encontrar área transversal de cada elemento de la armadura que minimice su peso sujeto a restricciones de esfuerzo y desplazamiento. El peso de la armadura está dado por:

$$F(\boldsymbol{x}) = \sum_{j=1}^{10} \gamma A_j L_j \tag{5.43}$$

donde $\boldsymbol{x}$ es una solución candidata, A_j es el área de la sección transversal del miembro j-ésimo. L_j es la longitud del miembro j y γ el peso volumétrico del material.

El desplazamiento máximo permitido para cada nodo (vertical y horizontal) se asume como 5.08 cm. En total son 10 restricciones de esfuerzo

y 8 de desplazamiento. El valor máximo y mínimo que puede tener cada sección transversal son 0.5062 y 999.0 respectivamente. Los datos asumidos son: módulo de elasticidad $E = 7.3 \cdot 10^5 kg/cm^3$, máximo esfuerzo permisible $= 1742.11 kg/cm^2$, $\gamma = 7.4239 \cdot 10^{-3} kg/cm^3$ y una carga de -45454.0 kg vertical aplicada en los nodos 2 y 4.

Resultados

En la tabla 5.17 se muestra el mínimo valor encontrado por diferentes algoritmos tomados de [14]. GSSA: algoritmo general de búsqueda estocástica, con población de tamaño cinco, probabilidad de cruza de cero, radio de mutación 0·10/elemento y recocido con $\alpha = 1.001$. VGA: Variable string length Genetic Algorithm de Rajeev y Krishamoorthy [28] con tamaño de población de 50. MC: Monte-Carlo annealing algorithm de Elperin [29]. SAARSR: Simulated Annealing with Automatic Reduction of Search Range de Tzan y Pantelides [30]. ISA: Iterate Simulated Annealing de Ackley [31]. SSO: State Space Optimal [32]. Como puede verse en la tabla, I-PAES encontró mejores resultados factibles que el resto de los métodos. MC encontró un valor de peso menor a I-PAES, pero como se muestra en las tablas 5.18 y 5.19, viola restricciones de esfuerzo y desplazamiento.

Peso en problema k1							
Elem.	I-PAES	GSSA	VGA	MC	SSO	ISA	SAARSR
1	190.53	205.17	206.46	200.01	193.75	269.48	201.35
2	0.6466	0.6452	0.6452	0.6452	0.6452	79.810	0.6452
3	146.33	134.20	151.62	129.04	150.15	178.45	161.55
4	95.07	90.973	103.23	90.328	98.62	152.90	95.68
5	0.6452	0.6452	0.6452	0.6452	0.6452	70.390	0.6452
6	3.0166	0.6452	0.6452	0.6452	3.23	10.260	4.19
7	47.677	55.487	54.84	51.616	48.18	147.87	49.16
8	129.826	127.75	129.04	145.17	136.64	14.710	131.55
9	133.282	133.56	132.27	96.78	139.47	156.06	134.32
10	0.6452	0.6452	0.6452	0.6452	0.6452	87.740	0.6452
Vol. (cm^3)	801624.5	805777	833258	765710	828956	1313131	833258
Peso (kg)	5951	6186	6186	5685	6155	9750	6187

Tabla 5.17: Comparación de resultados del problema k1

5.2.18 Problema k2

Descripción

Considere la armadura mostrada en la figura 5.5 tomada de [13]. El problema consiste en encontrar el área transversal de cada elemento de la armadura que minimice su peso sujeto a restricciones de esfuerzo y desplazamiento. El peso de la armadura está dado por:

$$F(\boldsymbol{x}) = \sum_{j=1}^{25} \gamma A_j L_j \tag{5.44}$$

donde $\boldsymbol{x}$ es una solución candidata, A_j es el área de la sección transversal del miembro j-ésimo. L_j es la longitud del miembro j y γ es el peso volumétrico del material. Las condiciones de carga se muestran en la tabla 5.20. Las coordenadas de los nodos se muestran en la tabla 5.21. Los grupos de barras se muestran en la tabla 5.22. Los datos asumidos son: módulo de elasticidad

Esfuerzos en problema k1							
Elem.	I-PAES	GSSA	VGA	MC	SSO	ISA	SAARSR
1	483.27	-447.65	-444.75	-460.10	-475.31	-209.75	-476.58
2	-73.37	0.41	3.41	-15.30	91.98	-111.35	43.99
3	-613.26	670.31	593.43	695.72	597.46	449.90	569.04
4	-478.62	499.60	440.30	503.06	461.46	239.13	485.80
5	1741.30	-1464.09	-1428.68	**-1757.16**	-1754.88	362.13	-1641.04
6	-15.72	0.41	3.41	-15.30	18.37	-866.13	14.83
7	1313.54	-1134.31	-1148.24	-1214.48	-1299.10	-763.45	-1311.60
8	-507.89	513.60	508.24	453.71	482.74	1064.60	528.83
9	482.80	-481.25	-485.97	-664.00	-461.46	-331.34	-492.79
10	103.985	0.58	-4.82	21.64	-130.07	143.23	-65.61

Tabla 5.18: Comparación de esfuerzos del problema k1. En negritas los casos que violan las restricciones.

$E = 7.3 \cdot 10^5 kg/cm^3$, máxima esfuerzo permisible $= 2787.38 kg/cm^2$ y $\gamma = 7.4239 \cdot 10^{-3} kg/cm^3$. Para este caso se comparó I-PAES con Coello [13].

El desplazamiento máximo permisible para cada nodo en cada dirección se asume como 0.889 cm. En total son 25 restricciones de esfuerzo y 18 de desplazamiento.

Resultados

Debido a las diferencias entre unidades que se manejan de un autor a otro, para este problema se decidió reportar el volumen en in^3 del mejor resultado obtenido, así como del valor de las variables de búsquda. Como se aprecia en la tabla 5.23, I-PAES obtuvo mejor resultado que [13].

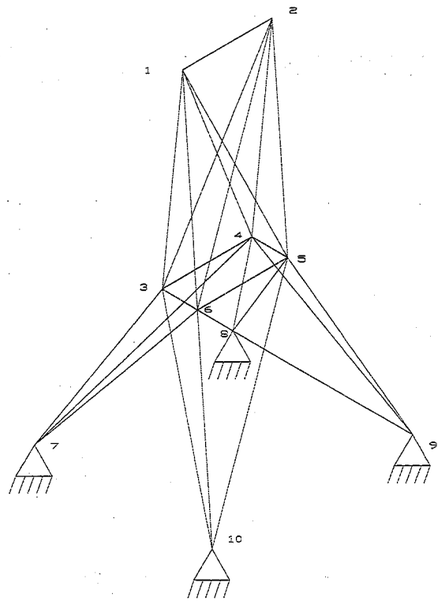

Figura 5.5: Armadura de acero del problema k2

Desplazamientos en problema k1							
Elem.	I-PAES	GSSA	VGA	MC	SSO	ISA	SAARSR
1	0.5134	0.5602	0.5528	0.5954	0.4802	0.4022	0.5419
2	-5.080	-5.0798	-4.9040	**-5.4352**	-4.9056	-3.8008	**5.0889**
3	-1.368	-1.4654	-1.2948	-1.5016	-1.3264	-0.8631	-1.3213
4	-5.060	-5.0792	-4.8997	**5.4543**	-4.8826	-4.8857	-5.0746
5	0.6053	0.5607	0.5571	0.5763	0.5954	0.2627	0.5970
6	-1.878	-1.8474	-1.8303	-1.7130	-1.8047	-2.9298	-1.9303
7	-0.768	-0.8396	-0.7433	-0.8715	-0.7484	-0.5636	-0.7129
8	-4.059	-3.6813	-3.6199	-3.9140	-4.0030	-2.4762	-3.9901

Tabla 5.19: Comparación de desplazamientos del problema k1. En negritas los casos que violan las restricciones.

Nodo	Fx(kg)	Fy(kg)	Fz(kg)
1	4540	-4540	-4540
2	0	-4540	-4540
3	227	0	0
4	272.4	0	0

Tabla 5.20: Condiciones de carga para el problema k2

5.2.19 Problema k3

Descripción

Considere la armadura mostrada en la figura 5.6. El problema consiste en encontrar el área transversal de cada elemento de la armadura que minimice su peso sujeto a restricciones de esfuerzo y desplazamiento. El peso de la armadura está dado por:

$$F(\boldsymbol{x}) = \sum_{j=1}^{49} \gamma A_j L_j \tag{5.45}$$

Nodo	X (cm)	Y (cm)	Z (cm)
1	-95.25	0	508.0
2	95.25	0	508.0
3	-95.25	95.25	254.0
4	95.25	95.25	254.0
5	95.25	-95.25	254.0
6	-95.25	-95.25	254.0
7	-254.0	254.0	0
8	254.0	254.0	0
9	254.0	-254.0	0
10	-254.0	-254.0	0

Tabla 5.21: Coordenadas de los nodos para el problema k2

Número de Grupo	Miembros
1	1-2
2	1-4, 2-3, 1-5, 2-6
3	2-5, 2-4, 1-3, 1-6
4	3-6, 4-5
5	3-4, 5-6
6	3-10, 6-7, 4-9, 5-8
7	3-8, 4-7, 6-9, 5-10
8	3-7, 4-8, 5-9, 6-10

Tabla 5.22: Grupos de barras para el problema k2

Problema k2		
Variables	I-PAES	Coello
x1 (in^2)	0.1030	0.1303
x2 (in^2)	0.1013	0.1201
x3 (in^2)	3.5594	3.4834
x4 (in^2)	0.1045	0.1102
x5 (in^2)	1.9140	1.6583
x6 (in^2)	0.7775	0.8373
x7 (in^2)	0.1379	0.1172
x8 (in^2)	3.9864	4.0900
vol (in^3)	4675.4	4700.9

Tabla 5.23: Comparación de resultados del problema k2

donde $\boldsymbol{x}$ es una solución candidata, A_j es el área de la sección transversal del miembro j-ésimo. L_j es la longitud del miembro j y γ es el peso volumétrico del material. El espacio de búsqueda para las barras es un catálogo de 233 entradas de Altos Hornos de México S.A.

Se manejarán dos casos para este problema; en el primero solo se considerará como restricción el esfuerzo en cada uno de los elementos de la estructura siendo el esfuerzo máximo permisible 3500 kg/cm^2. Para el segundo caso se agregan, además de las restricciones de esfuerzo, restricciones de desplazamiento para cada nodo. El desplazamiento total para cada nodo no debe ser mayor a 10 cm. Para el caso uno son en total 49 restricciones, mientras que para el caso dos son 73 restricciones. Los datos asumidos son: módulo de elasticidad $E = 2.1 \cdot 10^6 kg/cm^3$,máximo esfuerzo permisible $= 3500.00 kg/cm^2$, $\gamma = 7.4250 \cdot 10^{-3}$ y una carga de 4994.00 kg. horizontal aplicada en los nodos 3, 5, 7, 9, 12, 14, 16, 19, 21, 23, 25 y 27. El catálogo de donde se tomaron las secciones, fué ordenado previamente respecto al tamaño de las áreas.

Resultados

En las tablas 5.24 y 5.25 se muestra la comparación los promedios de las corridas de caso 1 y 2 respectivamente, de I-PAES y de diferentes métodos tomados de [14]. SA: recocido simulado estándar. GA50: algoritmo genético

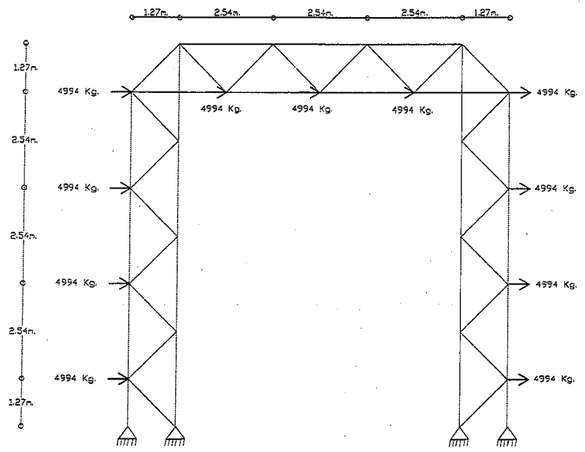

Figura 5.6: Armadura de acero del problema k3

con población de 50 individuos; probabilidad de cruza de 80 % y radio de mutación de 0.006/elemento. GSSA50: algoritmo de búsqueda estocástica general con población de 50; probabilidad de cruza de 80%; radio de mutación de 0.04/elemento y recocido esponencial con $\alpha = 1.01$. GSSA5: mismo caso que el anterior pero con población de 5 y $\alpha = 1.001$. Como puede verse en las tablas, I-PAES encuentra en promedio, mejores resultados que el resto de los algoritmos.

Problema k3 caso 1	
MÉTODO	**PESO PROMEDIO (Kg)**
I-PAES	610
SA	627
GA50	649
GSSA50	619
GSSA5	625

Tabla 5.24: Comparación de resultados del problema k3 caso 1

Problema k3 caso 2	
MÉTODO	**PESO PROMEDIO (Kg)**
I-PAES	725
SA	737
GA50	817
GSSA50	748
GSSA5	769

Tabla 5.25: Comparación de resultados del problema k3 caso 2

5.3 Poblemas Multiobjetivo

A continuación se muestra el resultado de dos puebas realizadas para el caso de problemas de mas de una función objetivo sujetas a restricciones usando I-PAES. Dado que este tipo problema tiene diferencias importantes se realizaron las siguientes modificaciones al procedimiento descrito en el capítulo 4.

1. Se elimina el elitismo extra. Dado que se trata de aproximar el frente de Pareto sujeto a restricciones, los mecanismos de elitismo del PAES son suficientes. Ya no es necesario preservar al mejor individuo encontrado que cumple con las restricciones.

2. No se escogen a los mejores de una sola función objetivo. Esto se refiere a los pasos 4.22, 4.23 y 4.24 de la función *seleccion* descrita en el capítulo cuatro donde se toman a los individuos factibles que minimzan la función objetivo, lo cual ya no aplica al ser varias las funciones objetivo. Por lo tanto estos pasos son eliminados.

3. No se reinicializan las σ. Dado que el espacio de búsqueda ya no tiende a convertirse en un punto, reinicializar las σ no permitirá que la búsqueda se vuelva fina, por lo que puede dar por resultado frentes muy "rugosos".

4. Cada restricción i se hace igual a $max(0, g_i)$

Se conservan el resto de la características del I-PAES como reducción del espacio de búsqueda. Esto con la intención de alejar a la población de zonas no factibles.

Los problemas fueron propuestos por los autores del presente trabajo dada la escacez en la bibliografía. El verdadero frente de Pareto para estos problemas no se conoce. Los resultados son buenos desde el punto de vista de uniformidad y formación de los frentes de Pareto.

5.3.1 Problema m1

Descripción

Considere la misma estructura del problema k1. Se trata de obtener un frente de Pareto del peso de la estructura contra la deformación vertical del nodo 2, sujeta a las mismas restricciones del problema original.

Resultados

En la figura 5.7 se muestra el resultado de la corrida. De la población final en el archivo, 81 individuos resultaron factibles.

5.3.2 Problema m2

Descripción

Considere la misma estructura del problema k2. Se trata de obtener un frente de Pareto del peso de la estructura contra la deformación en la dirección Y del nodo 1, sujetas a las mismas restricciones del problema original.

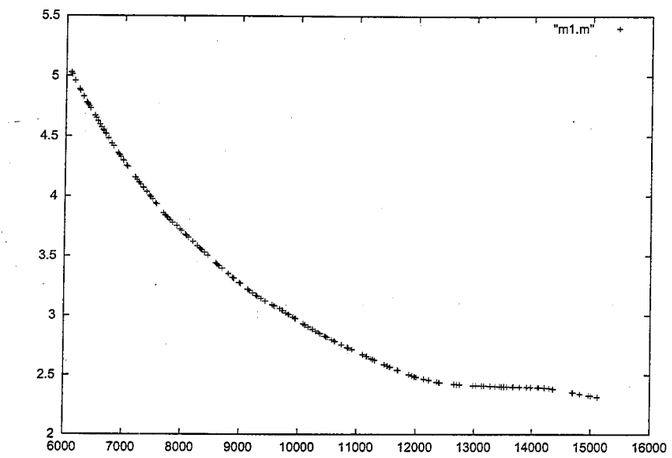

Figura 5.7: Resulado del problema m1

Resultados

En la figura 5.8 se muestra el resultado de la corrida. Del la población final en el archivo, 200 individuos resultaron factibles.

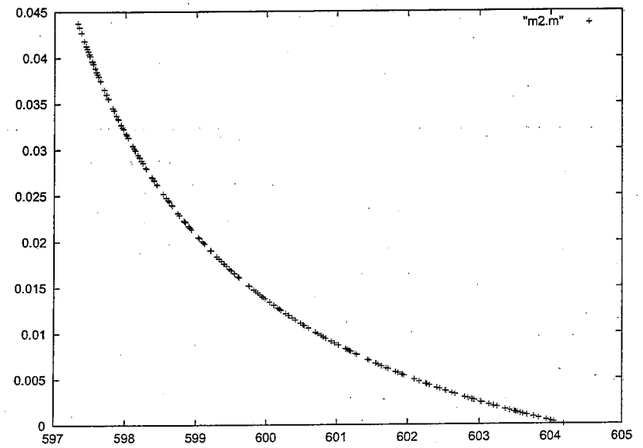

Figura 5.8: Resultado del problema m2

Conclusiones

El método propuesto I-PAES muestra tener un buen desempeño en la optimización de funciones sujetas a restricciones. Mostró un mejor desempeño que el otro algoritmo multiobjetivo para manejo de restricciones.

La implementación del mecanismo de malla probó ser muy robusto logrando manejar hasta 73 funciones objetivo simultaneamente.

El método es capaz de optimizar problemas tanto de una sola función objetivo como multiobjetivo, sujetas a restricciones.

En los casos g05 y g13 no se obtuvieron resultados. Sin embargo pruebas preliminares de I-PAES utilizando estrategias $(\mu + \lambda)$ dieron soluciones factibles.

Aunque el algoritmo parte de ideas sencillas es un poco complejo de implementar, pero no tanto como para que esto sea un impedimento para su uso.

En general mostro ser una buena opción para la resolución de problemas sujetos a restricciones dando para la mayoría de los casos resultados factibles de forma consistente.

Trabajo Futuro

Algunos de los posibles puntos a investigar a partir de este trabajo son:

- Probar el criterio de recortar el espacio de búsqueda usando otros métodos multiobjetivos como SPEA o NSGA-II.
- Probar la implementación $\mu + \lambda$ de PAES.
- Buscar mejores métodos de exploración.
- Disminuir el número de parámetros combinando, por ejemplo, la frecuencia de los cortes con la disminución del espacio de búsqueda para generar una variable que sea la pérdida de espacio de búsqueda por evaluación de función.
- Experimentar que cada individuo tenga su propio espacio de búsqueda el cual disminuirá independientemente y utilizar estos espacios en la cruza de individuos.

Apéndice A

Detalles del Control en la Reducción del Espacio de Búsqueda

Si deseamos que el hipervolumen del espacio de búsqueda no disminuya mas allá de cierta proporción α cada vez que se hace un corte, ¿que proporción β debe disminuir como máximo el rango para cada una de las variables de búsqueda? Primero tenemos la siguiente relación:

$$H_{\text{nuevo}} \geq \alpha H_{\text{anterior}} \tag{A.1}$$

donde H representa Hipervolumen y α es la proporción entre los espacios mencionada anteriormente. Luego tenemos:

$$\prod_{i=1}^{n}(\overline{x}_i^{t+1} - \underline{x}_i^{t+1}) = \alpha \prod_{i=1}^{n}(\overline{x}_i^{t} - \underline{x}_i^{t}) \tag{A.2}$$

ahora, si consideramos que la reducción del rango de cada x_i tienen todas la misma proporción a la que llamaremos β tenemos:

$$\prod_{i=1}^{n}\beta(\overline{x}_i^{t} - \underline{x}_i^{t}) = \alpha \prod_{i=1}^{n}(\overline{x}_i^{t} - \underline{x}_i^{t})$$

(A.3)

$$\beta^n \prod_{i=1}^{n} (\overline{x}_i^t - \underline{x}_i^t) = \alpha \prod_{i}^{n} (\overline{x}_i^t - \underline{x}_{i=1}^t)$$

(A.4)

$$\beta^n = \alpha \tag{A.5}$$

$$\beta = \alpha^{\frac{1}{n}} \tag{A.6}$$

La proporción β es el valor que nos interesa. Para cada variable x_i, el nuevo ancho del espacio de búsqueda debe cumplir la siguiente relación:

$$ancho_{nuevo} \geq \beta ancho_{anterior} \tag{A.7}$$

Apéndice B

Deducción de la Fórmula Para el Cálculo de DeltaMin

¿Como calculamos delta$_i$ mínimo que se debe usar para no disminuir en exceso el espacio de búsqueda? De A.7 tenemos:

$$\overline{x}_{pob,i} - \underline{x}_{pob,i} = ancho\_pob_i \tag{B.1}$$

$$\overline{x}_i^t - \underline{x}_i^t = ancho_i^t \tag{B.2}$$

$$ancho\_pob_i + 2deltaMin_i \geq \beta \cdot ancho_i^t \tag{B.3}$$

$$deltaMin_i = \frac{\beta \cdot ancho_i^t - ancho\_pob_i}{2} \tag{B.4}$$

donde deltaMin$_i$ es el valor mínimo que debe tener delta$_i$ para no violar la relación A.7 y β proviene de la deducción del apéndice A.

Bibliografía

[1] J. Nocedal y S. Wright, *Numerical Optimization*, Springer, New York, 1999.

[2] F. Hillier y G. Lieberman, *Introduction to Operations Research*, McGraw-Hill, EEUU, 1995.

[3] Mezura, E. (2001). *Uso de la Técnica Multiobjetivo NPGA para el Manejo de Restricciones en Algoritmos Genéticos.* Tesis de Maestría, Universidad Veracruzana.

[4] Zitzler, E., Deb, K., and Thiele, L. (2000). *Comparison of multiobjective evolutionary algorithms: Empirical results.* Evolutionary Computation, 8(2), 173–195.

[5] Deb. K ,Amrit Pratap, Sameer Agarwal and T. Meyarivan (2000). *A Fast and Elitist Multi-Objective Genetic Algorithm-NSGA-II.* KanGAL Report Number 2000001, Indian Institute of Technology, Kanpur, India, 2000.

[6] Eckart Zitzler and Lothar Thiele. *An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach, Technical Report 43.* Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.

[7] 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, June 1994. IEEE Service Center.

[8] Joshua D. Knowles and David W. Corne. *The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation* , In 1999 Congress on Evolutionary Computation, pages 98-105, Washington, D.C., July 1999. IEEE Service Center.

[9] Carlos A. Coello Coello. *A Short Tutorial on Evolutionary 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 21-40. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001 .

[10] Günter Rudolph. *Evolutionary Search under Partially Ordered Fitness Sets.* In Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001), pages 818-822. ICSC Academic Press: Millet Sliedrecht, 2001 .

[11] Srinivas, N. and Deb, K. (1995). *Multiobjective function optimization using nondominated sorting genetic algorithms*, Evolutionary Computation, 2(3), 221–248.

[12] T. P. Runarsson and X. Yao, *Stochastic Ranking for Constrained Evolutionary Optimization,* IEEE Transactions on Evolutionary Computation, 4(3):284-294, September 2000.

[13] Carlos A. Coello Coello. *Constraint handling through a multiobjective optimization technique* , In Annie S. Wu, editor, Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program, pages 117-118, Orlando, Florida, July 1999.

[14] Salvador Botello, Jose L. Marroquin, Eugenio Oñate y Johan Van Horebeek. *Solving Structural Optimization Problems with Genetic Algorithms and Simulated Annealing.* International Journal for Numerical Methods in Engineering 45, 1069-1084 (1999).

[15] Robert Hinterding. *Representation, Mutation and Crossover Issues in Evolutionary Computation.* Proceedings of Congress on Evolutionary Computation, pags. 916-923, San Diego, USA, July 16-19 2000.

[16] J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 2nd edition, 1992.

[17] A. Hernández, P. Buckles y C. Coello. *Estrategias Evolutivas: La Versión Alemana del Algoritmo Genético.* Soluciones Avanzadas, enero 1999, número 64, pags. 47-53.

[18] H-P. Schwefel. *Evolution and Optimum Seeking.* Wiley, New-York, 1995.

[19] C. A. Coello, G. Toscano y E. Mezura. *Current and Future Research Trends in Evolutionary Multiobjective Optimization.*

[20] Vilfredo Pareto. *Cours D'Economie Politique, volume I y II.* F. Rouge, Lausanne, 1896.

[21] J. N. Morse. *Reducing the size of the nondominated set: Pruning by clustering.* Computers and Operations Research, 7(1-2):55-66, 1980.

[22] Surry, P.D., Radcliffe, N.J., Boyd, I.D. *Patrick D. Surry, Nicholas J. Radcliffe, Ian D. Boyd: A Multi-objective Approach to Constrained Optimisation of Gas Supply Networks: the COMOGA Method*, In Terence C. Fogarty, editor, Evolutionary Computing. AISB Workshop. Selected Papers, Lecture Notes in Computer Science, pages 166-180, Sheffield, U.K., 1995. Springer-Verlag.

[23] Carlos A. Coello Coello and Gregorio Toscano. *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.

[24] Eduardo Camponogara and Sarosh N. Talukdar. *A Genetic Algorithm for Constrained and Multiobjective Optimization*, In Jarmo T. Alander, editor, 3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA), pages 49-62, Vaasa, Finland, August 1997. University of Vaasa.

[25] Porter D. Sherman. *Ranking Techniques in Multi-Criteria Genetic Algorithm-Based Optimization.* PhD thesis, Department of Computer and Information Science, Polytechnic University, Brooklyn, New York, 1995.

[26] Kalyanmoy Deb. *Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design*, In Kaisa Miettinen, Marko M. Mkel, Pekka Neittaanmki, and Jacques Periaux, editors, Evolutionary Algorithms in Engineering and Computer Science, chapter 8, pages 135-161. John Wiley & Sons, Ltd, Chichester, UK, 1999.

[27] 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, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[28] Rajeev S. y Krishamoorthy CS. *Genetic algorithms-based methodologies for design optimization of trusses.* Journal of Structural Engineering 1997; 123(3):350-358.

[29] Elperin T. *Monte-Carlo structural optimization in discrete variables with annealing algorithm.* International Journal for Numerical Methods in Engineering 1988; 26:815-821.

[30] Tzan S. y Pantelides CP. *Annealing strategy for optimal structural design.* Journal of Structural Engineering 1996; 122(7):815-827.

[31] Acckley DH. *An empirical study of bit vector function optimization.* In Genetic Algorithms and Simulated Annealing, Lawrence D (ed.) Norgan Kaufmann Publishers: Los Altos CA, 170-271.

[32] Haug EI. y Arora JS. *Applied Optimal Design Mechanical and Structural Systems.* Wiley: New Tork, 1979.

[33] Graham Kendall. *G5BAIM - Artificial Intelligence Methods.* Tomado de http://www.cs.nott.ac.uk/ gxk/courses/g5baim/002es/ES03-howtheyoperate.html, School of Computer Science and Information Technology University of Nottingham.

[34] Lawrence j. Fogel. *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming.* John Wiley & Sons, New York, 1999.

[35] Charles Darwin. *The Origin of Species by Means of Natural Selection or the preservation of Favored Races in the Struggle for life.* Random House, New York, 1993.