Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

"Optimización Multiobjetivo Usando Algoritmos Genéticos Culturales"

Tesis que presenta

Fernando Gutiérrez Méndez

Para Obtener el Grado de

Maestro en Ciencias

en Computación

Director de Tesis

Dr. Carlos A. Coello Coello

México, Distrito Federal — Diciembre, 2011

Resumen

La optimización multiobjetivo tiene amplias aplicaciones en distintas áreas de la ingeniería y las ciencias computacionales. Muchos de estos problemas tienen espacios de búsqueda muy grandes por lo que, en algunos casos, no pueden ser resueltos mediante técnicas exactas en un tiempo razonable. Para resolver este tipo de problemas suelen utilizarse metaheurísticas.

Dentro de las metaheurísticas destacan los algoritmos basados en computación evolutiva, los cuales simulan el proceso de selección del "más apto" en una computadora, a fin de resolver problemas (por ejemplo de optimización y clasificación). En los algoritmos evolutivos, las soluciones de un problema son modeladas como individuos de una población, a las cuales se le aplican operadores inspirados en la evolución biológica. Este tipo de algoritmos han sido capaces de obtener muy buenos resultados en diversos problemas del mundo real, de alta complejidad.

Una técnica de computación evolutiva son los algoritmos culturales. Esta clase de algoritmo incorpora, además del espacio de población, un espacio de creencias. El espacio de creencias es un repositorio de información extraída durante el proceso de optimización y utilizada para hacer la búsqueda más eficiente.

En esta tesis se diseñó e implementó un algoritmo cultural para resolver problemas de optimización multiobjetivo. El algoritmo propuesto utiliza el algoritmo llamado NSGA-II como su espacio de población. La propuesta consiste en incorporar un espacio de creencias a NSGA-II y utilizarlo como guía en la selección. Además, se incorpora un procedimiento de búsqueda local basado en la información contenida en el espacio de creencias.

El algoritmo propuesto fue evaluado utilizando problemas de prueba y medidas de desempeño estándar, reportados en la literatura especializada. Los resultados del algoritmo cultural multiobjetivo propuesto fueron comparados contra los obtenidos por el NSGA-II. Los resultados indicaron que el algoritmo cultural propuesto logró un mejor desempeño en la mayor parte de los problemas de prueba adoptados, indicando el potencial de este tipo de técnicas para resolver problemas con funciones objetivo computacionalmente costosas.

Abstract

Multiobjective optimization has wide applicability in different areas within engineering and computer science. Many of these problems have very large search spaces and, therefore, in some cases, cannot be solved using exact techniques, within a reasonable time. For solving these types of problems, it is common to adopt metaheuristics.

Within metaheuristics, algorithms based on evolutionary computation are a popular choice. Evolutionary algorithms simulate the "selection of the fittest" principle in a computer, with the aim of solving (for example, optimization and classification) problems. When using evolutionary algorithms, the solutions of a problem are modeled using individuals in a population, to which several operators inspired on biological evolution are applied. This type of approach has been able to obtain very good results in a variety of highly complex real-world problems.

A particular evolutionary computation technique are the so-called cultural algorithms. This type of algorithm incorporates, besides the population space, a belief space. The belief space is a repository of information which is extracted during the optimization process, and which is adopted to perform a more efficient search.

In this thesis, a cultural algorithm for solving multiobjective optimization problems is designed and implemented. The proposed approach adopts NSGA-II as its population space. This work incorporates a belief space into NSGA-II to guide the selection. Furthermore, it incorporates a local search procedure which is based on information contained in the belief space.

The proposed approach was assessed using several standard test problems and performance measures reported in the specialized literature. The results of the proposed multiobjective cultural algorithm were compared with respect to those obtained by NSGA-II. Our results indicate that the proposed approach had a better performance in most of the test problems adopted, which indicates the potential of this sort of approach for dealing with computationally expensive objective functions.

Agradecimientos

Agradezco a mis padres por el apoyo que siempre me han brindado. Gracias por su paciencia y dedicación.

A mi tía Juani, gracias por apoyarme con todo lo necesario para lograr mis metas.

Gracias a todos mis compañeros de generación por brindarme su amistad.

Al doctor Carlos Coello, sin su guía este trabajo no hubiera sido posible.

A mis sinodales, Dr. Ricardo Landa Becerra y Dr. Luis Gerardo de la Fraga por sus comentarios los cuales permitieron hacer de este un mejor trabajo.

A Sofi por ser siempre tan amable y paciente con nosostros. Gracias por apoyarnos en cada momento.

Al CINVESTAV por permitirme formar parte de esta gran institución.

Al CONACyT por el apoyo económico proporcionado.

Este trabajo de tesis se derivó del proyecto CONACyT titulado "Escalabilidad y Nuevos Esquemas Híbridos en Optimización Evolutiva Multiobjetivo" (Ref. 103570), cuyo responsable es el Dr. Carlos A. Coello Coello.

Índice general

Índice de figuras

Índice de Tablas

Lista de algoritmos

Capítulo 1

Introducción

La optimización multiobjetivo tiene amplias aplicaciones en distintas áreas de la ingeniería y las ciencias computacionales. Entre la amplia gama de algoritmos existentes para atacar este tipo de problemas, destacan los basados en la computación evolutiva.

Dentro de la computación evolutiva encontramos una clase de algoritmos llamados algoritmos culturales. Los algoritmos culturales incorporan conocimiento del dominio obtenido durante el proceso de búsqueda para hacerlo más eficiente. Sin embargo, no existen muchas propuestas sobre cómo aplicarlos a problemas multiobjetivo con y sin restricciones.

En este trabajo se plantea el desarrollo de un algoritmo cultural para resolver problemas multiobjetivo por medio de la incorporación de conocimiento del dominio a un algoritmo evolutivo multiobjetivo existente.

1.1. Antecedentes

Los algoritmos evolutivos utilizan mecanismos inspirados en la evolución biológica tales como la reproducción, mutación, recombinación y selección. Las soluciones son modeladas como individuos de una población. Una función de aptitud determina el entorno donde "viven" las soluciones.

En el modelo de un algoritmo cultural, un individuo es representado por dos términos: un conjunto de características distintivas y las experiencias que ha adquirido. Las características distintivas son transmitidas entre generaciones a través de operadores inspirados en mecanismos biológicos. Las experiencias de varios individuos son combinadas y modificadas para dar lugar a una experiencia de grupo. Esta experiencia de grupo recibe el nombre de *espacio de creencias*. Las experiencias de un individuo pueden ser incorporadas al espacio de creencias si se cumplen ciertas condiciones. La información contenida en el espacio de creencias es utilizada para modificar el desempeño de los individuos de la población [1].

En un algoritmo cultural se habla también de un canal de comunicación o protocolo que regula la forma en que la información contenida en el espacio de creencias afecta

a los individuos y también la forma en que los individuos contribuyen al espacio de creencias.

El espacio de población puede ser modelado de muchas formas, por ejemplo por medio de:

- Programación evolutiva
- Estrategias evolutivas
- Algoritmos genéticos
- Evolución diferencial

La representación del espacio de creencias también puede llevarse a cabo de muchas formas; algunos ejemplos son:

- Conjuntos
- Particiones del espacio
- Grafos
- Redes semánticas
- Espacios de versiones

Las posibles combinaciones dan lugar a muchas formas de plantear un algoritmo cultural. Sin embargo, son pocas las opciones que han sido exploradas a la fecha.

Reynolds [1] plantea un algoritmo cultural basado en un algoritmo genético para representar el espacio de población y un espacio de creencias basado en espacios de versiones [2]. El algoritmo resultante fue llamado *Version Space guided Genetic Algorithm (VGA)*. Los resultados obtenidos con este algoritmo fueron prometedores.

Ricardo Landa [3] propone un algoritmo cultural basado en programación evolutiva [4] (*Cultural Algorithm with Evolutionary Programming, CAEP*) para la solución de problemas multiobjetivo. El espacio de creencias está basado en una cuadrícula que contiene información usada para guiar la búsqueda hacia soluciones no dominadas distribuidas uniformemente sobre el frente de Pareto. Los resultados obtenidos son buenos salvo por algunas limitaciones como la pérdida de diversidad en algunos casos y el potencial para mejorar respecto a poder generar una distribución más uniforme de las soluciones.

Chrisopher Best [5] propone una extensión de CAEP. El algoritmo propuesto incluye varias fuentes de conocimiento para guiar la búsqueda: situacional, de dominio, normativo, histórico y topológico. Como trabajo futuro los autores del algoritmo recomiendan investigar más a fondo el conocimiento topológico para lograr una mejor distribución de las soluciones sobre el frente de Pareto.

Los algoritmos culturales han demostrado excelentes resultados en problemas del mundo real. Además, como se mostró al inicio de esta sección, existe un número considerable de formas de plantear un algoritmo cultural, lo que los convierte en un tema de investigación muy atractivo.

1.2. Objetivos

En esta tesis se propone un algoritmo cultural para optimización multiobjetivo. La técnica propuesta incorpora conocimiento de dominio al algoritmo llamado NSGA-II [6]. Se decidió utilizar NSGA-II como base para el algoritmo por ser un método probado en una amplia variedad de problemas. Además NSGA-II tiene la característica de lograr rápidamente una buena dispersión de las soluciones lo cual es ventajoso en nuestra propuesta.

La forma en que se logrará el objetivo de esta tesis es mediante el desarrollo de los siguientes temas:

- Investigación sobre algoritmos evolutivos existentes con el objetivo de decidir cuál es el mejor candidato para incorporarle un mecanismo de conocimiento de dominio en su proceso de búsqueda.
- Planteamiento del esquema de conocimiento de dominio. Esto incluye el estudio de las distintas fuentes de conocimiento posibles.
- Análisis de la forma más adecuada de modelar y representar el espacio de creencias.

Dentro de los objetivos también se contempla la implementación de los siguientes puntos:

- Algoritmo cultural a partir del esquema de conocimiento de dominio planteado.
- Conjunto de experimentos basados en un conjunto de problemas de prueba estándar para comprobar el desempeño del algoritmo propuesto.

1.3. Contribuciones

El producto principal de este trabajo de tesis es la propuesta de un algoritmo evolutivo culturizado para resolver problemas de optimización multiobjetivo.

Se realizó una implementación del algoritmo propuesto la cual fue utilizada para realizar un conjunto de experimentos que validaron su funcionamiento. Éstos resultados pueden verse en el capítulo 6.

Finalmente, se validó la técnica propuesta utilizando un conjunto de problemas de prueba estándar. Los resultados de estas pruebas permitieron identificar las fortalezas y debilidades del algoritmo propuesto, tal y como se describe en el capítulo 7.

1.4. Estructura de la tesis

Los capítulos que componen el presente trabajo se encuentran organizados de la siguiente forma:

- **Capítulo 2.** Se presenta una descripción de los conceptos básicos de optimización. Además, se describen los principios, conceptos y mecanismos biológicos que sirven de inspiración a los algoritmos evolutivos. También se muestra un breve panorama de algunas de las técnicas existentes dentro de la computación evolutiva.

- **Capítulo 3.** En este capítulo se describen los operadores de búsqueda local.

- **Capítulo 4.** Aquí se enumeran algunos de los algoritmos culturales multiobjetivo existentes exponiendo sus ventajas y limitaciones. Se describe a detalle la estructura de los llamados "CAEP" (*Cultural Algorithms with Evolutionary Programming*) pues el algoritmo propuesto en esta tesis es de este tipo.

- **Capítulo 5.** Se describe el algoritmo cultural con programación evolutiva propuesto. Primeramente se describe el algoritmo, después el espacio de creencias utilizado y, finalmente, los parámetros del algoritmo junto con una guía sobre cómo asignarlos.

- **Capítulo 6.** Se presentan los resultados obtenidos por el algoritmo propuesto basándose en una comparación contra NSGA-II y empleando un conjunto de problemas de prueba y medidas de desempeño estándar reportadas en la literatura especializada.

- **Capítulo 7.** Se presentan las conclusiones obtenidas con base en los resultados experimentales y se proponen posibles mejoras así como algunas áreas prometedoras de trabajo futuro.

Capítulo 2

Conceptos básicos

La optimización tiene amplias aplicaciones en diversas ramas del conocimiento. Su objetivo es encontrar las mejores soluciones a problemas del mundo real mediante el uso de modelos matemáticos. En optimización, existen dos tipos de formulaciones: en una de ellas se intenta optimizar un único objetivo y en la segunda se optimizan de forma simultánea dos o más objetivos.

Para resolver cualquiera de estos dos tipos de problemas, existe una enorme cantidad de técnicas de programación matemática y métodos numéricos. Sin embargo, todas ellas requieren de restricciones e información adicional del modelo como son continuidad y derivabilidad.

Una alternativa para resolver problemas de optimización son las heurísticas, las cuales pueden obtener soluciones razonables a un costo computacional razonable, pero sin garantizar optimalidad (o incluso, factibilidad) de las soluciones obtenidas [1] [7].

Dentro de las heurísticas podemos encontrar la computación evolutiva. En este tipo de algoritmos las soluciones de un problema son vistas como individuos de una población. Estos individuos son mejorados utilizando mecanismos inspirados en la evolución de los seres vivos.

Una propuesta para mejorar el desempeño de los algoritmos evolutivos son los algoritmos culturales. Esta clase de algoritmos se pueden ver como como un esquema de alto nivel en el que se incorpora conocimiento del dominio obtenido durante el proceso de búsqueda a fin de hacer más eficiente dicho proceso.

2.1. Optimización

Optimizar es un objetivo común en muchas áreas de la ciencia y la tecnología. Los ingenieros ajustan parámetros para lograr el mejor desempeño de un sistema. Los científicos buscan valores que mejor ajusten un modelo a un experimento.

Para lograr aplicar las técnicas de optimización primero es necesario identificar los **objetivos**. Estos son, por ejemplo, el costo de producción de un diseño, tiempo

[1] Aunque, en la práctica, dichas soluciones suelen ser aproximaciones razonablemente buenas de los óptimos deseables.

necesario para implementarlo, error de aproximación, etc.

El segundo elemento a identificar son los parámetros que permiten ajustar las características del problema. Nuestro objetivo será encontrar los valores de estos parámetros que optimicen el problema.

Finalmente, se deben definir las **restricciones** que deben cumplir los parámetros. Por ejemplo, si un parámetro es una distancia, entonces éste no puede tomar valores negativos.

En resumen, para modelar un problema se deben identificar los siguientes puntos:

- *Variables de decisión.* Son los parámetros que determinan una posible solución del problema. Los métodos propuestos en este trabajo son aplicables únicamente en problemas para los cuales sus variables de decisión son números reales. Las variables de decisión se denotan por el vector

$$\vec{x} = [x_1, x_2, \ldots, x_n]^T, \qquad x_i \in \mathbb{R}$$

  A una elección particular de los elementos de un vector de variables de decisión se le llama *solución* del problema de optimización.

- *Función objetivo.* Es una función vectorial

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

  El resultado de evaluar las variables de decisión en esta función es lo que se desea optimizar. En este trabajo se tratará únicamente con funciones reales ($f_i : \mathbb{R} \rightarrow \mathbb{R}$). La función objetivo describe cómo se comporta el sistema a ser optimizado bajo una elección particular de los valores de las variables de decisión. Cada uno de los componentes del vector resultante de evaluar la función objetivo en un vector de variables de decisión representa una medida de los criterios a optimizarse.

- *Restricciones.* Se denominan restricciones a cada uno de los elementos del conjunto de igualdades y/o desigualdades que involucran a una o más de las variables de decisión. Una solución del problema de optimización debe de satisfacer todas las restricciones. Se dice que una solución es factible si satisface todas las restricciones. Una solución que no cumple una o más restricciones es llamada infactible.

Decimos que en un **problema de optimización multiobjetivo** se busca el vector de variables de decisión $\vec{x}^* \in \mathbb{R}^n$ que optimice la función vectorial:

$$\vec{f}(\vec{x}) \qquad \vec{f} : \mathcal{F} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^k$$

Donde $\mathcal{F}$ es llamada la **región factible** y está determinada por l restricciones de desigualdad y m restricciones de igualdad:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, l$$
$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, m$$

Se debe observar que aunque en la definición anterior se considera un problema de minimización es posible transformar, sin pérdida de generalidad, el problema a uno de maximización cambiando en la función objetivo cada f_i por $-f_i$.

2.1.1. Dominancia de Pareto

Antes de hablar de soluciones óptimas en problemas de optimización multiobjetivo es necesario introducir el concepto de dominancia de Pareto. Este concepto se utiliza para comparar dos soluciones considerando más de una función objetivo.

Decimos que un vector $\vec{v} = [v_1, \ldots, v_k]^T$ es menor o igual que otro vector $\vec{w} = [w_1, \ldots, w_k]^T$ (denotado por $\vec{v} \leq_p \vec{w}$) si y sólo si:

$$\forall i \in \{1, \ldots, k\} : v_i \leq w_i$$

Una solución $\vec{x} = [x_1, \ldots, x_n]^T$ domina a otra solución $\vec{y} = [y_1, \ldots, y_n]^T$ (escrito como $\vec{x} \prec \vec{y}$) respecto a un problema de optimización multiobjetivo $\vec{f}$ si y sólo si:

$$\vec{f}(\vec{x}) \leq_p \vec{f}(\vec{y}) \qquad \wedge \qquad \vec{f}(\vec{x}) \neq \vec{f}(\vec{y})$$

O de forma equivalente:

$$\forall i \in \{1, \ldots, k\} : x_i \leq y_i \qquad \wedge \qquad \exists i \in \{1, \ldots, k\} : x_i < y_i$$

La definición de dominancia surge de la idea intuitiva de cómo una solución es mejor que otra si la supera en por lo menos un objetivo y no es peor en ninguno de los otros.

2.1.2. Optimalidad de Pareto

Cuando hablamos de soluciones óptimas en problemas multiobjetivo, éstas se deben entender bajo el concepto de **optimalidad de Pareto**. Decimos que una solución $\vec{x}^* \in \mathcal{F}$ es Pareto-óptima respecto a un problema de optimización multiobjetivo si y sólo si no existe otra solución $\vec{x}$ tal que $\vec{x}$ domina a $\vec{x}^*$.

Es importante notar que usando esta definición de optimalidad, la solución a un problema no es única sino que existe un conjunto de soluciones óptimas al problema de optimización multiobjetivo.

Al conjunto de soluciones Pareto óptimas se le denomina **conjunto de óptimos Pareto** ($\mathcal{P}$). La imagen de este conjunto se denomina el **frente de Pareto** ($\mathcal{PF}$).

2.1.3. Problemas de optimización

Existen problemas para los cuales las variables de decisión sólo son válidas si sus valores son enteros. A los problemas de este tipo se les denomina **problemas de programación entera**. Éstos son un caso particular de los llamados problemas de

optimización discreta. En dichos problemas los valores de las variables de decisión son tomados de un conjunto finito de elementos.

En contraste, para los **problemas de optimización continua**, las variables de decisión toman valores de un conjunto infinito de elementos. [2]

Los problemas de optimización continua suelen ser más sencillos de resolver pues la información sobre la función objetivo y las restricciones en un punto particular ayudan a deducir información sobre todos los puntos cercanos. En el caso de los problemas discretos el comportamiento de la función objetivo y las restricciones pueden variar considerablemente entre dos puntos considerados cercanos [8].

En este trabajo únicamente serán considerados los problemas de optimización continua donde las variables de decisión son números reales.

Un caso particular de los problemas de optimización ocurre cuando la función objetivo y las restricciones son funciones lineales. En este caso se dice que se trata de un **problema de programación lineal**. Existen varios métodos para resolver problemas de este tipo que garantizan encontrar la solución exacta, como por ejemplo el llamado método simplex [9].

Cuando por lo menos una de las restricciones o la función objetivo son funciones no lineales, decimos que el problema es de **programación no lineal**. Para este tipo de problemas no existe un método general que garantice encontrar la mejor solución posible en tiempo polinomial [10, 11].

La mayoría de los algoritmos para problemas de optimización no lineal encuentran únicamente óptimos locales, es decir, puntos óptimos respecto a una región cercana, en contraste con los óptimos globales, los cuales son los mejores respecto a todos los demás puntos factibles.

Existen métodos para resolver problemas de optimización no lineal pero generalmente requieren de la primera derivada de la función objetivo. Esta información no siempre se encuentra disponible o es costosa de calcular.

En la práctica, suelen utilizarse **heurísticas** para resolver problemas de programación no lineal. Un ejemplo de heurística es el **recocido simulado** [12].

Una familia de heurísticas muy popular es la **computación evolutiva**, la cual ha mostrado resultados competitivos en una amplia variedad de problemas [13, 11].

Los algoritmos de optimización son métodos iterativos. Comienzan con una aproximación (posiblemente aleatoria) y generan un secuencia de estimaciones hasta alcanzar una cierta condición de paro. La estrategia que decide cómo pasar de una estimación a otra es lo que distingue a un algoritmo de otro.

Un buen algoritmo de optimización debe tener las siguientes características [8]:

- *Robustez.* Debe desempeñarse bien en una amplia variedad de problemas.
- *Eficiencia.* No debe requerir de recursos de cómputo excesivos.
- *Exactitud.* Debe identificar la solución con precisión.

[2] En teoría, debido a que los reales se representan de forma finita en una computadora.

Un buen algoritmo debe ofrecer un compromiso adecuado entre estas características.

2.1.4. Objetivo deseado

Como se ha mencionado antes, la solución a un problema de optimización multiobjetivo no es única. La respuesta al problema es en realidad un conjunto de soluciones óptimas.

Por lo tanto, es deseable encontrar el mayor número de soluciones posibles y además es deseable que dicho conjunto cumpla con un par de características:

- Las soluciones deben encontrarse lo más cerca posible del verdadero frente de Pareto del problema a resolver.
- Las soluciones deben encontrarse distribuidas lo más uniformemente posible sobre el frente de Pareto.

2.2. Cómputo evolutivo

Los algoritmos evolutivos hacen uso de principios, conceptos y mecanismos inspirados en la evolución de los seres vivos. La simulación de estos procesos naturales en la computadora da origen a técnicas de optimización que suelen tener un buen desempeño en problemas del mundo real, si bien no pueden garantizar que siempre convergerán a la mejor solución posible [14].

En un algoritmo de computación evolutiva las soluciones potenciales de un problema son modeladas como individuos de una población. El entorno donde existen los individuos es determinado por la función objetivo y sus restricciones. Es el entorno el que determina la capacidad de supervivencia de los individuos.

A diferencia de los algoritmos clásicos, los algoritmos evolutivos trabajan con poblaciones, es decir con conjuntos de soluciones. Esto es una ventaja pues en una única ejecución del algoritmo podemos encontrar varias soluciones.

En cada iteración de un algorimo evolutivo se aplican algunos operadores de variación a los elementos de la población con el fin de generar nuevas soluciones las cuales son preservadas o descartadas de acuerdo a un mecanismo de selección. El proceso se detiene después de un número de iteraciones (llamadas generaciones en el contexto de los algoritmos evolutivos).

Los operadores más comunes son la *recombinación* y la *mutación*. En la recombinación se combinan dos o más padres para dar lugar a uno o más individuos nuevos llamados hijos. En la mutación se aplica una variación aleatoria a un individuo para dar lugar a uno nuevo.

Existen tres paradigmas dentro del cómputo evolutivo los cuales se describen brevemente a continuación.

2.2.1. Programación evolutiva

Propuesta por Lawrence J. Fogel [15] y adaptada por David Fogel [16] para la optimización numérica.

Inicialmente se tiene una población de tamaño μ. No se utiliza un operador de recombinación por lo que las variaciones son únicamente producto del operador de mutación.

El operador de mutación genera un hijo por cada individuo. El mecanismo de selección se realiza por medio de una serie de torneos entre las $\mu + \mu$ soluciones. Al final del torneo se tienen μ individuos que pasarán a la siguiente generación.

Algoritmo 2.1 Programación Evolutiva

```
Generar la población inicial
Evaluar la población inicial
repetir
  para todo Individuo en la poblacion hacer
    Generar hijo usando mutación
  Evaluar hijos
  Realizar selección por torneo
hasta que Condición de paro se cumpla
```

2.2.2. Estrategias evolutivas

Las estrategias evolutivas fueron propuestas por Ingo Rechenberg [17]. En la versión original llamada $(1 + 1) - \text{ES}$ se utiliza un único padre el cual genera un único hijo. El hijo sobrevive sólo si es mejor que el padre. Para la generación de hijos se utiliza un operador de mutación.

Algoritmo 2.2 Estrategia evolutiva $(1 + 1) - \text{ES}$

```
Generar una población inicial
repetir
  para todo Individuo en la población hacer
    Generar individuo hijo
    Agregar el mejor individuo (padre o hijo) a la nueva población
hasta que Condición de paro se cumpla
```

Existen también estrategias evolutivas multimiembro. En ellas se utilizan μ padres para generar λ hijos. En este caso existen dos formas para realizar la selección. En la primera, llamada (μ, λ) sólo se toman en cuenta los μ mejores individuos de entre los λ hijos. En la segunda se considera la unión de los μ padres con los λ hijos para realizar la selección; ésta es llamada estrategia $(\mu + \lambda)$.

En las estrategias evolutivas puede utilizarse la recombinación, sin embargo, el operador principal es la mutación.

2.2.3. Algoritmos genéticos

Inicialmente fueron propuestos como un método para resolver problemas de aprendizaje de máquina por John Holland [18]. Actualmente, son el paradigma más popular en los algoritmos evolutivos.

A diferencia de las estrategias evolutivas, en los algoritmos genéticos el operador de recombinación es el principal. La cruza permite la distribución de los segmentos de información de los individuos a sus descendientes lo cual hace posible que la información útil se propague a lo largo de las generaciones. El operador de mutación puede destruir esta información por lo que es relegado al papel de operador secundario.

Algoritmo 2.3 Algoritmo genético

```
Generar una población inicial
repetir
  Seleccionar padres
  Generar población de hijos usando el operador de cruza
  Aplicar el operador de mutación
  Evaluar hijos
hasta que Condición de paro se cumpla
```

2.3. Algoritmos culturales

Existen una infinidad de problemas para los cuales las técnicas clásicas de optimización no resultan prácticas debido a lo grande y accidentado de sus espacios de búsqueda.

Los algoritmos evolutivos surgieron de la necesidad de resolver este tipo de problemas. Usualmente, en estos problemas no existe información adicional que pueda ser útil o si existe, es muy difícil de incorporar en el proceso de búsqueda. La única fuente de conocimiento sobre el problema a resolver que es usada por un algoritmo evolutivo es la función de aptitud [19].

Robert Reynolds sugirió un nuevo tipo de algoritmos evolutivos con conocimiento del dominio. A diferencia de otras propuestas en las que el conocimiento se integra *a priori*, aquí el conocimiento se extrae durante el proceso de búsqueda. A este tipo de algoritmos se les denomina **algoritmos culturales** [1].

Los algoritmos culturales son técnicas que incorporan conocimiento del dominio a los métodos de computación evolutiva. Se basan en la extracción de conocimiento del dominio durante el proceso evolutivo mediante la evaluación de las soluciones generadas hasta el momento.

El fenómeno llamado **evolución cultural** ha sido objeto de estudio de varias teorías sociales y antropológicas y sirve como modelo base a los algoritmos culturales. Estas teorías proponen que la evolución de las sociedades ocurre a un ritmo más rápido que la evolución a nivel genético.

Bajo estas teorías, la evolución cultural es vista como un proceso de herencia en dos niveles: el nivel **micro-evolutivo**, que consiste en el material genético heredado de padres a hijos, y el nivel **macro-evolutivo**, que es el conocimiento adquirido por los individuos a través de las generaciones y que sirve para guiar el comportamiento de los individuos que pertenecen a una población.

La cultura puede verse como un conjunto de fenómenos ideológicos compartidos por una población por medio de los cuales un individuo puede interpretar sus experiencias y decidir su comportamiento. Pueden identificarse dos componentes de este sistema: el conocimiento, el cual es compartido por todos los individuos de la población, y la interpretación de dicho conocimiento, característica única en cada individuo.

La cultura afecta el éxito y la supervivencia de individuos y grupos, llevando hacia procesos evolutivos que son tan importantes como los causados por la variación genética [20].

Reynolds adoptó el modelo de herencia doble (micro-evolutivo, macro-evolutivo) como inspiración para crear los algoritmos culturales [1].

En un algoritmo cultural pueden identificarse tres componentes principales:

- El **espacio de población**. Es un conjunto de posibles soluciones al problema. Puede modelarse usando cualquier técnica basada en poblaciones, como por ejemplo, los algoritmos genéticos.

- El **espacio de creencias**. Es un depósito de información donde los individuos pueden almacenar sus experiencias. En un algoritmo cultural la información adquirida por un individuo puede ser compartida con toda la población a través del espacio de creencias.

- Un **canal de comunicación**. Los dos espacios antes mencionados se encuentran conectados por un canal de comunicación. Este canal establece reglas mediante las cuales los individuos pueden incorporar sus experiencias al espacio de creencias. El conjunto de reglas es implementado mediante la llamada **función de aceptación**. El canal de comunicación también define la forma en que el espacio de creencias puede afectar a los nuevos individuos. Dicha afectación se realiza por medio de la llamada **función de influencia**.

Las interacciones entre los espacios de población y creencias se muestran en la figura 2.3.

La primera etapa en un algoritmo cultural es la inicialización del espacio de creencias. Durante esta etapa se preparan las estructuras de datos necesarias para mantener la información (cultura) que guiará el proceso de búsqueda.

Posteriormente, en cada iteración del algoritmo se procede a elegir un selecto grupo de individuos los cuales aportarán experiencias positivas y negativas al espacio de creencias. A este proceso se le conoce como actualización del espacio de creencias.

La información que constituyen estas experiencias puede ser muy variada, yendo desde regiones prometedoras hasta la densidad de soluciones factibles en una región

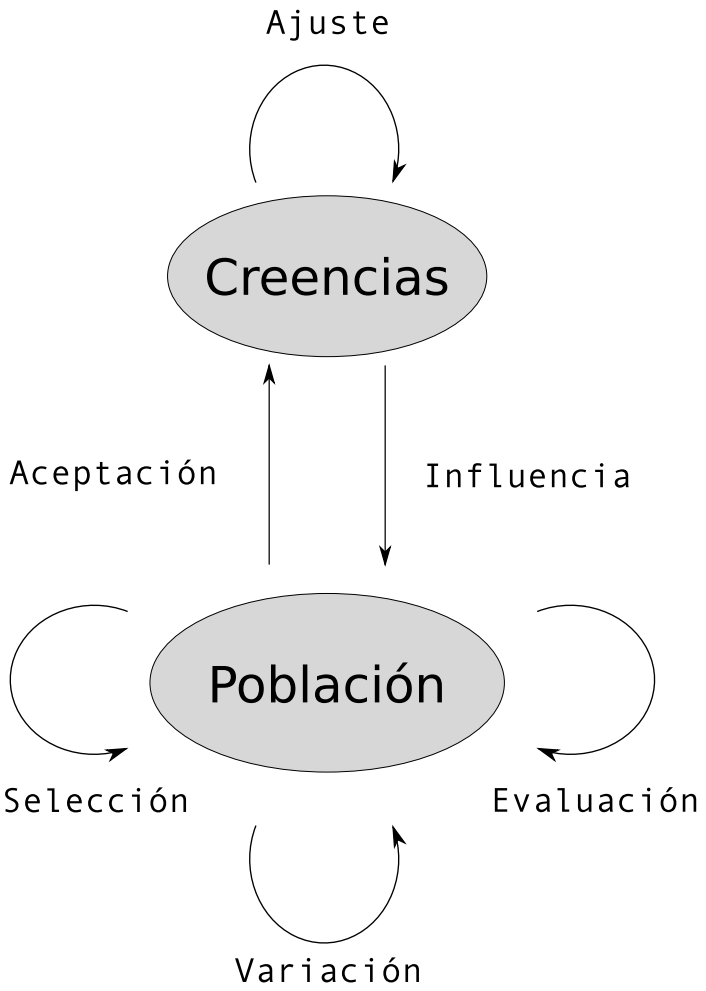

Figura 2.1: Interacciones entre los espacios de creencias y de población

particular del espacio de búsqueda. La elección de las fuentes de información correctas determina en gran medida el éxito de un algoritmo cultural.

En un algoritmo cultural, los operadores de recombinación y mutación son modificados por una función de influencia.

La función de influencia introduce un sesgo sobre los individuos generados de modo que éstos se acerquen a los comportamientos deseables de acuerdo a la información almacenada en el espacio de creencias y también aleja a los individuos generados de los comportamientos no deseables.

La estructura general de un algoritmo cultural se presenta en el algoritmo 2.4.

Algoritmo 2.4 Algoritmo cultural

```
Generar una población inicial
Evaluar población inicial
Inicializar el espacio de creencias
repetir
    Aceptar individuos de la población
    Actualizar espacio de creencias usando individuos aceptados
    Generar nueva población usando operadores influenciados por el espacio de creencias
    Evaluar la nueva población
    Seleccionar la población para la siguiente iteración
hasta que Condición de paro se cumpla
```

2.4. NSGA-II

Deb. et al. [6] propusieron un algoritmo evolutivo para optimización multiobjetivo llamado Nondominated Sorting Genetic Algorithm II (NSGA-II) el cual se ha vuelto muy popular debido a su eficiencia y eficacia. Dado que la propuesta descrita en esta tesis se basa en el NSGA-II, a continuación lo describiremos brevemente.

NSGA-II emplea una jerarquización de Pareto implementada eficientemente e incluye un mecanismo para la preservación de diversidad.

Además, incorpora la noción de elitismo, es decir, adopta un mecanismo para retener a las mejores soluciones generadas durante la búsqueda. Esto se logra combinando en cada generación la población de padres con la población de hijos de forma que compitan entre sí para pasar a la siguiente generación.

2.4.1. Jerarquización de Pareto

La jerarquización de Pareto adoptada por el NSGA-II consiste en clasificar los individuos en varias categorías (figura 2.4.1). Dicha clasificación se lleva a cabo de la siguiente forma:

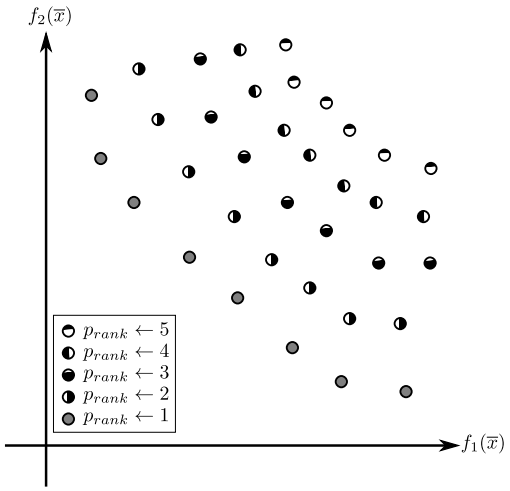

Figura 2.2: Ejemplo de una clasificación en jerarquías de Pareto en un problema con dos objetivos. Los puntos son las soluciones generadas hasta el momento. NSGA-II asigna un parámetro $p_{\text{rank}} = 1$ a los puntos no dominados, $p_{\text{rank}} = 2$ a los no dominados que quedan después de eliminar aquellos con $p_{\text{rank}} = 1$. El proceso se repite hasta clasificar todas las soluciones.

- El conjunto de individuos no dominados pertenece al primer nivel en la jerarquización de Pareto.
- Posteriormente, se elimina el conjunto de individuos no dominados de la población y se encuentra el nuevo conjunto de individuos no dominados en la población restante. Este conjunto constituye el segundo nivel en la jerarquía.
- El proceso continúa hasta que todos los individuos hayan sido clasificados.

Observamos que si k es el número de objetivos y m el número de individuos en la población entonces el procedimiento descrito anteriormente requiere $O(km)$ comparaciones para decidir si un individuo se encuentra en el primer nivel de la jerarquía. Para completar el primer nivel se habrán utilizado $O(km^2)$ comparaciones. Si en cada nivel existen $O(m)$ individuos, entonces para realizar toda la clasificación se requieren $O(km^3)$ comparaciones.

En NSGA-II se describe un método alternativo de realizar la clasificación el cual solo requiere de $O(km^2)$ comparaciones. Este método se describe en el algoritmo 2.5.

2.4.2. Preservación de diversidad

NSGA-II incluye un estimador de densidad simple y efectivo. El objetivo de este indicador es ordenar los individuos que tengan una misma categoría con respecto a la

Algoritmo 2.5 Ordenamiento no dominado rápido

Entrada: Población de individuos P
Salida: Conjunto $F_1, \ldots, F_s$ de frentes de Pareto

```
para todo p⃗ en P hacer
    S_p = ∅
    n_p = 0
    para todo q⃗ en P hacer
        si p⃗ ≺ q⃗ entonces
            S_p ← S_p ∪ {q⃗}
        si no, si q⃗ ≺ p⃗ entonces
            n_p ← n_p + 1
    si n_p = 0 entonces
        p_rank ← 1
        F_1 ← F_1 ∪ {p⃗}
i ← 1
mientras F_i ≠ ∅ hacer
    Q ← ∅
    para todo p⃗ en F_i hacer
        para todo q⃗ en S_p hacer
            n_q ← n_q − 1
            si n_q = 0 entonces
                q_rank ← i + 1
                Q ← Q ∪ {q⃗}
    i ← i + 1
    F_i ← Q
```

jerarquización de Pareto de manera que los individuos que estén en regiones menos pobladas del espacio de búsqueda sean favorecidos por el mecanismo de selección.

La densidad se estima calculando la distancia promedio a los dos individuos a cada lado de una solución para cada uno de los objetivos.

Esta medida se calcula utilizando el algoritmo 2.6. Un individuo con una medida mayor se encuentra en una región menos poblada que un individuo con menor medida.

Algoritmo 2.6 Distancia de agrupamiento

Entrada: Población de individuos P
Salida: Medida para cada individuo en P

$m \leftarrow |P|$
para todo $\vec{p}$ en P **hacer**
  $p_{\text{distance}} \leftarrow 0$
para todo objetivo f_j **hacer**
  $Q \leftarrow$ Conjunto ordenado integrado por los elementos de P de acuerdo al valor del objetivo f_j
  {$Q[i]$ se refiere al elemento en la posicion i del conjunto ordenado Q, siendo $Q[1]$ el primer elemento}
  $Q[1]_{\text{distance}} \leftarrow \infty$
  $Q[m]_{\text{distance}} \leftarrow \infty$
  $f_j^{\max} \leftarrow \text{máx}_{\vec{q} \in P}\, f_j(\vec{q}))$
  $f_j^{\min} \leftarrow \text{mín}_{\vec{q} \in P}\, f_j(\vec{q}))$
  para $i = 2$ hasta $(m-1)$ **hacer**
    $Q[i]_{\text{distance}} \leftarrow \frac{f_j(P[i+1]) - f_j(I[i-1])}{f_j^{\max} - f_j^{\min}}$

2.4.3. Selección

La fortaleza de NSGA-II radica en la forma en que se realiza la selección: al comparar dos individuos se elige al que pertenezca a la jeraraquía de Pareto más baja y en caso de que ambos individuos pertenezcan a la misma categoría, entonces se elige a aquél ubicado en una región con menor densidad de población. Dado que la selección se aplica a la unión de padres e hijos, esta es implícitamente elitista, ya que siempre se retendrán los individuos que resulten mejor clasificados, independientemente de la población a la que pertenezcan.

Capítulo 3

Operadores de búsqueda local

Para la solución de problemas de optimización multiobjetivo existen diferentes clases de algoritmos.

Por un lado, tenemos los algoritmos que sirven para encontrar una única solución de forma muy rápida. Un ejemplo clásico es el método de Newton, el cual converge de manera muy rápida [8] pero tiene dificultad para encontrar el frente de Pareto completo.

Por otro lado, existen los métodos globales, los cuales destacan en lograr encontrar la solución global de manera efectiva pero lo hacen a una velocidad de convergencia mucho menor. Dentro de esta clase de algoritmos destacan los algoritmos evolutivos.

Finalmente, existen los algoritmos que hibridizan un método de búsqueda global con una estrategia de búsqueda local. El objetivo de esta clase de algoritmos es lograr la robustez de los métodos globales pero con la mejora en la velocidad de convergencia de los métodos locales. El algoritmo propuesto en este trabajo entra dentro de esta última categoría.

La búsqueda local se lleva a cabo mediante el uso de un operador especial que se aplica sobre una solución. Un operador de búsqueda local es un procedimiento que se encarga de buscar en el vecindario de una solución particular, soluciones localmente óptimas.

Por vecindad de una solución se entiende un conjunto de soluciones cercanas a ella. Formalmente, la vecindad de una solución $\vec{x}$ se describe como:

$$B(\vec{x}, r) := \{\vec{y} \in \mathbb{R}^n : x_i - r < y_i < x_i + r,\ \forall i \in \{1, \ldots, n\}\}$$

donde r es el radio de la vecindad y determina qué tan cercanas son las soluciones.

El objetivo de un operador de búsqueda local es refinar soluciones existentes y acelerar la convergencia hacia el óptimo global.

Representaremos a un operador de búsqueda local como Φ donde $\Phi(\vec{x})$ es la nueva solución generada a partir de $\vec{x}$. Un ejemplo del comportamiento esperado de un operador de búsqueda local puede observarse en la figura 3.

Algunos de los comportamientos deseados para un operador de búsqueda local Φ de acuerdo a [21] son los siguientes:

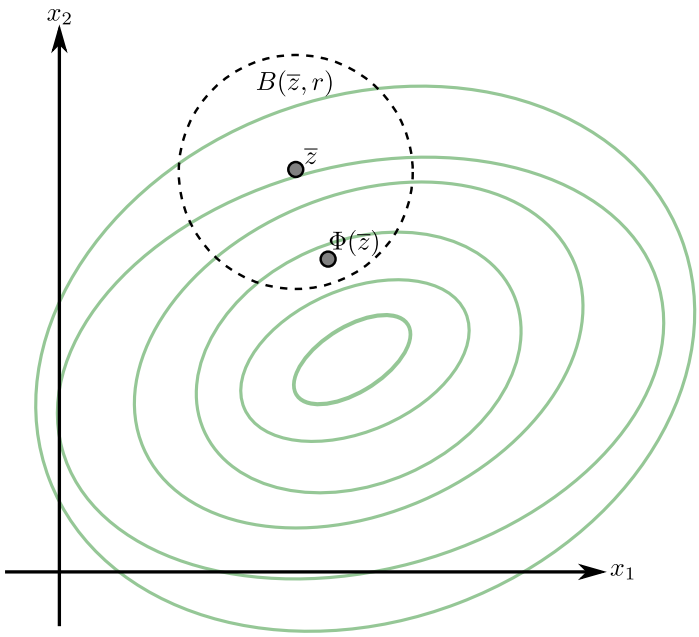

Figura 3.1: Ejemplo del comportamiento de un operador de búsqueda local en una función con dos variables de decisión y un objetivo

- Si una solución $\vec{x}$ no está cerca de $\mathcal{P}$ entonces es deseable que $\Phi(\vec{x}) \prec \vec{x}$.
- En caso de que la solución se encuentre cerca de $\mathcal{P}$, entonces es deseable realizar una búsqueda sobre $\mathcal{P}$.
- El cambio entre las dos situaciones descritas anteriormente debería ser automático y de acuerdo a la posición de la solución sobre la que se aplica Φ.

El funcionamiento deseable de un buscador local puede lograrse usando métodos numéricos tradicionales de optimización o mediante heurísticas. Estas últimas tienen la ventaja de no requerir de la información que proporciona la derivada de la función. El operador propuesto en este trabajo de tesis se basa en una heurística: la dirección y tamaño de paso son calculados a partir de información contenida en el espacio de búsqueda.

3.1. Trabajo previo

En esta sección revisaremos algunas de las propuestas encontradas en la literatura sobre la hibridación de algoritmos de búsqueda global con buscadores locales.

3.1.1. Multi-Objective Genetic Local Search

En [22] los autores proponen un algoritmo híbrido para encontrar las soluciones no dominadas de un problema de optimización multiobjetivo.

La propuesta consiste en un algoritmo genético el cual aplica un proceso de búsqueda local a cada solución después de los operadores del algoritmo genético.

Otra característica importante es el uso de una función de agregación lineal aleatoria para asignar la aptitud de los individuos.

Una función de agregación lineal es una suma ponderada de los k objetivos del problema de optimización:

$$a(\vec{x}) = w_1 f_1(\vec{x}) + \ldots + w_k f_k(\vec{x})$$

con w_i, $i \in \{1, \ldots, k\}$ no negativos y $\sum_{i=1}^{k} w_i = 1$.

Al vector $\vec{w} = [w_1, \ldots, w_k]^T$ se le llama *dirección* de la presión de selección pues al utilizar la función de agregación como medida de aptitud (con los ajustes necesarios dependiendo de si se trata de un problema de maximización o uno de minimización), la optimización de la función objetivo ocurrirá en la dirección del vector de pesos $\vec{w}$.

En el caso del algoritmo antes descrito, el vector de pesos $\vec{w}$ es elegido de forma aleatoria antes de seleccionar cada par de individuos para el operador de recombinación.

Posteriormente, se aplica un operador de mutación a las soluciones generadas y finalmente, el operador de búsqueda local.

El procedimiento de búsqueda local intenta maximizar el valor de la aptitud definida de acuerdo a la función de agregación. El vector $\vec{w}$ de la función de agregación es el mismo que el empleado en la selección de los padres que dieron origen a la solución sobre la cual se aplica el procedimiento de búsqueda local. La búsqueda local requiere de un parámetro j que controla el número de evaluaciones de la función objetivo utilizadas. El pseudocódigo de este procedimiento se muestra en el algoritmo 3.1.

Algoritmo 3.1 Procedimiento de búsqueda local utilizado en Multi-Objective Genetic Local Search

Entrada: Solución inicial $\vec{x}$
Entrada: Número máximo de soluciones revisadas j
  repetir
    Elegir de forma aleatoria una solución $\vec{y}$ en la vecindad de $\vec{x}$
    si $\vec{y}$ es mejor que $\vec{x}$ **entonces**
      $\vec{x} \leftarrow \vec{y}$
  hasta que se han revisado j soluciones vecinas

Otra característica importante del algoritmo es el uso de una estrategia elitista. Esto consiste en mantener dos conjuntos de soluciones: la población actual y un conjunto de soluciones no dominadas. En cada iteración del algoritmo las soluciones no dominadas de la población actual son incluidas en el conjunto de soluciones no dominadas, y las soluciones que ahora son dominadas por alguna de las recién agregadas son eliminadas del conjunto.

3.1.2. Pareto Memetic Algorithm

Propuesto por Jaszkiewicz [23] es también un algoritmo genético multiobjetivo que incorpora una búsqueda local.

En cada iteración del algoritmo se elige de forma aleatoria un vector de pesos $\vec{w}$ el cual determina una función agregativa. Posteriormente, dos soluciones muy buenas en relación a la función de agregación son elegidas para recombinarse y dar lugar a una nueva solución. El procedimiento de búsqueda local se aplica a la nueva solución.

Lo que distingue a este algoritmo es el uso de dos conjuntos de soluciones. El primero de ellos es el conjunto de soluciones actuales (denotado por CS). El segundo contiene las soluciones potencialmente óptimas de Pareto (representado por PP).

La selección se realiza de la siguiente forma:

1. En cada iteración se toma una muestra aleatoria con repetición de CS.

2. La primera y segunda mejor soluciones de acuerdo a la función agregativa son elegidas para recombinarse.

3. La función agregativa induce un orden en CS. La mejor solución en CS se encuentra en la posición 1. El tamaño de la muestra aleatoria se elige de forma que la posición esperada de la solución generada sea igual a un parámetro Er.

El parámetro Er controla la presión de selección. Valores bajos incrementan la presión de selección y pueden causar convergencia prematura.

El conjunto PP será la salida del algoritmo. Inicialmente, este conjunto está vacío, pero es actualizado cada vez que el procedimiento de búsqueda genera una nueva solución.

Una solución es incluida en PP si no es dominada por ningún otro punto de PP. Una solución es eliminada de PP cuando es dominada por una solución generada por el procedimiento de búsqueda local.

El procedimiento de búsqueda local también intenta mejorar una solución de acuerdo a la función agregativa utilizada.

A diferencia del algoritmo descrito en la subsección anterior, en este caso no se emplean funciones de agregación lineales, sino la función siguiente:

$$a(\vec{x}) = \max_{i=1,\ldots,k} \{w_i | f_i(\vec{x}) - z_i^*|\}$$

donde $\vec{z^*} = [z_1^*, \ldots, z_k^*]^T$ es un punto de referencia. Este tipo de función se conoce como función de agregación de Tchebycheff.

El punto de referencia $\vec{z^*}$ es calculado a partir del conjunto PP:

$$z_i^* = \text{mín}\{f_i(\vec{p}) | \vec{p} \in PP\} + 0.1\,(\text{máx}\{f_i(\vec{p}) | \vec{p} \in PP\} - \text{mín}\{f_i(\vec{p}) | \vec{p} \in PP\})$$

3.1.3. M-PAES

M-PAES [24] es un algoritmo de optimización multiobjetivo. También se caracteriza por incorporar un procedimiento de búsqueda local.

Algunas de las observaciones realizadas por los autores de M-PAES para justificar su propuesta son las siguientes:

- El elitismo mejora considerablemente el desempeño de los algoritmos genéticos multiobjetivo. El elitismo consiste en mantener por separado un conjunto de soluciones no dominadas que es actualizado en cada iteración del algoritmo.

- El uso de funciones de escalarización ayuda a encontrar soluciones particulares de acuerdo a los criterios de preferencia de un problema particular. Para el caso donde se desean conocer todas las soluciones, se han utilizado funciones de escalarización aleatorias las cuales no muestran un desempeño completamente satisfactorio.

- PAES es una propuesta anterior de los mismos autores. Este es un algoritmo que emplea únicamente búsqueda local y aún así es competitivo con respecto a otros algoritmos genéticos para optimización multiobjetivo. A pesar de esto, la búsqueda local es superada por los métodos poblacionales (como los algoritmos genéticos multiobjetivo) en problemas altamente multimodales o deceptivos.

Basado las observaciones anteriores, M-PAES no emplea funciones de agregación. En su lugar se emplea una selección basada en jerarquización de Pareto.

Otra característica de M-PAES es el uso de una partición del espacio de las funciones objetivo en forma de cuadrícula. Esta partición permite comparar dos soluciones y decidir cuál de ellas se encuentra en una región menos poblada.

M-PAES requiere de dos conjuntos (llamados archivos) de soluciones. El primero de ellos almacena las soluciones no dominadas encontradas hasta el momento. El segundo es utilizado por el procedimiento de búsqueda local. Cada vez que se aplica el procedimiento de búsqueda local sobre una solución $\vec{c}$, el segundo archivo es llenado con soluciones contenidas en el primero que no dominan a $\vec{c}$.

El procedimiento de búsqueda local de M-PAES se presenta en el algoritmo 3.2.

3.1.4. Búsqueda local con regla de reemplazo generalizada

Usualmente, los pasos a seguir en una iteración de un operador de búsqueda local son:

1. Se considera una solución inicial $\vec{x}$ a ser mejorada.

2. Generar de alguna forma una nueva solución $\vec{y}$ a partir de $\vec{x}$.

3. Comparar $\vec{y}$ con $\vec{x}$; si $\vec{y}$ es mejor, entonces se reemplaza a $\vec{x}$ con $\vec{y}$.

Algoritmo 3.2 Búsqueda local en M-PAES

Entrada: $\vec{c}$ solución que será mejorada
Entrada: G archivo con las soluciones no dominadas encontradas hasta ahora
Entrada: $H \subset G$ archivo con soluciones que no dominan a $\vec{c}$
  fallas $\leftarrow 0$
  movimientos $\leftarrow 0$
  mientras fallas $<$ limite_fallas y movimientos $<$ limite_movimientos **hacer**
    Mutar $\vec{c}$ para obtener una nueva solución $\vec{m}$
    si $\vec{c} \prec \vec{m}$ **entonces**
      Descartar $\vec{m}$
      fallas $\leftarrow$ fallas $+ 1$
    si no, **si** $\vec{m} \prec \vec{c}$ **entonces**
      $\vec{c} \leftarrow \vec{m}$
      fallas $\leftarrow 0$
    si no, **si** $\vec{m}$ es dominada por cualquier elemento de H **entonces**
      Descartar $\vec{m}$
    si no
      si H no está lleno **entonces**
        Incluir $\vec{m}$ en H
        si $\vec{m}$ está en una región menos poblada de H que $\vec{c}$ **entonces**
          $\vec{c} \leftarrow \vec{m}$
      si no
        si $\vec{m}$ está en una región menos poblada de H que algún elemento de H **entonces**
          Incluir $\vec{m}$ en H
          Quitar de H un elemento que se encuentre en la región más poblada
          si $\vec{m}$ está en una región menos poblada de H que $\vec{c}$ **entonces**
            $\vec{c} \leftarrow \vec{m}$
        si no
          si $\vec{m}$ está en una región menos poblada de H que $\vec{c}$ **entonces**
            $\vec{c} \leftarrow \vec{m}$
    movimientos $\leftarrow$ movimientos $+ 1$

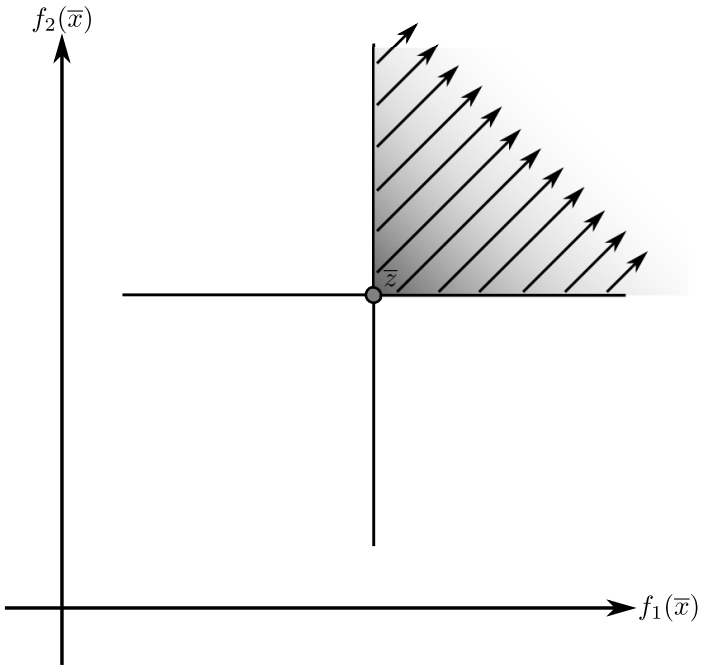

Figura 3.2: Supongamos que tenemos funciones objetivo $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})]^T$ y una solución particular $\vec{z}$. La región sombreada representa las soluciones dominadas por $\vec{z}$

Para decidir si la nueva solución reemplazará a la anterior se suelen utilizar dos criterios diferentes:

- Si la nueva solución domina a la existente, entonces la reemplaza (figura 3.1.4).
- Cuando la nueva solución no es dominada por la existente, entonces también la reemplaza (figura 3.1.4).

Se ha mencionado [25] que la regla de reemplazar una solución por otra no dominada no ejerce mucha presión de selección pues en problemas con muchos objetivos casi todas las parejas de soluciones serán no dominadas entre sí.

Por otro lado, reemplazar una solución sólo por otra que la domina no funciona muy bien pues es difícil encontrar soluciones que dominen a la solución actual.

En [26] se propone una alternativa a las reglas tradicionales de reemplazo basadas en dominancia de Pareto.

Esta regla se denomina *regla d* y establece que una solución $\vec{x}$ es reemplazada por otra solución $\vec{y}$ si $\vec{y}$ es mejor que $\vec{x}$ en d o más objetivos.

Variando el valor de d se obtiene un conjunto de reglas (Regla d, Regla (d-1) hasta Regla 1) que reemplazan las dos reglas mencionadas con anterioridad.

En [26] también se propone un procedimiento de búsqueda local que utiliza estas reglas. Su funcionamiento puede verse en el algoritmo 3.3.

En la literatura podemos encontrar otras propuestas de reglas de reemplazo. Algunas de ellas son:

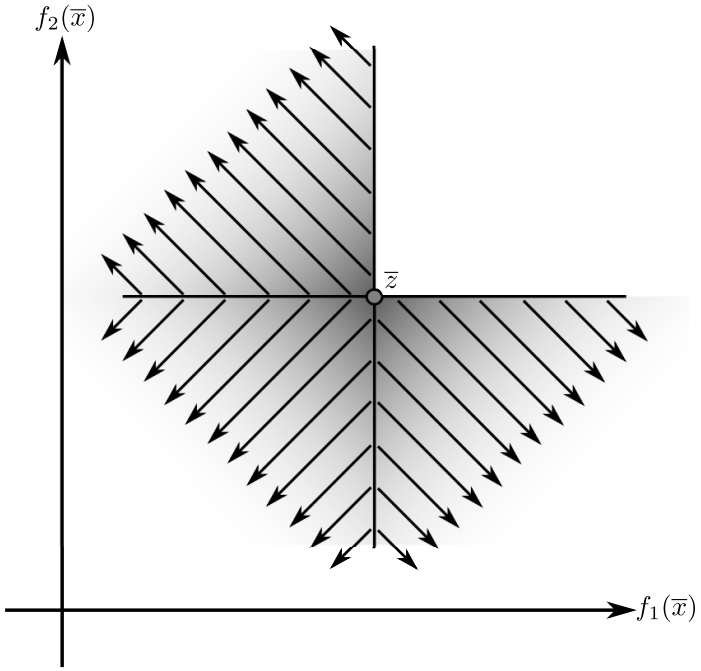

Figura 3.3: En esta figura la región sombreada representa las soluciones que no son dominadas por $\vec{z}$.

Algoritmo 3.3 Procedimiento de búsqueda local utilizando las reglas de reemplazo generalizadas.

Entrada: Solución que será mejorada $\vec{x}$
Entrada: Número de objetivos que deben ser mejorados d
Salida: La solución $\vec{x}$ mejorada
  Elegir de forma aleatoria dos soluciones de la población actual
  repetir
    Contar el número de objetivos en los que mejora cada solución
    Quedarse con la solución que mejora en un mayor número de objetivos
    Seleccionar de forma aleatoria otra solución de la población actual
  hasta que Se exceda el número máximo de intentos
  repetir
    Generar una solución $\vec{y}$ vecina a $\vec{x}$
    Contar el número de objetivos en los que $\vec{y}$ mejora a $\vec{x}$
    si El número de objetivos mejorados es mayor o igual a d **entonces**
      $\vec{x} \leftarrow \vec{y}$
  hasta que Se exceda el número máximo de intentos

- dominancia-ϵ [27] en donde se permite que una solución con una mejora pequeña en cada objetivo no reemplace a la solución actual.
- dominancia-α [28] en la cual un pequeño detrimento en un objetivo es permitido si en el resto de los objetivos se logra una mejora considerable.

3.2. Geometría del espacio de búsqueda

3.2.1. Conos direccionales

Consideremos el caso de optimizar una función de un solo objetivo. Aquí la definición de dominancia se reduce simplemente a que el valor del único objetivo sea menor al de la solución dominada.

Continuando con este ejemplo supongamos que tenemos una solución cualquiera. Esta solución divide el espacio de búsqueda en dos regiones. Una contiene a todos los puntos dominados por la solución. La otra contiene a todos los puntos que dominan a la solución.

En estas condiciones cualquier operador que modifique la solución sólo puede tener dos resultados: encontrar una mejor solución o una peor.

Para extender este concepto a funciones multiobjetivo primero debemos analizar cada objetivo por separado. Dados un punto cualquiera y un objetivo particular tenemos que existen dos subespacios del espacio de parámetros. Las soluciones que se encuentran en uno de los subespacios logran una mejora en el objetivo particular. Las que se encuentran en el otro subespacio son peores en el mismo objetivo.

Consideremos ahora las intersecciones de los subespacios formados para cada objetivo. Denotamos por $\{+,-\}$ a la intersección formada por el subespacio donde el primer objetivo incrementa su valor y el segundo la disminuye. Así, en el caso de una función de dos objetivos existen cuatro regiones formadas por las intersecciones de los subespacios: $\{+,+\}$, $\{-,-\}$, $\{+,-\}$, $\{-,+\}$. En general, para problemas con k objetivos existen 2^k de estas regiones (llamadas **conos direccionales**).

De acuerdo a sus características estas regiones son clasificadas en 3 tipos (figura 3.2.1):

- Los **conos de descenso** son las regiones donde todos los objetivos disminuyen; por ejemplo $\{-,-\}$ en una función de dos objetivos.
- Las regiones donde todos los objetivos aumentan su valor son llamadas **conos de ascenso**. Por ejemplo $\{+,+\}$ en una función de dos objetivos.
- El resto de las regiones son conocidas como **conos de diversidad**. En ellas ocurre que hay mejoras en algunos objetivos pero empeoramientos en otros.

De acuerdo a lo mencionado antes sobre las reglas de reemplazo en procedimientos de búsqueda local es natural asumir que un movimiento hacia cualquiera de los conos direccionales con excepción del cono de ascenso es aceptable.

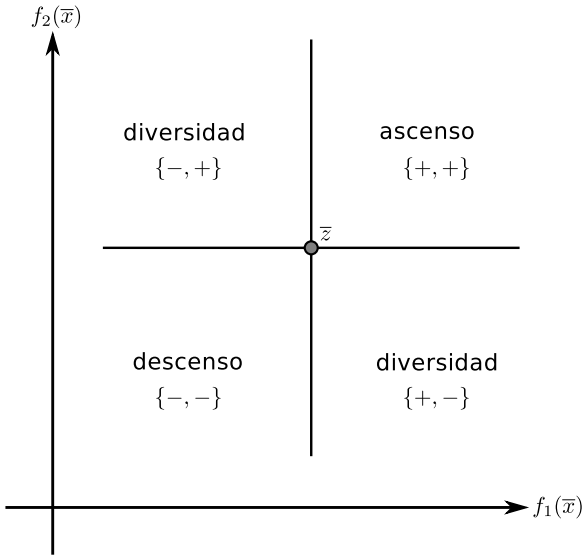

Figura 3.4: Ejemplo de los conos direccionales en una función de dos objetivos vistos en el espacio de las funciones objetivos. La figura se tomó de [29]

Por definición, las soluciones que se encuentran dentro de un cono de descenso dominarán a la solución original. También hay que observar que al acercarse hacia un punto óptimo de Pareto el tamaño de los conos de descenso y ascenso se reduce. En el caso que se esté sobre una solución óptima de Pareto entonces sólo existirán conos de diversidad.

3.2.2. Continuidad de Lipschitz

Un espacio métrico es una pareja formada por un conjunto M y una función $d : M \times M \to \mathbb{R}$ tal que para cualesquiera $x, y, z \in M$ se cumple lo siguiente:

- $d(x, y) \geq 0$
- $d(x, y) = 0$ si y sólo si $x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

En particular, un espacio vectorial $\mathbb{R}^n$ junto con una norma forman un espacio métrico.

Una función $F : X \to Y$, con (X, d_X) y (Y, d_Y) espacios métricos es Lipschitz continua si existe una constante $k \geq 0$ tal que para todo $p, q \in X$ se cumple que:

$$d_Y(F(p), F(q)) \leq k d_X(p, q)$$

Cualquier constante k que cumpla con la definición anterior es llamada constante de Lipschitz para la función F. Usualmente, al preguntar por la constante de Lipschitz de una función se responde con la más pequeña que cumpla la definición.

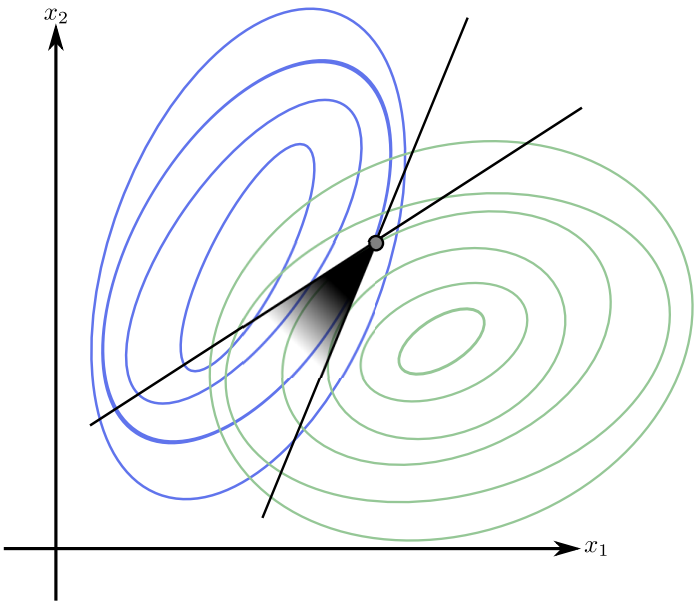

Figura 3.5: La región sombreada representa un cono de descenso visto en el espacio de las funciones objetivo. La gráfica corresponde a una función de dos objetivos y dos variables de decisión. La figura fue tomada de [29]

3.2.3. Cálculo del tamaño de paso

Al desarrollar un procedimiento de búsqueda local puede ocurrir que se conozca un método para encontrar una dirección prometedora hacia dónde dirigir la búsqueda (por ejemplo dentro de un cono de descenso).

Si se parte de una solución inicial $\vec{x}$ y se tiene una dirección $\vec{a}$ hacia donde uno se quiere desplazar, entonces la nueva solución está dada por

$$\vec{x}_{nuevo} = \vec{x} + t\vec{a} \tag{3.1}$$

donde $t > 0$ es el tamaño de paso (ver figura 3.2.3).

Usualmente, es deseable lograr una cierta distancia entre la solución original y la nueva solución. Esta distancia es medida en el espacio de las funciones objetivo:

$$||\vec{f}(\vec{x}) - \vec{f}(\vec{x}_{nuevo})||_\infty \approx \epsilon \tag{3.2}$$

La única forma de lograr obtener esta distancia es encontrando un valor adecuado para t.

Los autores de [30] proponen un método para calcular t. Este método se presenta a continuación.

Si F es Lipschitz continua entonces existe L tal que:

$$||\vec{f}(\vec{p}) - \vec{f}(\vec{q})||_\infty \leq ||p - q||_\infty \qquad \forall \vec{p}, \vec{q} \in \mathbb{R}^n \tag{3.3}$$

Combinando (3.1), (3.2) y (3.3) tenemos que.

$$\epsilon \leq Lt||a||_\infty$$

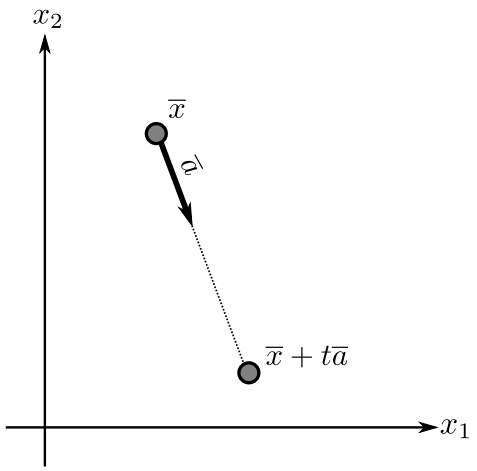

Figura 3.6: Una solución $\vec{x}$ será modificada usando la dirección $\vec{a}$ y un tamaño de paso t

de donde

$$t = \frac{\epsilon}{L||a||_\infty} \tag{3.4}$$

3.3. Observaciones

Los puntos importantes a considerar en el diseño de un procedimiento de búsqueda local son los siguientes:

- Encontrar un balance en el número de soluciones cercanas consideradas por el proceso de búsqueda local de forma que este no consuma una gran cantidad de evaluaciones de la función objetivo pero aún así logrando una mejora en la velocidad de convergencia.

- Elección correcta del tamaño de paso deseado. Este debe ser lo suficientemente grande para lograr una mejora en el desempeño pero evitando problemas como la convergencia prematura.

- Observar que el permitir movimientos de empeoramiento de la solución actual en algunos casos puede ayudar a evitar quedar estancados en óptimos locales de la función objetivo.

- El procedimiento de búsqueda local puede ser aplicado en distintos momentos de un algoritmo genético: después de cada iteración, cada cierto número de iteraciones, cuando el algoritmo genético haya terminado, etc. El momento en que se aplique tendrá mucha influencia en el desempeño final del algoritmo híbrido.

- Finalmente, existe el problema de decidir las soluciones sobre las cuales será aplicado el procedimiento de búsqueda local. En algunos casos es conveniente apli-

carlo en todos los individuos, pero en otros casos el desempeño mejora si se aplica sólo a las mejores soluciones.

Capítulo 4

Incorporación de Conocimiento en Algoritmos Evolutivos

La mayoría de las técnicas de optimización multiobjetivo y en particular los algoritmos evolutivos no utilizan conocimiento del dominio para guiar el proceso de búsqueda. Podemos decir que estos algoritmos no tienen memoria de las regiones del espacio de búsqueda que han sido vistadas previamente. Debido a esto, es posible que en muchos casos se realicen evaluaciones innecesarias de la función objetivo.

Los algoritmos evolutivos suelen consumir un número importante de evaluaciones de la función objetivo para lograr una aproximación buena del frente de Pareto. Esto puede causar problemas al intentar aplicar algoritmos evolutivos en la solución de problemas reales, pues generalmente, se dispone sólo de un número reducido de evaluaciones de la función objetivo.

En los últimos años se han realizado esfuerzos por reducir el costo computacional de los algoritmos evolutivos multiobjetivos. Algunos autores han logrado reducir la complejidad algorítmica de los algoritmos para identificar soluciones no dominadas [6, 10]. Otro enfoque, el cual es de nuestro interés en esta tesis, es la reducción del número de evaluaciones de la función de aptitud mediante el uso del conocimiento extraído durante la búsqueda.

4.1. Esquemas de incorporación de conocimiento

Un primer esquema para la reducción del número de evaluaciones de la función de aptitud consiste en utilizar una aproximación eficiente de la función de aptitud original. El uso de una aproximación resulta útil cuando la función de aptitud es computacionalmente costosa o el número de evaluaciones está limitado por otras razones.

La aproximación es utilizada para predecir soluciones prometedoras con un costo computacional menor que el del problema original. En [31] se mencionan varias estrategias adoptadas para aproximar la función de aptitud.

Otro esquema es el uso de conocimiento en diferentes etapas de la búsqueda con

el propósito de acelerar la convergencia o reducir el número de evaluaciones de la función de aptitud.

Un caso particular de este esquema es el uso de conocimiento del dominio obtenido previamente (posiblemente con la asistencia de un experto en el problema) para construir un conjunto de casos de referencia que hacen la búsqueda más eficiente.

Otra opción es utilizar el conocimiento para insertar algunas soluciones buenas en la población inicial de un algoritmo evolutivo o para guiar los operadores de variación hacia la generación de buenas soluciones.

Finalmente, dentro de los esquemas de incorporación de conocimiento en algoritmos evolutivos encontramos a los algoritmos culturales. Su principal ventaja es el no requerir de la incorporación del conocimiento *a priori*. En este tipo de algoritmos el conocimiento es extraído durante el proceso de búsqueda.

Uno de los primeros intentos por incorporar conocimiento del dominio en computación evolutiva se encuentra en el algoritmo llamado **EnGENEous** propuesto por Powell et al. [32]. En esta propuesta se incorpora un sistema experto a un algoritmo genético.

Una segunda propuesta de los mismos autores es el método de **interdigitación** [33]. En este método también se integra un sistema experto en un algoritmo genético.

Otra idea sugerida por Louis y Rawlins [34] fue usada para resolver problemas de diseño. Inicialmente, agregaron conocimiento del dominio de un problema a los operadores de recombinación [34] [35]. Posteriormente, incorporaron conocimiento a la inicialización de la población, a la codificación de las soluciones y al resto de los operadores evolutivos [36].

A diferencia de las propuestas anteriores, los algoritmos culturales no requieren de conocimiento previo sino que este es extraído durante el proceso de búsqueda.

En las siguientes secciones se presentan diversas propuestas de algoritmos culturales reportadas en la literatura que se consideran más representativas del trabajo realizado en el área.

4.2. Extensión de GENOCOP a un algoritmo cultural

En [37] Reynolds explora algunas extensiones a un paquete de software para la solución de problemas de optimización llamado GENOCOP [38]. Este software emplea algoritmos genéticos para trabajar con problemas de optimización continua con un alto número de restricciones.

En problemas con un alto número de restricciones, la región factible suele quedar dividida en múltiples regiones disjuntas. Por ello, GENOCOP trabaja ajustando el dominio de los operadores genéticos de acuerdo al conjunto de restricciones. La extensión propuesta por Reynolds es un algoritmo cultural el cual se encarga de recolectar información sobre la ubicación de las regiones factibles y ajusta, en consecuencia los operadores en GENOCOP.

En muchas técnicas para el manejo de restricciones como las funciones de penalización y los algoritmos de reparación, las restricciones del problema no tienen un papel activo en la generación de los individuos. Sólo se utilizan para decidir si un individuo es factible o no después de haber sido generado.

En la misma propuesta se sugiere que hacer uso de las restricciones para generar nuevos individuos puede acelerar la convergencia y precisión del algoritmo. Por ello el algoritmo cultural descrito extiende a GENOCOP de forma que utilice las restricciones para guiar el proceso de búsqueda.

El espacio de población del algoritmo descrito está representado por GENOCOP. El espacio de creencias está compuesto por intervalos de valores reales. Estos intervalos son extendidos o reducidos de forma que incorporen las experiencias de individuos prometedores.

4.3. Algoritmos culturales con programación evolutiva

En [39], Chung y Reynolds utilizan programación evolutiva con un operador de mutación influenciado por el mejor individuo encontrado hasta el momento y también usando los intervalos donde las mejores soluciones han sido encontradas. Su propuesta fue llamada CAEP ("Cultural Algorithms with Evolutionary Programming"). La estructura del procedimiento se muestra en el algoritmo 4.1.

Algoritmo 4.1 Estructura de un algoritmo cultural con programación evolutiva [40]

Seleccionar una población inicial de p soluciones usando una distribución uniforme en el dominio de cada parámetro
Evaluar el desempeño de cada solución de acuerdo a la función objetivo
Inicializar el espacio de creencias utilizando la población inicial
repetir
  Generar p nuevas soluciones aplicando un operador de variación influenciado por el espacio de creencias
  Evaluar el desempeño de las nuevas soluciones
  Para cada uno de los $2p$ individuos elegir de forma aleatoria c competidores
  Realizar competencias por pares entre cada individuo y sus c competidores
  Quedarse únicamente con las p soluciones con mayor número de victorias
hasta que se exceda el tiempo máximo de ejecución o se encuentre una solución aceptable

En este caso, se consideraron dos categorías de conocimiento para el espacio de creencias [40]:

- **Conocimiento normativo**. Este tipo de conocimiento establece estándares de comportamiento de los individuos y provee una guía sobre los ajustes posibles a cada individuo.

- **Conocimiento situacional**. Está conformado por un conjunto de experiencias de individuos modelo y es útil para interpretar las experiencias de cada individuo por separado.

Tanto el conocimiento normativo como el situacional pueden incluir experiencias positivas o negativas. En la propuesta de Chung sólo se considera conocimiento positivo.

4.3.1. Conocimiento situacional

El conocimiento situacional es representado por dos elementos:

- Una función de ajuste la cual es el operador del espacio de creencias aplicado para actualizar los e individuos ejemplares.
- Un conjunto de individuos modelo. Este es un conjunto ordenado con los e mejores individuos a lo largo del proceso evolutivo.

La función de ajuste trabaja incluyendo un individuo en el conjunto sólo si éste es el mejor en la generación actual y supera al primer individuo en el conjunto. Su funcionamiento detallado puede verse en el algoritmo 4.2.

Algoritmo 4.2 Función de ajuste para el conocimiento situacional

Entrada: Número de individuos en el conjunto e
Entrada: Conjunto ordenado de individuos $\vec{e}_1, \ldots, \vec{e}_e$
Entrada: El mejor individuo en la generación actual $\vec{x}_{\text{best}}$
Salida: El nuevo conjunto de individuos $\vec{e}_1, \ldots, \vec{e}_e$

```
si x_best ≺ e_1 entonces
  para i desde e disminuyendo hasta 2 hacer
    e_i ← e_{i-1}
  e_1 ← x_best
```

4.3.2. Conocimiento normativo

El conocimiento normativo está compuesto por cuatro elementos:

- Un conjunto de intervalos $I_1, \ldots, I_n$ cada uno correspondiente a la variable j. Un intervalo es un conjunto de números reales y se define como:

$$I_j = [l_j, u_j] = \{x | l_j \leq x \leq u_j, x \in \mathbb{R}\}$$

  donde l_j y u_j son los límites inferior y superior, respectivamente.

- Un conjunto de valores $L_1, \ldots, L_n$ que representan la aptitud del límite inferior l_j para el parámetro j.

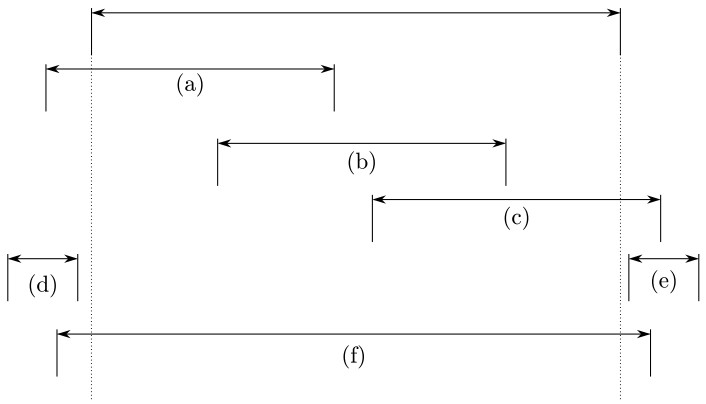

Figura 4.1: Los seis casos posibles para el nuevo intervalo al ajustar el conocimiento normativo.

- Un conjunto de valores $U_1, \ldots, U_n$ con la medida de aptitud del límite superior u_j para el parámetro j.

- Una función de ajuste encargada de actualizar el conocimiento normativo.

Antes de comenzar con la primera iteración del algoritmo, los intervalos I_j son definidos como el dominio de la variable correspondiente. Las medidas de aptitud L_j y U_j son inicializadas en $-\infty$ (suponiendo que se desea maximizar la aptitud).

La función de ajuste del conocimiento normativo recibe el conjunto de individuos aceptados y calcula los nuevos límites inferior y superior para cada una de las variables. En la figura 4.3.2 se presenta un ejemplo de lo que puede ocurrir. La primera línea representa el intervalo antes de aplicar la función de ajuste y las seis líneas inferiores representan los posibles casos que pueden presentarse al calcular el nuevo intervalo.

El caso (b) representa un intervalo que se encuentra contenido en el anterior. En los casos (a), (c) y (f) el nuevo intervalo se intersecta con el anterior. Finalmente, en los casos (d) y (e) el nuevo intervalo cae fuera del intervalo actual.

La idea principal al actualizar el conocimiento normativo se describe a continuación. Si el nuevo límite (inferior o superior) se encuentra fuera del intervalo actual entonces éste es extendido. Si el nuevo límite cae dentro del intervalo entonces éste se reduce únicamente si la aptitud de la solución límite es mayor que la del límite anterior.

4.3.3. Función de aceptación

Se consideran tres tipos de funciones de aceptación.

- **Estática**. Considera únicamente a un porcentaje de los individuos en la cima de la jerarquía definida por el valor de aptitud de los individuos. Su funcionamiento se describe en el algoritmo 4.3.

- **Relativa**. Los individuos aceptados son aquellos cuya aptitud sea mayor que el promedio. Se presenta en el algoritmo 4.4.
- **Estática con información temporal**. Es similar a la función de aceptación estática pero el porcentaje varía con el número de generaciones realizadas. Se describe en el algoritmo 4.5.

Algoritmo 4.3 Función de aceptación estática.

Entrada: P la población actual
Entrada: l porcentaje de aceptación (entre 1 y 50)
Salida: A el conjunto de individuos aceptados

```
c ← |P|
e ← ⌊(l/100)c⌋
Ordena P, P[1] será el individuo con mayor aptitud
A ← ∅
para k desde 1 hasta e hacer
    Incluye P[k] en A
```

Algoritmo 4.4 Función de aceptación relativa.

Entrada: P la población actual
Salida: A el conjunto de individuos aceptados

```
c ← |P|
a ← 0
para p⃗ en P hacer
    a ← a + aptitud(p⃗)
a ← a/c
A ← ∅
para p⃗ en P hacer
    si aptitud(p⃗) > a entonces
        Incluye p⃗ en A
```

4.3.4. Uso del conocimiento normativo

El proceso mediante el cual el conocimiento normativo aplica su influencia sobre las soluciones se realiza regulando el tamaño de paso en la mutación.

Dada una solución $\vec{x} = [x_1, x_2, \ldots, x_n]^T$, el operador de mutación modifica el valor de cada uno de sus componentes de la siguiente forma:

$$x_j = x_j + |u_j - l_j| N(0,1)$$

donde u_j, l_j son los límites inferior y superior correspondientes al intervalo I_j en el conocimiento normativo. $N(0,1)$ es una variable aleatoria con distribución normal y desviación estándar igual a 1.

Algoritmo 4.5 Función de aceptación estática con información temporal.

Entrada: P la población actual
Entrada: l porcentaje de aceptación (entre 1 y 50)
Entrada: t número de generaciones
Salida: A el conjunto de individuos aceptados

```
c ← |P|
β ← l/100
e ← ⌊βc + ⌊βc/t⌋⌋
Ordena P, P[1] será el individuo con mayor aptitud
A ← ∅
para k desde 1 hasta e hacer
  Incluye P[k] en A
```

4.3.5. Uso del conocimiento situacional

El conocimiento situacional es utilizado para elegir una dirección prometedora hacia dónde realizar la mutación del individuo a influenciar.

La forma de aplicar la información del conocimiento situacional a un componente x_j de una solución $\vec{x}$ particular es mediante una mutación con dirección, tal y como se describe en la siguiente fórmula:

$$x_j = \begin{cases} x_j + |\sigma_j N(0,1)| & x_j \prec e_{\text{best},j} \\ x_j - |\sigma_j N(0,1)| & e_{\text{best},j} \prec x_j \\ x_j + \sigma_j N(0,1) & \text{en otro caso} \end{cases}$$

donde $\vec{e}_{\text{best}}$ representa el mejor individuo dentro del conjunto de individuos que conforman el conocimiento situacional y σ_j es el tamaño de paso para la variable j.

4.4. Celdas de creencia

En un trabajo posterior, Jin y Reynolds [41] proponen un esquema basado en regiones n-dimensionales llamadas celdas de creencia. Éstas son un mecanismo que soporta la adquisición, almacenamiento e integración del conocimiento sobre las restricciones no lineales en un algoritmo cultural.

La idea en la propuesta de Jin y Reynolds consiste en construir un mapa del espacio de búsqueda el cual es utilizado para inferir reglas acerca de cómo guiar el proceso de búsqueda de un algoritmo evolutivo. El resultado esperado es que la búsqueda evite regiones no factibles y explore las regiones factibles.

En un problema con restricciones, éstas dividen el espacio de las variables en regiones de dos tipos distintos. Algunas regiones son factibles y otras no (ver ejemplo en la figura 4.4).

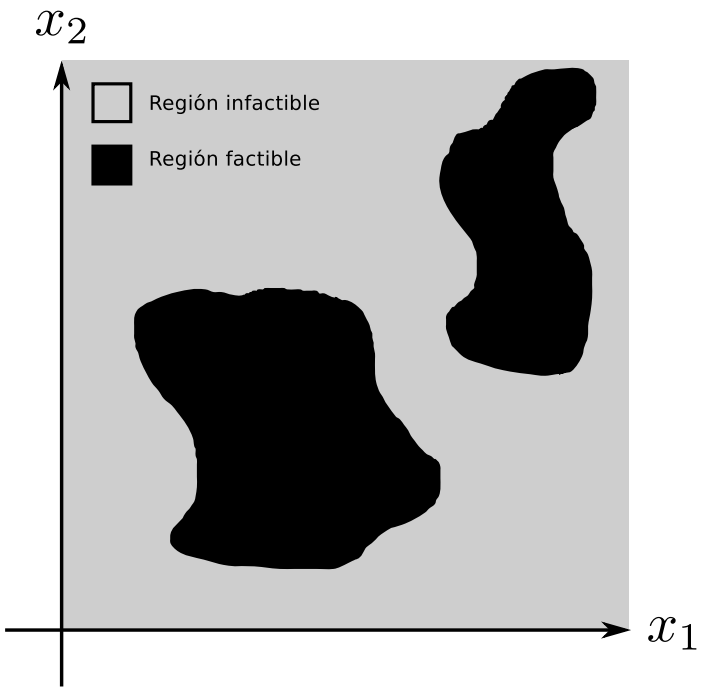

Figura 4.2: Ejemplo de zonas factibles y no factibles en una función de dos variables

4.4.1. Conocimiento de restricciones

El conocimiento de restricciones requiere almacenar información sobre la ubicación de las regiones factibles e infactibles. Sin embargo como en el ejemplo de la figura 4.4, las regiones n-dimensionales que dividen al espacio pueden no ser triviales de representar. Es por eso que una discretización con menos detalle es necesaria.

Debido a esto, la propuesta de Jin y Reynolds consiste en subdividir el espacio en subregiones llamadas **celdas de creencia**. Estas celdas pueden ser de tres tipos:

- **Celdas factibles**. Éstas cubren una región del espacio de variables completamente factibles.
- **Celdas infactibles**. Cubren una región conformada únicamente por valores infactibles.
- **Celdas semifactibles**. Se encuentran sobre regiones que contienen soluciones tanto factibles como infactibles.

Las celdas de creencias pueden proporcionar información importante sobre las regiones prometedoras del espacio de búsqueda. En un problema de optimización con restricciones las soluciones deben ser factibles por lo que las celdas factibles y semifactibles son las mejores candidatas para realizar la búsqueda.

Además, en muchos problemas con restricciones, las mejores soluciones se encuentran en la frontera entre las regiones factibles e infactibles. El espacio de creencias representado por celdas de creencias puede ayudar a encontrar estas fronteras.

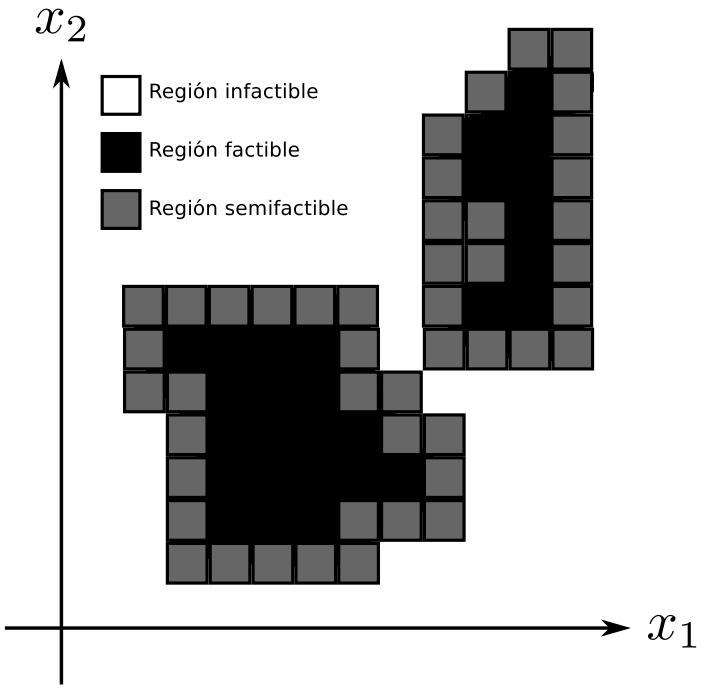

Figura 4.3: Celdas de creencia en para el ejemplo de la figura 4.4.

El contenido de cada celda de creencias consiste de dos números enteros. El primero de ellos cuenta la cantidad de soluciones factibles encontradas dentro de la celda. El segundo es el número de soluciones infactibles.

En la implementación de Jin [41] se utiliza un número fijo de celdas de creencia. Cada celda almacena los dos números mencionados antes, la posición de una de las esquinas de la celda y su tamaño para cada uno de los parámetros.

La inicialización y actualización de las celdas de creencia es similar a la de los intervalos en el conocimiento normativo.

4.4.2. Uso del conocimiento de restricciones

La función de influencia utilizada por Jin tiene como objetivo mover individuos hacia las regiones prometedoras del espacio de búsqueda. Su implementación es similar a la mutación usada como función de influencia en el conocimiento normativo.

4.5. Algoritmo cultural para entornos dinámicos

Usando un algoritmo de programación evolutiva como espacio de población, Saleem propuso un algoritmo cultural enfocado a entornos dinámicos [42]. En su propuesta añade tres formas más de incorporar conocimiento de dominio a CAEP además de extender la propuesta de Chung y Jin.

Los nuevos tipos de conocimiento propuestos son el topográfico, de dominio y el conocimiento histórico.

4.6. Uso de estructuras de datos eficientes

Usando como base el trabajo de Jin y Reynolds se propuso un algoritmo para optimización con restricciones en [43] y [44]. Esta propuesta consiste en el uso de una estructura de datos espacial para almacenar un mapa de las regiones factibles.

El conocimiento utilizado en este caso, se relaciona con las restricciones del problema y el espacio de búsqueda es dividido en subregiones factibles, infactibles o semifactibles.

4.7. Algoritmo cultural basado en un optimizador por cúmulo de partículas

Iacoban [45] propone un cambio en el algoritmo de programación evolutiva usado como espacio de población en un optimizador por cúmulo de partículas [46]. También analiza el efecto del espacio de creencias sobre el proceso evolutivo e indentifica las etapas del proceso de búsqueda con el espacio de creencias.

4.8. Algoritmo cultural basado en evolución diferencial

En [3] se propone un algoritmo cultural basado en programación evolutiva para la solución de problemas multi-objetivo. El espacio de creencias está basado en una cuadrícula que contiene información usada para guiar la búsqueda hacia soluciones no dominadas distribuidas uniformemente sobre el frente de Pareto. En un trabajo posterior de los mismos autores [47] se propone un algoritmo cultural el cual emplea como espacio de población a la evolución diferencial [48].

4.9. Uso de otras fuentes de conocimiento

Best et al. [5] proponen una extensión de CAEP. El algoritmo propuesto incluye además del conocimiento situacional y normativo, otras fuentes de conocimiento adicionales para guiar la búsqueda. Estas fuentes son el conocimiento de dominio, histórico y topográfico.

4.9.1. Conocimiento de dominio

Su objetivo es realizar una búsqueda incremental de una región del espacio de búsqueda. El trabajo que este conocimiento realiza es llevar las soluciones hacia óptimos locales. Puede pensarse como una búsqueda con gradiente que parte desde un individuo particular.

Para encontrar la mejor dirección primero se generan 3^n (donde n es el número de variables de decisión) individuos que rodean a la solución padre. Estos individuos se construyen incrementando, decrementando o manteniendo el valor de cada variable de decisión.

Cada una de las soluciones construidas es comparada con el padre y si alguna lo domina entonces ésta lo reemplaza.

4.9.2. Conocimiento histórico

Este conocimiento sigue el progreso de los mejores individuos a lo largo de las generaciones. La información guardada es independiente para cada uno de los k objetivos.

Para cada objetivo, el conocimiento histórico almacena una lista de los individuos con el mejor valor del objetivo particular en las últimas l generaciones, donde l es un parámetro del algoritmo.

Para influenciar un individuo primero se elige un objetivo. Después se selecciona de forma aleatoria una de las l soluciones almacenadas para el objetivo particular. Finalmente, el individuo es reemplazado por una nueva solución generada dentro de una vecindad del individuo seleccionado del conocimiento histórico.

El objetivo principal de este conocimiento es distribuir los individuos sobre el frente de Pareto.

4.9.3. Conocimiento topográfico

El conocimiento topográfico, al igual que el histórico, almacena información de forma independiente para cada uno de los objetivos.

Para cada objetivo se encuentra la aptitud promedio y se utiliza como un punto de comparación de los individuos. Cada una de las soluciones se encontrarán ya sea por arriba o por debajo del punto de comparación.

El conocimiento topográfico mantiene un registro de los intervalos donde las soluciones se encuentran por arriba o por debajo de la aptitud media en cada uno de los objetivos. Posteriormente, se elige una región que contenga buenos individuos y se genera una nueva solución en esa región.

Capítulo 5

Propuesta de algoritmo cultural para optimización multiobjetivo

La técnica propuesta en este trabajo de tesis es un algoritmo cultural diseñado para resolver problemas de optimización continua multiobjetivo sin restricciones.

En esta propuesta, el espacio de creencias es utilizado para influenciar la selección y también como fuente de información para un operador de búsqueda local.

La técnica propuesta se construye sobre el NSGA-II, modificando la selección e incorporando un procedimiento de búsqueda local.

El espacio de creencias está compuesto por un conjunto de individuos representativos organizados dentro de una estructura de datos eficiente que permite realizar consultas espaciales sobre ellos.

Este tipo de conocimiento es parecido al conocimiento situacional propuesto por Chung y Reynolds [39] pero diferente en cuanto al uso que se le da a la información. También difiere en que permite la aceptación de individuos con experiencias negativas (dominados) pues la información que aportan es de utilidad para el operador de búsqueda local.

Algunas de las propuestas existentes utilizan particiones del espacio de búsqueda como celdas donde el conocimiento es almacenado. Este tipo de enfoque suele ser costoso en memoria o puede requerir del uso de estructuras de datos complejas.

En contraste, la propuesta aquí presentada almacena el conocimiento en forma de puntos k-dimensionales. El tamaño del conjunto de puntos se mantiene acotado a lo largo de la ejecución del algoritmo, por lo que el uso de memoria y tiempo de cómputo se mantienen dentro de rangos aceptables.

5.1. Estructura del espacio de creencias

Como se ha mencionando anteriormente, el espacio de creencias se compone de un conjunto de puntos k-dimensionales. Un punto en el espacio de creencias es un vector con los valores de cada objetivo para una solución particular del problema. Junto con cada punto se almacena también el vector de parámetros que dieron lugar

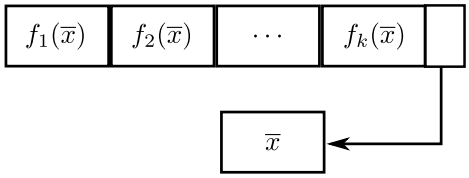

Figura 5.1: Cada punto en el espacio de creencias es un vector de valores de los objetivos. Además, cada punto incluye una referencia al vector de variables de decisión que lo generó.

a la solución particular (ver figura 5.1).

El conocimiento disponible en el espacio de creencias sobre el espacio de búsqueda del problema responde a la pregunta sobre la distancia de cualquier individuo hacia el punto más cercano en nuestro conjunto de individuos representativos hallados hasta el momento. También puede responder a la pregunta sobre cuáles son los j vecinos más cercanos en el conjunto representativo hacia un individuo cualquiera.

Este conocimiento es empleado para guiar la búsqueda hacia la construcción de un conjunto solución distribuida uniformemente sobre el frente de Pareto así como también para mejorar la velocidad de convergencia.

5.1.1. Árboles KD

Los árboles KD [49] son una estructura de datos que permite almacenar puntos k-dimensionales y realizar consultas sobre ellos de manera eficiente.

Se podría usar cuadrículas, pero una de las desventajas de su uso es que estas no escalan bien conforme la dimensión del espacio de búsqueda se incrementa [43]. En contraste, los árboles-KD mantienen un buen desempeño aunque se incremente la dimensión de los puntos. El costo promedio de una consulta de vecinos más cercanos es $O(\log n)$ (con n el total de puntos almacenados) siempre que n sea mucho mayor que la dimensión de los puntos [49].

Otra ventaja es que el costo de insertar o eliminar puntos de la estructura es bajo, evitando así tener que reconstruir la estructura cada vez que se requiera modificar.

En un árbol KD cada nodo representa un punto. Los nodos con uno o dos hijos determinan una partición del espacio en dos subespacios. La división entre dos subespacios está delimitada por un hiperplano que pasa por el punto en el nodo. Una de las ramas del nodo contiene únicamente puntos que se encuentran en uno de los subespacios y la otra rama contiene puntos en el subespacio complementario.

La dirección de los hiperplanos se determina de la siguiente forma: el hiperplano que pasa por el nodo raíz es perpendicular al eje que determina la primera dimensión y los hiperplanos en el segundo nivel del árbol son perpendiculares al eje de la segunda dimensión, continuando de esta forma, hasta el nivel k. Para el nivel $k+1$, los hiperplanos serán perpendicuales de nuevo al eje de la primera dimensión.

Un ejemplo de un árbol KD puede observarse en la figura 5.1.1. La partición del espacio para el mismo ejemplo se presenta en la figura 5.1.1.

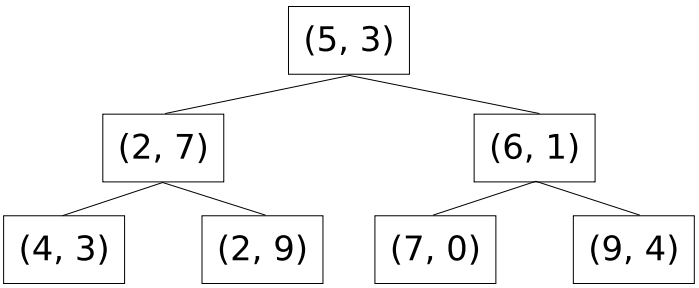

Figura 5.2: Ejemplo de un árbol KD para puntos bidimensionales.

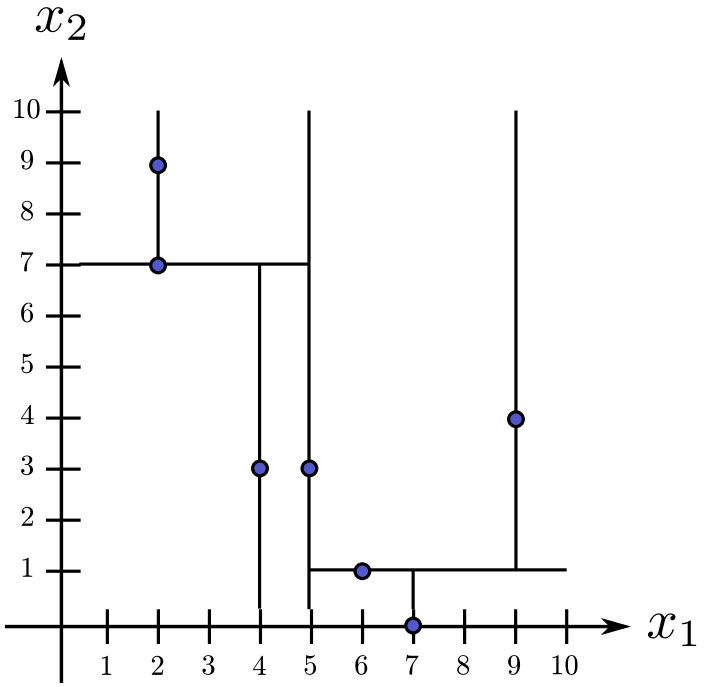

Figura 5.3: Partición del espacio determinada por el árbol del ejemplo 5.1.1.

5.1.2. Iniciación del espacio de creencias

Después de generar la primera población de individuos se procede a construir el árbol que representa al espacio de creencias utilizando los individuos de esta primera generación.

Puesto que el número máximo de elementos contenidos en el árbol es acotado y este límite es siempre mayor o igual que el tamaño de la población (tal y como se describe en las secciones siguientes), entonces el árbol se construye con todos los individuos de la primera generación.

Para mantener un buen desempeño del algoritmo es deseable construir y mantener un árbol balanceado [50].

5.1.3. Actualización del espacio de creencias

Partiendo del conjunto de individuos aceptados, el proceso de actualización del espacio de creencias puede ser resumido en tres pasos:

1. Insertar los elementos aceptados en el árbol. Es posible que después de realizar esta operación el número de individuos en el árbol exceda el límite máximo.

2. Se calcula el número de individuos excedentes en el árbol. En las primeras generaciones muy probablemente este número sea cero.

3. Se proceden a eliminar individuos del árbol hasta que el total sea igual al máximo permitido.

El objetivo del conocimiento almacenado en el espacio de creencias es servir como una muestra representativa del espacio de búsqueda. Por ello es deseable que los individuos almacenados en el árbol se encuentren distribuidos lo más uniformemente posible. Una forma de lograrlo es conservando únicamente los puntos más alejados entre sí. De tal forma que el criterio de eliminación al actualizar el espacio de creencias es precisamente eliminar primero los puntos más cercanos entre sí.

Eliminar los puntos más cercanos entre sí requiere de un poco más de explicación. Consideremos un ejemplo con 4 puntos (figura 5.1.3). En este caso, es sencillo observar cuáles son los dos puntos más cercanos. Es claro que no resulta adecuado eliminar los dos puntos pues así sólo se logra tener una mala distribución de las muestras. Lo mejor es conservar uno de ellos y eliminar el otro.

Basándose en la observación anterior se propone una heurística para eliminar puntos en el árbol. Este procedimiento recibe como entrada el árbol y el número de puntos a ser eliminados. Su funcionamiento se describe en el algoritmo 5.1.

Una observación sobre la implementación es acerca de la eliminación de nodos en los árboles KD. A diferencia de otros árboles de búsqueda binarios, los árboles KD no pueden ser rebalanceados fácilmente sin destruir su estructura. Teniendo en cuenta su buen desempeño aún cuando contienen un gran número de puntos, en [51] se sugiere una alternativa para la eliminación de puntos.

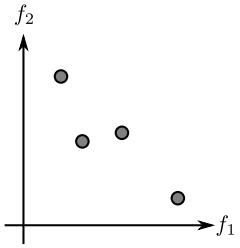

Figura 5.4: Ejemplo de puntos en el espacio de creencias. Los dos puntos más cercanos son candidatos a ser eliminados de acuerdo al algoritmo de actualización.

Algoritmo 5.1 Procedimiento de eliminación de puntos del árbol. La subrutina flip() devuelve 0 o 1 con probabilidad $\frac{1}{2}$ en cada caso.

Entrada: $A = \{\vec{p}_1, \ldots, \vec{p}_l\}$ los l puntos en el árbol
Entrada: i el número de puntos que deben eliminarse del árbol
  para cada $\vec{p}_j$ en A **hacer**
    $\vec{q} \leftarrow$ vecino más cercano de $\vec{p}_j$ en $A \setminus \{\vec{p}_j\}$
    $d_j \leftarrow |\vec{q} - \vec{p}_j|$
  $[\vec{q}_1, \ldots, \vec{q}_l] \leftarrow$ ordena A de menor a mayor de acuerdo a d_j
  $j \leftarrow 1$
  para $k = 1$ hasta i **hacer**
    si $0 =$ flip() **entonces**
      Elimina $\vec{q}_j$ del árbol
    si no
      Elimina $\vec{q}_{j+1}$ del árbol
    $j \leftarrow j + 2$

Dicha alternativa consiste en reemplazar la operación de eliminación por otra más simple. Cuando un punto debe ser eliminado, simplemente se le añade una marca que indique su cualidad de haber sido removido del árbol pero sin realmente quitarlo de este. La mayoría de las operaciones de búsqueda en un árbol KD requieren calcular distancias entre dos puntos. Cuando una de estas operaciones involucra a un punto eliminado, esta distancia se considera como infinita. Así, un nodo eliminado nunca será devuelto como resultado de una operación de búsqueda.

Es claro que este cambio tiene como consecuencia incrementar el uso de memoria. Por ello, en la implementación del algoritmo, el árbol es reconstruido completamente cada cierto número de generaciones. Esto provee además la ventaja de rebalancear periódicamente el árbol, sin incurrir en un costo de cómputo demasiado alto.

El número de generaciones entre reconstrucciones del árbol no afecta la calidad de las soluciones encontradas por el algoritmo, únicamente controla el balance entre el uso de memoria y el tiempo de cómputo.

5.2. Canal de comunicación

Teniendo la estructura del espacio de creencias es necesario decidir cuáles son los individuos utilizados para actualizarlo. También se requiere establecer los mecanismos de influencia en la búsqueda con base en la información del espacio de creencias.

En las siguientes subsecciones se exponen estos puntos.

5.2.1. Función de aceptación

La más común en las funciones de aceptación es elegir únicamente individuos que aporten experiencias positivas. En el caso de problemas de optimización multiobjetivo los individuos más aptos son aquellos con una posición alta en la jerarquía de Pareto (es decir, los más cercanos al frente de Pareto).

Sin embargo, también es posible considerar los individuos con experiencias negativas (por ejemplo, los que ya fueron dominados por una nueva solución) pues pueden servir de guía en la generación de mejores soluciones. Por ejemplo, supongamos que tenemos una solución $\vec{x}_0$ que desea ser mejorada y supongamos también que tenemos una solución anterior $\vec{x}_{\text{prev}}$ que ya fue dominada por la nueva solución ($\vec{x}_0 \prec \vec{x}_{\text{prev}}$). Entonces, la dirección $\vec{d} = \vec{x}_0 - \vec{x}_{\text{prev}}$ es una dirección de descenso. Por dirección de descenso se entiende que existe un tamaño de paso t tal que:

$$f_i(\vec{x}_0 + t\vec{d}) < f_i(\vec{x}_0) \qquad i = 1, \ldots, k$$

Una búsqueda sobre la dirección $\vec{d}$ es candidata a hallar una mejor solución.

Tomando en cuenta lo antes mencionado, la función de aceptación utilizada en esta propuesta elige una muestra aleatoria de la población. Su funcionamiento se muestra en el algoritmo 5.2.

Algoritmo 5.2 Función de aceptación utilizada en esta propuesta. El procedimiento rand() devuelve un número aleatorio entre 0 y 1 con distribución uniforme.

Entrada: P conjunto de individuos que forman la población actual
Entrada: $0 < a < 1$ porcentaje de aceptación
Salida: A conjunto de individuos aceptados
  $A \leftarrow \emptyset$
  para cada $\vec{x}$ en P **hacer**
    si rand() $\leq a$ **entonces**
      $A \leftarrow A \cup \{a\}$

5.2.2. Influencia en la selección

En un problema de optimización siempre es necesario contar con un método para comparar soluciones y decidir cuál es mejor. En NSGA-II se cuenta con un operador de comparación que toma en cuenta no sólo la dominancia de Pareto sino también la distribución de las soluciones. Dos soluciones que no son comparables utilizando la dominancia de Pareto (ninguna domina a la otra) pueden evaluarse utilizando la densidad de individuos en la región donde se encuentran. Una solución en una región menos poblada es preferible sobre otra que se encuentra en una región con mayor densidad de población.

Continuando con esta idea es posible proponer un operador de comparación basado en el espacio de creencias el cual favorezca a soluciones ubicadas en regiones poco pobladas. El operador se detalla en el algoritmo 5.3.

5.2.3. Operador de búsqueda local

El principal uso que se le da al espacio de creencias en esta propuesta es el de aportar conocimiento para un operador de búsqueda local.

Muchos de los operadores de búsqueda local requieren de varias evaluaciones de la función objetivo para lograr mejorar una solución. En contraste, en esta propuesta se utiliza el espacio de creencias para obtener la información que normalmente saldría de estas evaluaciones.

El procedimiento de búsqueda local puede resumirse en tres etapas:

1. Buscar en el espacio de creencias 2^k soluciones más cercanas a la solución sobre la cual se aplica el procedimiento.

2. Clasificar las soluciones vecinas en tres categorías: soluciones que son dominadas por la solución actual, soluciones que dominan a la solución actual y soluciones que no dominan ni son dominadas por la solución actual.

3. Dependiendo del número de soluciones de cada tipo, el operador puede realizar movimientos hacia un cono de descenso o hacia un cono de diversidad.

Algoritmo 5.3 Operador de comparación basado en el espacio de creencias.

Entrada: $\vec{p}$ (primera solución a comparar)
Entrada: $\vec{q}$ (segunda solución a comparar)
Entrada: A (árbol KD que compone al espacio de creencias)
Entrada: ϵ (un valor de tolerancia)
Salida: $\vec{b}$ mejor solución.

si $\vec{p} \prec \vec{q}$ **entonces**
  $\vec{b} \leftarrow \vec{p}$
si no, **si** $\vec{q} \prec \vec{p}$ **entonces**
  $\vec{b} \leftarrow \vec{q}$
si no
  $[\vec{r}_1, \vec{r}_2] \leftarrow$ primer y segundo vecinos más cercanos a $\vec{p}$ en A
  $[\vec{s}_1, \vec{s}_2] \leftarrow$ primer y segundo vecinos más cercanos a $\vec{q}$ en A
  si $|\vec{p} - \vec{r_1}| > \epsilon$ y $|\vec{q} - \vec{s_1}| > \epsilon$ **entonces**
    si $|\vec{p} - \vec{r_1}| > |\vec{q} - \vec{s_1}|$ **entonces**
      $\vec{b} \leftarrow \vec{p}$
    si no, **si** $|\vec{q} - \vec{s_1}| > |\vec{p} - \vec{r_1}|$ **entonces**
      $\vec{b} \leftarrow \vec{q}$
    si no
      si $|\vec{p} - \vec{r_2}| > \epsilon$ y $|\vec{q} - \vec{s_2}| > \epsilon$ **entonces**
        si $|\vec{p} - \vec{r_2}| > |\vec{q} - \vec{s_2}|$ **entonces**
          $\vec{b} \leftarrow \vec{p}$
        si no, **si** $|\vec{q} - \vec{s_2}| > |\vec{p} - \vec{r_2}|$ **entonces**
          $\vec{b} \leftarrow \vec{q}$
        si no
          si $0 =$ flip() **entonces**
            $\vec{b} \leftarrow \vec{p}$
          si no
            $\vec{b} \leftarrow \vec{q}$

Utilizando la solución actual y una solución vecina, puede generarse un vector de dirección que lleve la búsqueda hacia una dirección de descenso o diversidad. Supongamos que $\vec{x}_0$ es la solución sobre la cual se aplica el operador de búsqueda y $\vec{x}_{\text{vec}}$ es una solución vecina. Considerando el vector dirigido de $\vec{x}_0$ hacia $\vec{x}_{\text{vec}}$:

$$\vec{v} = \frac{1}{|\vec{x}_{\text{vec}} - \vec{x}_0|}\vec{x}_{\text{vec}} - \vec{x}_0$$

podemos considerar las siguientes direcciones de búsqueda para cada uno de los tres casos mencionados anteriormente:

- Si $\vec{x}_{\text{vec}} \prec \vec{x}_0$ entonces $\vec{v}$ es una dirección de descenso.
- Cuando $\vec{x}_0 \prec \vec{x}_{\text{vec}}$ la dirección de descenso es $-\vec{v}$.
- Finalmente, si $\vec{x}_{\text{vec}}$ no domina ni es dominada por $\vec{x}_0$ entonces tanto $\vec{v}$ como $-\vec{v}$ son direcciones de diversidad.

Como se mencionó en la sección 3.2, mientras más cerca se encuentre una solución del frente de Pareto, más estrechos se volverán los conos de descenso y ascenso. Debido a esto, entre los vecinos de una solución alejada del frente de Pareto se espera encontrar un mayor número de soluciones dominadas o que la dominan. Por otro lado, si la solución ya está cerca del frente de Pareto habrá más soluciones no comparables (es decir, que no dominan ni son dominadas por ninguna otra).

En la implementación del operador, el primer caso que se considera es cuando todas las soluciones vecinas no son comparables. Cuando esto ocurre, la solución se encuentra cerca del frente de Pareto. En este caso, la dirección utilizada es el vector que parte desde la solución actual hacia el punto medio entre dos soluciones vecinas $\vec{x}_{\text{vec}_1}$ y $\vec{x}_{\text{vec}_2}$. Donde $\vec{x}_{\text{vec}_1}$ es el vecino más cercano a $\vec{x}_{\text{vec}_2}$ y estas dos soluciones son la pareja de vecinos más cercanos con la distancia más grande entre ellos.

Cuando existen dos o más vecinos que dominan a la solución actual se emplea un procedimiento similar: la dirección de búsqueda se elige como el vector desde la solución actual hacia el punto medio entre las dos soluciones más cercanas con la mayor distancia entre sí.

El objetivo de elegir la dirección de búsqueda de esta forma es lograr una mejor distribución de las soluciones generadas.

Finalmente, si no se ha aplicado ninguno de los dos casos anteriores entonces se procede a verificar si existen una o más soluciones vecinas que sean dominadas por la solución actual. En caso de haberlas, se calcula el vector de dirección que parte desde cada una de ellas hacia la solución actual. Cada uno de estos vectores son direcciones de descenso. La dirección utilizada por el operador es el promedio de estos vectores de descenso.

Para el cálculo del tamaño de paso sería deseable que la solución se moviera por lo menos tan lejos como la distancia máxima entre sus vecinos más cercanos. Si $\vec{x}_0$ es la solución original y $\vec{d}$ el vector de dirección, entonces se debe elegir el tamaño

de paso t tal que la nueva solución $(\vec{x}_0 + t\vec{d})$ cumpla con el requisito mencionando al inicio de este párrafo.

Para el cálculo del tamaño de paso con las características deseadas se emplea el procedimiento descrito en la sección 3.2.3. Para el cálculo de la constante de Lipschitz se emplea una aproximación basada en los vecinos más cercanos obtenidos con el espacio de creencias:

$$L = \max_j \frac{||F(\vec{x}_0) - F(\vec{x}_{\text{vec}_j})||_\infty}{||\vec{x}_0 - \vec{x}_{\text{vec}_j}||_\infty}$$

5.3. Integración con NSGA-II

En nuestra propuesta, durante el primer tercio de las generaciones se permite que NSGA-II se ejecute sin modificaciones. El procedimiento de búsqueda local propuesto trabaja mejor cuando las soluciones ya han comenzado a distribuirse sobre el espacio de búsqueda. Es por eso que se permite a NSGA-II iniciar su operación sin modificaciones.

El algoritmo 5.4 muestra la sección correspondiente a un ciclo para una generación de NSGA-II.

Algoritmo 5.4 Ciclo principal de NSGA-II. Este fragmento representa las operaciones realizadas en una generación de NSGA-II.

$R_t = P_t \cup Q_t$
$\mathcal{F} = \text{ordenamientoNoDominadoRapido}(R_t)$
mientras $|P_{t+1}| \geq N$ **hacer**
  $\text{distanciaDeAgrupamiento}(\mathcal{F}_i)$
  $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
$\text{ordena}(P_{t+1}, \geq_n)$
$P_{t+1} = P_{t+1}[0 : N]$
$Q_{t+1} = \text{generaNuevaPoblacion}(P_{t+1})$
$t = t + 1$

Los cambios realizados para integrar el esquema del algoritmo cultural pueden observarse en el algoritmo 5.5.

El operador de selección ($\geq_{\text{espacioDeCreencias}}$) es empleado en el torneo binario utilizado para elegir a los padres usados en la generación de los nuevos individuos. El operador de búsqueda local se aplica a las soluciones después de los operadores clásicos de NSGA-II.

Algoritmo 5.5 Modificaciones realizadas al ciclo principal de NSGA-II para integrar el esquema de un algoritmo cultural.

$R_t = P_t \cup Q_t$
$A_t = \emptyset$
$\mathcal{F} = \text{ordenamientoNoDominadoRapido}(R_t)$
mientras $|P_{t+1}| \geq N$ **hacer**
  $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
$\text{ordena}(P_{t+1}, \geq_{\text{espacioDeCreencias}})$
$P_{t+1} = P_{t+1}[0 : N]$
$Q_{t+1} = \text{generaNuevaPoblacion}(P_{t+1})$
$A_{t+1} = \text{acceptaIndividuos}(P_{t+1}, A_t)$
$\text{actualizaEspacioDeCreencias}(A_{t+1})$
$t = t + 1$

5.4. Parámetros del algoritmo

En conjunto con los parámetros requeridos por NSGA-II, la técnica aquí descrita requiere que se proporcionen ciertos valores que determinan su desempeño. En la tabla 5.1 se mencionan los parámetros requeridos junto con una breve explicación de su función así como de los valores recomendados para ellos.

Tabla 5.1: Parámetros requeridos por la técnica.

Parámetro	Descripción	Valor recomendado
aceptacion	Porcentaje de individuos aceptados para aportar información al espacio de creencias	No debe ser mayor a 50 %. Un valor adecuado es alrededor de 40 %
$multiplicador_{arbol}$	El número máximo de puntos en el árbol es igual al tamaño de la población multiplicado por este parámetro.	No debe ser menor a 2 pues el espacio de creencias no tendrá suficiente información para lograr una mejora en el desempeño. Tampoco se recomienda que sea mayor a 5 pues únicamente se incrementaría el uso de memoria sin mejorar el desempeño. El valor sugerido es 4.

$multiplicador_{distancia}$	La distancia de separación deseada entre una solución y la solución generada por el operador de búsqueda local es multiplicada por este parámetro.	Si no se conoce el problema se sugiere dejar este parámetro en 1. Es posible lograr una convergencia más rápida incrementando este valor (en nuestras pruebas valores de hasta 2 generaban buenos resultados). Se debe tener en cuenta que en algunos problemas al incrementar el valor de este parámetro la distribución de las soluciones puede verse negativamente afectada.
$probabilidad_{operador}$	Probabilidad de aplicación del operador de búsqueda local.	No se recomienda aplicar el operador a todas las soluciones; porcentajes de hasta 80 % son aceptables.
reconstruir	Indica el número de generaciones que deben pasar antes de que el árbol sea reconstruido.	El valor elegido para este parámetro no afecta en ningún modo la calidad de las soluciones generadas. Su función es controlar el uso de memoria. Cualquier valor entre 1 y el número máximo de generaciones es aceptable. El valor utilizado en la implementación del algoritmo es igual a 20.

5.5. Observaciones finales

El algoritmo propuesto en esta tesis es un algoritmo cultural para resolver problemas de optimización multiobjetivo. Este fue construido utilizando como espacio de población a NSGA-II. El espacio de creencias consiste de un conjunto de soluciones representativas contenidas en una estructura de datos que permite realizar consultas espaciales sobre puntos. El uso principal del conocimiento contenido en el espacio de creencias es guiar a un operador de búsqueda local hacia la generación de mejores soluciones.

Los puntos principales en el diseño del algoritmo propuesto son los siguientes:

- Como base se utilizó a NSGA-II. Las razones por las que se eligió a este algoritmo como motor de búsqueda global son el ser un algoritmo probado en gran cantidad de problemas, además, tiene la propiedad de obtener una buena dispersión de las soluciones lo cual resulta útil en lograr que el espacio de creencias contenga una muestra representativa de las soluciones encontradas.

- La estructura del espacio de creencias se diseñó como un conjunto de soluciones representativas almacenadas en un árbol KD. En cada generación del algoritmo un nuevo conjunto de soluciones es incluído en el espacio de creencias y algunas soluciones son eliminadas de éste. Los criterios bajo los cuales las soluciones son incluidas o eliminadas tienen por objetivo mantener en el espacio de creencias un conjunto de muestras representativas del espacio de búsqueda.

- Se realizaron cambios en el mecanismo de comparación utilizado por NSGA-II para llevar a cabo la selección y el ordenamiento de las soluciones. El operador de comparación propuesto en este trabajo utiliza los puntos de referencia contenidos en el espacio de creencias para favorecer a las soluciones ubicadas en regiones menos pobladas del espacio de búsqueda.

- Para acelerar la convergencia del algoritmo se incorporó un operador de búsqueda local el cual es aplicado a algunas soluciones despúes de los operadores empleados por NSGA-II.

- El espacio de creencias es utilizado por el operador de búsqueda local para decidir la dirección y tamaño de paso utilizados. Las soluciones de referencia son utilizadas para calcular direcciones de descenso y diversidad. El tamaño de paso también es calculando utilizando las soluciones contenidas en el espacio de creencias.

- El utilizar las soluciones contenidas en el espacio de creencias para el operador de búsqueda local logra un reducción en el número de evaluaciones requeridas.

El objetivo principal de la técnica propuesta es lograr una reducción del número de evaluaciones de la función objetivo, y con ese objetivo en mente, el algoritmo cultural aquí descrito será validado en el capítulo siguiente.

Capítulo 6

Resultados

En este capítulo se presentan las pruebas realizadas para evaluar el desempeño del algoritmo propuesto en el capítulo anterior.

Para comparar los resultados obtenidos por dos algoritmos de optimización multiobjetivo diferentes, es necesario comparar las soluciones obtenidas por cada uno de ellos en el mismo conjunto de problemas, y estableciendo condiciones que permitan una comparación justa. En este caso se decidió utilizar el conjunto de funciones de prueba Zitzler-Deb-Thiele (ZDT) [52], que ha sido utilizado frecuentemente en la literatura especializada.

En los inicios de la optimización multiobjetivo usando cómputo evolutivo eran comunes las comparaciones visuales entre los frentes de Pareto obtenidos por dos algoritmos distintos. Sin embargo, para realmente lograr una comparación objetiva es necesario definir una medida cuantitativa de la calidad de las soluciones obtenidas.

En el caso de los problemas de optimización de un único objetivo es sencillo decidir cuándo una solución es mejor que otra. Por otro lado, para problemas de optimización multiobjetivo donde no existe una solución única sino un conjunto de soluciones, no es fácil definir un criterio para determinar cuándo un conjunto es mejor que otro.

En la literatura han sido propuestos varios indicadores de calidad o desempeño. Cada uno de estos indicadores se basan en diferentes metodologías y definiciones de calidad de las soluciones. En la siguiente sección se realiza un breve resumen de los indicadores utilizados en esta tesis para validar nuestros resultados.

6.1. Medidas de desempeño

Antes de comenzar a definir las medidas de desempeño utilizadas en este trabajo es necesario definir los requisitos necesarios de un conjunto de soluciones generadas por un algoritmo de optimización multiobjetivo.

Se espera que un conjunto de soluciones se encuentre compuesto únicamente por soluciones mutuamente no comparables; es decir, para cualquier par de soluciones $\vec{x}_1$, $\vec{x}_2$ no ocurre que $\vec{x}_1 \prec \vec{x}_2$ ni $\vec{x}_2 \prec \vec{x}_1$. A este tipo de conjuntos se les llama **aproximación del conjunto de Pareto** y a su imagen bajo la función objetivo se

le conoce como **aproximación del frente de Pareto**.

Se debe observar que bajo esta definición un conjunto de aproximación no tiene por que ser igual al conjunto de Pareto. Encontrar el verdadero conjunto de óptimos de Pareto es, en general, un problema difícil por lo que generalmente sólo se obtienen conjuntos de aproximación.

Como se mencionó en la sección 2.1.4, las soluciones generadas por un buen algoritmo de optimización deben cumplir dos objetivos:

- Encontrarse lo más cerca posible del verdadero frente de Pareto.
- Tener una buena distribución sobre el frente de Pareto.

Las medidas de desempeño disponibles para comparar algoritmos evolutivos multiobjetivo miden la calidad de las soluciones de acuerdo a estos dos aspectos.

Aun cuando el resultado de un algoritmo de optimización es un conjunto de vectores de decisión, se ha hecho la observación [53] [54] de que resulta más natural realizar la evaluación del desempeño en el espacio de las funciones objetivo. Es por ello que las medidas de desempeño aquí mencionadas trabajan con conjuntos de aproximación del frente de Pareto.

Un indicador de calidad es una función $I : \Omega \rightarrow \mathbb{R}$ que asigna un valor real a un conjunto de aproximación del frente de Pareto.

Un indicador debe definir un orden total en los conjuntos de aproximación del frente de Pareto. Si tenemos dos conjuntos de aproximación A, B tales que $I(A) > I(B)$ entonces el conjunto A es preferible sobre B.

Un indicador particular considera aceptable un conjunto de aproximación bajo algunos criterios específicos considerados en su diseño. Es por esto que puede ocurrir que $I(A) > I(B)$ utilizando algún indicador pero $I'(B) > I'(A)$ bajo un indicador distinto.

6.1.1. Hipervolumen

Este indicador fue propuesto por Zitzler y Thiele [55]. Consiste en la medida de la región dominada por el conjunto de aproximación y acotada por un punto de referencia $\vec{r} \in \mathbb{R}^k$. Será denotado por I_H.

Si el conjunto de aproximación A está compuesto por los puntos $A = \{\vec{x}_1, \ldots, \vec{x}_i\}$, entonces $I_H(A)$ es igual volúmen k-dimensional (área en el caso de dos objetivos, volumen para tres objetivos) de la región formada por la unión de los hiperrectángulos delimitados por cada $\vec{x}_j \in A$ y el punto de referencia $\vec{r}$ (ver ejemplo en la figura 6.1.1).

Una particularidad importante de este indicador es que entre todos los posibles conjuntos de aproximación para un problema particular, aquél que maximiza el hipervolumen está compuesto únicamente por puntos Pareto óptimos [56].

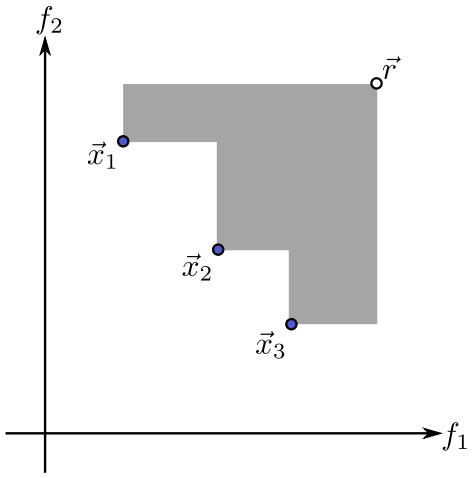

Figura 6.1: Ejemplo del hipervolumen para un problema de dos objetivos. El área de la región sombreada es el valor del indicador.

6.1.2. Distribución

Utilizado inicialmente para evaluar la distribución de las soluciones en NSGA-II, este indicador fue propuesto por Deb [6]. Para su cálculo se utiliza la siguiente ecuación:

$$I_D(A) = \frac{\sum_{m=1}^{k} d_m^e + \sum_{i=1}^{|A|} |d_i - d|}{\sum_{m=1}^{k} d_m^e + |A|d}$$

Donde cada d_i son las distancias euclidianas entre cada solución y su vecino más cercano; d es el promedio de estas distancias.

Un conjunto de aproximación puede tener una buena distribución pero no cubrir por completo el frente de Pareto. A manera de penalizar este caso los parámetros d_m^e son la distancia entre las soluciones extremas del verdadero frente de Pareto y el conjunto de aproximación en cada una de las funciones objetivo.

6.1.3. Distancia generacional invertida

Antes de describir la distancia generacional invertida debemos explicar la distancia generacional. Esta medida de desempeño fue propuesta por Van Veldhuizen y Lamont [57] [58].

Su finalidad es medir qué tan lejos se encuentra el conjunto de aproximación A del verdadero frente de Pareto $\mathcal{PF}$. Se calcula utilizando la siguiente ecuación:

$$I_{DG}(A) = \frac{\left(\sum_{i=1}^{|A|} d_i^2\right)^{\frac{1}{2}}}{|A|}$$

Los valores d_i corresponden a la distancia euclidiana entre cada elemento de A y el miembro más cercano en $\mathcal{PF}$.

En este indicador, un conjunto de aproximación A es mejor que otro conjunto B si $I_{DG}(A) < I_{DG}(B)$.

La distancia generacional invertida difiere en que las distancias d_i son medidas desde cada elemento de una discretización P de $\mathcal{PF}$ hacia el elemento más cercano en el conjunto de aproximación A:

$$I_{DGI}(A) = \frac{\left(\sum_{i=1}^{|P|} d_i^2\right)^{\frac{1}{2}}}{|P|}$$

6.1.4. Cobertura de conjuntos

A diferencia de las medidas de desempeño mencionadas anteriormente, la cobertura de conjuntos es un indicador binario, es decir, es una función $I : \Omega \times \Omega \to \mathbb{R}$.

La cobertura de conjuntos fue propuesta por Zitzler y Thiele [59]. Denotado por I_{CC}, este indicador representa la proporción de soluciones dominadas o iguales entre dos conjuntos de aproximación.

$$I_{CC}(A, B) = \frac{\left|\left\{\vec{b} \in B : \exists \vec{a} \in A : \vec{a} \prec \vec{b} \vee \vec{a} = \vec{b}\right\}\right|}{|B|}$$

Si $I_{CC}(A, B) = 1$ entonces todos los elementos de B son dominados por los de A y, por lo tanto, A es preferible sobre B.

Este indicador no es simétrico, es decir, $I_{CC}(A, B)$ puede ser diferente de $I_{CC}(B, A)$ por lo que es necesario considerar ambos casos para comparar dos conjuntos de aproximación.

6.2. Evaluación del desempeño

6.2.1. Funciones de prueba

El conjunto de funciones de prueba Zitzler-Deb-Thiele (ZDT) ha sido utilizado previamente para validar el desempeño de NSGA-II [6]. Es por esto que se decidió utilizarlo también para evaluar el algoritmo propuesto en el capítulo anterior.

Las cinco funciones de prueba utilizadas se presentan en el apéndice A. [1]

6.2.2. Resultados

El objetivo del algoritmo cultural es reducir el número de evaluaciones. El algoritmo propuesto es comparado contra NSGA-II a fin de poder determinar si efectivamente se reduce el número de evaluaciones requeridas por el algortimo evolutivo multiobjetivo al que se aplique (el NSGA-II en este caso). La metodología utilizada para obtener los resultados presentados fue la siguiente:

[1] Se hace notar que ZDT5 se omitió de este estudio comparativo, por ser un problema binario.

Tabla 6.1: Parámetros utilizados para el problema ZDT1

Parámetro	Algoritmo Cultural	NSGA-II
Tamaño de la población	100	100
Probabilidad de mutación	0.033	0.033
Probabilidad de cruza	0.9	0.9
Distribución de la mutación	20	20
Distribución de la cruza	15	15
Multiplicador de distancia	1.3	-
Probabilidad operador	0.6	-

Tabla 6.2: Estadísticas del hipervolumen en el problema ZDT1.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**118**	**115**	**119**	**0.93**
NSGA-II	116	114	117	0.65

- Se fijó un número máximo de evaluaciones de la función objetivo.
- Para cada problema de prueba se realizaron 100 ejecuciones de los dos algoritmos.
- Se calcularon las medidas de desempeño antes mencionadas para cada conjunto de aproximación obtenido.

En todos los casos, el parámetro porcentaje de aceptación del algoritmo cultural se fijó en 40 % y el tamaño del árbol en cuatro veces el tamaño de la población.

Las estadísticas de estas pruebas se presentan en las secciones siguientes. Las gráficas de las soluciones obtenidas por cada algoritmo en algunos de los problemas de prueba se presentan en el apéndice C.

Problema ZDT1

El número de evaluaciones se fijó en 3100. Los parámetros utilizados se muestran en la tabla 6.1. Para el hipervolumen se utilizó como punto de referencia $\vec{r} = [11, 11]$. El valor máximo del hipervolumen en este problema es $120\frac{2}{3}$.

Basados en los valores obtenidos en el cálculo del hipervolumen (tabla 6.2) podemos observar que el algoritmo cultural logra acercarse más al frente de Pareto que NSGA-II utilizando el mismo número de evaluaciones de la función objetivo. Por otro lado, para NSGA-II el hipervolumen se mantiene más próximo al promedio en todas las corridas.

Tabla 6.3: Estadísticas del indicador de distribución en el problema ZDT1.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	0.69	0.50	0.92	0.084
NSGA-II	**0.66**	**0.51**	**0.83**	**0.057**

Tabla 6.4: Estadísticas de la distancia generacional invertida en el problema ZDT1.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**0.0051**	**0.0023**	**0.0103**	**0.00177**
NSGA-II	0.0107	0.0077	0.0142	0.00135

En el caso de la distribución (tabla 6.3) se observa que NSGA-II en promedio logra obtener una mejor distribución. Sin embargo, existen algunos casos donde el algoritmo cultural logra superar a NSGA-II.

Respecto a la distancia generacional invertida (tabla 6.4) el conjunto de aproximación generado por el algoritmo cultural, en promedio, se encuentra más cercano al verdadero frente de Pareto.

Finalmente se observa en la tabla 6.5 que las soluciones producidas por el algoritmo cultural, en su mayoría, dominan a las obtenidas por NSGA-II.

Problema ZDT2

Para este problema, el número máximo de evaluaciones de la función objetivo también se fijó en 3100. Los parámetros utilizados se muestran en la tabla 6.6. El valor máximo del hipervolumen en este problema es $120\frac{1}{3}$ usando como punto de referencia $\vec{r} = [11, 11]$.

Los resultados para el hipervolumen (tabla 6.7) son similares a los del problema ZDT1. El algoritmo cultural supera a NSGA-II en cuanto a la cercanía de las soluciones al frente de Pareto.

La distribución de las soluciones (tabla 6.8) obtenidas por el algoritmo cultural es

Tabla 6.5: Cobertura de conjuntos para el problema ZDT1. P_C representa el conjunto de aproximación obtenido por el algoritmo cultural y P_N el obtenido por NSGA-II.

	Promedio	Mínimo	Máximo	Desviación estándar
$I_{CC}(P_N, P_C)$	0.00023	0.0	0.02380	0.002369
$I_{CC}(P_C, P_N)$	0.99403	0.80645	1.0	0.026293

Tabla 6.6: Parámetros utilizados para el problema ZDT2

Parámetro	Algoritmo Cultural	NSGA-II
Tamaño de la población	100	100
Probabilidad de mutación	0.033	0.033
Probabilidad de cruza	0.9	0.9
Distribución de la mutación	20	20
Distribución de la cruza	15	15
Multiplicador de distancia	1.2	-
Probabilidad operador	0.6	-

Tabla 6.7: Estadísticas del hipervolumen en el problema ZDT2.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**111**	**105**	**117**	**2.46**
NSGA-II	107	100	111	2.39

mejor en el caso promedio. Sin embargo, existen casos donde la distribución obtenida po el algoritmo cultural es peor que la lograda por NSGA-II.

De acuerdo a la distancia generacional invertida (tabla 6.9), el algoritmo cultural resulta mejor que NSGA-II, logrando siempre una menor distancia al frente de Pareto y manteniéndose estable en alcanzar esa distancia en todas las ejecuciones del algoritmo.

Las soluciones encontradas por el algoritmo cultural para el problema ZDT2 no fueron nunca dominadas por las encontradas por NSGA-II (tabla 6.10). Por otro lado, las soluciones obtenidas por NSGA-II fueron en su mayoría dominadas por las del algoritmo cultural.

Problema ZDT3

La comparación de los algoritmos usando el problema ZDT3 se realizó limitando el número de evaluaciones a 3100. Los parámetros de ambos algoritmos se encuentran en la tabla 6.11. El hipervolumen se calculó utilizando como referencia el punto $\vec{r} =$

Tabla 6.8: Estadísticas del indicador de distribución en el problema ZDT2.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**0.78**	**0.56**	**1.14**	**0.238**
NSGA-II	0.88	0.65	1.01	0.076

Tabla 6.9: Estadísticas de la distancia generacional invertida en el problema ZDT2.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**0.0132**	**0.0058**	**0.0302**	**0.0056**
NSGA-II	0.025	0.0151	0.0422	0.00562

Tabla 6.10: Cobertura de conjuntos para el problema ZDT2. P_C representa el conjunto de aproximación obtenido por el algoritmo cultural y P_N el obtenido por NSGA-II.

	Promedio	Mínimo	Máximo	Desviación estándar
$I_{CC}(P_N, P_C)$	0.0	0.0	0.0	0.0
$I_{CC}(P_C, P_N)$	0.8484	0.0588	1.0	0.29544

$[11, 11]$, siendo su valor máximo de 128.77811.

El algoritmo cultural logra acercarse más al frente de Pareto que NSGA-II de acuerdo al hipervolumen (tabla 6.12).

La distribución de las soluciones calculadas por el algoritmo cultural es más variable que la de las generadas por NSGA-II (tabla 6.13). Sin embargo, en la mayoría de los casos, el algoritmo cultural logra una mejor distribución.

Respecto a la distancia generacional invertida, los resultados son similares a las dos funciones de prueba anteriores: el algoritmo cultural logra acercarse más al frente de Pareto pero con una mayor variabilidad en las distintas ejecuciones del algoritmo.

Finalmente, de acuerdo al indicador de cobertura de conjuntos, sólo en pocos casos las soluciones obtenidas por el algoritmo cultural son dominadas por las obtenidas por NSGA-II.

Tabla 6.11: Parámetros utilizados para el problema ZDT3

Parámetro	Algoritmo Cultural	NSGA-II
Tamaño de la población	100	100
Probabilidad de mutación	0.033	0.033
Probabilidad de cruza	0.9	0.9
Distribución de la mutación	20	20
Distribución de la cruza	15	15
Multiplicador de distancia	1.2	-
Probabilidad operador	0.6	-

Tabla 6.12: Estadísticas del hipervolumen en el problema ZDT3.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**125**	**120**	**127**	**1.22**
NSGA-II	123	120	125	0.70

Tabla 6.13: Estadísticas del indicador de distribución en el problema ZDT3.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**0.72**	**0.53**	**0.9**	**0.0742**
NSGA-II	0.74	0.6	0.87	0.053

Tabla 6.14: Estadísticas de la distancia generacional invertida en el problema ZDT3.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**0.0056**	**0.0023**	**0.0092**	**0.00117**
NSGA-II	0.0077	0.0059	0.0106	0.00092

Tabla 6.15: Cobertura de conjuntos para el problema ZDT3. P_C representa el conjunto de aproximación obtenido por el algoritmo cultural y P_N el obtenido por NSGA-II.

	Promedio	Mínimo	Máximo	Desviación estándar
$I_{CC}(P_N, P_C)$	0.0415	0.0	0.3913	0.06572
$I_{CC}(P_C, P_N)$	0.8794	0.3478	1.0	0.13244

Tabla 6.16: Parámetros utilizados para el problema ZDT4

Parámetro	Algoritmo Cultural	NSGA-II
Tamaño de la población	100	100
Probabilidad de mutación	0.1	0.1
Probabilidad de cruza	0.9	0.9
Distribución de la mutación	20	20
Distribución de la cruza	15	15
Multiplicador de distancia	1.5	-
Probabilidad operador	0.45	-

Tabla 6.17: Estadísticas del indicador de distribución en el problema ZDT4.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**0.85**	**0.44**	**1.11**	**0.132**
NSGA-II	1.03	0.51	1.47	0.171

Problema ZDT4

El problema ZDT4 es considerado difícil. NSGA-II no logra cubrir una región amplia del espacio de búsqueda en las primeras generaciones. Es por ello que el algoritmo cultural no logra obtener buenos resultados en este problema.

En la mayoría de las ejecuciones de ambos algoritmos, existen soluciones con valores de los objetivos mayores que el punto de referencia utilizado comúnmente en la literatura para calcular el hipervolumen de este problema. Debido a ello, no se presentan los valores del hipervolumen para este problema.

Los parámetros utilizados para realizar las pruebas en este problema se muestran en la tabla 6.16.

Los valores del indicador de distribución (tabla 6.17) son muy variables cuando se realiza un número reducido de evaluaciones en ambos algoritmos. A pesar de ello, en promedio, el algoritmo cultural logra una mejor distribución.

NSGA-II supera al algoritmo cultural en el problema ZDT4 de acuerdo a la distancia generacional invertida (tabla 6.18).

Tabla 6.18: Estadísticas de la distancia generacional invertida en el problema ZDT4.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	0.3295	0.1320	0.7346	0.10106
NSGA-II	**0.1801**	**0.0480**	**0.3334**	**0.05391**

Tabla 6.19: Cobertura de conjuntos para el problema ZDT4. P_C representa el conjunto de aproximación obtenido por el algoritmo cultural y P_N el obtenido por NSGA-II.

	Promedio	Mínimo	Máximo	Desviación estándar
$I_{CC}(P_N, P_C)$	0.4285	0.0	1.0	0.25014
$I_{CC}(P_C, P_N)$	0.3762	0.0	1.0	0.26278

Tabla 6.20: Parámetros utilizados para el problema ZDT6

Parámetro	Algoritmo Cultural	NSGA-II
Tamaño de la población	100	100
Probabilidad de mutación	0.1	0.1
Probabilidad de cruza	0.9	0.9
Distribución de la mutación	20	20
Distribución de la cruza	15	15
Multiplicador de distancia	1.2	-
Probabilidad operador	0.6	-

De acuerdo a la cobertura de conjuntos NSGA-II es mejor que el algoritmo cultural en el problema ZDT4.

Problema ZDT6

Para el problema ZDT6, los parámetros utilizados se presentan en la tabla 6.20. Los resultados presentados son con un límite de 5100 evaluaciones de la función objetivo.

El punto de referencia para el hipervolumen es $\vec{r} = [11, 11]$ y su valor máximo es 119.51857.

El algoritmo cultural supera a NSGA-II en el problema ZDT6 de acuerdo al hipervolumen (tabla 6.21). Al igual que en los otros problemas, NSGA-II logra un valor de hipervolumen más cercano al promedio en todas las ejecuciones.

NSGA-II obtiene una mejor distribución que el algoritmo cultural en este problema (tabla 6.22).

Tabla 6.21: Estadísticas del hipervolumen en el problema ZDT6.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**109**	**93**	**117**	**6.61**
NSGA-II	98	91	103	1.95

Tabla 6.22: Estadísticas del indicador de distribución en el problema ZDT6.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	0.91	0.28	1.69	0.277
NSGA-II	**0.80**	**0.41**	**1.07**	**0.105**

Tabla 6.23: Estadísticas de la distancia generacional invertida en el problema ZDT6.

	Promedio	Mínimo	Máximo	Desviación estándar
Alg. Cultural	**0.018**	**0.0001**	**0.0558**	**0.0135**
NSGA-II	0.0423	0.0297	0.0627	0.00543

Con base en la distancia generacional invertida, el algoritmo cultural supera a NSGA-II al acercarse más al frente de Pareto.

6.2.3. Reducción del número de evaluaciones

A fin de mostrar de forma más clara la reducción del número de evaluaciones necesarias se realizó un segundo experimento utilizando los problemas de prueba ZDT1, ZDT2, ZDT3 y ZDT6. Se realizaron 10 ejecuciones del algoritmo cultural y de NSGA-II en cada problema. Cada ejecución de los algoritmos fue detenida en la generación donde el hipervolumen fue igual o excedió el 98 % de su valor máximo. Los parámetros utilizados por ambos algoritmos son los mismos que los del experimento de la sección anterior.

Las tablas 6.25, 6.26, 6.27 y 6.28 muestran los resultados de este experimento para los problemas ZDT1, ZDT2, ZDT3 y ZDT6, respectivamente.

En los cuatro casos, el algoritmo cultural logró, de acuerdo al indicador hipervolumen, acercarse al verdadero frente de Pareto tanto como NSGA-II pero utilizando un número de evaluaciones de la función objetivo significativamente menor.

En promedio el algoritmo cultural requiere realizar únicamente el 63.3 % del total de evaluaciones requeridas por NSGA-II en el problema ZDT1, 56.6 % para ZDT2,

Tabla 6.24: Cobertura de conjuntos para el problema ZDT6. P_C representa el conjunto de aproximación obtenido por el algoritmo cultural y P_N el obtenido por NSGA-II.

	Promedio	Mínimo	Máximo	Desviación estándar
$I_{CC}(P_N, P_C)$	0.2947	0.0	0.8235	0.16095
$I_{CC}(P_C, P_N)$	0.5962	0.0	1.0	0.25059

Tabla 6.25: Número de evaluaciones requeridas para alcanzar un valor de hipervolumen igual o superior a 118.2533 en el problema ZDT1.

	Promedio	Mínimo	Máximo
Algo. Cultural	**2908.4**	**2645**	**3136**
NSGA-II	4590.0	4100	5000

Tabla 6.26: Número de evaluaciones requeridas para alcanzar un valor de hipervolumen igual o superior a 117.9266 en el problema ZDT2.

	Promedio	Mínimo	Máximo
Algo. Cultural	**3951.3**	**3228**	**4728**
NSGA-II	6977.7	6400	7500

Tabla 6.27: Número de evaluaciones requeridas para alcanzar un valor de hipervolumen igual o superior a 126.2025 en el problema ZDT3.

	Promedio	Mínimo	Máximo
Algo. Cultural	**2872.1**	**2615**	**3477**
NSGA-II	4740.0	4200	5200

Tabla 6.28: Número de evaluaciones requeridas para alcanzar un valor de hipervolumen igual o superior a 117.1282 en el problema ZDT6.

	Promedio	Mínimo	Máximo
Algo. Cultural	**9417.3**	**3499**	**17696**
NSGA-II	18750.0	17700	19700

Tabla 6.29: Parámetros utilizados para el problema DTLZ2

Parámetro	Algoritmo Cultural	NSGA-II
Tamaño de la población	100	100
Probabilidad de mutación	0.083	0.083
Probabilidad de cruza	0.9	0.9
Distribución de la mutación	20	20
Distribución de la cruza	15	15
Multiplicador de distancia	1.1	-
Probabilidad operador	0.5	-

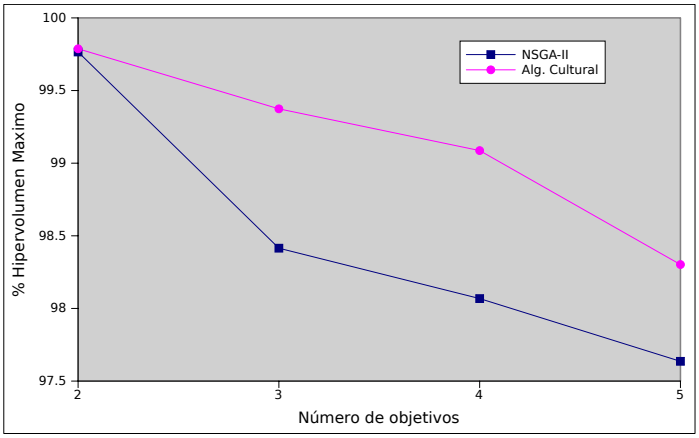

Figura 6.2: Porcentajes del hipervolumen máximo logrados por el algoritmo cultural y NSGA-II en la función DTLZ2, con 2, 3, 4 y 5 objetivos.

60.6 % para ZDT3 y 50.2 % para ZDT6.

6.2.4. Escalabilidad

Finalmente, se llevó a cabo un tercer experimento utilizando el problema DTLZ2 [60] (ver apéndice B) para evaluar el desempeño del algoritmo al incrementar el número de objetivos.

El experimento consiste en realizar 10 ejecuciones independientes del algoritmo cultural y NSGA-II para las versiones con 2, 3, 4 y 5 objetivos del problema DTLZ2. Para cada ejecución se calculó el porcentaje del hipervolumen máximo alcanzado por cada algoritmo. Los parámetros utilizados por ambos algoritmos se muestran en la tabla 6.29; el número de evaluaciones de la función se limitó a 3000.

Se observa en la gráfica 6.2.4 que el algoritmo cultural logra, en promedio, superar a NSGA-II (con base en el hipervolumen) en el problema DTLZ2 aún al incrementar el número de objetivos hasta 5.

6.3. Sumario

En la mayoría de los problemas de prueba adoptados, el algoritmo cultural cumplió su objetivo de acercarse más al frente de Pareto que NSGA-II utilizando un número reducido de evaluaciones de la función objetivo.

Para el caso del problema ZDT4 aún cuando el algoritmo cultural no logró superar a NSGA-II en un número reducido de evaluaciones es posible observar una pequeña ventaja del algoritmo cultural si se permite un número más elevado de evaluaciones. Cabe mencionar que para este problema, NSGA-II requiere de un número considerable de evaluaciones para lograr converger al verdadero frente de Pareto.

Finalmente, aún cuando en algunos casos el algoritmo cultural logra obtener una mejor distribución de las soluciones que NSGA-II, sigue siendo una debilidad del algoritmo propuesto el no lograr consistentemente una distribución igual o mejor que la lograda por NSGA-II.

Otra debilidad del algoritmo es en el caso de funciones altamente multimodales. En funciones con un número elevado de frentes de Pareto locales el operador de búsqueda local presenta la tendencia a quedar atrapado.

Capítulo 7

Conclusiones y trabajo a futuro

7.1. Conclusiones

La computación evolutiva y, en particular, los algoritmos culturales, han mostrado un buen desempeño en una gran cantidad de problemas difíciles. En esta tesis se realizó una propuesta de un algoritmo cultural el cual utiliza el conocimiento del dominio adquirido durante el proceso de búsqueda para guiar a un buscador local y lograr una reducción en el número de evaluaciones de la función objetivo.

Producto del trabajo realizado podemos mencionar las siguientes conclusiones:

- El algoritmo propuesto consiste en incorporar un espacio de creencias a NSGA-II [6]. El espacio de creencias esta compuesto por un conjunto de soluciones de referencia. La soluciones de referencia se almacenan en una estructura de datos llamada árbol KD [49]. Esta estructura permite realizar de forma eficiente consultas espaciales (vecinos más cercanos, puntos en una región, etc). El espacio de creencias es utilizado para comparar puntos en la selección y para calcular la dirección y tamaño de paso en un operador de búsqueda local incorporado a NSGA-II.

- El uso de un buscador local es una técnica útil para acelerar la convergencia de un algoritmo de optimización multiobjetivo.

- Muchas propuestas de buscadores locales requieren de un número importante de evaluaciones adicionales de la función objetivo. El incorporar un buscador local dentro del marco de un algoritmo cultural permite obtener la información que normalmente sería producto de esas evaluaciones adicionales directamente del espacio de creencias.

- El éxito de la técnica propuesta depende en gran medida del uso de un mecanismo de búsqueda global que logre rápidamente una buena dispersión de las soluciones. Para este trabajo, NSGA-II cumplió ese objetivo logrando que el espacio de creencias fuera un buen repositorio de las experiencias de muchos individuos representativos.

- En otras propuestas de algoritmos culturales el espacio de creencias es estructurado en forma de particiones del espacio las cuales logran buenos resultados pero suelen ser costosas en el uso de memoria. En contraste, para este trabajo se decidió estructurar el espacio de creencias como un conjunto de soluciones de referencia.

- El uso de una estructura de datos eficiente (árbol KD) permitió darle al espacio de creencias (compuesto por soluciones de referencia) una forma que cumple con los objetivos de proporcionar conocimiento del dominio a la selección y el buscador local. El conocimiento consiste en los datos que proporcionan las soluciones de referencia sobre la distribución de las soluciones, además, sirven como guía en la búsqueda de direcciones de descenso y para el cálculo del tamaño de paso en el procedimiento de búsqueda local.

- En problemas donde no se logra una buena dispersión de las soluciones en las primeras etapas del proceso de optimización puede ocurrir que el conocimiento incorporado al espacio de creencias no sea suficiente para lograr una mejora en la velocidad de convergencia. Aún en esos casos es posible obtener buenos resultados si se permite un número mayor de evaluaciones.

- El buscador local puede llegar a tener problemas en funciones altamente multimodales (como es el caso de ZDT4). Por tanto, es necesario encontrar otro mecanismo que explote el conocimiento extraído del espacio de creencias para lograr mover el buscador local fuera de los óptimos locales.

7.2. Trabajo a futuro

Los siguientes puntos son algunas ideas, inquietudes y estrategias a seguir para extender el trabajo presentado en esta tesis:

- Un estudio más detallado del parámetro multiplicador de distancia. En las pruebas del algoritmo se observó que en algunos problemas se puede incrementar el valor del parámetro, logrando así una convergencia más rápida, mientras que en otros, al aumentar el parámetro, la distribución de las soluciones se ve afectada negativamente. Determinar en qué tipo de problemas resulta adecuado utilizar valores mayores a 1.0 para este parámetro puede ser de gran utilidad en la aplicación del algoritmo a problemas del mundo real.

- Explorar si otras estrategias para decidir las soluciones sobre las cuales se debe aplicar el operador de búsqueda local, logran obtener mejores resultados.

- Utilizar un conjunto de problemas de prueba más grande para lograr tener una visión más amplia de los casos para los cuales resulta adecuado el algoritmo propuesto.

- Cambiar el buscador global sobre el cual se construyó el algoritmo y observar qué resultados se obtienen al incorporar el buscador local aquí presentado.
- Incorporar una técnica para el manejo de restricciones. Una posible idea puede surgir de la observación de la estructura del espacio de creencias utilizada por el algoritmo, la cual puede ser modificada fácilmente para incorporar conocimiento, similar a las celdas de creencia [41].

Apéndice A

Problemas ZDT

Cada una de las funciones es definida como $\vec{f} : \mathbb{R}^n \to \mathbb{R}^k$, es decir, $\vec{f}(\vec{x}) \in \mathbb{R}^k$ y $\vec{x} \in \mathbb{R}^n$. Además $\vec{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_k(\vec{x})]$ y $\vec{x} = [x_1, \ldots, x_n]$.

A.1. Problema ZDT1

$$\vec{f} : \mathbb{R}^{30} \to \mathbb{R}^2$$

Donde

$$\begin{aligned} & f_1 = x_1 \\ & g = 1.0 + \tfrac{9}{n-1} \textstyle\sum_{i=2}^{n} x_i \\ & f_2 = g \left(1.0 - \sqrt{\tfrac{f_1}{g}} \right) \\ & 0 \leq x_i \leq 1 \qquad i = 1, \ldots, 30 \end{aligned}$$

En este problema el frente de Pareto verdadero es convexo y conectado.

A.2. Problema ZDT2

$$\vec{f} : \mathbb{R}^{30} \to \mathbb{R}^2$$

Donde

$$\begin{aligned} & f_1 = x_1 \\ & g = 1 + \tfrac{9}{n-1} \left(\textstyle\sum_{i=2}^{n} x_i \right) \\ & f_2 = g \left(1.0 - \left(\tfrac{x_1}{g} \right)^2 \right) \\ & 0 \leq x_i \leq 1 \qquad i = 1, \ldots, 30 \end{aligned}$$

El frente de Pareto verdadero es, en este caso, convexo y conectado.

A.3. Problema ZDT3

$$\vec{f} : \mathbb{R}^{30} \rightarrow \mathbb{R}^2$$

Donde

$$\begin{aligned}
&f_1 = x_1 \\
&g = 1 + \tfrac{9}{n-1}\left(\textstyle\sum_{i=2}^{n} x_i\right) \\
&f_2 = g\left(1 - \sqrt{\tfrac{x_1}{g}} - \tfrac{x_1}{g}\operatorname{sen}(10\pi x_1)\right) \\
&0 \leq x_i \leq 1 \qquad i = 1, \ldots, 30
\end{aligned}$$

El frente de Pareto verdadero de este problema está dividido en cinco partes.

A.4. Problema ZDT4

$$\vec{f} : \mathbb{R}^{10} \rightarrow \mathbb{R}^2$$

Donde

$$\begin{aligned}
&f_1 = x_1 \\
&g = 1 + 10(n-1) + \textstyle\sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x_i)\right) \\
&f_2 = g\left(1 - \left(\tfrac{x_i}{g}\right)^2\right) \\
&0 \leq x_1 \leq 1 \\
&-5 \leq x_i \leq 5 \qquad i = 2, \ldots, 10
\end{aligned}$$

Es un problema difícil pues tiene 21^9 frentes de Pareto locales. El frente de Pareto verdadero es convexo y conectado.

A.5. Problema ZDT6

$$\vec{f} : \mathbb{R}^{10} \rightarrow \mathbb{R}^2$$

Donde

$$\begin{aligned}
&f_1 = 1 - e^{-4x_1}\operatorname{sen}^6(6\pi x_1) \\
&g = 1 + 9\left(\tfrac{\sum_{i=2}^{n} x_i}{n-1}\right)^{\frac{1}{4}} \\
&f_2 = g\left(1 - \left(\tfrac{f_1}{g}\right)^2\right) \\
&0 \leq x_i \leq 1 \qquad i = 1, \ldots, 10
\end{aligned}$$

El frente de Pareto verdadero, en este caso, no es convexo.

Apéndice B

Problema DTLZ2

El problema DTLZ2 pertenece al conjunto de pruebas Deb-Thiele-Laumanns-Zitzler (DTLZ) y es utilizado frecuentemente en la literatura para evaluar la escalabilidad de los algoritmos de optimización multiobjetivo.

El frente de Pareto de este problema es la sección de una esfera ubicada en el cuadrante positivo del espacio.

Es una función

$$\vec{f} : \mathbb{R}^{12} \to \mathbb{R}^k$$

Donde

$$\begin{aligned}
&g = \textstyle\sum_{i=k}^{12} (x_i - 0.5)^2 \\
&f_1 = (1+g)\cos(x_1 \tfrac{\pi}{2}) \ldots \cos(x_{k-1} \tfrac{\pi}{2}) \\
&f_2 = (1+g)\cos(x_1 \tfrac{\pi}{2}) \ldots \operatorname{sen}(x_{k-1} \tfrac{\pi}{2}) \\
&\vdots \\
&f_k = (1+g)\operatorname{sen}(x_1 \tfrac{\pi}{2})
\end{aligned}$$

$$0 \leq x_i \leq 1 \qquad i = 1, \ldots, 12$$

Apéndice C

Gráficas de resultados

En este apéndice se presentan las gráficas de las soluciones generadas por el algoritmo cultural y NSGA-II en los problemas donde el algoritmo cultural superó a NSGA-II.

Para realizar las gráficas se realizaron 11 ejecuciones del algoritmo cultural y NSGA-II en cada uno de los problemas de prueba. Los parámetros utilizados por ambos algoritmos son los mismos que los utilizados en la sección 6.2.2.

El algoritmo cultural se detuvo en la generación donde el hipervolumen alcanzó o superó el 99 % de su valor máximo. Se permitió que NSGA-II se ejecutara hasta la generación donde el número de evaluaciones de la función objetivo fuera igual o superior al número de evaluaciones realizadas por el algoritmo cultural.

Las gráficas representan las soluciones correspondientes a la mediana del hipervolumen en las 11 ejecuciones de cada algoritmo.

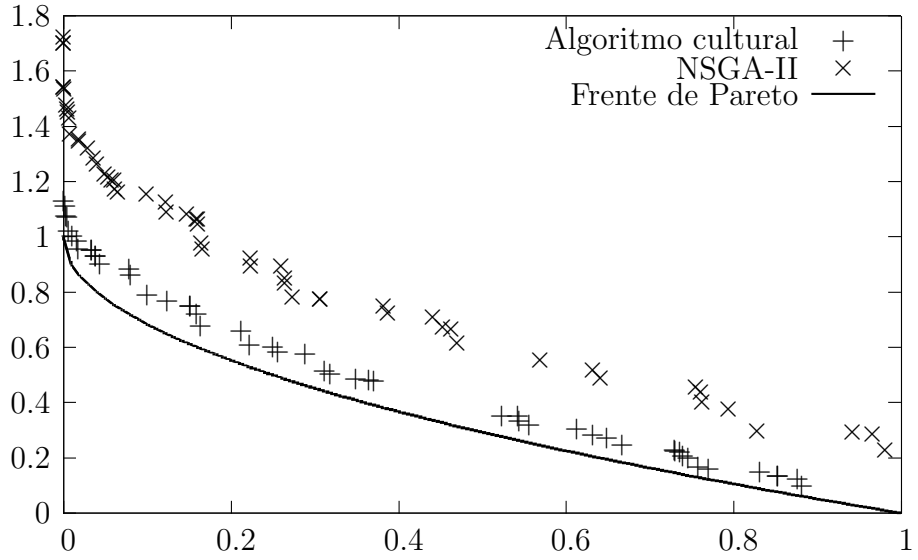

Figura C.1: Soluciones generadas por el algoritmo cultural y NSGA-II para el problema ZDT1.

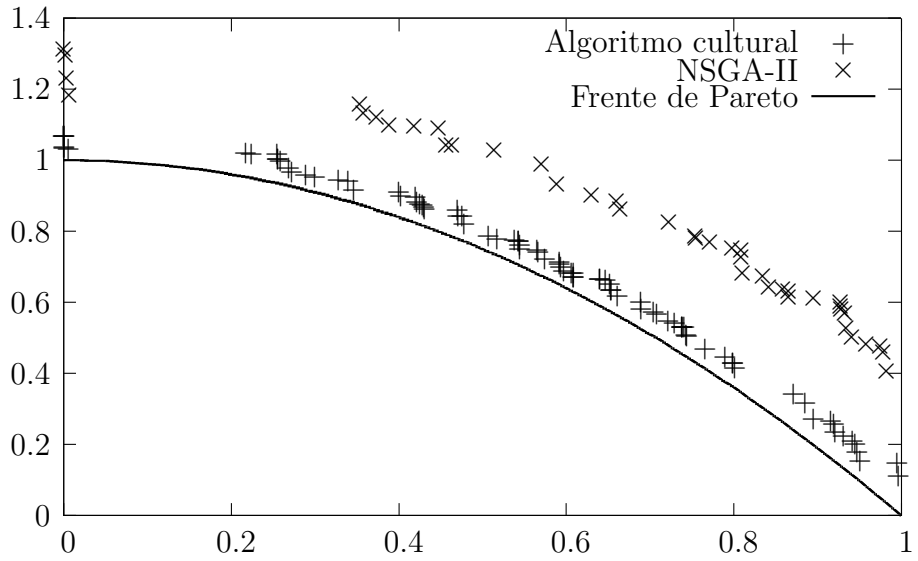

Figura C.2: Soluciones generadas por el algoritmo cultural y NSGA-II para el problema ZDT2.

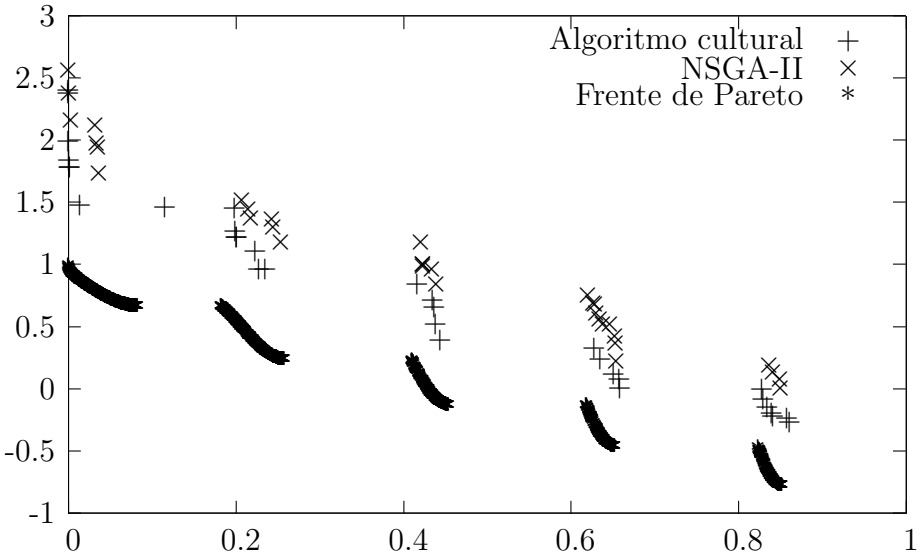

Figura C.3: Soluciones generadas por el algoritmo cultural y NSGA-II para el problema ZDT3.

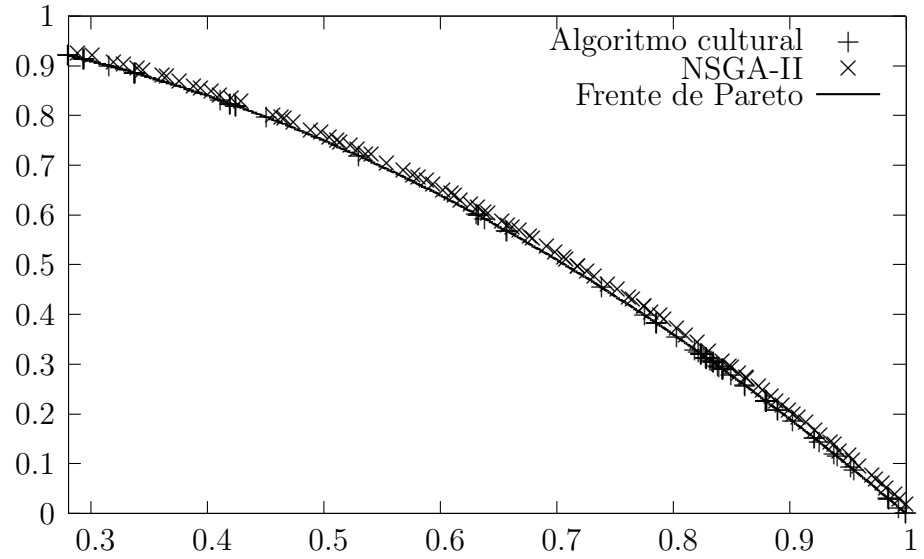

Figura C.4: Soluciones generadas por el algoritmo cultural y NSGA-II para el problema ZDT6.

Apéndice D

Indicadores de desempeño

Las gráficas mostradas en este apéndice corresponden a los valores de los indicadores de desempeño para el algoritmo cultural y NSGA-II en los problemas de prueba.

Los datos utilizados para realizar las gráficas son los presentados en la sección 6.2.2. Los valores se muestran en forma de diagrama de caja.

Los extremos inferior y superior de los rectángulos o cajas delimitan el primer y tercer cuartiles del conjunto de valores del indicador en las 100 ejecuciones de cada algoritmo. La línea que corta a cada caja corresponde a la mediana de los valores del indicador. Los extremos de las líneas delgadas son el mínimo y máximo.

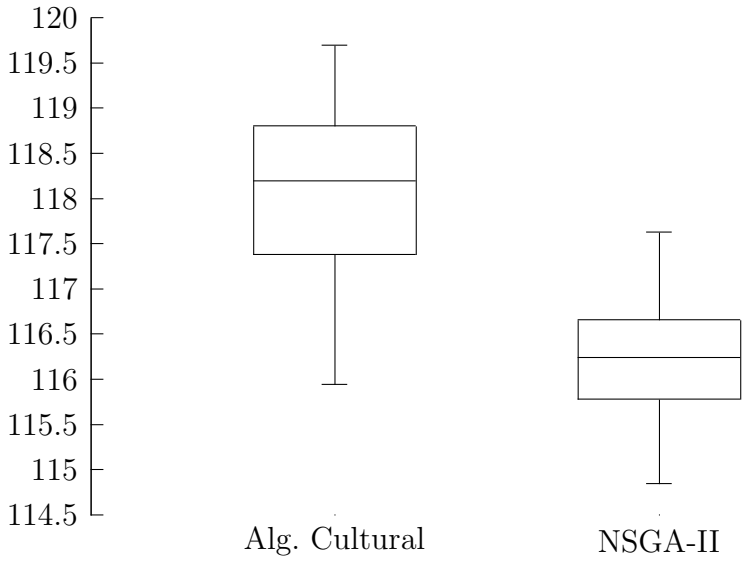

Figura D.1: Resultados del hipervolumen en el problema ZDT1.

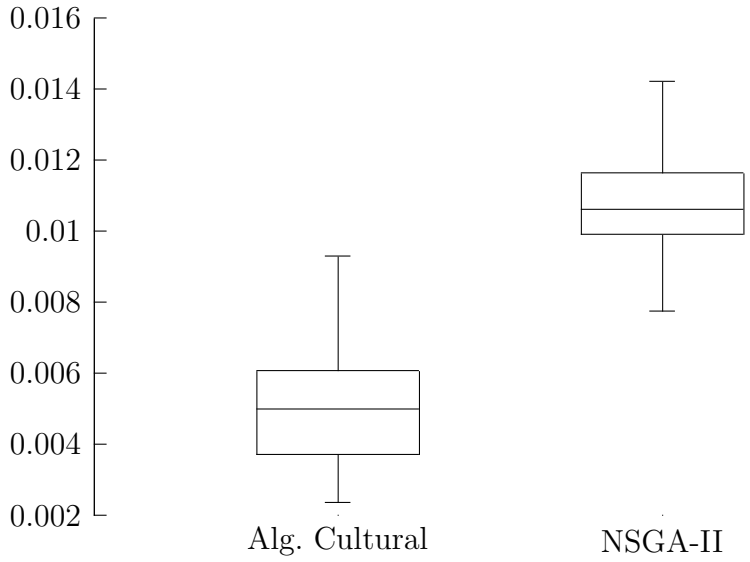

Figura D.2: Resultados de la distancia generacional invertida en el problema ZDT1.

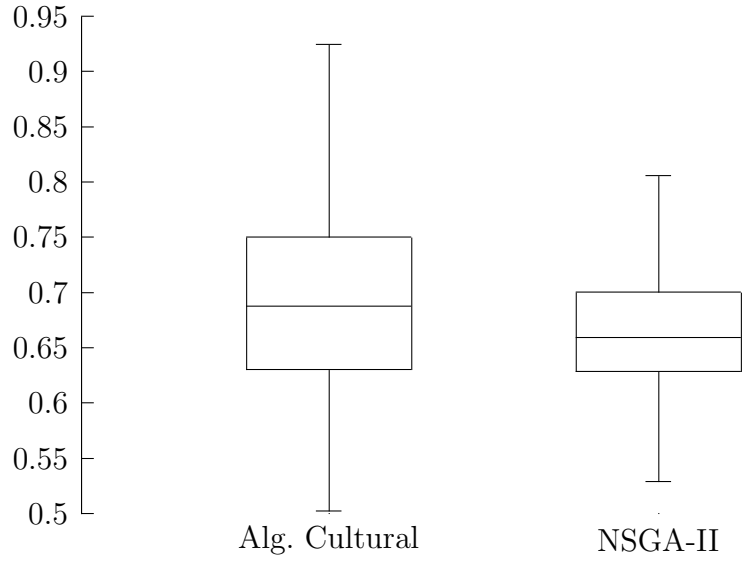

Figura D.3: Resultados del indicador de distribución en el problema ZDT1.

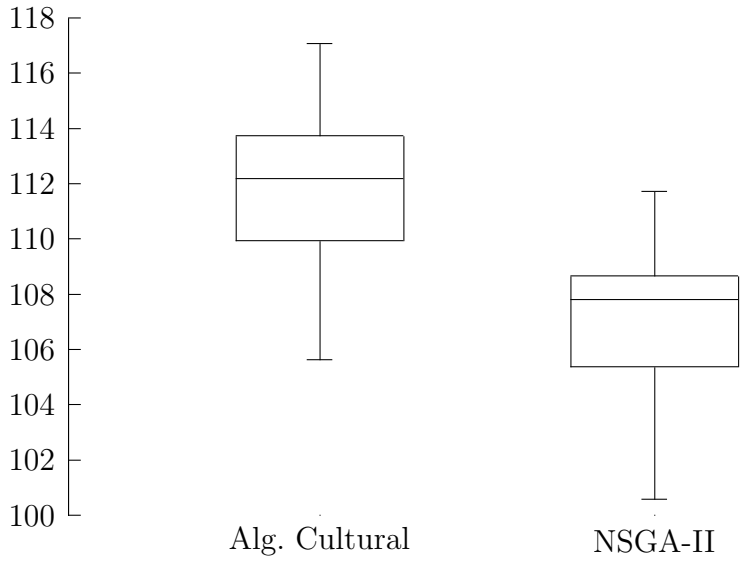

Figura D.4: Resultados del hipervolumen en el problema ZDT2.

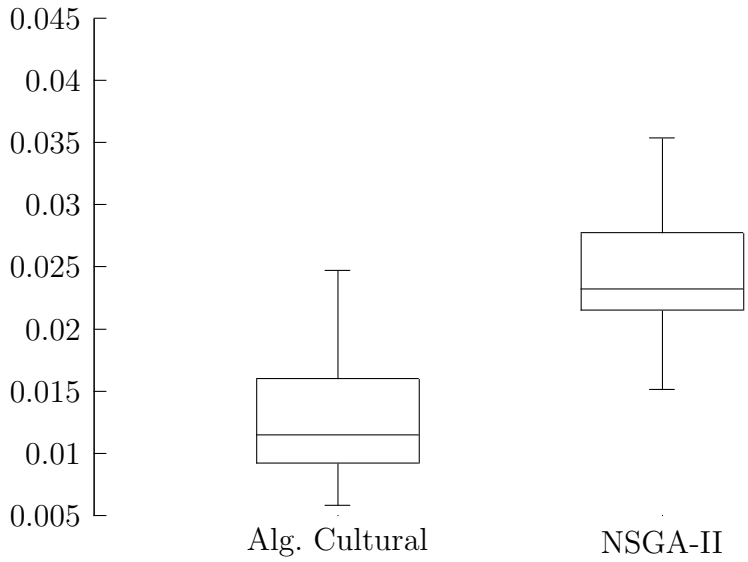

Figura D.5: Resultados de la distancia generacional invertida en el problema ZDT2.

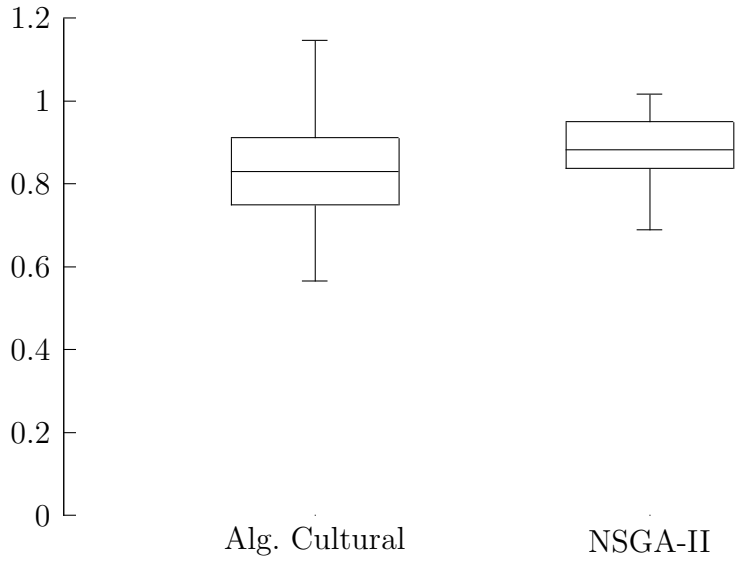

Figura D.6: Resultados del indicador de distribución en el problema ZDT2.

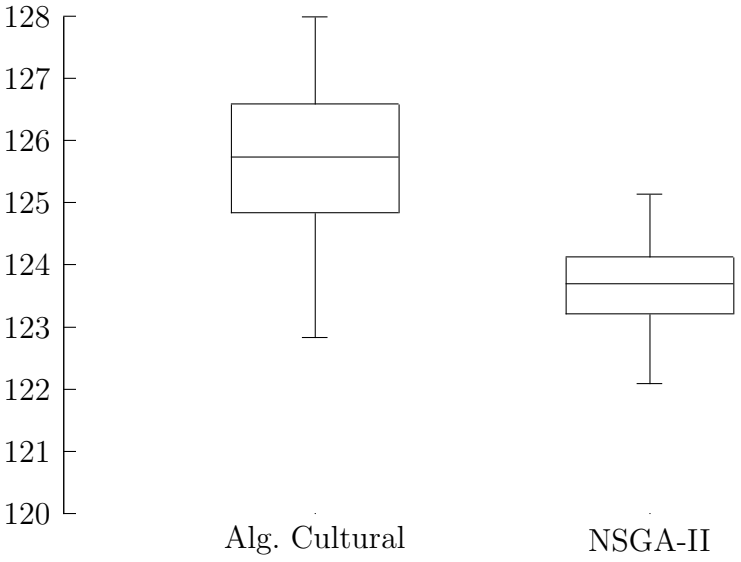

Figura D.7: Resultados del hipervolumen en el problema ZDT3.

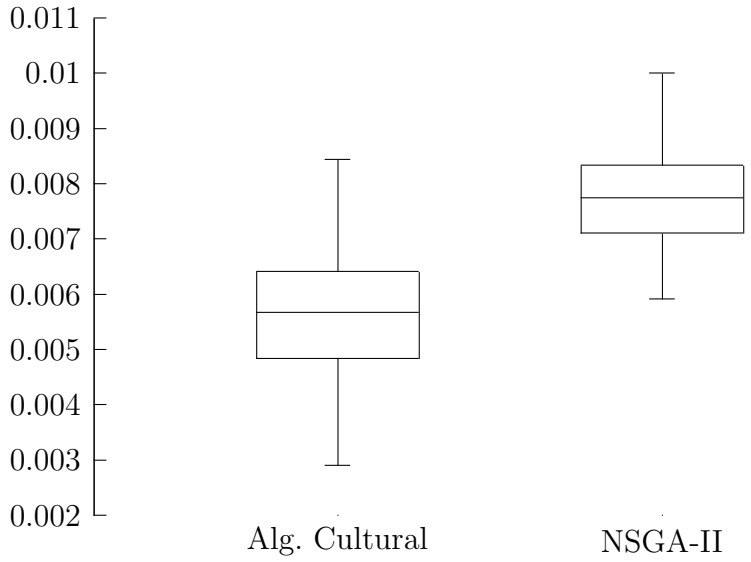

Figura D.8: Resultados de la distancia generacional invertida en el problema ZDT3.

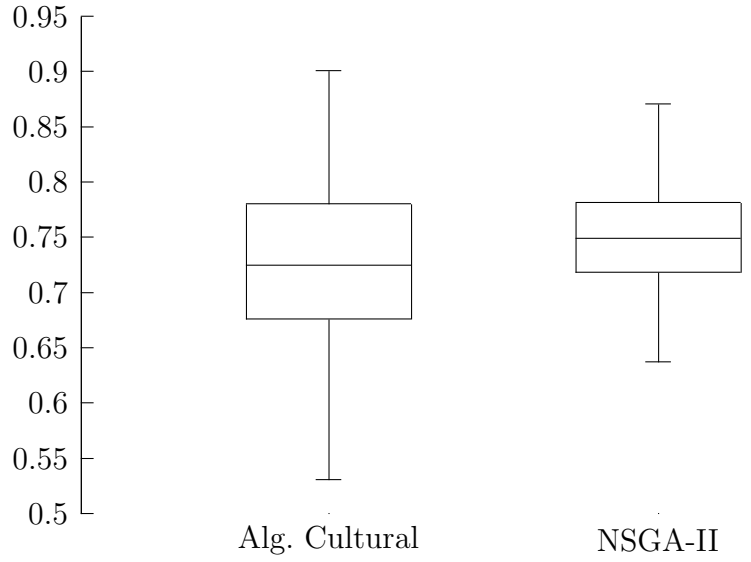

Figura D.9: Resultados del indicador de distribución en el problema ZDT3.

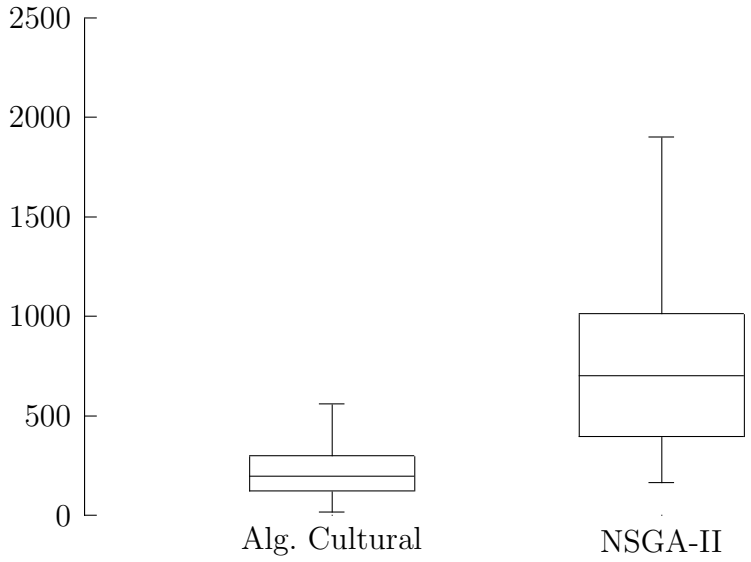

Figura D.10: Resultados del hipervolumen en el problema ZDT4.

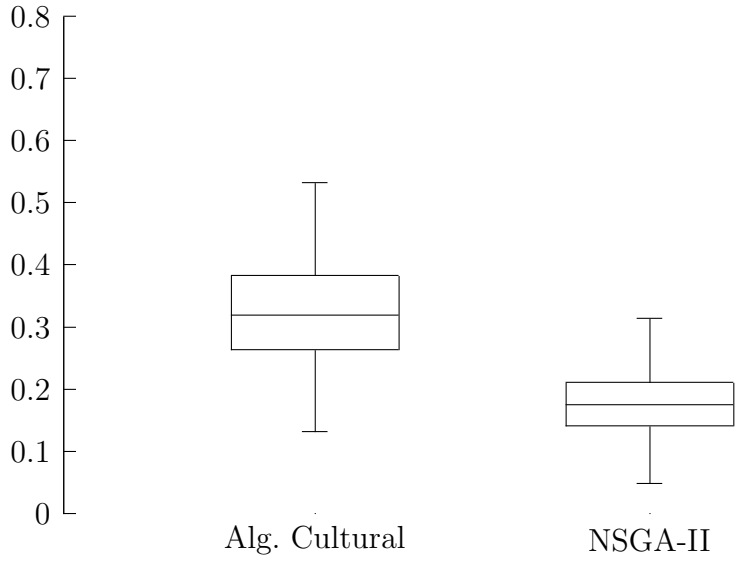

Figura D.11: Resultados de la distancia generacional invertida en el problema ZDT4.

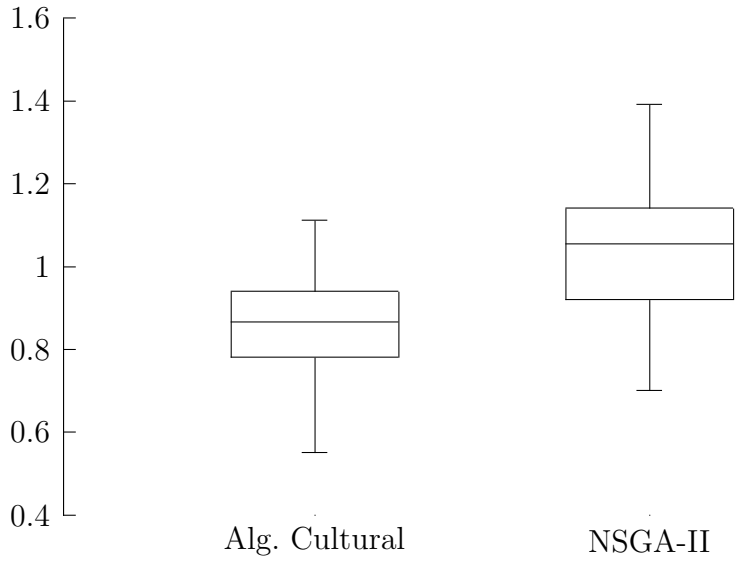

Figura D.12: Resultados del indicador de distribución en el problema ZDT4.

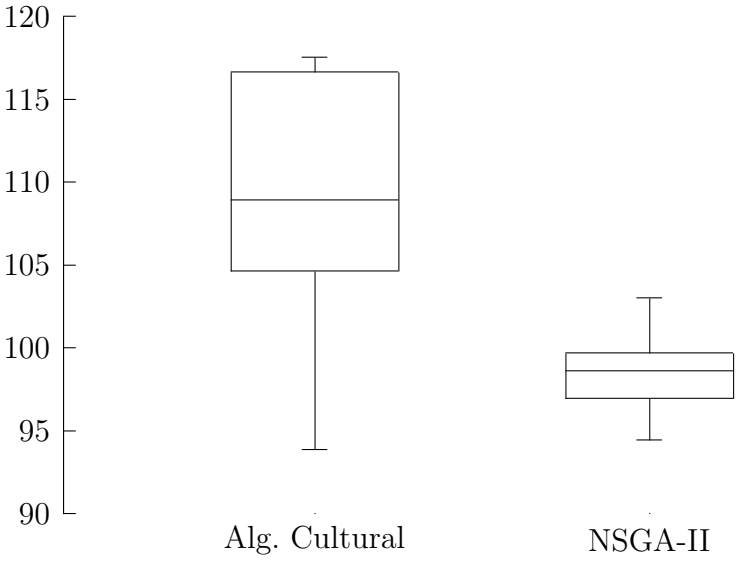

Figura D.13: Resultados del hipervolumen en el problema ZDT6.

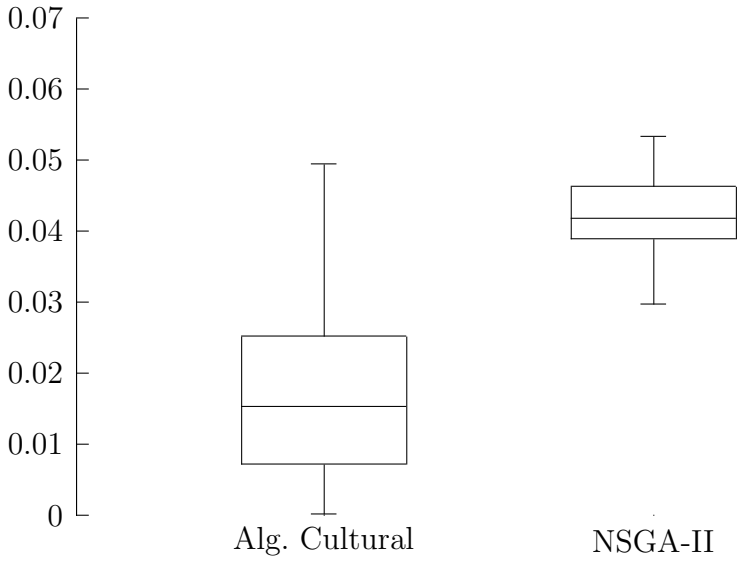

Figura D.14: Resultados de la distancia generacional invertida en el problema ZDT6.

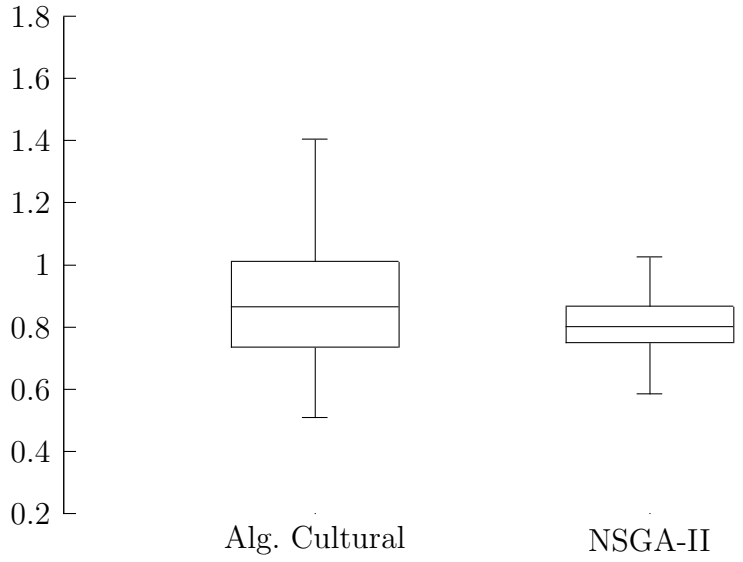

Figura D.15: Resultados del indicador de distribución en el problema ZDT6.

Bibliografía

[1] Robert G. Reynolds. An introduction to cultural algorithms. In A. V. Sebald and L. J. Fogel, editors, *Third Annual Conference on Evolutionary Programming*, pages 131–139, River Edge, New Jersey, 1994. World Scientific.

[2] Tom Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Computer Science Department, Stanford University, 1978.

[3] Ricardo Landa Becerra. Algoritmos culturales aplicados a optimización con restricciones y optimización multiobjetivo. Master's thesis, Departamento de Ingeniería Eléctrica, Sección Computación, CINVESTAV IPN, 2002.

[4] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. *A Tutorial on Evolutionary Multiobjective Optimization*. Swiss Federal Institute of Technology (ETH) Zurich.

[5] C. Best, Xiangdong Che, R.G. Reynolds, and Dapeng Liu. Multi-objective cultural algorithms. In *2010 IEEE Congress on Evolutionary Computation (CEC '2010)*, pages 1 –9, july 2010.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182 –197, apr 2002.

[7] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67 –82, apr 1997.

[8] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer New York, 2006.

[9] George B. Dantzig and Mukund N. Thapa. *Linear Programming 1: Introduction*. Springer New York, 1997.

[10] M.T. Jensen. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503 – 515, oct. 2003.

[11] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[12] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[13] Gary B. Lamont Carlos A. Coello Coello and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Second Edition)*. Kluwer Academic Publishers, New York, March 2002.

[14] D.B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3 –14, jan 1994.

[15] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA, 1966.

[16] David B. Fogel. An analysis of evolutionary programming. In *Proc. of the First Annual Conference on Evolutionary Programming*, pages 43–51, 1992.

[17] I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.

[18] John H. Holland. Concerning efficient adaptive systems. In M. C. Yovitis, G. T. Jacobi, and G. D. Goldstein, editors, *Self-Organizing Systems*, pages 215–230, Washington, D.C., 1962. Spartan Books.

[19] Gregory J. E. Rawlins. *Foundations of genetic algorithms*. Morgan Kaufmann, 1991.

[20] Peter J. Richerson and Robert Boyd. *Not by genes alone: how culture transformed human evolution*. The University of Chicago Press, 2005.

[21] A. Lara, G. Sanchez, C.A. Coello Coello, and O. Schutze. Hcs: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):112 –132, feb. 2010.

[22] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 119 –124, may 1996.

[23] Andrzej Jaszkiewicz. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the pareto memetic algorithm. *Annals of Operations Research*, 131:135–158, 2004.

[24] J.D. Knowles and D.W. Corne. M-paes: a memetic algorithm for multiobjective optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 325 –332 vol.1, 2000.

[25] J.D. Knowles and D.W. Corne. A comparison of diverse approaches to memetic multiobjective combinatorial optimization. In *Proc. of 2000 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 103–108, 2000.

[26] Tadahiko Murata, Shiori Kaige, and Hisao Ishibuchi. Generalization of dominance relation-based replacement rules for memetic emo algorithms. In Erick Cantú-Paz, James Foster, Kalyanmoy Deb, Lawrence Davis, Rajkumar Roy, Una-May O'Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitch Potter, Alan Schultz, Kathryn Dowsland, Natasha Jonoska, and Julian Miller, editors, *Genetic and Evolutionary Computation - GECCO 2003*, volume 2723 of *Lecture Notes in Computer Science*, pages 202–202. Springer Berlin / Heidelberg, 2003.

[27] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[28] K. Ikeda, H. Kita, and S. Kobayashi. Failure of pareto-based moeas: does non-dominated really mean near to optimal? In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 957 –962 vol. 2, 2001.

[29] M. Brown and R. E. Smith. Directed multi-objective optimisation. *International Journal of Computers, Systems and Signals*, 6(1):3–17, 2005.

[30] Oliver Schütze, Marco Laumanns, Emilia Tantar, Carlos A. Coello Coello, and El-ghazali Talbi. Convergence of stochastic search algorithms to gap-free pareto front approximations. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 892–901, New York, NY, USA, 2007. ACM.

[31] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9:3–12, 2005. 10.1007/s00500-003-0328-5.

[32] David J. Powell, Siu Shing Tong, and Michael M. Skolnick. EnGENEous domain independent machine learning for design optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 151–159, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[33] David J. Powell, Michael M. Skolnick, and Siu Shing Tong. Interdigitation : Hybrid technique for engineering design optimization employing genetic algorithms, expert systems, and numerical optimization. *Handbook of Genetic Algorithms*, pages 312–331, 1991.

[34] Sushil Louis and Gregory Rawlins. Designer genetic algorithms: Genetic algorithms in structure design. In Richard K. Belew and Lashon B. Booker, editors, *Fourth International Conference on Genetic Algorithms*, pages 53–60. Morgan Kauffman Publishers, jul 1991.

[35] Sushil Louis. *Genetic Algorithms as a Computational Tool for Design.* PhD thesis, Department of Computer Science, Indiana University, aug 1993.

[36] Sushil J. Louis and Fang Zhao. Domain knowledge for genetic algorithms. *International Journal of Expert Systems*, 8(3):195–211, 1995.

[37] Robert G. Reynolds, Zbigniew Michalewicz, and Michael J. Cavaretta. Using cultural algorithms for constraint handling in GENOCOP. In John R. McDonnell, Robert G. Reynolds, and David B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 298–305, Cambridge, Massachussetts, 1995. MIT Press.

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

[39] Chan-Jin Chung and Robert G. Reynolds. CAEP: An evolution-based tool for real-valued function optimization using cultural algorithms. *Journal on Artitificial Intelligence Tools*, 7(3):239–292, 1998.

[40] Chan-Jin Chung. *Knowledge-based approaches to self-adaptation in cultural algorithms*. PhD thesis, Wayne State University, United States - Michigan, 1997.

[41] Xidong Jin and R.G. Reynolds. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach. In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation (CEC '99).*, volume 3, pages 3 vol. (xxxvii+2348), 1999.

[42] Saleh M. Saleem. *Knowledge-Based Solution to Dynamic Optimization Problems using Cultural Algorithms*. PhD thesis, Wayne State University, Detroit, Michigan, 2001.

[43] Carlos A. Coello Coello and Ricardo Landa Becerra. Adding knowledge and efficient data structures to evolutionary programming: A cultural algorithm for constrained optimization. In Erick Cantú-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 201–209, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[44] Carlos A. Coello Coello and Ricardo Landa Becerra. Efficient evolutionary optimization through the use of a cultural algorithm. *Engineering Optimization*, 36(2):219–236, April 2004.

[45] R. Iacoban, R.G. Reynolds, and J. Brewster. Cultural swarms: modeling the impact of culture on social interaction and problem solving. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 205 – 211, 2003.

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

[47] Ricardo Landa Becerra. *Use of Domain Information to Improve the Performance of an Evolutionary Algorithm*. PhD thesis, Departamento de Ingeniería Eléctrica, Sección Computación, CINVESTAV IPN, 2007.

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

[49] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517, September 1975.

[50] Donald E Knuth. *Art of Computer Programming, Volume 1: Fundamental Algorithms (3rd Edition)*. Addison-Wesley Professional, 1997.

[51] Andrew W. Moore. *Efficient Memory-based Learning for Robot Control*. PhD thesis, University of Cambridge, 1991.

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

[53] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117 – 132, april 2003.

[54] M.P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, 1998.

[55] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms a comparative case study. In Agoston Eiben, Thomas Bck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer Berlin / Heidelberg, 1998.

[56] M. Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In Carlos Fonseca, Peter Fleming, Eckart Zitzler, Lothar Thiele, and Kalyanmoy Deb, editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 74–74. Springer Berlin / Heidelberg, 2003.

[57] D.A. Van Veldhuizen and G.B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, 1998.

[58] D.A. Van Veldhuizen and G.B. Lamont. On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 204 –211 vol.1, 2000.

[59] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257 –271, nov 1999.

[60] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC '02.*, volume 1, pages 825 –830, may 2002.