Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Uso de Algoritmos Evolutivos Multi-Objetivo para Resolver Problemas con un Gran Número de Variables de Decisión

Tesis que presenta

Luis Miguel Antonio

para obtener el Grado de

Maestro en Ciencias en Computación

Director de la Tesis

Dr. Carlos Artemio Coello Coello

México,D.F. **Septiembre 2013**

Resumen

Muchos problemas multiobjetivo del mundo real tienen cientos e incluso miles de variables de decisión, por lo cual la escalabilidad es un tema de gran importancia. Esto, sin embargo, contrasta con los modelos evolutivos existentes para optimización multiobjetivo los cuales suelen validarse con problemas de prueba con un número relativamente bajo de variables de decisión (normalmente no más de 30). En la actualidad la investigación existente en el área de algoritmos evolutivos multiobjetivo suele enfocar su atención a la escalabilidad en el espacio de las funciones objetivo (esta área se conoce como "many-objective optimization"), sin poner mucha atención a la escalabilidad en el espacio de las variables de decisión.

En esta tesis se propone un esquema que permite que un algoritmo evolutivo multiobjetivo sea capaz de lidiar con problemas de de alta dimensionalidad en el espacio de las variables de decisión. Para validar el algoritmo propuesto, se usó un conjunto de prueba que es escalable en el número de variables de decisión y se compararon resultados con respecto a los obtenidos por dos algoritmos representativos del estado del arte en al área (GDE3 y NSGA-II) utilizando un número de variables de decisión que va de las 200 hasta las 5000. Los resultados obtenidos muestran claramente que la propuesta presentada es tanto efectiva como eficiente para resolver problemas de optimización multiobjetivo a gran escala en el espacio de las variables de decisión.

Abstract

Many real-world multiobjective optimization problems have hundreds and even thousands of decision variables, which turns scalability into a very important topic. This, however, contrasts with the currently available evolutionary models for multiobjective optimization, which are normally validated with test problems having a relatively low number of decision variables (normally, no more than 30). Nowadays, research on multiobjective evolutionary algorithms has focused on scalability in objective function space (this area is known today as "many-objective optimization"), disregarding scalability in decision variable space.

In this thesis, we propose a scheme that allows a multi-objective evolutionary algorithm to be able to deal with problems having a large dimensionality in decision variable space. In order to validate the proposed approach, we adopted a set of test problems that are scalable in decision variable space, and we compared results with respect to those obtained by two algorithms which are representative of the state-of-the-art in the area (GDE3 and NSGA-II) using a number of decision variables that goes from 200 up to 5000. The obtained results clearly show that our proposed approach is both effective and efficient for solving large scale multiobjective optimization problems.

Agradecimientos

La realización de esta tesis marca la culminación de una etapa de mi vida la cual no habría sido posible sin la presencia, cariño, apoyo y guía de muchas personas. Aunque no puedo mencionar el nombre de cada uno de ellos, les doy mis más sinceros agradecimientos.

Me gustaría agradecer a mis padres por el amor y apoyo incondicional que me han brindado durante toda mi vida. También a todos mis seres queridos, por todo lo que han compartido conmigo y por ser el motor que me impulsa cada día. De igual forma quiero agradecer a todos los que se han cruzado en mi camino y me han brindado su cariño.

También quiero agradecer a todas aquellas personas que en algún momento han sido mis guías, pues sin sus enseñanzas y consejos ninguno de mis logros hubiese sido posible. A todos los Drs. que me impartieron clases durante esta etapa de mi vida por compartir sus conocimientos, en especial al Dr. Debrup Chakraborty.

Quiero agradecer a mi asesor el Dr. Carlos A. Coello Coello por haberme dado el honor de trabajar bajo su guía, por transmitirme su pasión por la investigación y con ello sembrar en mi un gran deseo de aprender, por todas sus enseñanzas, por su apoyo y por extralimitar sus funciones de guía compartiendo importantes consejos y experiencias de vida.

Finalmente quiero agradecer al CINVESTAV por permitirme formar parte de esta gran institución y al CONACyT por brindarnos un apoyo económico, permitiendo que muchas personas podamos concluir los estudios de maestría.

El tema de esta tesis se derivó del proyecto CONACyT titulado “Escalabilidad y Nuevos Esquemas Híbridos en Optimización Evolutiva Multiobjetivo” (Ref. 103570), cuyo investigador responsable es el Dr. Carlos A. Coello Coello.

Índice general

Índice de figuras

Lista de Algoritmos

Índice de tablas

Capítulo 1

Introducción

En el mundo real, a menudo se presentan problemas de optimización numérica y combinatoria. Ejemplos comunes pueden encontrarse en áreas tales como: finanzas, predicción, robótica y teoría de juegos, entre otras. Estos problemas de optimización del mundo real muchas de las veces requieren de optimizar más de una función objetivo al mismo tiempo, y dichas funciones suelen estar en conflicto entre sí. Dichos problemas se denominan Problemas de Optimización Multiobjetivo (POMs) y su solución consiste en un conjunto de soluciones que representen los mejores compromisos entre las funciones objetivo que están siendo optimizadas. Estas soluciones se denominan *conjunto de óptimos de Pareto*, y su respectiva proyección en el espacio de los objetivos se denomina *frente de Pareto*.

Los POMs son comúnmente resueltos por técnicas clásicas de programación matemática como el método del criterio global, la programación por metas, el método de satisfacción de metas, la optimización min-max y combinaciones lineales de pesos, entre otros [2]. A pesar de la considerable cantidad de técnicas existentes para atacar este tipo de problemas, el hecho de que los problemas del mundo real tienden a ser no lineales y con funciones objetivo las cuales resultan muy costosas para evaluar, hacen que el uso de estas técnicas no nos lleven a buenos resultados o que éstas no sean capaces de ofrecer una solución adecuada en un tiempo razonable, lo que conlleva a buscar otras alternativas, tales como las metaheurísticas, posibles. Entre las metaheurísticas disponibles en la actualidad, los algoritmos evolutivos (AEs) son uno de los métodos más utilizados en la literatura especializada. Los algoritmos evolutivos multiobjetivo (AEMO) tienen la ventaja de generar varios elementos del conjunto de óptimos de Pareto en una sola ejecución, en contraste con las técnicas de programación matemática que usualmente solo producen una solución por cada ejecución. Ademas, los AEMO son menos sensibles ante aspectos como la forma y continuidad del frente de Pareto, mientras que dichas características usualmente causan serias dificultades a las técnicas de programación matemática a la hora de resolver estos problemas.

La motivación de este trabajo es que existen problemas multiobjetivo del mundo real que tienen cientos e incluso miles de variables de decisión, por lo cual la escalabilidad es un tema de importancia en los algoritmos de optimización. Esto, sin embargo,

contrasta con los modelos evolutivos existentes para optimización multiobjetivo, los cuales suelen validar su desempeño con problemas de prueba con un número relativamente bajo de variables de decisión (usualmente no más de 30 variables). De hecho, la escalabilidad, en cuanto al número de variables de decisión de un POM, es un tema que ha sido escasamente estudiado en el contexto de optimización multiobjetivo utilizando metaheurísticas.

Hasta donde sabemos, no existe ningún AEMO diseñado específicamente con el propósito de lidiar con POMs que tengan un gran número de variables de decisión. Esto quizás, motivado por el hecho de que la mayoría de los investigadores suponen que los AEMO existentes deberían de funcionar correctamente con un gran número de variables de decisión. Sin embargo, existe evidencia empírica que indica que la mayoría de las metaheurísticas existentes, tienen una reducción significativa en su desempeño confirme incrementa el número de variables de decisión de los problemas [3, 4]. En esta tesis se propone un esquema basado en coevolución cooperativa, el cual permite lidiar con POMs que tengan un gran número de variables de decisión. La razón del uso de tal esquema es la existencia de evidencia que indica que los esquemas basados en coevolución cooperativa han demostrado ser efectivos para resolver problemas de optimización global a gran escala [5].

1.1. Objetivos de la tesis

En esta sección se describen los objetivos de esta tesis, abarcando tanto el general como los particulares.

Objetivo General

Proponer un algoritmo evolutivo multiobjetivo que tenga un mejor desempeño que los algoritmos del estado del arte al resolver problemas de optimización multiobjetivo con una alta dimensionalidad en el espacio de las variables de decisión.

Objetivos Particulares

- Diseñar un modelo algorítmico para atacar de mejor manera problemas de optimización multiobjetivo con una alta dimensionalidad en el espacio de las variables mediante algoritmos evolutivos.

- Realizar una implementación del modelo propuesto.

- Realizar comparativas de desempeño del algoritmo propuesto con respecto a algoritmos representativos del estado del arte con el fin de observar las fortalezas y debilidades del algoritmo propuesto con respecto a los modelos existentes.

1.2. Organización de la tesis

El resto de esta tesis está organizado de la siguiente forma. Los cuatro primeros capítulos describen los antecedentes y los conceptos básicos requeridos para comprender el trabajo que aquí se presenta. Los últimos tres capítulos están dedicados a describir el esquema propuesto, su implementación y los resultados obtenidos con el mismo.

El capítulo 2 presenta una breve introducción a la optimización multiobjetivo y a la computación evolutiva. Al inicio del capítulo se proporcionan los conceptos básicos de optimización multiobjetivo. Posteriormente se describen la computación evolutiva y los algoritmos evolutivos más representativos del estado del arte y sus componentes principales. Asimismo se presentan conceptos y terminología que se necesitan en los capítulos siguientes.

En el capítulo 3 se presenta el trabajo existente referente a la escalabilidad del número de variables de decisión de un POM en el contexto de optimización multiobjetivo utilizando metaheurísticas. Se presentan también los trabajos de optimización global a gran escala más representativos del estado del arte. Se describen los mecanismos auxiliares, las técnicas desarrolladas y los modelos existentes para lidiar con el problema.

El capítulo 4 describe la coevolución dentro de la computación evolutiva. Se muestran los tipos de coevolución existentes y la forma en que dichos esquemas se han aplicado dentro de la computación evolutiva, así como los modelos desarrollados específicamente para la optimización multiobjetivo.

En el capítulo 5 se describe nuestra propuesta. También se mencionan cuáles fueron los criterios que se consideraron para el diseño y se describe la forma de atacar el problema de la alta dimensionalidad. Asimismo, se detalla el funcionamiento general del algoritmo propuesto y de las técnicas en él utilizadas. Finalmente, se habla de los detalles de la implementación.

El capítulo 6 se dedica a presentar los resultados obtenidos por el algoritmo. Se comienza con una descripción de los problemas de prueba y de las métricas utilizadas. El capítulo finaliza con la evaluación de los resultados realizando un análisis del desempeño de los algoritmos comparados para cada problema de prueba.

En el capítulo 7 se presentan las conclusiones y el trabajo a realizarse a futuro. También se analizan brevemente los puntos fuertes y débiles del algoritmo propuesto. Para terminar, se describen algunas líneas de trabajo futuro que resultan de nuestro interés en virtud de los resultados reportados en esta tesis.

Capítulo 2

Optimización multiobjetivo

En este capítulo se da una introducción a la optimización multiobjetivo. Así mismo se propone el planteamiento del problema general de optimización. Se introducen también conceptos básicos de Computación Evolutiva incluyendo una descripción breve de sus paradigmas principales.

2.1. Problemas de optimización multiobjetivo

Un problema de optimización multiobjetivo (POM) consiste en encontrar un vector de variables de decisión que satisfagan un cierto conjunto de restricciones y optimice un conjunto de funciones objetivo [3]. Estas funciones forman una descripción matemática de los criterios de desempeño que suelen estar en conflicto unos con otros y que se suelen medir en unidades diferentes. El término "optimizar" en este caso significa encontrar soluciones que den valores a las funciones objetivo que sean aceptables para los tomadores de decisiones.

Un POM se define formalmente de la siguiente manera:

Encontrar un vector

$$\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T \tag{2.1}$$

El cual satisface las m restricciones de desigualdad

$$g_i(\vec{x}) \leq 0 \qquad\qquad i = 1, 2, \ldots, m \tag{2.2}$$

las p restricciones de igualdad

$$h_i(\vec{x}) = 0 \qquad\qquad i = 1, 2, \ldots, p \tag{2.3}$$

y optimice la función vectorial

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{2.4}$$

donde

$$\vec{x} = [x_1, x_2, \ldots, x_n]^T \tag{2.5}$$

es el vector de variables de decisión que definen el problema.

Al tener más de una función objetivo, la noción de óptimo cambia, ya que no se busca obtener una solución única, como se hace en la optimización mono-objetivo, sino que se aspira a obtener los mejores compromisos posibles entre las funciones objetivo, presentándose entonces más de una solución. La noción de "óptimo" que se suele adoptar en optimización multiobjetivo es la propuesta originalmente por Francis Ysidro Edgeworth y que más tarde generalizara Vilfredo Pareto (en 1896), la cual se conoce como óptimo de Pareto [6].

Definición 1 (Optimalidad de Pareto): Decimos que un vector de variables de decisión $\vec{x}^* \in \Omega$ (donde Ω es la zona factible) es un óptimo de Pareto si (suponiendo que se minimizan todas las funciones objetivo):

$$\forall \vec{x} \in \Omega \wedge \forall i \in \{1, \ldots, k\}, f_i(\vec{x}) = f_i(\vec{x}^*) \vee \nexists i \in \{1, \ldots, k\} : f_i(\vec{x}) < f_i(\vec{x}^*) \tag{2.6}$$

Definición 2 (Dominancia de Pareto): Se dice que un vector $\vec{u} = [u_1, \ldots, u_k]^T$ domina a otro vector $\vec{v} = [v_1, \ldots, v_k]^T$ (denotado por $\vec{u} \preceq \vec{v}$) si y sólo si $\vec{u}$ es parcialmente menor que $\vec{v}$, es decir:

$$\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i \tag{2.7}$$

Definición 3 (Conjunto de óptimos de Pareto): Para un problema multiobjetivo dado $\vec{f}(\vec{x})$, el conjunto de óptimos de Pareto P^* se define como:

$$P^* := \{\vec{x} \in \Omega | \nexists \vec{x'} \in \Omega, \vec{f}(\vec{x'}) \preceq \vec{f}(\vec{x})\} \tag{2.8}$$

Definición 4 (Frente de Pareto): Para un problema multiobjetivo dado $\vec{f}(\vec{x})$ y su conjunto de óptimos de Pareto P^*, el frente de Pareto se define como:

$$PF^* := \{\vec{f}(\vec{x}), \vec{x} \in P^*\} \tag{2.9}$$

La obtención del frente de Pareto de un problema de optimización multiobjetivo es la meta principal de la optimización multiobjetivo. Es deseable que la aproximación obtenida esté tan cerca como sea posible del verdadero frente de Pareto y que se distribuya a lo largo del frente de Pareto de la manera más uniforme posible [6].

El problema a tratar en este trabajo se enfoca en proponer un modelo el cual pueda lidiar de mejor manera con problemas de optimización multiobjetivo de alta dimensionalidad en el espacio de las variables de decisión (o sea, con problemas que tengan mucho más que 30 variables).

2.2. Técnicas tradicionales para la optimización multiobjetivo

A continuación se muestran algunas de las técnicas tradicionales más representativas que existen para resolver problemas multiobjetivo. Éstas se clasifican en tres categorías principales [6]:

- Técnicas *a priori*: En éstas, las especificaciones de preferencia son proporcionadas antes del proceso de búsqueda de soluciones. Esto quiere decir que se toman decisiones "antes" de comenzar la búsqueda.

- Técnicas *a posteriori*: En éstas, se producen varias soluciones al problema antes de tomar una decisión. Se produce un conjunto de óptimos de Pareto y entonces el tomador de decisiones selecciona las soluciones que prefiera de entre todas las disponibles.

- Técnicas de articulación progresiva de preferencias: En éstas, la búsqueda y la toma de decisiones se van haciendo a la par mientras ocurre el proceso de optimización. Éstas normalmente operan en tres fases: (1) encontrar una solución no dominada, (2) obtener la reacción del tomador de decisiones con respecto a esta solución, modificando las preferencias de los objetivos de acuerdo a esta reacción, y (3) repetir los dos pasos previos hasta que el tomador de decisiones esté satisfecho o no se puedan obtener más mejoras.

A continuación se describen algunas técnicas representativas de estas tres categorías.

2.2.1. Técnicas *a priori*

Método del criterio global

En esta técnica el objetivo es minimizar una función que define un criterio global el cual mide qué tanto se puede acercar el tomador de decisiones al vector ideal f^0. La forma más usual de esta función es la siguiente:

$$f(x) = \sum_{i=1}^{k} \left(\frac{f_i^0 - f_i(x)}{f_i^0} \right)^p \tag{2.10}$$

donde k es el número de objetivos y p suele tomar como valor 1 ó 2, aunque también puede adoptar otros valores. Obviamente, los valores cambian considerablemente dependiendo del valor de p que se escoja y puede darse el caso de que cualquier p produzca una solución inaceptable.

Programación por metas

En este método, el tomador de decisiones tiene que asignar objetivos o metas que desee alcanzar para cada objetivo. Estos valores se incorporan al problema como restricciones adicionales. La función objetivo trata entonces de minimizar las desviaciones absolutas entre las metas y los objetivos. La forma más simple de este método se muestra a continuación:

$$\min \sum_{i=1}^{k} |f_i(x) - T_i|, \text{sujeto a } x \in \Omega \tag{2.11}$$

donde T_i denota el conjunto de objetivos o metas establecidos por el tomador de decisiones para la i-ésima función objetivo $f_i(x)$ y Ω representa la región factible. El objetivo es minimizar la suma de los valores absolutos de las diferencias entre los valores deseables para cada objetivo y los valores que se han alcanzado hasta el momento.

Método lexicográfico

En este método, los objetivos son jerarquizados en orden de importancia, por el tomador de decisiones (del mejor al peor). La solución óptima x^* se obtiene entonces minimizando las funciones objetivo, comenzando con la de mayor importancia y procediendo de acuerdo al orden de importancia de los objetivos. Haciendo que los subíndices de los objetivos indiquen no sólo el número de la función objetivo, sino también su prioridad, entonces, $f_1(x)$ y $f_k(x)$ denotan la función objetivo más importante y la menos importante, respectivamente. De tal forma, el primer problema es formulado de la siguiente forma:

$$\text{Minimizar } f_1(x) \tag{2.12}$$

sujeto a

$$g_i(\vec{x}) \leq 0; \qquad i = 1, 2, \ldots, m \tag{2.13}$$

y su solución x_1^* y $f_1^* = f_1(x_1^*)$ es obtenida. Posteriormente, el segundo problema se formula como:

$$\text{Minimizar } f_2(x) \tag{2.14}$$

sujeto a

$$g_i(\vec{x}) \leq 0; \qquad i = 1, 2, \ldots, m \tag{2.15}$$

$$f_1(x) = f_1^* \tag{2.16}$$

y la solución de este problema se obtiene como x_2^* y $f_2^* = f_2(x_2^*)$. Este procedimiento se repite hasta que todos los k objetivos han sido considerados. Entonces, el j-ésimo problema esta dado por

$$\text{Minimizar } f_j(x) \tag{2.17}$$

sujeto a

$$g_i(\vec{x}) \leq 0; \qquad i = 1, 2, \ldots, m \tag{2.18}$$

$$f_l(x) = f_l^*, \qquad l = 1, 2, \ldots, j-1 \tag{2.19}$$

2.2.2. Técnicas *a posteriori*

Combinación linear de pesos

En este método se crea una función objetivo la cual es una suma de pesos de los componentes del vector de valores de la función original. El problema a resolver es el siguiente:

$$\min \sum_{i=1}^{k} w_i f_i(x) \tag{2.20}$$

sujeto a

$$x \in \Omega \tag{2.21}$$

donde $w_i \geq 0$ para toda i y es estrictamente positivo para al menos un objetivo. El conjunto de soluciones no dominadas puede ser generado mediante la variación paramétrica de los pesos w_i. Se hace notar que los valores de pesos no necesariamente reflejan la importancia relativa de los objetivos, ya que son sólo factores los cuales, al ser variados, localizan puntos diferentes en el conjunto de óptimos de Pareto.

Método de la restricción ϵ

Este método es una de las técnicas de escalarización mas conocidas para resolver problemas multiobjetivo. La idea de este método es minimizar una de las funciones objetivo (la de mayor preferencia o importancia) mientras que las otras son utilizadas como límites restrictivos para algunos niveles permitidos ϵ_l. Variando los niveles ϵ_l

se pueden obtener las soluciones no dominadas del problema. Para obtener valores adecuados de ϵ_l, se lleva a cabo una optimización mono-objetivo para cada función objetivo en turno. Para poder obtener varias soluciones óptimas de Pareto, se necesita entonces resolver el problema de restricción ϵ (ϵ-*constraint problem*) utilizando diferentes valores de ϵ_l. El esquema general es un proceso de optimización interactivo en el cual el usuario debe proporcionar el rango de la referencia objetivo. Además, debe proporcionar el incremento para las restricciones impuestas por ϵ. Este incremento determina el número de soluciones de Pareto que se han de generar.

2.2.3. Técnicas de articulación de preferencias progresivas

Método de desarrollo probabilístico de compromisos

La principal motivación de este método es poder manejar los riesgos en el desarrollo de los compromisos de los objetivos y al mismo tiempo ser capaces de acomodar las preferencias de los tomaderos de decisiones de una forma progresiva. En este caso, se presupone que el problema de optimización multiobjetivo tiene funciones objetivo probabilísticas y restricciones también probabilísticas. Al principio, se encuentra una solución inicial utilizando una función objetivo sustituta. Después, se forma una función de utilidades multiatributo llevando a un nuevo objetivo sustituto y a una nueva solución. La solución es entonces verificada para ver si es satisfactoria para el tomador de decisiones. El proceso se repite hasta encontrar una solución satisfactoria.

Lo interesante de este método es que el tomador de decisiones jerarquiza objetivos en orden de importancia al principio del proceso, mediante una función de utilidad multiatributo, y después utiliza comparaciones por pares para conciliar estas preferencias con el comportamiento real de los atributos. Esto permite no sólo una participación interactiva del tomador de decisiones sino que además permite obtener conocimiento acerca de los compromisos del problema.

2.3. Algoritmos Evolutivos Multiobjetivo

El uso de algoritmos evolutivos para resolver problemas multiobjetivo se ha popularizado en años recientes [7]. Los algoritmos evolutivos tienen como objetivo principal evolucionar individuos, los cuales, en este caso, representan soluciones a un cierto problema de optimización multiobjetivo. De esta manera, los individuos se irán renovando generación tras generación bajo el principio de supervivencia del más apto, por lo que cada nueva generación debe poseer características mejores que las anteriores, acercándonos con cada iteración a las mejores soluciones posibles al problema.

Antes de hablar de los algoritmos evolutivos multiobjetivo (AEMO), repasaremos algunos conceptos básicos de computación evolutiva.

2.3.1. Computación Evolutiva

La Computación Evolutiva es un área de las ciencias de la computación inspirada en los procesos de la evolución natural. Ésta concibe la evolución como un proceso de optimización que es simulado para resolver problemas. La idea fundamental de la computación evolutiva es evolucionar una población de posibles soluciones a un problema, usando operadores inspirados en la variación genética y en la selección natural (reproducción, mutación, competencia y selección). La evolución opera entonces como un método de búsqueda entre una inmensa cantidad de soluciones posibles. El conjunto de opciones de búsqueda consiste en las posibles secuencias genéticas, y las soluciones deseables son los organismos más aptos para sobrevivir y reproducirse en su ambiente. Las técnicas que conforman la computación evolutiva son procedimientos inspirados en el principio natural de la evolución de las especies, las cuales modelan la evolución como un proceso de optimización.

Conceptos Básicos de Computación evolutiva

En las técnicas de la computación evolutiva existen individuos (entidades) los cuales tienen la capacidad de reproducirse, con una determinada probabilidad. Cada uno de estos individuos representa una posible solución al problema que se desea resolver. Al reproducirse los individuos, se realiza un proceso de exploración en el espacio de búsqueda para mejorar cada vez más las posibles soluciones y con esto acercarnos a la solución óptima del problema. A cada individuo se le asocia una aptitud (habilidad) mediante la cual se evalúa su propia supervivencia en cada generación. Esto se relaciona con el valor de la función objetivo para la correspondiente interpretación de éste en términos de soluciones del problema. En su conjunto, estos individuos representan posibles soluciones al problema en cuestión (a este conjunto se le denomina población), los cuales se renuevan generación tras generación bajo el principio de la supervivencia del más apto. La población cuenta con diversidad entre sus individuos, con lo cual se busca obtener un muestreo representativo del espacio de búsqueda del problema, es decir, contar con un mecanismo de exploración.

Entre los problemas apropiados para resolver con estas técnicas se encuentran aquellos en los que la región factible no se puede caracterizar de manera sencilla, así como aquellos en que no se cumplen condiciones de convexidad o conexidad, o cuando no se cuenta con conocimiento previo de las características del espacio de búsqueda. A continuación se definen algunos conceptos básicos de la computación evolutiva.

Población: es el conjunto de individuos que representan las posibles soluciones a un problema. Estos individuos contienen un cromosoma o genoma cuya codificación es llamada genotipo. La descodificación de dicho cromosoma se denomina fenotipo. Los espacios fenotípicos y genotípicos son distintos, si bien el mapeo del genotipo al fenotipo es único.

Operadores: son aquellos procedimientos que manipulan la información genética de

la población para producir nuevos individuos a partir de la población original. Los operadores genéticos más comunes en los algoritmos evolutivos (AEs) son la cruza y la mutación.

Mutación: es la perturbación sobre uno o más alelos del cromosoma de un individuo. Nos permite alcanzar una región del espacio de búsqueda diferente a la del individuo original, lo que ayuda a que la búsqueda no se estanque en un óptimo local. En los algoritmos genéticos la mutación es considerada un operador secundario, debido a que se aplica con menor frecuencia que la cruza.

Cruza: la cruza tiene la finalidad de heredar la información genética de dos o más padres a la descendencia que éstos crean. En la computación evolutiva, la cruza entre cromosomas se simula intercambiando segmentos de cadenas lineales de longitud fija. Los genes de los padres se combinan para formar las cadenas cromosómicas de sus descendientes, buscando preservar los buenos genes y obtener mejores soluciones al problema. El principio detrás de la cruza es combinar dos o más individuos con características deseadas para producir uno o mas individuos que combinen tales cualidades. La recombinación es entonces el procedimiento del algoritmo que se dedica a la explotación de las zonas más prometedoras en el espacio de búsqueda. Éste es el operador principal de los algoritmos genéticos.

Selección: la selección es el proceso mediante el cual se determina qué individuos pasarán a formar parte de la siguiente generación. Cada uno de los individuos cuenta con un valor de aptitud determinado, el cual indica qué tan bueno es dicho individuo con respecto a los demás. A la determinación de dichos valores se le conoce como evaluación de la función de aptitud y dado que el proceso de selección toma en cuenta esta función, se imita en el algoritmo el principio de supervivencia del más apto.

2.3.2. Principales Paradigmas de la Computación evolutiva

Existen tres paradigmas dentro de la computación evolutiva: los algoritmos genéticos (AGs), la programación evolutiva (PE) y las estrategias evolutivas (EE). Éstos, en su conjunto componen los algoritmos de la computación evolutiva y son llamados, de forma genérica, algoritmos evolutivos (AEs). Dichos algoritmos tienen en común el uso de operadores que simulan la reproducción, la mutación, la competencia, y la selección de individuos dentro de una población. A continuación se da una explicación breve acerca del funcionamiento de cada uno de estos paradigmas [8].

Algoritmos Genéticos

Los algoritmos genéticos (AG) fueron propuestos por John H. Holland [9], motivado por su interés en resolver problemas de aprendizaje de máquina. Con los AGs se introducen conceptos existentes en la naturaleza, como el genotipo y fenotipo. El genotipo representa la información genética de un individuo, que ha heredado de sus

antepasados y que a su vez transmite a sus descendientes. El fenotipo son las características visibles de cada individuo. Dentro de la computación evolutiva, el fenotipo es la representación decodificada de una posible solución al problema y el genotipo es la cadena cromosómica que codifica dicha solución. El algoritmo genético opera a nivel de genotipo de las soluciones, enfatiza la importancia de la cruza sexual (la cual es su operador principal) sobre el de la mutación y utiliza una selección probabilística. La cadena binaria es la representación tradicional de los algoritmos genéticos, y se le denomina *cromosoma*. A cada posición dentro del cromosma correspondiente a una variable de decisión se le denomina *gene* y cada uno de sus valores específicos es llamado *alelo*. De tal forma, para poder aplicar el algoritmo genético se requiere de los siguientes componentes básicos:

- Una representación de las soluciones al problema.
- Un procedimiento para crear una población de posibles soluciones (estas soluciones suelen generarse inicialmente de forma aleatoria).
- Una función de evaluación que simule el ambiente, clasificando las soluciones (los individuos) en términos de su *aptitud*
- Operadores genéticos que alteren la composición de los descendientes que se producirán para las siguientes generaciones.
- Valores para los parámetros utilizados por el algoritmo genético como lo son: el tamaño de población, la probabilidad de cruza, probabilidad de mutación, número máximo de generaciones, etc.

El algoritmo genético básico se muestra a continuación:

1. Crear una población inicial, usualmente de manera aleatoria.
2. Calcular la aptitud de cada individuo.
3. Realizar selección de individuos para la cruza.
4. Aplicar los operadores genéticos de cruza y mutación a la población actual para generar a la población de la siguiente generación.
5. Ir al paso 2 hasta que se cumpla la condición de paro. Generalmente, la condición de paro es un número máximo de generaciones (iteraciones).

Estrategias Evolutivas

Las estrategias evolutivas (EE) fueron propuestas por varios investigadores alemanes encabezados por Ingo Rechenberg en 1964 [10]. Son un método de ajustes discretos aleatorios inspirado en el mecanismo de mutación que ocurre en la naturaleza. Es por eso que en las EEs el operador principal es la mutación y la cruza es

un operador secundario el cual se utiliza con menor frecuencia que en los AGs. La versión original de la EE, denotada por (1 + 1)-EE, usa un solo padre y produce en cada iteración un solo hijo usando mutación. El mejor de entre los dos (el padre y el hijo) pasará a formar parte de la siguiente generación. En la (1 + 1)-EE, a partir de un padre $x^t = (x_1^*, x_2^*, \ldots, x_n^*)^T$ se genera un hijo mediante:

$$x_i^{t+1} = x_i^t + N_i(0, \sigma_i^t) \tag{2.22}$$

donde t es la generación en la que se encuentra el algoritmo y $N_i(0, \sigma_i^t)$ es un vector de números Gaussianos independientes con media cero y desviación estándar σ_i. De tal forma, la mutación opera como una perturbación entre los individuos.

Más tarde, Rechenberg [11] introdujo el concepto de población en las EEs, al proponer una estrategia evolutiva llamada $(\mu + 1)$-EE, en la cual se tienen μ padres y se genera un solo hijo a partir de éstos, el cual puede reemplazar al peor padre de la población, en una selección extintiva.

Posteriormente, Hans-Paul Schwefel usó las estrategias $(\mu + \lambda)$-EE y (μ, λ)-EE, donde μ son los padres y λ los hijos. La primera estrategia selecciona μ individuos de la unión de padres e hijos, mientras que la segunda selecciona μ individuos de la población de hijos.

Las EEs han sido empleadas para resolver problemas en áreas tales como ingeniería, bioquímica, óptica, redes, etc, [12].

Programación Evolutiva

La programación evolutiva (PE) es una técnica que fue originalmente propuesta por Lawrence J. Fogel [13], en la cual la inteligencia se ve como un comportamiento adaptativo. Está inspirada en el principio de la evolución al nivel de las especies. Utiliza una selección probabilística y no requiere de la cruza u otro tipo de recombinación, puesto que, en la naturaleza, una especie no puede mezclarse con otra y este algoritmo evolutivo simula este principio. Enfatiza los nexos de comportamiento entre padres e hijos, en vez de buscar emular operadores genéticos específicos, como lo hacen los algoritmos genéticos.Originalmente, se definió el algoritmo de la PE de la siguiente manera:

1. Crear una población inicial de manera aleatoria.

2. Aplicar el operador de mutación (hay diversos tipos de operadores de mutación disponibles).

3. Calcular la aptitud de cada descendiente y aplicar selección mediante torneo para retener a las mejores soluciones.

La PE ha sido utilizada en diferentes áreas tales como: reconocimiento de patrones, predicción, identificación, control automático y juegos, entre otras [13, 14].

2.4. Algoritmos Evolutivos multi-objetivo

Existen varias ventajas al utilizar las técnicas evolutivas. Una de ellas es que la computación evolutiva involucra métodos que son poblacionales, lo que significa que trabajan con varias soluciones a la vez, y no con una, como lo hacen la mayoría de las técnicas de programación matemática. Esto permite que los AEMOs generen varios elementos del conjunto de óptimos de Pareto en una sola ejecución. Ademas, los AEMOs no requieren que el problema a resolver cumpla condiciones de continuidad y/o derivabilidad en sus funciones objetivo, ni que éstas estén dadas en forma algebraica.

En general, las características fundamentales de un AEMO son mantener un conjunto de soluciones potenciales, el cual es sometido a un proceso de selección y es manipulado por operadores genéticos (generalmente la recombinación y la mutación [15]).

Los AEs y los AEMOs son estructuralmente similares. La diferencia radica en que un AEMO calcula $i(i \geq 2)$ funciones de aptitud. Sin embargo, el operador de selección de un AE trabaja con un solo valor escalar. La forma más sencilla de conservar la estructura de un AE simple al abordar problemas multiobjetivo consiste en transformar el vector de aptitudes en un valor escalar.

En una reseña de AEMOs Fonseca y Fleming [16] presentan las principales estrategias para llevar a cabo la asignación de aptitud de los individuos. Estas estrategias se muestran a continuación:

- **Métodos de agregación:** combinan los objetivos en una función escalar que es utilizada para calcular la aptitud un individuo. Tienen la ventaja de producir una sola solución. Por otro lado, definir una función objetivo de esta forma requiere un profundo conocimiento del dominio del problema, el cual usualmente no está disponible.

- **Métodos basados en población:** son capaces de evolucionar múltiples soluciones no dominadas al mismo tiempo en una sola ejecución. Cambian el criterio de selección durante la fase de reproducción y la búsqueda es guiada en varias direcciones al mismo tiempo. A menudo, se seleccionan fracciones del conjunto de individuos para ser recombinados (*mating pool*) de acuerdo a uno de los n objetivos del problema.

- **Métodos basados en jerarquización de Pareto:** estos métodos asignan la aptitud con base en la dominancia de Pareto. Utilizan la dominancia de Pareto para determinar la probabilidad de reproducción de cada individuo. Hoy en día los esquemas basados en la jerarquización de Pareto son los más populares en el campo de la optimización multiobjetivo.

Existen además otras clasificaciones de los AEMOS, las cuales se basan en las técnicas clásicas de optimización multiobjetivo, descritas anteriormente en este capítulo, que los AEMOs utilizan para llevar a cabo la optimización. Enseguida se muestran algunas de éstas, clasificadas por Coello [6] de la siguiente forma:

- Enfoques Simples:
  - Funciones agregativas.
  - Programación por metas.
  - Método ϵ-*constraint*.
- Enfoques que no incorporan el concepto de optimalidad de Pareto en su mecanismo de selección.
  - Ordenamiento lexicográfico.
  - Uso de pesos y elitismo generados aleatoriamente.
  - Métodos basados en población.
- Enfoques que incorporan el concepto de optimalidad de Pareto en su mecanismo de selección.
  - MOGA.
  - NSGA y NSGA-II.
  - SPEA y SPEA2.

Además de éstas se encuentran también los AEMOs basados en indicadores.

AEMOs basados en indicadores

Recientemente se ha considerado, dentro del diseño de los AEMOs, utilizar métricas de desempeño o indicadores durante el proceso evolutivo, ya sea en el mecanismo de selección de parejas para la cruza y/o en el mecanismo de diversidad. Son entonces estos tipos de AEMOs los cuales se dice que están basados en indicadores, como lo ejemplifican el *indicator-based evolutionary algorithm* (IBEA) [17] y el *S metric selection evolutionary multi-objective algorithm* (SMS-EMOA) [18]. La mayor ventaja del uso de indicadores radica en que son valores escalares y existe evidencia de que escalan mejor que la optimalidad de Pareto en problemas con muchas funciones objetivo.

Elementos clave de los AEMOs

De las definiciones previas proporcionadas, podemos distinguir dos características principales que debemos alcanzar en la aproximación del conjunto de óptimos de Pareto: una es minimizar la distancia al verdadero frente de Pareto y la segunda es maximizar la diversidad dentro del conjunto de óptimos de Pareto. La mayoría de los AEMOs más populares utilizan la jerarquización de individuos con respecto a la optimalidad de Pareto, para la selección y adoptan algún estimador de densidad como nichos [19] y *crowding* [1] para producir varias soluciones distintas en una sola ejecución. En esta sección se describen algunos conceptos importantes de los AEMOs, que se toman en consideración para su diseño, a fin de lograr cubrir estas características.

Elitismo A diferencia de la optimización global, en la cual se retiene a las mejores soluciones de la población de una generación a otra sin alteración, la implementación del elitismo en los AEMOs es más compleja. En lugar de un solo mejor individuo, hay un conjunto de élite de un tamaño que puede ser mayor que el de la población. En la actualidad, hay dos enfoques principales para llevar a cabo el mecanismo de elitismo. Uno de ellos es combinar la población de padres y la de hijos, para posteriormente retener la mejor mitad. Otra alternativa es mantener una población secundaria llamada archivo histórico, en el cual se retienen las soluciones no dominadas encontradas en el transcurso del proceso de búsqueda [15].

Diversidad poblacional En un AE, el mecanismo de selección toma las soluciones con aptitud alta y descarta aquellas con aptitud baja, lo cual permite que la población converja a una solución única. Sin embargo, en los AEMOs la meta es encontrar un conjunto de soluciones bien distribuidas a lo largo del frente de Pareto en una sola ejecución. Esto requiere de estimadores de densidad que eviten convergencia a una solución única (*fitness sharing* [20], *clustering* [21], nichos [19], *crowding* [1], entropía [22], etc.). A continuación se describen algunos de ellos:

- **Vector de pesos:** En este caso, un conjunto de vectores en el espacio aptitud/objetivo es utilizado para diversificar los puntos en la superficie del frente de Pareto (es decir, generar una contribución uniforme del frente de Pareto). Cambiando los pesos, se definen diferentes direcciones de búsqueda, permitiendo así mover las soluciones lejos de sus vecindarios.
- ***Fitness sharing*/Enfoque de nichos:** En este enfoque, el tamaño (o radio) de un vecindario (o nicho) es controlado por el valor σ_{share} (radio de nicho). Para ello, se hace un conteo de las soluciones que se encuentran localizadas en el mismo nicho. Esto busca promover la generación de soluciones en las regiones menos pobladas del espacio de búsqueda.
- ***Crowding/Clustering*:** En este caso, se seleccionan las soluciones que sobreviven de acuerdo a una métrica de hacinamiento de una región, medida en el espacio de las funciones objetivo. Esta es una idea similar a la presente en *fitness sharing*, pero más eficiente y sin requerir parámetros adicionales. Este es el esquema adoptado en el NSGA-II [1].

2.4.1. Algoritmos Evolutivos Multiobjetivo representativos

En esta sección se describen brevemente algunos de los AEMOs que se encuentran entre los más representativos en el área:

***Multi-Objective Genetic Algorithm* (MOGA).** Propuesto por Fonseca y Fleming [23]. Este algoritmo utiliza un procedimiento de jerarquización basado en la no dominancia, en el cual la jerarquía de un individuo es igual al número de

soluciones que lo dominan. El procedimiento para asignar la aptitud consta de los siguientes pasos:

1. Ordenar la población de acuerdo con la jerarquía de los individuos.
2. Asignar la aptitud de los individuos interpolando del mejor (jerarquía 1) al peor individuo (jerarquía $n \leq M$, donde M es el tamaño de la población). La interpolación se hace usualmente por medio de una función lineal.
3. Promediar y compartir la aptitud de los individuos con la misma jerarquía.

Este algoritmo no adopta elitismo en su version original.

***Non-dominated Sorting Genetic Algorithm* II (NSGA-II).** Srinivas y Deb propusieron un método que utiliza una selección basada en jerarquización de Pareto, para favorecer a las soluciones no dominadas, y un método de nichos para mantener la diversidad el cual nombraron como *Non-dominated Sorting Genetic Algorithm* (NSGA) [24]. NSGA está basado en la clasificación por capas de los individuos, de acuerdo a su no dominancia. Antes de realizar la selección se jerarquiza la población para obtener los individuos que dominan al resto de la población. Este grupo de individuos conforma el primer frente no dominado y se les asigna un valor de aptitud falso. Para mantener la diversidad en la población, estas soluciones comparten entre ellas su aptitud. Posteriormente, estos individuos son ignorados y se genera el segundo frente no dominado de individuos, a los cuales se les asigna una aptitud menor que la del primer frente. Este proceso continúa hasta que todos los individuos son clasificados. A partir de esta idea, Deb et al. [1] propusieron una nueva versión de este algoritmo, llamado NSGA-II. El nuevo algoritmo es más eficiente en términos computacionales, además de incorporar elitismo y un operador de crowding para mantener la diversidad, sin la necesidad de elegir un parámetro adicional (como los nichos). NSGA-II es eficiente, pero parece tener dificultades para generar soluciones no dominadas que se encuentren en ciertas regiones aisladas del espacio de búsqueda, además de degradar rápidamente su desempeño al aumentar la cantidad de funciones objetivo.

***Strength Pareto Evolutionary Algorithm* 2 (SPEA2).** El Strength Pareto Evolutionary Algorithm 2 fue propuesto por Zitzler et al. [25], y se basa en su predecesor SPEA [26], en el que se combinan las técnicas más exitosas de los diferentes AEMOs, existentes hasta ese entonces. SPEA [26] introduce elitismo al almacenar a los individuos no dominados en un archivo externo. Para cada individuo de este conjunto externo se calcula un valor llamado fortaleza. Este valor es proporcional al número de soluciones que domina un cierto individuo del conjunto externo. En este algoritmo, cada individuo tiene un valor de aptitud el cual es la suma de su fortaleza más una estimación de densidad. El algoritmo aplica los operadores de selección, cruza y mutación para llenar un archivo de individuos, para después copiar a los individuos no dominados tanto

de la población original como la de dicho archivo. Si el número de individuos no dominados es mayor que el tamaño de población establecido, se aplica un operador de truncamiento basado en las distancia de los k-ésimos vecinos más cercanos. De esta forma, los individuos con la menor distancia a otros individuos son seleccionados. SPEA2 es la versión mejorada de SPEA y tiene tres diferencias principales con respecto a su predecesor: (1) incorpora una estrategia de asignación de aptitud de grano fino, la cual toma en cuenta para cada individuo el número de individuos que lo dominan y el número de individuos que éste domina; (2) utiliza una técnica de estimación de densidad del vecino más cercano, la cual guía la búsqueda de forma más eficiente, y (3) tiene una mejor técnica de truncamiento de archivo que garantiza la preservación de soluciones de frontera.

***S-Metric Selection Evolutionary Multi-Objective Algorithm* (SMS-EMOA).** Fue propuesto por Beume et al. [18]. Para este AEMO el *hipervolumen*[1] (o métrica S) [27] es utilizado en su proceso de selección. En el SMS-EMOA sólo una solución es creada cada vez y es agregada a la población actual, para entonces ejecutar la selección de individuos que pasarán a la siguiente generación. Posteriormente, para cada solución en la población extendida, se calcula su contribución a la métrica de hipervolumen, como la diferencia de su valor estando dentro y fuera de la población. La diferencia es entonces asignada como aptitud a cada individuo (solución) en la población. Al inicio del proceso evolutivo, muchas soluciones dentro de la población actual pueden ser dominadas y, por lo tanto, no contribuyen a la métrica del hipervolumen de la aproximación del frente de Pareto. Para estos casos, SMS-EMOA utiliza una jerarquizaron de Pareto como la que utiliza NSGA-II [1] y la contribución al hipervolumen es calculada para cada capa de la jerarquía de las soluciones. En consecuencia, las soluciones descartadas serán seleccionadas como las que menos contribuyen en la métrica del hipervolumen pero que se encuentran en la capa más alta de la jerarquización de Pareto.

***Multiobjective Evolutionary Algorithm Based on Decomposition* (MOEA/D).** Fue propuesto por Zhang y Li [28]. Es un algoritmo evolutivo multiobjetivo cuyo mecanismo principal es la descomposición de un problema de optimización multiobjetivo en un conjunto de subproblemas de optimización escalar, los cuales optimiza simultáneamente mediante la evolución de una población de soluciones. Utiliza el algoritmo de Tchebycheff para mantener una población de N puntos, donde cada uno es la representación de un subproblema. Así mismo, adopta una población externa (EP), la cual es usada para guardar las soluciones no dominadas encontradas durante la búsqueda. Se da una descripción más

[1]El hipervolumen es un indicador que se utiliza para medir y comparar la calidad de las soluciones finales producidas por un AEMO. Fue propuesto originalmente por Zitzler y Thiele [27], quienes lo definen como el tamaño del espacio cubierto o del espacio dominado. Se da una definición más formal del hipervolumen en la subsección 6.1.1 de esta tesis.

a fondo de este algoritmo en la sección 3.1.1 de esta tesis.

Capítulo 3

Optimización a gran escala en el espacio de las variables de decisión

Los algoritmos evolutivos (AEs) han sido aplicados con éxito en un gran número de problemas de optimización numérica y combinatoria [29]. Sin embargo la mayoría de las veces pierden su efectividad y ventajas a la hora de ser aplicados a problemas de un gran número de variables de decisión, sufriendo de la llamada *"maldición de la dimensionalidad"*, la cual implica que el desempeño de un algoritmo evolutivo se deteriora rápidamente a medida que aumenta la dimensionalidad del espacio de búsqueda [30]. Algoritmos evolutivos que tienen un buen desempeño en problemas de baja dimensionalidad, usualmente fallan a la hora de buscar buenas soluciones en problemas de alta dimensionalidad. En este capítulo se presentan los trabajos relacionados con la optimización de problemas mutiobjetivo con un gran número de variables de decisión existentes en el estado del arte. Así mismo se presenta el trabajo relevante de optimización de problemas a gran escala en el área de optimización global.

3.1. Optimización evolutiva multiobjetivo a gran escala

Muchos problemas multiobjetivo del mundo real tienen cientos e incluso miles de variables de decisión, por lo cual la escalabilidad es un tema relevante para los algoritmos de optimización. Esto, sin embargo, contrasta con los modelos evolutivos existentes para optimización multiobjetivo los cuales suelen validarse con problemas de no más de 30 variables. Esto nos hace ver que la escalabilidad en el espacio de las variables ha sido un tema raramente abordado en el dominio de los algoritmo evolutivos multiobjetivo, a diferencia de la escalabilidad del número de funciones objetivo que ha sido abordado por varios investigadores en años recientes [31, 32]. En esta sección se da una introducción al lector al área de la computación evolutiva y al trabajo realizado en problemas multiobjetivo de gran escala, en el espacio de las variables de decisión. Se describen los estudios realizados y el desempeño de algunos

paradigmas evolutivos, así como su funcionamiento.

3.1.1. MOEA/D

MOEA/D [28] es un algoritmo evolutivo multiobjetivo que consiste en descomponer un problema de optimización multiobjetivo en un conjunto de subproblemas de optimización escalar y los optimiza simultáneamente, mediante la evolución de una población de soluciones. Utiliza el algoritmo de Tchebycheff para mantener una población de N puntos, donde cada uno es la representación de un subproblema. Asi mismo adopta una población externa (EP), la cual es usada para guardar las soluciones no dominadas encontradas durante la búsqueda. Su funcionamiento se describe a continuación.

Sean $\lambda^1, \cdot, \lambda^N$ un conjunto de vectores de pesos pares y z^* un punto de referencia. El problema de optimización multiobjetivo se puede descomponer en N subproblemas de optimización escalar, utilizando el enfoque de Tchebycheff y la función objetivo del j-ésimo subproblema, de la siguiente forma:

$$g^{te}(x|\lambda^j, z^*) = \max_{l \leq i \leq m} \{\lambda_i^j | f_i(x) - z_i^*|\} \tag{3.1}$$

donde $\lambda^j = (\lambda_1^j, \cdot, \lambda_m^j)^T$. MOEA/D minimiza estas N funciones objetivo en una sola ejecución. Aquí, un vecindario de vectores de peso λ^i está definido como un conjunto de sus vectores de pesos más cercanos en $\{\lambda_1, \cdot, \lambda_N\}$. Así, el vecindario del i-ésimo subproblema consiste en todos los subproblemas con los vectores de pesos del vecindario de λ^i. La población está compuesta de las mejores soluciones encontradas hasta el momento para cada problema. Sólo las soluciones actuales a sus subproblemas vecinos, son explotadas para optimizar un subproblema. En cada generación t, MOEA/D con el enfoque de Tchebycheff mantiene:

- una población de N puntos $x^1, \cdot, x^N$, donde x^i es la solución actual del i-ésimo subproblema.
- $FV^1, \cdot, FV^N$, donde FV^i es el valor de F de x^i, es decir, $FV^i = f(x^i)$ para cada $i = 1, \cdot, N$;
- $z = (z_1, \cdot, z_m)^T$, donde z_i es el mejor valor encontrado hasta el momento para el objetivo f_i;
- una población externa EP, la cual es usada para almacenar las soluciones no dominadas encontradas hasta el momento.

El algoritmo funciona de la siguiente manera, teniendo como entrada un POM, un número N de subproblemas a considerar, una distribución uniforme de vectores de pesos: $\lambda^1, \cdot, \lambda^N$ y un número T, que indique el numero de vectores de pesos en el vecindario de cada vector de pesos:

Paso 1) Inicialización:

Paso 1.1) Establecer $EP = 0$

Paso 1.2) Calcular las distancias Euclidianas entre todos los pares de vectores de pesos y entonces calcular los T vectores de pesos más cercanos a cada vector de pesos. Para cada $i = 1, \cdot, N$ establecer $B(i) = \{i_1, \cdot, i_T\}$ donde $\lambda^{i_1}, \cdot, \lambda^{i_T}$ son los T vectores de pesos más cercanos a λ^i.

Paso 1.3) Generar una población inicial $x^1, \cdot, x^N$ de forma aleatoria y establecer $FV^i = F(x^i)$.

Paso 1.4) Inicializar $z = (z_1, \cdot, z_m)^T$

Paso 2) Actualización:

Para $i = 1, \cdot, N$ hacer

Paso 2.1) Reproducción: Escoger dos indices k, l de forma aleatoria de $B(i)$ y generar una nueva solución y de x^k y x^l usando operadores genéticos.

Paso 2.2) Mejora: Aplicar una reparación o mejora del problema en especifico sobre y para producir y'.

Paso 2.3) Actualizar z: Para cada $j = 1, \cdot, m, if z_j < f_j(y')$, entonces establecer $z_j = f_j(y')$

Paso 2.4) Actualizar el vecindario de soluciones: Para cada indice $j \in B(i)$, si $g^{te}(y'|\lambda^j, z) < g^{te}(x^j|\lambda^j, z)$, entonces establecer $x^j = y'$ y $FV^j = F(y')$

Paso 2.5) Actualizar EP: Descartar de EP todos los vectores dominados por $F(y')$ y agregar esta misma a EP si ningún vector en EP domina $F(y')$

Paso 3) Criterio de paro: Si el criterio de paro se cumple, detener el algoritmo y dar como salida EP. De otra manera, ir al paso 2.

El interés en este trabajo, es el estudio que presenta acerca de la escalabilidad de las variables de decisión sobre el problema ZDT1, descrito en 7. Ellos estudian como el costo computacional, en términos de números de evaluaciones sobre el vector objetivo, incrementa a la hora de incrementar el número de variables de decisión del problema. Utilizan un número máximo de 10,0000 evaluaciones sobre el vector de funciones objetivo y un número de variables de decisión que va desde las 10 hasta 100 variables. Observan el comportamiento de su propuesta (MOEA/D) por medio de 30 ejecuciones independientes sobre cada número de variables de decisión, utilizando como medida de desempeño la minimización de la distancia entre representantes del frente de Pareto, la llamada *métrica-D* [33].

Concluyen que el número de evaluaciones promedio necesarios para cumplir con su medida de desempeño establecida se incrementa de forma lineal a medida que el número de variables de decisión utilizadas aumenta. Atribuyen este comportamiento a que el número de subproblemas que se utilizan es siempre 100,

sin importar cuantas variables de decisión sean utilizadas, y que la complejidad de cada subproblema podría escalarse linealmente con el número de variables de decisión.

A pesar de estos resultados obtenidos, se tiene que tomar en cuenta que éste estudio es demasiado pequeño para reflejar un comportamiento general del algoritmo (MOEA/D) sobre problemas de optimización multiobjetivo a gran escala (en el espacio de las variables de decisión). Ademas, como se refleja en el trabajo de Durillo et al. [4, 3], descrito más adelante, así como en los resultados observados en esta tesis, es a partir de un número mayor que 100 variables de decisión en donde el deterioro del desempeño de las metaheurísticas es aún más evidente, al menos en el caso de los problemas ZDT.

3.1.2. Estudios de escalabilidad de metaheuristicas multiobjetivo

Hasta ahora solo se conocen dos trabajos los cuales han prestado atención al comportamiento de las metaheurísticas y su comportamiento ante la escalabilidad del número de variables de un problema multiobjetivo. En los trabajos de Durillo et al. [4, 3] se realizó un estudio de escalabilidad de 8 metaheurísticas multiobjetivo conformadas por tres algoritmos genéticos (NSGA-II [1], SPEA2 [25], y PESA-II [34]), una estrategia evolutiva (PAES [35]), un PSO (OMOPSO [36]), un Algoritmo Genético Celular (MOCell [37]), un algoritmo basado en Evolución Diferencial (GDE3 [38]) y un algoritmo de búsqueda dispersa (AbYSS [39]), en el cual se toman los problemas de prueba del conjunto Zitzler-Deb-Thiele (ZDT) [40] y se escalan en el espacio de las variables de decisión en un rango de las 8 a las 2048 variables. Se analiza el comportamiento de dichas metaheurísticas sobre éstos.

Los problemas del conjunto ZDT son escalables en términos del número de variables de decisión al mismo tiempo que mantienen invariante el frente de Pareto de cada problema. Ademas, este conjunto de problemas ofrece funciones que tienen distintas propiedades, como problemas convexos, no convexos, discontinuos, multifrontales, etc. En este estudio evaluaron dichos problemas con 8, 16, 32, 64, 128, 256, 512, 1024 y 2048 variables de decisión. Esto tratando de estudiar no sólo qué técnicas escalan mejor, sino también, si sus capacidad de búsqueda permanecen contantes o no cuando el número de variables aumenta. Para esto, adoptaron como métrica el indicador del hipervolumen (HV) [27]. Se tomó el 98 % del HV del verdadero frente como criterio para considerar que un POM había sido resuelto exitosamente. Sin embargo teniendo en cuenta de que es posible que algunos algoritmos no logren nunca alcanzar esta meta se tomó ademas un número máximo de 10,000,000 evaluaciones sobre las funciones objetivo. De las ocho metaheurísticas del estudio, se concluye que GDE3 [38] y OMOPSO [36] son las que tienen el mejor desempeño. Sin embargo este estudio evidenció empíricamente que todas las metahurísticas estudiadas decrementan significativamente su eficiencia a medida que el número de variables de decisión aumenta.

Zhang et al. [28] realizaron un pequeño estudio de escalabilidad usando el problema ZDT1 [40] con hasta 100 variables de decisión. En dicho estudio se buscaba estudiar el promedio de evaluaciones necesarias para reducir el valor de la métrica D en el problema ZDT1 conforme se aumenta el número de variables de decisión de dicho problema. Empíricamente, se logró determinar que el promedio de evaluaciones aumenta linealmente conforme aumenta el número de variables de decisión. Esto indica que el algoritmo no se especializa en problemas de gran dimensión, en el espacio de las variables de decisión, lo cual es obvio ya que fue en realidad diseñado para problemas con un gran número de funciones objetivo.

Hasta ahora, son estos trabajos los únicos de los que se tiene conocimiento que presten atención a la escalabilidad de las variables del problema en cuanto al número de variables de decisión. Cabe mencionar que el trabajo de Zhang et al. [28] sólo se centra en un problema y no se centra en el diseño de técnicas para alta dimensionalidad. Aunque en el trabajo de Durillo et al. [3], se realiza un estudio más profundo e incluso se proponen algunas mejoras a los algoritmos más relevantes de dicho estudio, aún no se conoce un esquema específicamente diseñado para problemas multiobjetivo de alta dimensionalidad en el espacio de las variables. Por lo tanto, esta tesis busca contribuir al estado del arte actual, planteando un nuevo algoritmo evolutivo multiobjetivo que tenga mejor desempeño que los tradicionales en problemas de alta dimensionalidad.

Es importante mencionar que en el área optimización evolutiva mono-objetivo, a diferencia de la multiobjetivo, existe mucha más investigación con respecto a la escalabilidad de las variables de decisión de un problema. Los diversos criterios que se toman para el desarrollo de técnicas, que se ocupan de problemas de alta dimensionalidad en el espacio de las variables de decisión de problemas mono-objetivo, se pueden dividir de forma general en aquellas que dividen el problema en grupos de variables (dividen el problema original para hacerlo más fácil de resolver) y las que no lo dividen (atacan el problema perturbando la población del algoritmo evolutivo o combinando métodos evolutivos existentes) [5]. La razón por la cual esto se debe de tener en cuenta es debido a que algunas de las ideas de estas técnicas, a pesar de haber sido diseñadas para el caso de optimización mono-objetivo, pudieran extenderse a la optimización evolutiva multiobjetivo para lidiar con la alta dimensionalidad en el espacio de las variables de decisión. Dentro de estos criterios el modelo de coevolución cooperativa es uno de los que mejores resultados ha demostrado. Es por eso la propuestas de este trabajo es un modelo basado en coevolución cooperativa llevado a la optimización evolutiva multiobjetivo, el cual se describe más adelante.

3.2. Optimización global a gran escala

En esta sección se describen los diversos algoritmos del estado del arte para optimizar problemas a gran escala sin restricciones. La mayoría de estos algoritmos se enfocan a optimizar en un rango que va desde las 100 hasta las 1000 variables de

decisión. A continuación se da una introducción sobre este tipo de algoritmos.

Para poder hablar sobre este tipo de problemas definiremos de manera breve a que nos referimos con optimizar en el ámbito de la optimización global. Podemos entonces definir el problema de optimización global de la siguiente manera:

Minimizar:

$$f(\vec{x}) \tag{3.2}$$

donde $f : \mathbb{R}^D \rightarrow \mathbb{R}$ es la función objetivo, misma que mapea un espacio D-dimensional a una dimensión. Al evaluarla con el vector $\vec{x} \in \mathbb{R}^D$, tal vector se compone de las variables de decisión.

Existen diversas técnicas enfocadas a la optimizacion de este tipo de problemas con un gran número de variables. Estas técnicas se dividen en dos grandes categorías:

Técnicas que dividen el problema en grupos de variables: algoritmos que optimizan mediante la división de variables o la división de la población

Técnicas que no dividen el problema: algoritmos que hacen uso de heurísticas con alguna modificación, por ejemplo, con una forma especial de inicializar la población, basándose en el balance de exploración y explotación mediante algoritmos meméticos, búsquedas locales, etc.

3.2.1. Técnicas que no dividen el problema

Estas técnicas pueden operar de la siguiente manera: Perturbando la población o combinando métodos. En ambos casos se utilizan operadores básicos (inicializar la población, realizar la selección, mutación y cruza) y algún método auxiliar para mejorar el desempeño. A continuación se describen algunos algoritmos que caen en esta clasificación.

Perturbar la población: Cuando se inicializa la población, existen algunos métodos que resultan eficientes en algunos casos para cubrir mayor espacio de búsqueda, agilizando el proceso de encontrar la solución. Otra variante es el cambio de valores en parámetros o la reducción de la población.

Evolución diferencial basada en oposición: El enfoque de evolución diferencial basado en oposición (opposition-based differential evolution, EDO) [41], es una técnica cuya principal diferencia con respecto a la evolución diferencial es en la inicialización de la población y en la producción de nuevas generaciones [42].

La principal característica y ventaja de esta técnica es que, al inicializar la población, se crea un espejo de los individuos, teniendo finalmente unos individuos positivos y otros negativos. Esto resulta eficiente cuando la función es simétrica, el espacio de búsqueda se reduce a la mitad considerando que la mitad de la función es un espejo de la otra mitad. La desventaja de este algoritmo es, cuando la función es asimétrica,

el límite superior e inferior tienen diferentes valores, o bien, cuando la función es rotada, pues la población original y la población opuesta no abarcarían todo el espacio de búsqueda, considerando que se tendría que hacer una ordenación de ambas poblaciones para desechar a los peores individuos. Otra desventaja es que si la función es simétrica y los óptimos locales también son simétricos, un individuo puede llegar a caer en el óptimo local al igual que su espejo, por lo que el algoritmo se podría estancar temporalmente.

Evolución diferencial con adaptación automática y reducción de la población: La adaptación automática es un mecanismo donde los parámetros de control de la mutación (F) y la cruza (CR) van cambiando durante su ejecución, y el tamaño de la población se va reduciendo en cada generación. Este algoritmo, jDEdynNP [43], funciona con los siguientes puntos:

- **Parámetros de control auto-adaptativos.** Se utiliza un mecanismo de adaptación de parámetros de control CR y F. Estos parámetros de control son calculados con las siguientes ecuaciones:

$$F_i^{G+1} = \begin{cases} F_l + rand_1 \times F_u & \text{si} \quad rand_2 < \tau_1 \\ F_i^G & \text{en otro caso} \end{cases}$$

$$CR_i^{G+1} = \begin{cases} rand_3 & \text{si} \quad rand_4 < \tau_2 \\ CR_i^G & \text{en otro caso} \end{cases}$$

  donde $rand_j \in [0,1], j \in \{1,2,3,4\}$, es un número aleatorio con distribución uniforme, $\tau_1 = 0.1$ y $\tau_2 = 0.1$ son probabilidades ajustadas a los parámetros de control para F y CR. $F_l = 0.1$, y $F_u = 0.9$ son valores asignados por el usuario que limitan los valores que puede tomar F.

- **Tamaño dinámico de la población.** Se utiliza una variante de la evolución diferencial [44] con tamaño dinámico de población. Al inicio, se establece un tamaño de población inicial, pero conforme va avanzando el proceso evolutivo, la población se va reduciendo. Antes de decrementar el tamaño de la población se seleccionan los individuos mediante lo siguiente:

$$\vec{x_i}^G = \begin{cases} \vec{x}^G_{\frac{NP}{2+i}} & \text{si} \quad f(\vec{x}^G_{\frac{NP}{2+i}}) < f(\vec{x_i}^G) \wedge G = G_R \\ \vec{x_i}^G & \text{en otro caso} \end{cases}$$

  Donde GR es la generación en la cual la población es reducida. La siguiente generación se conforma de la siguiente manera: se toma un individuo de la primera mitad de la población, y otro de la segunda, y luego se toma entonces al mejor para formar la población actual.

- **Autoadaptación de F.** Usa un mecanismo para cambiar el signo al parámetro de control F con una probabilidad de 0.75 cuando $f(\vec{x}_r 2) > f(\vec{x}_r 3)$.

La ventaja del algoritmo, es que cuando se reduce la población, se aplica el método de selección de la evolución diferencial, por lo que no se requiere de un método de ordenamiento, ya que utiliza un método voraz ({lem greedy}).

Algoritmo de evolución diferencial auto-adaptativo: Es una mejora del algoritmo jDEdynNP, la cual fue nombrada jDElsgo [45]. En este algoritmo agregaron los siguientes mecanismos a jDEdynNP:

- Aleatoriedad a los dos primeros individuos de la población.
- Un valor de F más pequeño para la mutación, donde $F \in [0.01, 0.1]$.
- Se usa un factor de búsqueda local cuando se obtiene mejor resultado que el de el mejor individuo actual.
- Se utiliza una cruza exponencial.

Multi-agentes: El algoritmo genético multiagente MAGO [46] se basa en algoritmos genéticos [47] y agentes computacionales. Los agentes han tenido un gran uso en computación como un método para resolver problemas de satisfacción de restricciones, extracción de imágenes, etc, sobre todo en el área de inteligencia artificial [48, 49]. Un agente es una entidad física o virtual que cumple con las siguiente propiedades:

- Es apto para vivir y actuar en un ambiente.
- Es capaz de percibir su entorno.
- Tiene propósitos específicos.
- Es de un comportamiento reactivo. Los sistemas multi-agentes son sistemas computacionales cuyos agentes trabajan juntos para lograr una meta.

Un agente, para los problemas de optimización numérica, es aquel que representa una posible solución al problema a optimizar, como es en el caso de MAGO. El valor de su energía es igual al valor negativo de la función objetivo, en el caso de minimización.

MAGO es muy similar a un algoritmo genético celular [50] pues utiliza una red como su población. Cada individuo está localizado en una celda. En esencia, un algoritmo genético celular tiene una estructura de vecindad apta para una implementación paralela de grano fino. En MAGO, cada individuo es considerado como un agente con un propósito y comportamiento propios.

Combinación de métodos: Los algoritmos evolutivos han ganado popularidad como métodos de optimización durante los últimos años. Entre las razones de este

hecho podemos mencionar la facilidad de implementación y un buen rendimiento obtenido para un amplio número de funciones. Sin embargo, los diversos desarrollos han mostrado que la hibridación con otras estrategias como las búsquedas locales pueden mejorar el desempeño en ciertos casos, refinando los valores de los individuos que han realizado una exploración y después una explotación del espacio de búsqueda. A continuación, se describen algunos algoritmos que han sido propuestos con una búsqueda local y una global para problemas de optimización a gran escala.

3.2.2. Técnicas que dividen el problema

Este tipo de técnicas dividen las variables en sub-poblaciones, implementando en la mayoría de los casos la técnica de co-evolución cooperativa [51]. Existe un tipo de técnicas que crea grupos estáticos. En una de éstas, el usuario debe determinar cuántos grupos son necesarios. Existen otras en las que se calcula automáticamente el número de dimensiones para crear grupos. A continuación, se discuten brevemente algunos algoritmos que caen en esta categoría.

CCGA: Es el primer modelo de coevolución cooperativa. Fue propuesto por Mitchell A. Potter y Kenneth A. De Jong [51], se basa en modelar un ecosistema que consiste de dos o más especies, las cuales tienen una relación ecológica de mutualismo. Dichas especies cooperan entre ellas y son recompensadas basándose en qué tan bien trabajan juntas en la solución de un problema. Cada ciclo consiste en una evolución completa de cada una de las subpoblaciones. La idea principal de esta técnica es descomponer un problema de alta dimensionalidad en subcomponentes de baja dimensionalidad y evolucionar estos subproblemas cooperativamente por un número predefinido de ciclos. La cooperación ocurre tan solo cuando se evalúan a los individuos, ya que un individuo es más apto entre más ayude a los demás a mejorar la solución al problema.

La idea principal de CCGA es la siguiente:

- Una especie representa un subcomponente de una solución potencial.
- Las soluciones completas se obtienen de la unión de miembros representativos de cada especie.
- La asignación de crédito a nivel de especie se define en términos del resultado de la colaboración entre especies.
- Cuando se requiere, el número de especies (subpoblaciones) se incrementa o decrementa de forma automática.
- La evolución de cada especie es manejada por un algoritmo evolutivo.

La estructura del modelo original de coevolución propuesto por Mitchell A. Potter y Kenneth A. De Jong en [51] se puede resumir en el siguiente procedimiento:

1. Descomponer el vector objetivo en m subcomponentes de menor dimensión
2. Iniciar un índice i=1 para comenzar un nuevo ciclo
3. Optimizar el i-ésimo subcomponente con un cierto algoritmo evolutivo por un número predefinido de evaluaciones de la función de aptitud.
4. Si $i \leq m$ entonces volver al paso 3
5. Detener si el criterio de paro se satisface. De lo contrario, volver al paso 2 para comenzar otro ciclo.

A continuación se mencionan algunos algoritmos basados en esta idea.

CCGA-I y CCGA-II [51]: Modelos basados en coevolución cooperativa que consisten en dividir el problema de N variables de decisión en N subpoblaciones (especies), para así reducir la dificultad del problema original y poder llegar a una mejor solución. Este esquema fue propuesto originalmente para algoritmos genéticos. La diferencia entre CCGA-I y CCGA-II es que CCGA-I utiliza una técnica voraz para la coevolución, mientras que CCGA-II permite la coevolución con valores aleatorios.

CCDE-O y CCDE-H [52]: Modelo de coevolución cooperativa con motor de búsqueda de evolución diferencial. En el caso de CCDE-O, se subdivide la población inicial de tal forma que cada subpoblación tenga una sola variables de decisión. Para el caso de CCDE-H, se divide en dos subpoblaciones, cada una de las cuales se encarga de optimizar la mitad de las variables del problema.

DECC-I y DECC-II [53]: Modelos que utilizan la estrategia de coevolución diferencial con algunas variantes. En este caso, se generan subgrupos los cuales tienen asignados cierto nuero de variables de forma aleatoria, lo que quiere decir que cualquier variable tiene la misma posibilidad de pertenecer a cualquier subgrupo existente. Además, aquí no sólo se evoluciona con respecto al mejor de cada ciclo, sino que se toman también al peor y a un miembro aleatorio de las subpoblaciones. La diferencia en DECC-II es que en vez de evolucionar todas las subpoblaciones cada ciclo, sólo se toma una al azar y es ésta la única que se evoluciona mientras las demás permanecen estáticas.

MLCC [54]: Modelo de coevolución cooperativa multinivel. Trabaja con la idea de subpoblaciones de tamaño variable para atacar la división de problemas de alta dimensionalidad. El problema que se presenta con la mayoría de estas técnicas es a la hora de optimizar funciones de alta dimensionalidad las cuales son no separables [55], pues la coevolución original ha demostrado tener éxito en las funciones separables,

debido a que es posible optimizar de forma independiente las variables de decisión. Sin embargo en las funciones no separables, al presentarse una influencia entre las variables, la separabilidad se vuelve un poco más difícil de realizar. Para fines de nuestro estudio definiremos la separabilidad de la siguiente forma:

Una función $f(x)$ es separable si $\forall k \in \{1, n\}$ y

$$\left.\begin{array}{ll} x \in S, & x = (x_1, \cdots, x_k, \cdots, x_n) \\ x' \in S, & x' = (x_1, \cdots, x'_k, \cdots, x_n) \end{array}\right\} \Rightarrow f(x) < f(x')$$

implica

$$\left.\begin{array}{ll} \forall y \in S, & y = (y_1, \cdots, x_k, \cdots, y_n) \\ \forall y' \in S, & y' = (y_1, \cdots, x'_k, \cdots, y_n) \end{array}\right\} \Rightarrow f(y) < f(y')$$

de otra forma es llamada función no separable.

Básicamente la no-separabilidad de una función quiere decir que el vector de variables de decisión consiste de variables que interactúan entre si, mientras que la separabilidad significa que el vector de variables en el valor de aptitud depende tan solo de si mismo; es decir, las variables son independientes entre si.

DEwSAcc [56]: Es un algoritmo autoadaptativo con coevolución cooperativa y evolución diferencial. Se auto-adapta la cruza y la mutación. Para adaptar el factor de amplificación del i-ésimo individuo de la generación $G + 1$ (denotado por $F_{i,G+1}$) se aplica:

$$F_{i,G+1} = Fi, G \cdot e^{\tau N(0,1)} \tag{3.3}$$

donde τ es el factor de aprendizaje y $N(0, 1)$ es un número aleatorio con distribución gaussiana.

La co-evolución cooperativa que utiliza se basa en la división de las sub-poblaciones de DECC [53]. En ésta, se cambian algunos individuos en un cierto número de generaciones. La cruza CR se adapta con respecto al mecanismo de coevolución cooperativa utilizando una cruza binaria. La ventaja de este algoritmo es que tiene mayor posibilidad de explorar y explotar el espacio de búsqueda, realizando una combinación para que la mutación y la cruza tengan diferentes caminos con diferentes soluciones. Sin embargo, al estar basado en el algoritmo DECC, no existe una interacción constante entre las variables de decisión del problema, por estar éstas distribuidas en las diversas sub-poblaciones. Por ende, este método no tiene mucho éxito en dar solución a problemas no separables.

Como se ha podido observar, en el área optimización evolutiva mono-objetivo, a diferencia de la multiobjetivo, existe mucha más investigación con respecto a la escalabilidad de las variables de decisión de un problema. Los diversos criterios que se toman para el desarrollo de técnicas que se ocupan de problemas de alta dimensionalidad en el espacio de las variables de decisión de problemas mono-objetivo se pueden

dividir de forma general en aquellas que dividen el problema en grupos de variables (o sea, dividen el problema original para hacerlo más fácil de resolver) y las que no lo dividen (atacan el problema perturbando la población del algoritmo evolutivo o combinando métodos evolutivos existes) [5]. La razón por la cual esto se debe de tener en cuenta es debido a que algunas de las ideas de estas técnicas, a pesar de haber sido diseñadas para el caso de optimización mono-objetivo, pudieran extenderse a la optimización evolutiva multiobjetivo para lidiar con la alta dimensionalidad en el espacio de las variables de decisión.

Capítulo 4

Coevolución en Algoritmos Evolutivos Multiobjetivo

Como se ha mencionado anteriormente el estudio de escalabilidad de variables de problemas multiobjetivo no ha sido un área muy estudiada, más sin embargo en el área de optimzación global, si se ha hecho un análisis más a fondo de este tipo de problemas, como se describió en el capítulo previo. Los diversos criterios que se toman para el desarrollo de técnicas que se ocupan de problemas de alta dimensionalidad en el espacio de las variables de decisión de problemas mono-objetivo se pueden dividir de forma general en aquellas que dividen el problema en grupos de variables (dividen el problema original para hacerlo más fácil de resolver) y las que no lo dividen (atacan el problema perturbando la población del algoritmo evolutivo o combinando métodos evolutivos existes) [5]. Dentro de estos criterios el modelo de coevolución cooperativa es uno de los que mejores resultados ha demostrado [52, 54, 53, 55], es por eso que la propuesta presentada en esta tesis un modelo basado en coevolución cooperativa, técnica de la cual se hace descripción más detallada en esta sección.

En este capítulo se explica el paradigma general de la coevolución, dentro del ámbito de la computación evolutiva. Se describe la forma en que se ha adaptado y utilizado el esquema de coevolución cooperativa para lidiar con problemas de optimización multi-objetivo y las formas de descomposición de un problema que se han propuesto para esta área. Se presentan también algunos algoritmos evolutivos multiobjetivo más representativos del estado del arte que han utilizado esta técnica como esquema base y se hace una descripción de su funcionamiento.

4.1. Coevolución

En la naturaleza, la coevolución es el proceso de cambio genético recíproco en una especie, o grupo, en respuesta de otro. Es decir, la coevolución se refiere al cambio evolutivo recíproco entre especies que intenractuan entre si. Este termino surge de un estudio acerca de la interacción entre plantas y mariposas, realizado por Ehrlich y Raven [57], en el cual se examinan los patrones de interacción entre dos grandes

Interacción	**A**	**B**	**Descripción**
Neutralismo	0	0	Especies A y B son independientes y no intereactuan entre si.
Mutualismo	+	+	Ambas especies se benefician de la interacción
Competencia	-	-	Ambas especies tienen efectos negativos entre si ya que compiten por los mismos recursos
Comensalismo	+	0	Una de las especies se beneficia de la realación mientras que la otra no se ve ni afectada ni beneficiada.
Amenalismo	-	0	Una de las especies se afectada en la realación mientras que la otra no se ve ni afectada ni beneficiada.
Depredación	+	-	La especie A (depredador) se beneficia de la especie B (presa), viendose esta última afectada en la relación.

Tabla 4.1: Posibles interacciones entre dos posibles especies.

grupos de organismos como los son las plantas y los herbívoros.

Las interacciones entre dos posibles grupos de seres vivos, especies, pueden ser descritas dependiendo de sus posibles tipos de relaciones, las cuales pueden ser tanto en forma positiva como negativa, dependiendo en las consecuencias resultantes que la interacción refleja sobre los individuos de las poblaciones de cada especie. Estas interacciones se resumen en la Tabla 4.1.

Este proceso puede ser también utilizado dentro de los algoritmos evolutivos. En años recientes muchos investigadores han dirigido sus esfuerzos a estudiar mejor la coevolución como un modelo alternativo para resolver problemas multiobjetivo en computación evolutiva. Este cambio recíproco observado en la coevolución es considerado, dentro del computo evolutivo, ya sea como una carrera competitiva, como la coevolución de casos de prueba para un problema (predadores) con sus soluciones (presa) [58], o más recientemente, como enfoques cooperativos en donde sub-poblaciones

evolucionan de forma separada componentes de la solución [59]. A continuación se describen ambos esquemas.

4.1.1. Coevolución Competitiva

La coevolución competitiva es un tipo de coevolución en la cual dos especies que interactuan entre si tienen efectos negativos entre si, ya que compiten por los mismos recursos dentro de su entorno. Dentro de los algoritmos evolutivos se aplica este hecho con la idea de que la aptitud de un individuo de una población, que pertenece a una especie, se obtiene por la competencia directa de este con alguno de otra especie. Esta interacción ha sido modelada en construcciones de teoría de juegos tales como la de Mayard Smith [60] y el dilema del prisionero [61]. También es utilizada en la evolución de estrategias de juego de inteligencia artificial, donde el rango de oponentes potenciales hace dificil de establecer una sola, fija y exógena, función de aptitud, como es utilizada usualmente dentro de los algoritmos genéticos [62].

De forma más general, dentro de aplicaciones complejas de ingeniería de software, surgen numerosas situaciones en las que la construcción de funciones de prueba que verifiquen y demuestren la confiabilidad del software, es considerada casi tan importante como el mismo desarrollo del software. Es entonces que viendo a las soluciones de software como una población y a los conjuntos de prueba como otra, la coevolución competitiva permite la búsqueda de ambas al mismo tiempo [63]. Las ventajas de la coevolución competitiva son:

- Incremento de adaptabilidad por medio de la creación de una evolución de carrera de armamento.
- Surgimiento incremental de soluciones cada vez más complejas, debido a que cada población trata de ser mejor que su oponente.
- El diseño de la función de aptitud tiene un rol menos importante, lo cual conlleva a un sistema autónomo.
- El cambio continuo del paisaje de aptitud, lo cual puede ayudar a prevenir estancamiento en óptimos locales.

A continuación se mencionan algunos trabajos que se han realizado basados en el esquema competitivo:

- El primer trabajo que utiliza un esquema competitivo fue el propuesto por Hillis [64]. Este trabajo fue aplicado al problema de evolucionar redes mínimas de ordenamiento para listas de un número finito de elementos. Se observó que la coevolución de los casos de prueba en conjunto con los resultados de las redes de ordenamiento dan como resultado una mejora al procedimiento. Se utilizan dos poblaciones independientes: una de redes de ordenamiento (tomada como el huésped) y la otra como los casos de prueba (parasitos). La aptitud de

cada red de ordenamiento es medida por su habilidad de resolver de forma correcta los problemas de prueba, mientras que la aptitud para caso de prueba es proporcional al numero de veces que fue ordenado de forma incorrecta por las redes. Ambas poblaciones evolucionan simultáneamente, interactuando a través de la función de aptitud.

- Barbosa y Barreto proponen un algoritmo competitivo el cual coevoluciona un vector de pesos el cual pesa las funciones objetivo [65]. Se utilizan dos poblaciones las cuales escogen las parejas de reproducción con una función de aptitud. Asignando pesos para favorecer una solución en particular es dificil y es entonces en donde la coevolución es utilizada para facilitar dicha tarea.

- Rosin y Belew [66] utilizan las coevolución competitiva para problemas de aprendizaje de juegos (*game-learning*) aplicados a tres juegos: el juego del gato, mim y una versión reducida del juego de *go*. Consideran un esquema competitivo en el cual la aptitud de un individuo de una población "huésped" esta basada en la competencia directa con individuos de una población de "parasitos". Asi, los individuos de cada población son turnados para ser probados y para probar a otros, para la población huésped cuando la aptitud se esta midiendo y para la de parasitos cuando esta esta siendo usada para medir la aptitud. Entonces, como los parasitos también evolucionan por medio de un aptitud basada en competencia, el éxito del huésped depende del fracaso de sus parasitos y viceversa. El objetivo principal de este trabajo es entonces, el descubrimiento de diversos y al mismo tiempo consistentes de diversos conjuntos de parasitos que valgan la pena ser derrotados.

4.1.2. Coevolución Cooperativa

La coevolución cooperativa es un proceso en el cual dos especies interactúan entre sí de tal forma que ambas resultan beneficiadas en la interacción. En el caso de los algoritmos evolutivos, se aplica este concepto para la asignación de aptitud de los individuos, teniendo que este es el resultado de una *colaboración* entre los individuos de las especies. El primer modelo de coevolución cooperativa fue propuesto por Mitchell A. Potter y Kenneth A. De Jong [51], se basa en modelar un ecosistema que consiste de dos o más especies, las cuales tienen una relación ecológica de mutualismo. Dichas especies cooperan entre ellas siendo recompensadas basándose en qué tan bien trabajan juntas en la solución de un problema. Se busca automatizar la forma en que las especies son creadas, para que así estas emerjan de acuerdo a las necesidades de búsqueda del problema en lugar de ser creadas de forma predefinida por el usuario.

La idea principal de esta técnica es descomponer un problema de alta dimensión en subcomponentes de baja dimensión y evolucionar estos cooperativamente por un número predefinido de ciclos. Un ciclo consiste en una evolución completa de cada una de las subpoblaciones por un número de generaciones establecido. La cooperación ocurre tan solo cuando se evalúan a los individuos, ya que es aquí en donde se lleva

a cabo la interacción entre los individuos de cada una de las especia. Un individuo se considera más apto entre más ayude a los demás a mejorar la solución al problema. Por lo general, la mayoría de las propuestas que utilizan un esquema basado en coevolución cooperativa tienden a utilizar una especie por cada variable de decisión que contenga el problema.

La idea principal de un algoritmo con coevolución cooperativa es la siguiente:

- Una especie representa un subcomponente de una solución potencial.
- Las soluciones completas se obtienen de la unión de miembros representativos de cada especie.
- La asignación de crédito a nivel especie se define en términos del resultado de la colaboración entre especies.
- Cuando se requiere el número de especies evoluciona por sí misma.
- La evolución de cada especie es manejada por un algoritmo evolutivo.

La estructura del modelo original de coevolución propuesto por Mitchell A. Potter y Kenneth A. De Jong en [51] se puede resumir entonces en lo siguiente:

1. Descomponer el vector objetivo en m subcomponentes de menor dimensión.
2. Iniciar un índice i=1 para comenzar un nuevo ciclo.
3. Optimizar el i-esimo subcomponente con un cierto algoritmo evolutivo por un número predefinido de evaluaciones de *fitness* (FE).
4. Si $i < m$ entonces volver al paso 3.
5. Parar si el criterio de parada es satisfecho, en caso contrario volver al paso 2 para comenzar otro ciclo.

Aquí, cada ciclo consiste en una evolución completa de cada una de las subpoblaciones (especies) por un número de generaciones establecido.

La primer versión de esta propuesta fue nombrada como *Cooperative Coevolutionary Genetic Algorithm* 1 (CCGA-1). En ésta comienzan creando una especie de individuos por cada variable de decisión del problema. El valor de aptitud inicial de cada individuo para cada subpoblación se asignaba mediante combinar algún miembro de las demás especies, tomados de forma aleatoria, y aplicando el vector resultante de variables de decisión sobre la función objetivo. Después de esta inicialización,

cada subpoblación es *coevolucionada* con una planificación estilo Round-Robin [51], utilizando como motor de búsqueda un Algoritmo Génetico (AG) [47] tradicional.

Se realiza la interacción entre los individuos de cada especie, a la hora de realizar la evaluación de la función objetivo, tomando al individuo siendo evaluado y haciendo que éste interactúe con los mejores individuos de cada especie restante para conformar una posible solución al problema. Se asigna una aptitud a cada individuo dependiendo que tan bien interactúa con los de las otras especies especies. Para esta versión se realizaron pruebas con las funciones de Rastrigin, Schwefel, Griewangk, y Ackley, descubriendo que la forma de interacción entre las especies de esta primer versión tiene problemas como convergencia prematura y baja diversidad.

Como respuesta a estos problemas, se presentó una segunda versión nombrada como *Cooperative Coevolutionary Genetic Algorithm* 2 (CCGA-2), en donde el único cambio realizado fue que a la hora de la interacción entre las especies para formar una posible solución. En esta versión se toman individuos de las especies de forma aleatoria, en lugar de tomar específicamente al mejor de cada una como lo hacen en CCGA-1. Con esto se lograron disminuir los problemas de convergencia prematura y baja diversidad poblacional en las especies que se presentaban en CCGA-1. Ambas técnicas demostraron ser más eficientes que el AG tradicional ya que requerían un menor número de evaluaciones sobre las funciones objetivo para alcanzar un resultado óptimo [51].

4.1.3. Coevolución Cooperativa en Algoritmos Evolutivos Multiobjetivo

La coevolución cooperativa se ha extendido al plano de optimización multiobjetivo desde años recientes, esto debido a que los conceptos son adaptables a este campo y a que se han buscado nuevas rutas para dar solución a este tipo de problemas. En esta sección se presentan los principales modelos dentro de la optimización multiobjetivo con algoritmos evolutivos que utilizan un esquema basado en coevolución cooperativa. Esto nos concierne debido a que el modelo presentado en esta tesis se basa en este esquema.

- Uno de los primeros modelos conocidos que han propuesto un esquema basado en coevolución cooperativa para la optimización multiobjetivo es el MOCC-GA [67] (Multi-objective Cooperative Coevolutionary Genetic Algorithm). El MOCCGA es un algoritmo coevolutivo multiobjetivo que utiliza como motor de búsqueda un algoritmo genético multiobjetivo conocido como MOGA [68] (Multi-objective Genetic Algorithm). La idea principal de este algoritmo es descomponer el problema, por medio de la asignación de una función objetivo a cada especie y mediante una interacción de cooperación coevolutiva, para así evolucionar las especies hasta obtener un conjunto de posibles soluciones. En este esquema se crean tantas especies como variables de decisión para el problema existan, ya que en este cada especie representa una sola variable de

decisión del problema. Sin embargo, en lugar de asignar de forma directa un valor de aptitud al individuo de interés, el cual participa en la construcción de una solución completa, un valor de rango es determinado primero, el cual realiza un conteo del número de individuos que dominan a un individuo para entonces poder asignar una aptitud a este.

Al igual que en MOGA, el rango de cada individuo se obtiene después de compararlo con los demás individuos de su especie. En MOCCGA los objetivos son evaluados dos veces para cada individuo, una vez colaborando con un individuos al azar de las demás especies y otra con los mejores, llevando a la practica la propuesta de Potter y De Jong presentada en el CCGA [51] para minimizar la convergencia prematura que ocurría en ciertos problemas y poder determinar cual de las dos colaboraciones da un mejor resultado. Esto sin embargo, hace que aumente al doble el número de evaluaciones sobre el vector de funciones objetivo. En el MOCCGA la asignación de rangos en cada especie se hace solo dentro de cada una de sus poblaciones, lo cual limita potencialmente la valoración de los individuos. Cabe mencionar que no se reportan el número de evaluaciones necesarias para dar soluciones a los problemas que presentan.

- Tan et. al. [69] propone un esquema cooperativo en el cual, al igual que en MOCCGA, una población es definida por cada variable de decisión. En este, la evolución de cada una especie es controlada por el método de Fonseca y Fleming presentado en su propuesta MOGA [68]. Ademas, se hace uso de un archivo externo para guardar y actualizar las soluciones no-dominadas encontradas hasta el momento y también para guiar la búsqueda a las zonas menos explotadas en el espacio de búsqueda.

- Lorio y Li utilizan el esquema de NSGA-II [1] para evolucionar las poblaciones de cada una de las especies, creando un algoritmo al cual nombran NSCC-GA [70]. El algoritmo comienza inicializando a los individuos de cada población de forma aleatoria. Cada especie es responsable de una variable de decisión del problema, ósea que, habrán tantas especies como variables de decisión. Aquí, la colaboración se realiza tomando individuos que estén al frente del ordenamiento de no-dominados, para conformar una solución completa al problema. Se aplica un operador de elitismo el cual remueve un cierto numero de peores individuos de cada especia, el cual sea proporcional al número de descendencia que pasara a formar parte de la siguiente generación de cada subpoblación de cada especie. Permitiendo así que el tamaño de las poblaciones sea constante al mismo tiempo que conserva a los mejores individuos. Después de cada generación se aplica el ordenamiento de no-dominados para continuar con el proceso de interacción de la cooperación coevolutiva.

- Parmee y Watson [71] proponen un algoritmo genético coevolutivo multiobjetivo para el diseño de la estructura de un avión. Esta esquema coevolutivo divide el problema por número de funciones objetivo, es decir, asigna a cada especie

una función objetivo del problema multiobjetivo. Los autores proponen utilizar algoritmos genéticos (AGs) individuales para la optimización de cada objetivo, con lo cual tratan de hacer concurrente cada proceso que se encarga en especifico de cada uno de los objetivos por separado, haciendo uso de una maquina virtual para la paralelización de estos. Para cada generación, las soluciones relacionadas con los otros objetivos son comparadas con los mejores individuos de los AGs de las otras especies. Se utiliza la penalización de funciones para el control de los limites de las funciones.

Este método es diseñado con el objetivo especifico de converger a una sola e ideal solución compromiso. Sin embargo, a pesar de que el uso de las penalizaciones permite mantener diversidad en las poblaciones de cada especie, estas penalizaciones están relacionadas con la variabilidad de los valores de las variables de decisión. Este esquema también guarda las soluciones producidas durante el proceso de evolución para que así el usuario pueda analizar el camino recorrido por el algoritmo para llegar a la solución. Las ventajas de esta propuesta son las siguientes:

- Las soluciones óptimas locales pueden ser identificadas después de las primeras generaciones.
- Una region factible para el problema es identificada.
- Se tiene disponible información sobre la importancia y la redundancia de los parámetros de entrada durante la ejecución.
- Toda la información sobre el problema se produce en una sola ejecución del algoritmo.

- Tan et. al. presentan un algoritmo de coevolución cooperativa distribuido llamado DCCEA [72] (*Distributed Cooperative Coevolutionary Multiobjective Evolutionary Algorithm*). Este utiliza una técnica de asignación de rangos similar a la presentada en MOCCGA [67], con la principa diferencial de que el conteo de dominancia se realiza con los mejores individuos encontrados hasta el momento, los cuales son almacenados en un archivo. Cada subpoblación (especie) optimiza un solo parámetro por lo que un individuo de una población perteneciente a una especie es tan solo un componente de una solución. Aquí, el representante tomado para la colaboración es el mejor de cada especie y al igual que en las propuestas mencionadas previamente, para poder evaluar a cada individuo se forman colaboraciones en las cuales se toman a los representantes de cada especie y se forma una solución completa al problema.

  DCCEA adopta una arquitectura de paralelización de grano grueso. Para adaptarse a una estructura distribuida, se toman en cuenta muchas características de computo distribuido como las variantes de comunicación, distintas velocidades de computo y restricciones de red. En esta propuesta, se utiliza un número relativamente grande de evaluaciones sobre la función objetivo para lograr demostrar mejoras sobre NSGA-II [1].

- Coello et. al. [73] propone un algoritmo cooperativo coevolutivo para problemas de optimización multiobjetivo. Este a diferencia de los otros esquemas, subdivide el espacio de variables de decisión en lugar de separar el problema por medio de la asignación de las variables de decisión a cada una de las especies como lo hacen las propuestas anteriores. Por medio de esta subdivisión del espacio de búsqueda del problema, trata de encontrar las regiones mas prometedoras para enfocar su búsqueda dentro de ellas. Se determina que proporciones de los intervalos de las variables de decisión están siendo utilizadas y se descartan entonces las que considera que no están siendo utilizadas en el proceso de búsqueda. Para esto realiza un análisis sobre el frente de Pareto conseguido hasta el momento. Asi mismo, subdivide el espacio de búsqueda en intervalos, para que así las subpoblaciones (especies) puedan operar en porciones de éstos y prestar mayor atención a las cuales se considere que contribuyen a la búsqueda. En esta propuesta, las especies que se considera que no contribuyen en el proceso son eliminadas de manera automática.

Estas son algunas de las técnicas más relevantes que se basan en coevolución cooperativa para dar solución a problemas multiobejtivo. Como se puede observar, algunos de estos esquemas necesitan de algún conocimiento previo del problema o un números conjunto de parámetros de entrada. Es importante observar que la característica principal de la mayoría de estas propuestas basadas en coevolucion cooperativa es que optimizan las distintas variables de decisión del problema utilizando poblaciones distintas, una para cada variable de decisión, razón por la cual, la principal desventaja de éstas es que la posible interdependencia que podría existir entre las variables pudiera causar algunas dificultades cuando las variables son optimizadas de forma separada. Las otras propuestas dividen el problema en termino de los objetivos, esto sin embargo no seria de mucha utilidad para lidiar con la escalabilidad del número de variables, más bien quizás pudiese ser de ayuda para escalar el número de funciones objetivos, si es que ese fuera el caso. Es importante resaltar que a pesar de que estos algoritmos presentados están basadas en coevolución cooperativa, ninguno de ellos ha sido diseñado con el propósito especifico de lidiar con la escalabilidad (en el espacio de las variables de decisión) de problemas de optimización multiobjetivo.

Capítulo 5

Modelo Propuesto

Tomando en cuenta los mecanismos existentes para abordar el problema de la alta dimensionalidad en el espacio de las variables de decisión, en el área de optimización mono-objetivo, se decidió crear un modelo multiobjetivo que utilice el paradigma de coevolución cooperativa. Esto debido a su éxito en problemas de optimización global de alta dimensionalidad [5]. La idea principal del modelo propuesto es hacer uso de la técnica de divide y vencerás que se utiliza en el esquema de coevolución cooperativa (como el utilizado en CCGA [51]), pero transfiriendo este concepto a la optimización multiobjetivo. Se adoptaron también algunos conceptos adicionales tomados de modelos basados en coevolución cooperativa encontrados en la literatura especializada [5].

En este capítulo se describe un modelo que es capaz de resolver, de una mejor manera, problemas de optimización multiobjetivo de alta dimensionalidad en el espacio de las variables de decisión. Nuestra propuesta es un algoritmo basado en coevolución cooperativa, el cual, utiliza como motor de búsqueda el algoritmo GDE [38] (*Generalized Differential Evolution 3*). Hemos nombrado a esta propuesta como *Cooperative Coevolutionary Generalized Differential Evolution Algorithm 3* (CCGDE3), y la describimos a detalle a continuación.

5.1. Descripción del esquema

En esta sección se describe el esquema general del modelo propuesto. Se muestra la forma en que es aplicada la coevolución cooperativa, lo cual involucra la creación de las especies, la interacción entre éstas y la forma en que se realiza la colaboración, así como las adaptaciones de estos conceptos para lidiar con problemas de optimización multiobjetivo.

5.1.1. Creación de las especies

Por medio de los resultados que se encuentran en la literatura especializada, que proceden de técnicas que utilizan un modelo de coevolución cooperativa, se sabe que

dicho modelo tiene éxito en problemas de alta dimensionalidad de funciones separables, debido a que es posible optimizar de forma independiente las variables de decisión. Sin embargo, en su versión original, este modelo presenta problemas en funciones no separables [53]. Esto quiere decir que el vector de variables de decisión consiste de variables que interactúan entre sí, o sea que de algún modo no son del todo independientes. Esto no lo toma en cuenta el modelo original, ya que este separa en su totalidad cada una de las variables y las coloca en una subpoblación distinta, como es el caso de CCGA [51]. Por tanto, que dado un problema de optimización en el cual se tenga una función no separable, y si todas sus variables son altamente interdependientes entre sí, entonces ningún algoritmo de coevolución cooperativa podrá tener un buen desempeño en un caso tan extremo. Entonces, al presentarse una influencia entre las variables, la separabilidad se vuelve un poco más difícil de realizar. Para nuestro estudio definiremos la separabilidad de una función de la siguiente forma:

Una función $f(x)$ es separable si $\forall k \in \{1, n\}$ y

$$\left.\begin{array}{ll} x \in S, & x = (x_1, \cdots, x_k, \cdots, x_n) \\ x' \in S, & x' = (x_1, \cdots, x'_k, \cdots, x_n) \end{array}\right\} \Rightarrow f(x) < f(x')$$

implica

$$\left.\begin{array}{ll} \forall y \in S, & y = (y_1, \cdots, x_k, \cdots, y_n) \\ \forall y' \in S, & y' = (y_1, \cdots, x'_k, \cdots, y_n) \end{array}\right\} \Rightarrow f(y) < f(y')$$

de otra forma, diremos que se trata de una función no separable.

Básicamente, la no separabilidad de una función quiere decir que el vector de variables consiste de variables que ínteractúan entre sí, mientras que la separabilidad significa que el vector de variables en el valor de aptitud depende tan solo de sí mismo, es decir, las variables son independientes entre sí.

Dado este problema, se ha reportado que dividir el vector de variables de decisión de un problema en grupos aleatorios, tal que existan dos o más subpoblaciones, las cuales son además asignadas con cualesquiera variables de decisión del problema en cuestión. Esto proporciona mejores resultados que utilizar un esquema de división determinista, cuando se lidia con problemas de optimización, cuyas funciones son no separables [53, 55].

Motivados por esta investigación previa, nuestro modelo propuesto divide el vector de variables de decisión en S subpoblaciones, cada una representando un subconjunto del vector de variables de decisión a la vez en lugar de sólo representar una sola variable de decisión. Nuestro esquema asigna cada variable de decisión a un grupo (subpoblación) correspondiente, de forma aleatoria. Esto significa que una variable de decisión tendrá la misma probabilidad de pertenecer a una especie dada que todas

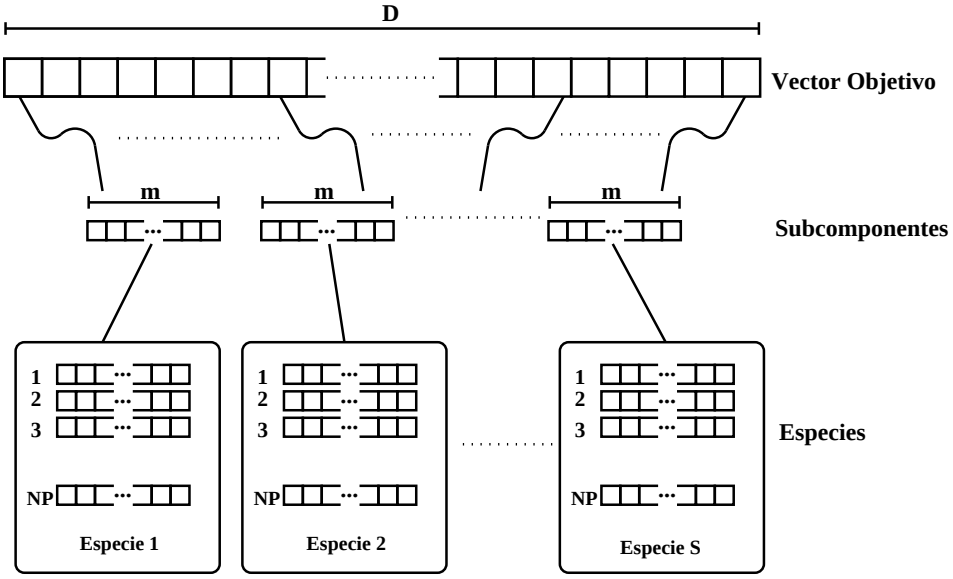

Figura 5.1: Representación gráfica de la creación de especies. Se tiene un vector de variables de decisión de dimensión D el cual se divide en S subcomponentes de dimensión m los cuales son creados de forma aleatoria a partir del vector de variables de decisión original y son asignados a las S especies existentes, donde $D = m * S$.

las demás variables. Esto permitirá entonces incrementar la probabilidad de que las variables de decisión que tengan interdependencia entre sí, sean optimizadas juntas.

Tomando en cuenta lo descrito anteriormente, nuestro modelo realiza lo siguiente para la creación de las especies: divide el vector de variables de decisión $\vec{x}$ de dimensión $D \in \mathbb{N}$ en $S \in \mathbb{N}$ subcomponentes del mismo tamaño. Cada subcomponente es creado a partir de un agrupamiento aleatorio de variables de decisión, para así incrementar la probabilidad de agrupar variables que necesitan interactuar entre sí y ser asignadas a una misma especia en problemas no separables. A las vez, se crean S subpoblaciones (especies), cada una de las cuales está compuesta por NP individuos los cuales tienen las variables asignadas de forma aleatoria a esa población, vectores de dimensión m, lo cual indica que los individuos de esa subpoblación serán de una determinada especie y que serán distintos a los de las demás subpoblaciones. Esto se muestra de forma gráfica en la Figura 5.1.

5.1.2. Proceso principal del esquema

Una vez teniendo las especies creadas, se realiza una inicialización aleatoria de cada uno de los NP individuos, de cada una de las poblaciones que pertenecen a cada una de S las especies, las cuales representan una parte del vector de variables de decisión. Después, el algoritmo ejecuta *ciclos*, dentro de los cuales se realiza la evolución de cada una de las S subpoblaciones por un número determinado de *generaciones*.

Esto continúa hasta cumplirse con un criterio de paro establecido, y al final,

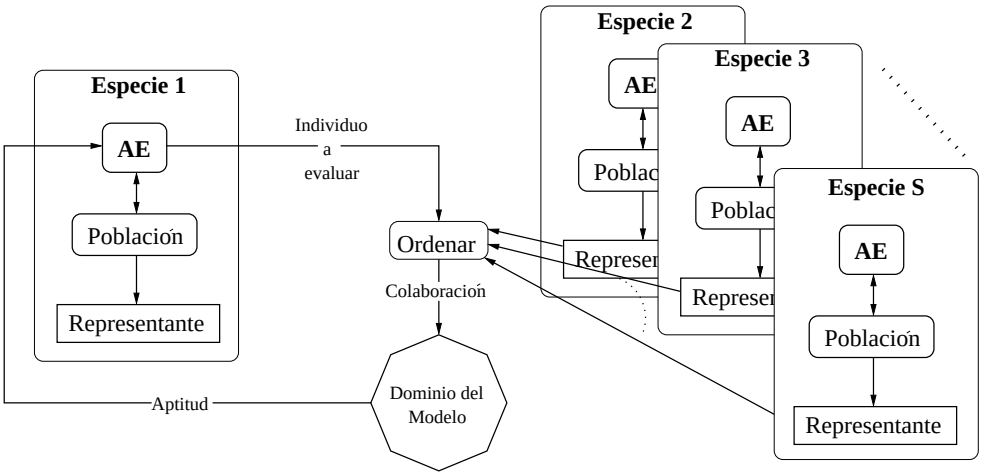

Figura 5.2: Estructura de interacción de especies de la coevolución cooperativa, para el modelo propuesto, desde la perspectiva de la especie número 1. Se tienen S especies, en donde los representantes de cada una para la colaboración son tomados al azar del mejor nivel de no dominados de cada especie.

las soluciones que son globalmente *no dominadas*(es decir, con respecto a todas las especies), constituyen la salida del algoritmo.

5.1.3. Colaboración de individuos

La colaboración entre las subpoblaciones (especies) se realiza de la siguiente manera: En la primera generación se forman colaboraciones aleatorias para ser evaluadas. Cada una de estas colaboraciones se hace tomando un individuo al azar ,de cada una de las demás subpoblaciones, y uniéndolo con el individuo que está siendo evaluado, para así formar una solución completa que es evaluada en el vector de funciones objetivo del problema. Posteriormente el resultado obtenido de dicha evaluación es devuelto al individuo que está siendo evaluado, para así obtener su aptitud y saber a partir de ésta, qué tan buena fue su colaboración con los demás individuos de las otras especies.

Después de esta primera generación, las poblaciones descendientes resultantes, desde Q_1 hasta Q_S (resultado de los operadores específicos del algoritmo evolutivo multiobjetivo que se usa como optimizado básico) de cada una de las especies, serán evaluadas formando colaboraciones con los componentes, seleccionados de forma aleatoria, del *mejor* nivel de no dominados en cada una de las poblaciones, de cada una de las especies, desde P_1 hasta P_S, de la generación previa. Una vez formadas estas colaboraciones se evalúan en las funciónes objetivo del problema, para asignar la aptitud correspondiente al individuo siendo evaluado. Este proceso se muestra de forma gráfica en la Figura 5.2.

El algoritmo itera de esta forma hasta que se cumpla alguna condición (usualmente un número predefinido de *ciclos*). Al final, se aplica un procedimiento de no

dominados, como el presentado en NSGA-II [1] a los mejores niveles de no dominados de cada una de las poblaciones de las especies, para obtener un conjunto final de soluciones para el problema que está siendo resuelto.

En el Algoritmo 1 se muestra el pseudocódigo del modelo propuesto. El esquema general del modelo de optimización evolutiva multiobjetivo con coevolución cooperativa, para optimización a gran escala en el espacio de las variables de decisión, en el cual se basa el algoritmo propuesto, se presenta en la Figura 5.3.

Algoritmo 1: Pseudocódigo de modelo de Coevolución Cooperativa para larga escala donde NP es el tamaño de cada subpoblación, $Ciclos$ es el número de ciclos a ejecutar en la coevolución, $Gmax$ son el número máximo de generaciones que se evolucionarán las especies por ciclo y $NumEsp$ es el número de especies a crear.

Entrada: NP, $Ciclos$, $Gmax$, $NumEsp$
Salida: $ConjuntoDeSoluciones$

```
1 Pobs = crearPoblaciones(NP, NumEsp);
2 inicializarEspecies(Pobs);
3 for j = 1 to Ciclos do
4     for i = 1 to NumEsp do
5         for k = 1 to Gmax do
6             AEMO(Pobs[i]);
7 ConjuntoDeSoluciones =obtenerConjuntoDeNoDominados(Pobs);
8 return ConjuntoDeSoluciones ;
```

El modelo aquí presentado puede adaptarse fácilmente a cualquier algoritmo evolutivo multiobjetivo (AEMO), ya que éste utiliza principalmente los operadores de cruza y mutación para el proceso de evolución, por lo cual ambos operadores de cualquier AEMO pueden ser adaptados para trabajar en conjunto con el modelo propuesto.

5.2. GDE3

La propuesta presentada en esta tesis adopta el algoritmo GDE3 como su optimizador multiobjetivo básico. Esto se debe a que en los estudios presentados por Durillo et. al [4, 3] acerca de las metaheurísticas y su comportamiento al incrementarse el número de variables de un problema multiobjetivo, GDE3 fue uno de los algoritmos que mejores resultados obtuvieron. De tal forma, consideramos adecuado usar GDE3, ya que además de mostrar buen desempeño, es relativamente fácil de implementar y conserva varias características de un algoritmo mono-objetivo, lo que facilita su acoplamiento al esquema co-evolutivo adoptado. A continuación se da una descripción

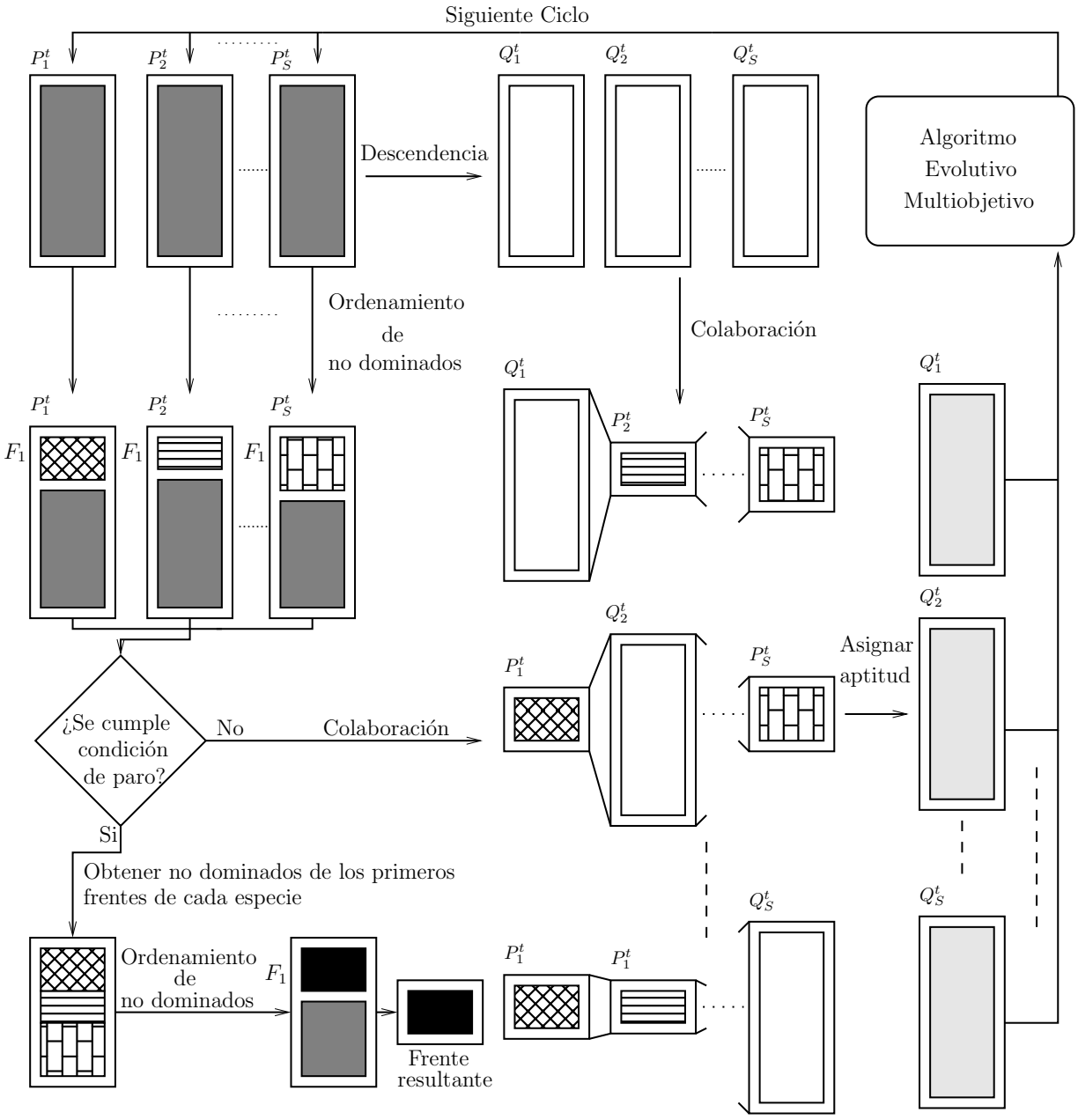

Figura 5.3: Esquema general del modelo propuesto. Q_i^t son los descendientes de cada especie y P_i^t son los padres de esos descendientes en el ciclo t. Hay S subpoblaciones donde S es el número de especies creadas y en cada una de las cuales se han asignado las distintas variables de decisión con el método descrito en la Figura 5.1.

de este algoritmo.

GDE3 [38] es la tercera versión del algoritmo conocido como *Generalized Differential Evolution* (GDE) [74]. GDE3 es capaz de resolver problemas de optimización con múltiples objectivos y en caso de tener un problema con un solo objetivo sin restricciones funciona como el algoritmo de Evolución Diferencial original, propuesto por Storn y Price en 1995 [44] (DE). GDE3 mejora su primera versión GDE en el caso multiobjetivo dando una mejor distribución en su conjunto de soluciones.

Comienza con una población de soluciones aleatorias, la cual se vuelve la población actual. En cada generación, se crea una población de descendientes utilizando los operadores de la evolución diferencial; después, la población actual para la siguiente generación es actualizada utilizando las soluciones de la población de padres y la población de hijos. La selección de GDE3 se basa en las siguientes reglas:

- En el caso de vectores no factibles, el vector de prueba nuevo, $\vec{u}$, creado a partir del los operadores de cruza y mutación de la evolución diferencial, es seleccionado si éste *domina débilmente*[1] al vector original $\vec{x}$, en el espacio en donde se violan las restricciones; de otra forma, el vector original es seleccionado.

- En el caso de que uno de los dos vectores sea factible y el otro no, se selecciona al vector factible.

- Si ambos vectores son factibles, se selecciona a aquél que domina al otro. Si ninguno de los vectores se dominan entre sí en el espacio de las funciones objetivo, entonces ambos son seleccionados para la siguiente generación.

Antes de pasar a la siguiente generación, el tamaño de la población quizás es mayor que NP. Si este es el caso, el tamaño de la población se reduce utilizando un ordenamiento de no dominados y una técnica de "poda" similar a la utilizada en NSGA-II [1]. Los vectores son ordenados con base en la no dominacia y el hacinamiento (*crowding*). Los peores miembros de la población, según estos dos criterios, son removidos para decrementar el tamaño de la población a su tamalo original. El funcionamiento de GDE3 se muestra en el Algoritmo 2.

[1]La relación de dominancia débil entre dos vectores esta definida tal que $\vec{x}_1$ domina débilmente a $\vec{x}_2$, si y sólo si $\forall i : f_i(\vec{x_1}) \leq f_i(\vec{x_2})$.

Algoritmo 2: Pseudocódigo del algoritmo GDE3, donde NP es el número de individuos de la población, $Gmax$ son el número máximo de generaciones, CR es la probabilidad de cruza, F la magnitud de la mutación y CD es la notación para *Crowding Distance* [1].

Entrada: D, $Gmax$,$NP \geq 4$, $F \in (0, 1+]$, $CR \in [0, 1]$
Salida: $ConjuntoDeSoluciones$

Inicializar población de NP individuos de forma aleatoria;
$G = 0$;
while $G < Gmax$ **do**
  `/* Mutación y recombinación */`
  for $i = 1$ **to** NP **do**
    Tomar $r1, r2, r3 \in \{1, 2, \ldots, NP\}$, seleccionados aleatoriamente, diferentes entre si y distintos de i;
    Tomar $j_{rand} \in \{1, 2, \ldots, D\}$;
    for $j = 1$ **to** D **do**
      if $rand_j \in [0, 1) < CR \vee j = j_{rand}$ **then**
        $u_{j,i,G} = x_{j,r3,G} + F \cdot (x_{j,r1,G} - x_{j,r2,G})$;
      else
        $u_{j,i,G} = x_{j,i,G}$;
  `/* Selección */`
  for $i = 1$ **to** NP **do**
    if $\vec{u}_{i,G} \preceq_c \vec{x}_{i,G}$ **then**
      $\vec{x}_{i,G+1} = \vec{u}_{i,G}$;
    else
      $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
  $m = 0$;
  for $i = 1$ **to** NP **do**
    if $\forall j : g_j(\vec{u}_{i,G}) \leq 0 \wedge \vec{x}_{i,G+1} == \vec{x}_{i,G} \wedge \vec{x}_{i,G} \not\prec \vec{u}_{i,G}$ **then**
      $m = m + 1$;
      $\vec{x}_{NP+m,G+1} = \vec{u}_{i,G}$;
  `/* Poda */`
  while $m > 0$ **do**
    Seleccionar $\vec{x} \in \{\vec{x}_{1,G+1}, \vec{x}_{2,G+1}, \ldots, \vec{x}_{NP+m,G+1}\}$ **tal que** $\{\forall i : \vec{x} \not\prec_c \vec{x}_{i,G+1} \wedge \forall(\vec{x}_{i,G+1} : \vec{x}_{i,G+1} \not\prec_c \vec{x})$ `CD(`$\vec{x}$`)`$\leq$ `CD(`$\vec{x}_{i,G+1}$`)`$\}$;
    `Remover(`$\vec{x}$`)`;
    m = m - 1;
  $G = G + 1$
return $ConjuntoDeSoluciones = \{\vec{x}_{1,G}, \vec{x}_{2,G}, \ldots, \vec{x}_{NP,G}\}$;

Capítulo 6

Resultados Experimentales

En este capítulo se presentan los resultados de los estudios experimentales que se realizaron sobre nuestro modelo propuesto, al cual decidimos llamar *Cooperative Coevolutionary* GDE3 (CCGDE3). Para validar el desempeño del algoritmo propuesto, se adoptó un conjunto de problemas de prueba escalable en el número de variables de decisión: el conjunto *ZitzlerDeb-Thiele (ZDT)* [40]. Además, este conjunto de problemas de prueba incluye características diversas del frente de Pareto, lo cual es también deseable.

6.1. Problemas de prueba

Se tomaron cuatro problemas de este conjunto de prueba: los denominados ZDT1, ZDT2, ZDT3 y ZDT6. Las principales características de estos problemas se describen a continuación.

- ZDT1: Tiene un frente de Pareto convexo el cual es continuo y uniformemente distribuido. Tiene ademas una densidad uniforme de soluciones en el espacio de búsqueda.
- ZDT2: Tiene un frente de Pareto no convexo el cual es continuo y uniformemente distribuido. Tiene además una densidad uniforme de soluciones en el espacio de búsqueda.
- ZDT3: Tiene un frente de Pareto que está compuesto de cinco regiones discontinuas. Presenta también una densidad de soluciones uniforme en el espacio de búsqueda. La principal complejidad de este problema es encontrar todas las regiones discontinuas del frente de Pareto.
- ZDT6: Tiene un mapeo no uniforme entre el espacio de las funciones objetivo y el espacio de las variables de decisión. Tiene un frente de Pareto no convexo.

En el apéndice 7 se muestran más a detalle los problemas de prueba utilizados.

Se compara la propuesta CCGDE3 con respecto a dos AEMOs: GDE3 [38] y NSGA-II [1]. En los experimentos, se utilizó un gran número de variables de decisión que va desde las 200 hasta las 5000.

6.1.1. Metodología

El principal objetivo de este trabajo es evaluar el comportamiento de nuestra propuesta (CCGDE3) al momento de resolver problemas de optimización multiobjetivo (POMs) con un gran número de variables de decisión. Para ello se decidió analizar la tasa de convergencia del algorítmo con respecto a la de los otros dos AEMOs contra los que se comparó (GDE3 y NSGA-II). Para calcular dicha tasa de convergencia, se adoptó el indicador de desempeño denominado hipervolumen [27].

Hipervolumen

El hipervolumen se obtiene calculando el volumen (en el espacio de las funciones objetivo) del conjuntos de soluciones no dominadas Q que minimizan el POM. Para cada solución $i \in Q$, un hipercubo v_i es generado con un punto de referencia W y la solución i como la esquina diagonal del hipercubo. El punto de referencia W se puede generar mediante la construcción de un vector de los peores valores del vector de funciones objetivo. El hipervolumen (HV) se calcula entonces como la unión de todos los hipercubos encontrados, como se muestra a continuación:

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{6.1}$$

Ya que el verdadero frende de Pareto de los problemas ZDT es conocido, se decidió ejecutar cada uno de los AEMOs a compararse, hasta que éstos obtuvieran una aproximación del frente verdadero de cada problema que tuviera un 95 % del hipervolumen con respecto al hipervolumen total del verdadero frente.

Esto representa una aproximación razonable al verdadero frente en términos de convergencia y anchura del frente, que son dos de los objetivos principales que se persiguen al resolver un POM. El propósito de este estudio fue identificar cuál de los AEMOs que estábamos comparando es capaz de alcanzar de forma más rápida el verdadero frente de Pareto. Los puntos de referencia utilizados para cada problema se presentan en la Tabla 6.1.

Debido a que era posible que algún AEMO no pudiera alcanzar la convergencia deseada, decidimos usar como una condición de paro adicional alternativa, un número máximo de evaluaciones de las funciones objetivo. De tal forma, establecimos un número máximo de 10 millones de evaluaciones para cada problema. En nuestros experimentos, se verificó la condición de paro de convergencia cada 100 evaluaciones sobre el vector de funciones objetivo para GDE3 y NSGA-II, lo que significa que la condición fue verificada a cada iteración, y cada ciclo para CCGDE3 (todo esto

Tabla 6.1: Puntos de referencia

Problema	**Punto de Referencia**
ZDT1	(1.1, 1.1)
ZDT2	(1.1, 1.1)
ZDT3	(0.9, 1.1)
ZDT6	(1.1, 1.1)

debido a los parámetros iniciales establecidos, los cuales se describen a detalle más adelante).

Se efectuaron 25 ejecuciones independientes para cada algoritmo e instancia de cada problema, usando un número de variables de decisión que va de las 200 a las 5000. Debido a que estamos tratando con algoritmos estocásticos, necesitamos realizar un análisis estadistico de los resultados obtenidos para así poder tener cierto nivel de confiabilidad estadística en ellos. Para ello, se realizó una prueba conocida como *bootstraping* [75].

Bootstrap

El análisis de datos tradicional, que consiste en derivar una distribución muestral y calcular con ella la probabilidad de la muestra, se basa en diferentes suposiciones, tales como que los datos son independientes entre sí, o que los datos se ajustan a una cierta distribución específica, o a que un estimador posee una distribución dada, etc. La mayoría de estas suposiciones se justifica cuando existe conocimiento previo del problema o cuando se a realizado una verificación empírica. Sin embargo, en los casos donde tales suposiciones no se cumplen, su relevancia es cuestionable.

Existe una clase de métodos basados en la simulación del proceso de muestreo, los cuales son llamados métodos estadísticos de cómputo intensivo. En general, las suposiciones en las que se basan estos métodos son menos que las de los métodos tradicionales. A continuación se muestran algunas de las ventajas de estos métodos basados en la simulación del proceso de muestreo [75]:

- Se necesitan menores suposiciones ya que no se requiere que las muestras se ajusten a una distribución normal o que las muestras sean de gran tamaño.
- Son más precisos en la práctica que los métodos clásicos.
- Los métodos de remuestreo son muy similares para un amplio rango de estadísticas, y no requieren una fórmula por cada una de ellas, por lo cual tienen mayor generalidad.

Dentro de estas técnicas se encuentra el método de Bootstrap. Los métodos de *bootstrap* son procedimientos de simulación del proceso de muestreo que han sido

ampliamente utilizados en la determinación de estimadores estadísticos, en la construcción de intervalos de confianza, en el cómputo de probabilidades en una prueba de hipótesis, etc. La idea de estos métodos consiste en generar una distribución muestral para una estadística dada, empleando un remuestreo de Monte Carlo y considerando a la distribución de la muestra como la distribución de la población misma. La distribución resultante es llamada Distribución Muestral de Bootstrap (DMB) y puede ser calculada para casi cualquier estadística; el único requerimiento es que la muestra sea representativa de la población [76, 77]. La validez de *bootstrap* consiste en que la distribución muestral es el mejor estimador de la distribución de la población. Si una muestra contiene N puntos muestrales, entonces el estimador de máxima probabilidad de la distribución de la población es determinado asignando a cada punto muestral una probabilidad de $1/N$ [76, 77].

Ya que el procedimiento de *boostrap* puede determinar distribuciones para muchas estadísticas, tales como: la media, la media recortada, la distancia intercuartil, etc., sin requerir normalidad en las muestras, los estudios estadísticos presentados en este capítulo se basan en este método. Esto debido a que, al tratar con problemas de alto costo computacional (como lo son los POMs de alta dimensionalidad en el espacio de las variables de decisión), cada ejecución consume cantidades importantes de tiempo, por lo cual, no se puede contar con un gran número de muestras de cada problema de prueba para cada algoritmo, debido al tiempo disponible que se tiene para las pruebas el cual es limitado. Para ello se han realizado cálculos basados en 1000 re-muestreos de nuestro conjunto original de 25 ejecuciones independientes de cada algoritmo e instancia de cada problema. Esto para poder obtener la media y el error estándar de la distribución de *bootstrap* (DBM) y así calcular intervalos de confianza. Para ello se utilizó medio intervalo del percentil ajustado de bootstrap (BCa) [75] a fin de obtener resultados más exactos, con un nivel de confianza del 95 % para la media.

6.1.2. Parametrización

Los parámetros para cada AEMO utilizado en nuestro estudio fueron seleccionados de tal forma que se pudiera realizar una comparación lo más justa posible entre ellos. Por esto, para NSGA-II y GDE3, utilizamos una población de 100 individuos. Para CCGDE3, se utilizaron tamaños de población de 40 individuos para cada especie, ya que para tener un balance en el número de individuos de todos los algoritmos se utilizaron dos especies para el esquema coevolutivo de nuestra propuesta. En el caso de NSGA-II, los índices de distribución para los operadores SBX y *polynomial-based mutation* [1], fueron establecidos como: $\eta_c = 20$ y $\eta_m = 20$, respectivamente. La probabilidad de cruza es $p_c = 0.7$ y la probabilidad de mutación $p_m = 1/L$, donde L es el número de variables de decisión en el POM. Para el caso de GDE3 y CCGDE3, los valores para ambos operadores F y CR [38], fueron establecidos con el mismo valor de 0.5 para los dos algoritmos, para así poder observar el impacto de nuestro esquema sobre el algoritmo base. Nuestra propuesta utiliza 2 especies, cada una con un tamaño de población de 40 individuos, utilizando solo una generación por cada

especia para cada ciclo en la evolución. Esto con el fin de tener más o menos el mismo tamaño de población que los otros dos AEMOs y así poder realizar una comparación lo más justa posible. Finalmente como se mencionó anteriormente, se utilizaron las siguientes cantidades de variables de decisión: 200, 500, 1000, 2000, 3000, 4000 y 5000.

6.1.3. Análisis de Resultados

En nuestros experimentos, se obtuvo la media del número de evaluaciones necesarias para GDE3, NSGA-II y CCGDE3 para generar una aproximación del 95 % de hipervolumen del verdadero frente de Pareto de cada problema, calculada con un procedimiento de *bootstrap* con 1000 remuestreos.

También se reporta la media del tiempo que le toma a cada algoritmo resolver cada problema, medido sobre la muestra original de 25 corridas independientes. Las tablas 6.2, 6.3, 6.4 y 6.5 muestran la media, el error estándar (*standard error*) y los intervalos de confianza del número del número de evaluaciones, obtenido por medio del procedimiento de *bootstrap*. Las tablas 6.6, 6.7, 6.8 y 6.9 muestran el tiempo promedio, en minutos, necesario para obtener una aproximación al verdadero frente de Pareto de cada problema con un 95 % del hipervolumen real. En estas tablas se muestra los resultados para los problemas ZDT1, ZDT2, ZDT3 y ZDT6, respectivamente. Cuando un valor de 10,000,000 aparece en la tabla para alguno de los algoritmos en alguno de los problemas, quiere decir que no fue capaz de obtener una aproximación aceptable para el problema en ninguna de las 25 corridas independientes originales. En este caso, el tiempo promedio reportado aparece en *itálicas*, debido a que, en estos casos, se requeriría de mucho más tiempo del reportado para obtener la convergencia deseada, ya que este tiempo reportado es el que le toma a los algoritmos en cuestión para llegar al criterio de paro adicional, sea, a las 10,000,000 evaluaciones sobre el vector de funciones objetivo.

Tabla 6.2: Evaluaciones para ZDT1

	NSGA II			**GDE3**			**CCGDE3**		
V. D.	Media	SE	IC	Media	SE	IC	Media	SE	IC
200	119556.66	1189.41	(117276,121879)	54055.30	258.13	(53525,54544)	**24323.79**	**210.59**	**(23895,24710)**
500	253903.03	2190.67	(249937,258559)	239592.64	1225.13	(237008,241923)	**64723.36**	**303.95**	**(64059,65283)**
1000	10000000	0	_	2212089.16	120619.60	(2026548,2517188)	**155135.41**	**1197.78**	**(152405,157114)**
2000	10000000	0	_	10000000	0	_	**387793.43**	**3733.05**	**(380450,395144)**
3000	10000000	0	_	10000000	0	_	**627913.55**	**5982.13**	**(615268,638379)**
4000	10000000	0	_	10000000	0	_	**862118.13**	**6104.23**	**(851877,876888)**
5000	10000000	0	_	10000000	0	_	**1183792.22**	**12822.72**	**(1160979,1210149)**

Tabla 6.3: Evaluaciones para ZDT2

	NSGA II			GDE3			CCGDE3		
V. D.	Media	SE	IC	Media	SE	IC	Media	SE	IC
200	139511.64	926.74	(137609,141263)	74300.78	383.90	(73589,75152)	**31766.35**	**236.42**	**(31331,32251)**
500	287201.97	1439.25	(284480,289913)	506293.02	11795.19	(486781,533661)	**87862.07**	**620.00**	**(86613,88959)**
1000	10000000	0	_	10000000	0	_	**221829.77**	**6897.47**	**(214008,254278)**
2000	10000000	0	_	10000000	0	_	**508944.26**	**5491.12**	**(498824,520553)**
3000	10000000	0	_	10000000	0	_	**797833.51**	**14679.82**	**(772387,830117)**
4000	10000000	0	_	10000000	0	_	**1088563.11**	**21295.44**	**(1052706,1140226)**
5000	10000000	0	_	10000000	0	_	**1544704.96**	**56609.77**	**(1450560,1685531)**

Tabla 6.4: Evaluaciones para ZDT3

	NSGA II			GDE3			CCGDE3		
V. D.	Media	SE	IC	Media	SE	IC	Media	SE	IC
200	119416.07	1240.83	(116495,121591)	69935.36	228.51	(69448,70378)	**24525.18**	**188.76**	**(24112,24886)**
500	245161.60	1639.91	(242088,248536)	316581.42	1484.31	(313621,319528)	**63566.92**	**463.12**	**(62674,64525)**
1000	10000000	0	_	1408341.00	8834.41	(1390794,1425559)	**145131.07**	**900.46**	**(143306,146744)**
2000	10000000	0	_	5492656.00	33479.38	(5419861,5552752)	**345831.12**	**2234.16**	**(341108,349741)**
3000	10000000	0	_	9632953.07	54638.04	(9495504,9718228)	**562077.37**	**4226.78**	**(553467,570352)**
4000	10000000	0	_	10000000	0	_	**777169.50**	**6922.90**	**(764062,791999)**
5000	10000000	0	_	10000000	0	_	**1006945.83**	**7946.20**	**(995261,1027586)**

Tabla 6.5: Evaluaciones para ZDT6

	NSGA II			GDE3			CCGDE3		
V. D.	Media	SE	IC	Media	SE	IC	Media	SE	IC
200	584371.092	1666.53	(581109,587773)	461942.38	2534.61	(457304,467485)	**157878.21**	**783.76**	**(156356,159437)**
500	1150362.232	2223.03	(1146032,1154919)	10000000	0	_	**481930.17**	**3064.22**	**(475202,487641)**
1000	10000000	0	_	10000000	0	_	**1283136.32**	**7555.97**	**(1266427,1296133)**
2000	10000000	0	_	10000000	0	_	**2810482.71**	**16512.01**	**(2778592,2843568)**
3000	10000000	0	_	10000000	0	_	**3892644.27**	**35764.36**	**(3805150,3947065)**
4000	10000000	0	_	10000000	0	_	**4859792.40**	**45586.92**	**(4785931,4964634)**
5000	10000000	0	_	10000000	0	_	**5768502.88**	**19085.49**	**(5723900,5800079)**

Tabla 6.6: Tiempos de ejecución para ZDT1

D.V.	NSGA II	GDE3	CCGDE3
200	0.1412 mins	0.0488 mins	**0.0230 mins**
500	0.5174 mins	0.2865 mins	**0.0707 mins**
1000	*14.8514 mins*	2.9855 mins	**0.2060 mins**
2000	*89.0180 mins*	*11.8222 mins*	**0.8269 mins**
3000	*92.7289 mins*	*23.7238 mins*	**1.9190 mins**
4000	*159.6966 mins*	*59.8936 mins*	**2.7693 mins**
5000	*174.4521 mins*	*78.7402 mins*	**7.1859 mins**

Tabla 6.7: Tiempos de ejecución para ZDT2

Algorithm	NSGA II	GDE3	CCGDE3
200	0.1964 mins	0.6402 mins	**0.0281 mins**
500	0.6001 mins	0.7082 mins	**0.0997 mins**
1000	*6.6401 mins*	*1.0876 mins*	**0.2968 mins**
2000	*14.7021 mins*	*17.4891 mins*	**1.2929 mins**
3000	*89.6921 mins*	*40.4873 mins*	**2.3151 mins**
4000	*159.5900 mins*	*59.7938 mins*	**3.5489 mins**
5000	*175.0103 mins*	*78.4983 mins*	**7.8487 mins**

Tabla 6.8: Tiempos de ejecución para ZDT3

Algorithm	NSGA II	GDE3	CCGDE3
200	0.1495 mins	0.3203 mins	**0.0241 mins**
500	0.4471 mins	0.3448 mins	**0.0654 mins**
1000	*10.3974 mins*	2.0606 mins	**0.2886 mins**
2000	*15.7690 mins*	12.3773 mins	**0.8323 mins**
3000	*77.6378 mins*	29.1153 mins	**1.3393 mins**
4000	*159.5612 mins*	*59.2153 mins*	**2.2788 mins**
5000	*174.8563 mins*	*78.1015 mins*	**6.2366 mins**

Ahora, prestaremos atención a la tasa de convergencia, es decir, el número de evaluaciones sobre las funciones necesitadas para que el algoritmo encontrara una aproximación al verdadero frente de Pareto de acuerdo a nuestra condición de éxito. En las figuras 6.1, 6.2, 6.3 y 6.4, se muestran los resultados de la media del número

Tabla 6.9: Tiempos de ejecución para ZDT6

Algorithm	NSGA II	GDE3	CCGDE3
200	0.7987 mins	14.0197 mins	**0.1507 mins**
500	2.2383 mins	*15.4108 mins*	**0.5872 mins**
1000	*7.1561 mins*	*17.2080 mins*	**1.9982 mins**
2000	*42.4543 mins*	*33.3090 mins*	**7.0168 mins**
3000	*102.4043 mins*	*34.0594 mins*	**9.6578 mins**
4000	*159.2021 mins*	*58.7624 mins*	**31.1226 mins**
5000	*174.6141 mins*	*77.6254 mins*	**39.1256 mins**

de evaluaciones requeridas por GDE3, NSGA-II y CCGDE3 para obtener una aproximación al verdadero frente de Pareto con un valor del hipervolumen del 95 %. Estas gráficas se muestran para ZDT1, ZDT2, ZDT3 y ZDT6. Se han conectado por medio de líneas los resultados de los algoritmos, para cada número de variables de decisión bajo consideración en este trabajo. Podemos observar entonces, por medio de estas gráficas, que CCGDE3 es el algoritmo más rápido de todos, lo cual quiere decir que escala mejor que los otros dos AEMOs cuando el número de variables de decisión es mayor. Las gráficas claramente muestran que CCGDE3 tiende a ser más rápido a medida que el número de variables de decisión aumenta, mientras que NSGA-II y GDE3 experimentan un comportamiento opuesto.

Es claro entonces que CCGDE3 es mucho más rápido que NSGA-II y GDE3, no solo en términos de números de evaluaciones, sino también en términos de tiempo, ya que el tiempo necesarios para que CCGDE3 obtenga un frente con un hipervolumen del 95 % para cada problema de prueba utilizado, es mucho menor que el requerido para NSGA-II y GDE3, como se muestra en los resultados reportados en las tablas 6.6, 6.7, 6.8 y 6.9. Así mismo, las tablas 6.2, 6.3, 6.4 y 6.5, muestran valores estrechos para los intervalos de confianza, relativamente conforme a los valores de los valores promedio del número de evaluaciones sobre las funciones obtenidos, por lo que podemos decir que los resultados son estadísticamente confiables como estimadores del comportamiento promedio de los algoritmos.

6.1.4. Análisis de varianza

Se realizó un análisis de varianza para determinar la sensibilidad de nuestra propuesta ante los parámetros de entrada utilizando la función ZDT1 (indicada en el Apéndice 7) con 1000 variables de decisión. Los parámetros analizados fueron: el número de $Ciclos$, el número máximo de generaciones por ciclo $GenMax$, el número de $Especies$ y el tamaño de población NP de cada una de ellas. Los niveles de cada uno de éstos son los siguientes:

- $Ciclos$: 10, 50, 100.

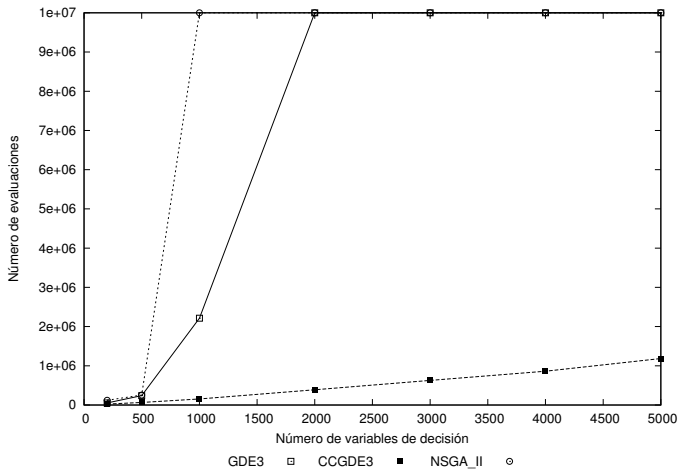

Figura 6.1: Gráfica del número de evaluaciones requeridas por cada algoritmo para obtener una aproximación con un 95 % de hipervolumen del verdadero frente de Pareto, para ZDT1.

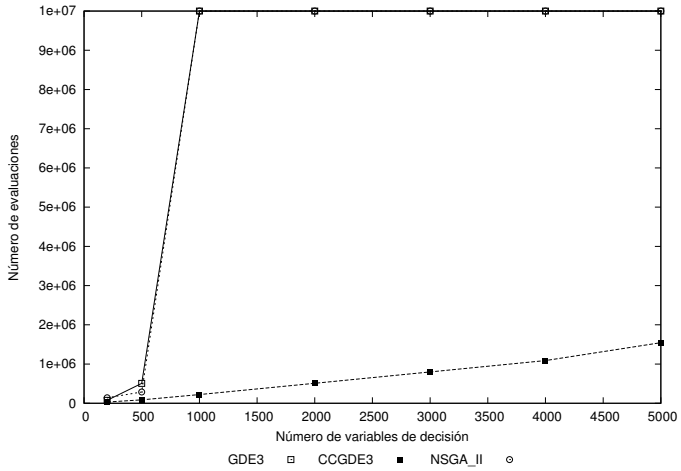

Figura 6.2: Gráfica del número de evaluaciones requeridas por cada algoritmo para obtener una aproximación con un 95 % de hipervolumen del verdadero frente de Pareto, para ZDT2.

- $GenMax$: 10, 50, 100.
- $Especies$: 2, 4, 8.
- NP: 20, 50, 100.

Se obtuvieron entonces 81 combinaciones diferentes, de las cuales se realizaron 25 ejecuciones independientes por cada una, lo cual dio en total 2025 ejecuciones. De estas 25 ejecuciones para cada combinación se obtuvieron 1000 remuestreos a travez del procedimiento de *bootstrap*. De esta forma, se produjo una distribución de *bootstrap* a partir de la cual se obtuvieron la media, el error estándar, el sesgo (*bias*) y los intervalos de confianza del 95 % para cada muestra. Todos estos resultados pueden

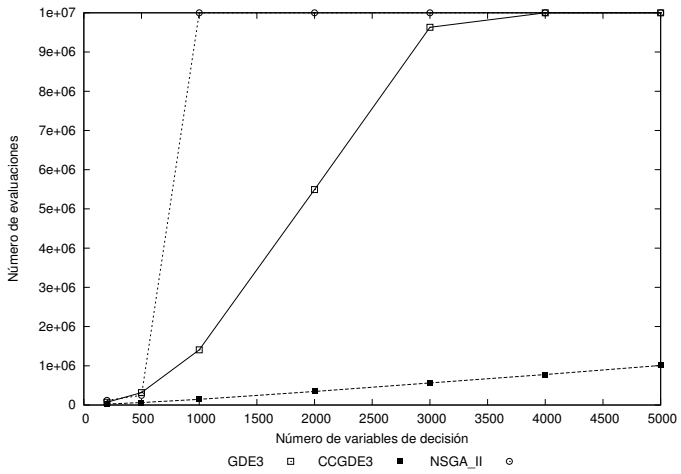

Figura 6.3: Gráfica del número de evaluaciones requeridas por cada algoritmo para obtener una aproximación con un 95 % de hipervolumen del verdadero frente de Pareto, para ZDT3.

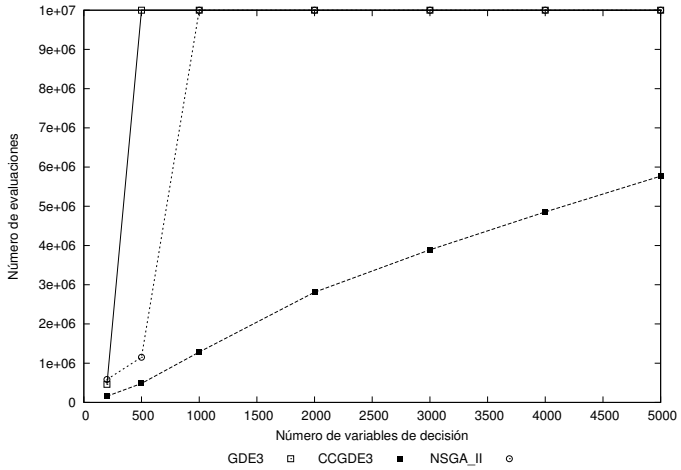

Figura 6.4: Gráfica del número de evaluaciones requeridas por cada algoritmo para obtener una aproximación con un 95 % de hipervolumen del verdadero frente de Pareto, para ZDT6.

observare a detalle en el Apéndice .4. Las hipótesis para estos experimentos fueron las siguientes:

Hipótesis nula: No existe una diferencia significativa entre los promedios de los resultados obtenidos y si existen diferencias, se deben a efectos aleatorios.

Hipótesis alternativa: Hay una combinación de parámetros en la que los promedios son distintos y no se debe a efectos aleatorios.

Los resultados que se obtuvieron prueban la *Hipótesis nula* para algunas combinaciones de parámetros como lo es para la relación entre el número de especies y el tamaño de población de cada una de éstas, en donde al parecer, la diferencia es

marginal cuando solo se cambian los valores para estos dos parámetros. Sin embargo, para la relación existente con el número de ciclos y el número de generaciones utilizadas para cada una de estas se prueba la *Hipótesis alternativa*. De aquí se observa que sí se tiene que elegir entre número de ciclos y número de generaciones por ciclo, teniendo en mente que lo que se toma como costo es el número de evaluaciones sobre las funciones objetivo, es preferible hacer uso de las evaluaciones con mayor número de ciclos que de generaciones. Esto se debe a que, al tener una mayor interacción entre las especies, más que un mayor progreso en la evolución individual de cada especie, se obtienen mejores resultados, ya que se obtiene un mayor valor para el hipervolumen del problema en cuestión (en este caso, ZDT1).

Capítulo 7

Conclusiones y trabajo a futuro

Este trabajo de tesis propone un nuevo esquema de coevolución cooperativa para resolver problemas de optimización multiobjetivo con un gran número de variables de decisión. Se ha presentado un nuevo algoritmo evolutivo multiobjetivo basado en el uso de GDE3 y un esquema de coevolución cooperativa al cual llamamos CCGDE3 (*Cooperative Coevolutionary* GDE3). El algoritmo propuesto mostró ser capaz de lidiar de forma exitosa con problemas multiobjetivo con un gran número de variables de decisión (hasta 5000). Se realizaron estudios para evaluar el desempeño de CCGDE3 con un conjunto de problemas de prueba escalable en el número de variables de decisión: el conjunto de *Zitzler-Deb-Thiele (ZDT)* [40]. Se estudió la tasa de convergencia del algoritmo propuesto y se comparó con respecto a la de NSGA-II y GDE3 al resolver los problemas antes mencionados. En otras palabras, se midió el número de evaluaciones de las funciones objetivo requeridas para que un algoritmo encontrara una aproximación al verdadero frente de Pareto de acuerdo a una condición de éxito dada. La condición de paro establecida fue alcanzar una aproximación al frente de Pareto de cada problema con un valor de hipervolumen de al menos el 95 %, o en su defecto haber realizado un máximo de 10 millones de evaluaciones sobre las funciones objetivo (esta segunda condición se aplica cuando el algoritmo no alcanzó a cumplir la condición de éxito establecida). En nuestros experimentos, se obtuvo la media del número de evaluaciones necesarias para GDE3, NSGA-II y CCGDE3 para generar una aproximación del 95 % de hipervolumen del verdadero frente de Pareto de cada problema, calculada con un procedimiento de *bootstrap* con 1000 remuestreos.

Los resultados confirman que CCGDE3 es muy eficiente y efectivo al resolver problemas multiobjetivo de gran escala (en el espacio de las variables de decisión), pues CCGDE3 resultó ser el algoritmo más rápido de todos, y el que escaló mejor. Los resultados claramente muestran que CCGDE3 tiende a ser más rápido a medida que el número de variables de decisión aumenta, mientras que NSGA-II y GDE3 experimentan un comportamiento opuesto.

Aunque se utilizó GDE3 como el optimizador básico, es relativamente fácil incorporar cualquier otro algoritmo evolutivo multiobjetivo (AEMO) en el esquema aquí presentado. Además, podría también usarse más de un tipo de AEMO como el optimizador básico, con el objetivo de combinar características complementarias

de distintos motores de búsqueda. Por lo tanto, como parte del trabajo a futuro, se tiene la propuesta de enfocarse en más técnicas que puedan ser incorporadas en el esquema de coevolución cooperativa presentado en esta tesis y encontrar una forma de disminuir la configuración de parámetros, mediante la auto-adaptación de algunos de ellos. Por ejemplo, sería posible: adaptar el número de especies que se necesitan y el tamaño de población para cada una de ellas dentro del algoritmo de una forma automática, así como el número de generaciones de cada ciclo de la evolución. Otra posibilidad es utilizar especies que sean capaces de seleccionar al mejor AEMO, de un conjunto disponible, para resolver problemas de optimización multiobjetivo con ciertas características específicas. En resumen, las dos áreas de investigación más prometedoras son la descomposición del problema (o sea, optimizar cada especie con un AEMO específico), y la auto-adaptación de los parámetros de control del algoritmo propuesto.

Bibliografía

[1] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[2] W.J. Cook. *Mathematical Programming Computation*. Springer, mathematical optimization society edition, 2009.

[3] J.J. Durillo, A.J. Nebro, C.A. Coello Coello, J. Garcia-Nieto, F. Luna, and E. Alba. A Study of Multiobjective Metaheuristics When Solving Parameter Scalable Problems. *IEEE Transactions on Evolutionary Computation*, 14(4):618–635, August 2010.

[4] Juan J. Durillo, Antonio J. Nebro, Carlos A. Coello Coello, Francisco Luna, and Enrique Alba. A Comparative Study of the Effect of Parameter Scalability in Multi-Objective Metaheuristics. In *2008 Congress on Evolutionary Computation (CEC'2008)*, pages 1893–1900, Hong Kong, June 2008. IEEE Service Center.

[5] Yazmin Rojas García. Optimización Evolutiva a Gran Escala con y sin Restricciones. Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, dic 2011.

[6] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[7] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

[8] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK, 1996.

[9] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Bradford Books. MIT Press, 1992.

[10] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[11] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, Alemania, 1973.

[12] Schwefel Hans Paul. *Numerical Optimization of Computer Models*. John Wiley & Sons Ltd, New York, USA, 1981.

[13] David B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. The Institute of Electrical and Electronic Enginers, New York, NY, USA, 1998.

[14] Carlos M. Fonseca and Peter J. Fleming. Multiobjective Optimization. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, volume 1, pages C4.5:1–C4.5:9. Institute of Physics Publishing and Oxford University Press, 1997.

[15] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.

[16] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U. K., 1994.

[17] Eckart Zitzler and Simon Künzli. Indicator-based Selection in Multiobjective Search. In Xin Yao et al., editor, *Parallel Problem Solving from Nature - PPSN VIII*, pages 832–842, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

[18] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 16 September 2007.

[19] Jianjun Hu and Erik Goodman. Robust and Efficient Genetic Algorithms with Hierarchical Niching and a Sustainable Evolutionary Computation Model. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 1220–1232, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[20] David Edward Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In John. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, Hillsdale, Nueva Jersey, EE. UU., 1987. Lawrence Erlbaum.

[21] Thomas E. Koch and Andreas Zell. MOCS: Multi-Objective Clustering Selection Evolutionary Algorithm. In W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 423–430, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[22] Ali Farhang-Mehr and Shapour Azarm. Entropy-based multi-objective genetic algorithm for design optimization. *Structural and Multidisciplinary Optimization*, 24(5):351–361, November 2002.

[23] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[24] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[25] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

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

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

[28] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007.

[29] R Sarker, M Mohammadian, and X Yao, editors. *Evolutionary Optimization*. Kluwer Academic Publishers, Boston, 2002.

[30] F. van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimization. *Trans. Evol. Comp*, 8(3):225–239, June 2004.

[31] Johannes Bader and Eckart Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation*, 19(1):45–76, Spring, 2011.

[32] Andre B. de Carvalho and Aurora Pozo. Measuring the convergence and diversity of CDAS Multi-Objective Particle Swarm Optimization Algorithms: A study of many-objective problems. *Neurocomputing*, 75(1):43–51, January 1 2012.

[33] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. Technical Report 139, Computer Engineering and Networks Laboratory, ETH Zurich, June 2002.

[34] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 283–290, San Francisco, California, 2001. Morgan Kaufmann Publishers.

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

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

[37] Antonio J. Nebro, Juan J. Durillo, Francisco Luna, Bernabe Dorronsoro, and Enrique Alba. MOCell: A Cellular Genetic Algoritm for Multiobjective Optimization. *International Journal of Intelligent Systems*, 24(7):726–746, July 2009.

[38] Saku Kukkonen and Jouni Lampinen. GDE3: The third Evolution Step of Generalized Differential Evolution. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 1, pages 443–450, Edinburgh, Scotland, September 2005. IEEE Service Center.

[39] Antonio J. Nebro, Francisco Luna, Enrique Alba, Bernabé Dorronsoro, Juan J. Durillo, and Andreas Beham. AbYSS: Adapting Scatter Search to Multiobjective

Optimization. *IEEE Transactions on Evolutionary Computation*, 12(4):439–457, August 2008.

[40] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.

[41] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama. Opposition-Based Differential Evolution. *Evolutionary Computation, IEEE Transactions on*, 12(1):64–79, 2008.

[42] Shahryar Rahnamayan and G. Gary Wang. Investigating in scalability of opposition-based differential evolution. In *Proceedings of the 8th conference on Simulation, modelling and optimization*, SMO'08, pages 105–111, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[43] Janez Brest, Ales Zamuda, Borko Boskovic, Mirjam Sepesy Maucec, and Viljem Zumer. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. *Evolutionary Computation, IEEE Transactions on*, 12(2032-2039):2032–2039, June 2008.

[44] Rainer Storn and Kenneth Price. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, December 1997.

[45] Janez Brest, Ales Zamuda, Iztok Fister, and Mirjam Sepesy Maucec. Large scale global optimization using self-adaptive differential evolution algorithm. *Evolutionary Computation, IEEE Transactions on*, 12(2032-2039):1–8, July 2010.

[46] Weicai Zhong, Jing Liu, Mingzhi Xue, and Licheng Jiao. A multiagent genetic algorithm for global numerical optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(2):1128–1141, 2004.

[47] L. D. Davis and Melanie Mitchell. Handbook of Genetic Algorithms. *Van Nostrand Reinhold*, 1991.

[48] Zhang Zheng, Ma Shu-gen, Cao Bing-gang, Zhang Li-ping, and Li Bin. Multiagent reinforcement learning for a planetary exploration multirobot system. In *Proceedings of the 9th Pacific Rim international conference on Agent Computing and Multi-Agent Systems*, pages 339–350, Berlin, Heidelberg, 2006. Springer-Verlag.

[49] Jiming Liu. *Autonomous Agents and Multi-agent: Explorations in Learning, Self-Organization and Adaptive Computation.* World Scientific Publishing Co. Pte. Ltd., 2001.

[50] Antonio J. Nebro, Juan J. Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Design issues in a multiobjective cellular genetic algorithm, 2007.

[51] Mitchell A. Potter and Kenneth A. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, PPSN III, pages 249–257, London, UK, UK, 1994. Springer-Verlag.

[52] Yan-jun Shi, Hong-fei Teng, and Zi-qiang Li. Cooperative co-evolutionary differential evolution for function optimization, 2005.

[53] Xin Yao Zhenyu Yang, Ke Tang. Differential evolution for high-dimensional function optimization. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, volume 1, sept. 2007.

[54] Zhenyu Yang, Ke Tang, and Xin Yao. Multilevel cooperative coevolution for large scale optimization. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1663–1670, 2008.

[55] Xin Yao Mohammad Nabi, Zhenyu Yang. Cooperative co-evolution for large scale optimization through more frequent random grouping. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, volume 1, pages 1– 8 Vol.1, sept. 2010.

[56] Ales Zamuda, Janez Brest, Borko Boskovic, and Viljem Zumer. Large scale global optimization using differential evolution with self-adaptation and cooperative coevolution. In *IEEE Congress on Evolutionary Computation*, pages 3718–3725. IEEE, 2008.

[57] Paul R. Ehrlich and Peter H. Raven. Butterflies and Plants: A Study in Coevolution. *Evolution*, 18(4):586–608, 1964.

[58] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *Proceedings of the ninth annual international conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent computation*, CNLS '89, pages 228–234, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[59] Carlos A. Coello Coello and Margarita Reyes Sierra. A Coevolutionary Multi-Objective Evolutionary Algorithm. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 1, pages 482–489, Canberra, Australia, December 2003. IEEE Press.

[60] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, UK, 1982.

[61] P. J. Darwen and X. Yao. On evolving robust strategies for iterated prisoner's dilemma. In *IN PROGRESS IN EVOLUTIONARY COMPUTATION*, pages 276–292. Springer-Verlag, 1995.

[62] Peter J. Angeline and Jordan B. Pollack. Competitive environments evolve better solutions for complex tasks, 1993.

[63] Daniel W. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *CNLS '89: Proceedings of the ninth annual international conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent computation*, pages 228–234, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[64] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *Proceedings of the ninth annual international conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent computation*, CNLS '89, pages 228–234, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[65] Helio JC Barbosa and André MS Barreto. An interactive genetic algorithm with co-evolution of weights for multiobjective problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 203–210, 2001.

[66] Christopher D. Rosin and Richard K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 373–381, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[67] Nattavut Keerativuttiumrong, Nachol Chaiyaratana, and Vara Varavithya. Multi-objective Co-operative Co-evolutionary Genetic Algorithm. In Juan Julián Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, José-Luis Fernández-Villaca nas, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN VII*, pages 288–297, Granada, Spain, September 2002. Springer-Verlag. Lecture Notes in Computer Science No. 2439.

[68] C.M. Fonseca and P.J. Fleming. Multiobjective genetic algorithms. In A.M.S. Zalzala and P.J. Fleming, editors, *Genetic Algorithms in Engineering Systems*, chapter 3, pages 63–78. The Institution of Electrical Engineers. Control Engineering Series 55, Bath, UK, 1997.

[69] K.C. Tan, Y.H. Chew, T.H. Lee, and Y.J. Yang. A Cooperative Coevolutionary Algorithm for Multiobjective Optimization. In *Proceedings of the 2003 IEEE*

International Conference on Systems, Man and Cybernetics, volume 1, pages 390–395. IEEE Press, 2003.

[70] Antony W. Iorio and Xiaodong Li. A Cooperative Coevolutionary Multiobjective Algorithm Using Non-dominated Sorting. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 537–548, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[71] Ian Parmee and Andrew H. Watson. Preliminary airframe design using coevolutionary multiobjective genetic algorithms. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1657–1665, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[72] K.C. Tan, Y.J. Yang, and T.H. Lee. A Distributed Cooperative Coevolutionary Algorithm for Multiobjective Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 4, pages 2513–2520, Canberra, Australia, December 2003. IEEE Press.

[73] Carlos A. Coello Coello and Margarita Reyes Sierra. A Coevolutionary Multi-Objective Evolutionary Algorithm. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 1, pages 482–489, Canberra, Australia, December 2003. IEEE Press.

[74] J. Lampinen. DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology, 2001.

[75] George P. Mccabe and David S. Moore. *Introduction to the Practice of Statistics*. W.H. Freeman & Company, 2009.

[76] Thomas Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism*. Natural Computing Series. Springer, 2006.

[77] Paul R. Cohen. *Empirical methods for artificial intelligence*. MIT Press, Cambridge, MA, USA, 1995.

Funciones de prueba

.1. ZDT1

El problema ZDT1 tiene un frente de Pareto convexo:

$$f_1(x_1) = x_1 \tag{1}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1) \tag{2}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g} \tag{3}$$

donde m es el número de variables de decisión, y $x_i \in [0, 1]$. El frente de Pareto se obtiene cuando $g(\vec{x}) = 1$.

.2. ZDT2

El problema ZDT2 tiene un frente de Pareto no convexo:

$$f_1(x_1) = x_1 \tag{4}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1) \tag{5}$$

$$h(f_1, g) = 1 - (f_1/g)^2 \tag{6}$$

donde m es el número de variables de decisión, y $x_i \in [0, 1]$. El frente de Pareto se obtiene cuando $g(\vec{x}) = 1$.

.3. ZDT3

El problema ZDT3 tiene un frente de Pareto que consiste en varias partes convexas no continuas:

$$f_1(x_1) = x_1 \tag{7}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1) \tag{8}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\,\mathrm{sen}(10\pi f_1) \tag{9}$$

donde m es el número de variables de decisión, y $x_i \in [0, 1]$. La introducción de la función de seno en h causa discontinuidad en el frente de Pareto verdadero. Sin embargo, no existe discontinuidad en el espacio de las variables de decisión.

.4. ZDT6

El problema ZDT6 tiene dos dificultades causadas por la no uniformidad en el espacio de búsqueda. En primer lugar, las soluciones del frente de Pareto están distribuidas de forma no uniforme a lo largo del frente de Pareto verdadero (el frente se encuentra sesgado por las soluciones para las que $f_1(\vec{x})$ tiene un valor cercano a uno). En segundo lugar, la densidad de las soluciones es más baja cerca del frente de Pareto verdadero y más alta lejos de él:

$$f_1(x_1) = 1 - exp(-4x_1)\,\mathrm{sen}^6(6\pi x_1) \tag{10}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \cdot ((\sum_{i=2}^{m} x_i)/(m-1))^{0.25} \tag{11}$$

$$h(f_1, g) = 1 - (f_1/g)^2 \tag{12}$$

donde m es el número de variables de decisión, y $x_i \in [0, 1]$. El frente de Pareto verdadero se obtiene cuando $g(\vec{x}) = 1$ y es no convexo.

Resultados para las pruebas de varianza

	Ciclos	GenMax	Especies	NP	Media (del hipervolumen)	bias	std. error	CI
1	10	10	2	20	1.807477	0.0004545207	0.01560495	(1.777,1.838)
2	10	10	2	50	1.995695	0.0002619201	0.005604757	(1.983,2.005)
3	10	10	2	100	1.959178	3.842284e-05	0.004392873	(1.951,1.969)
4	10	10	4	20	1.838261	-6.442968e-05	0.01077686	(1.816,1.858)
5	10	10	4	50	2.031795	-0.0003515716	0.008395259	(2.017,2.050)
6	10	10	4	100	2.062945	-0.0004311314	0.007700285	(2.047,2.078)
7	10	10	8	20	1.7852	-0.0001966355	0.008886161	(1.770,1.806)
8	10	10	8	50	1.97617	-5.6772e-05	0.008580817	(1.957,1.992)
9	10	10	8	100	2.012781	1.046604e-05	0.007828721	(1.997,2.028)
10	10	50	2	20	3.592366	-3.13654e-05	0.01029905	(3.570,3.612)
11	10	50	2	50	3.660673	0.0002285975	0.006872279	(3.648,3.675)
12	10	50	2	100	3.371767	-0.0003515128	0.005996515	(3.361,3.383)
13	10	50	4	20	3.569563	0.0002649983	0.01329552	(3.539,3.593)
14	10	50	4	50	3.606601	0.0004073638	0.01030298	(3.587,3.627)
15	10	50	4	100	3.392531	-0.0003238037	0.009244337	(3.374,3.411)
16	10	50	8	20	3.15898	-0.0002847717	0.01345306	(3.136,3.191)
17	10	50	8	50	3.194163	-0.000202112	0.01111986	(3.172,3.216)
18	10	50	8	100	3.065075	-0.0004249289	0.007930794	(3.051,3.082)
19	10	100	2	20	3.984387	0.0002051786	0.005621215	(3.969,3.993)
20	10	100	2	50	4.039567	-0.0001427301	0.002034135	(4.036,4.043)
21	10	100	2	100	3.907749	6.688808e-05	0.003509131	(3.901,3.915)
22	10	100	4	20	3.92323	2.492948e-05	0.008951794	(3.903,3.941)
23	10	100	4	50	4.042006	-0.000116361	0.0034055	(4.035,4.048)
24	10	100	4	100	3.883146	0.000229676	0.008587203	(3.864,3.899)
25	10	100	8	20	3.645712	0.0002907803	0.01056073	(3.624,3.664)
26	10	100	8	50	3.658262	0.0004608626	0.008223337	(3.641,3.674)
27	10	100	8	100	3.478011	-6.326244e-05	0.008270267	(3.462,3.496)
28	50	10	2	20	3.478451	-4.991284e-05	0.009952128	(3.457,3.496)
29	50	10	2	50	3.633461	3.808896e-05	0.005368332	(3.623,3.644)
30	50	10	2	100	3.372927	9.72046e-05	0.004309906	(3.364,3.381)
31	50	10	4	20	3.548152	6.424228e-05	0.007877458	(3.533,3.564)
32	50	10	4	50	3.670775	-3.9528e-06	0.005024643	(3.661,3.681)
33	50	10	4	100	3.476734	-0.0001031421	0.005195063	(3.466,3.487)
34	50	10	8	20	3.505287	0.0004009647	0.0101113	(3.484,3.525)
35	50	10	8	50	3.549054	-8.506228e-05	0.005548602	(3.538,3.561)
36	50	10	8	100	3.373508	1.222384e-05	0.005418259	(3.362,3.383)
37	50	50	2	20	4.104778	-0.0001069078	0.00349331	(4.098,4.112)
38	50	50	2	50	4.140019	9.12616e-06	0.0005228439	(4.139,4.141)
39	50	50	2	100	4.128574	-2.9908e-07	0.0004108734	(4.128,4.129)
40	50	50	4	20	4.107655	2.3994e-06	0.002820966	(4.102,4.113)
41	50	50	4	50	4.15742	2.15002e-05	0.0004711896	(4.156,4.158)
42	50	50	4	100	4.140447	1.193556e-05	0.0003311301	(4.140,4.141)
43	50	50	8	20	4.068871	-7.77432e-06	0.004015087	(4.061,4.077)
44	50	50	8	50	4.134104	-8.4796e-06	0.0005566457	(4.133,4.135)
45	50	50	8	100	4.100476	1.396124e-05	0.0006361898	(4.099,4.102)
46	50	100	2	20	4.118106	5.963136e-05	0.002016656	(4.114,4.122)
47	50	100	2	50	4.161482	-9.8552e-06	0.0003063973	(4.161,4.162)
48	50	100	2	100	4.150987	1.87288e-06	0.0002651879	(4.150,4.151)
49	50	100	4	20	4.126149	-5.178296e-05	0.002986494	(4.121,4.132)
50	50	100	4	50	4.168704	1.00442e-05	0.0002251761	(4.168,4.169)
51	50	100	4	100	4.165554	9.32608e-06	0.0001991716	(4.165,4.166)
52	50	100	8	20	4.122611	4.262876e-05	0.003356618	(4.114,4.128)
53	50	100	8	50	4.163366	7.70744e-06	0.0002335743	(4.163,4.164)
54	50	100	8	100	4.153419	1.105568e-05	0.0003142625	(4.153,4.154)
55	100	10	2	20	3.910708	-0.0001351722	0.005674257	(3.900,3.922)
56	100	10	2	50	4.015001	-3.47196e-06	0.001601845	(4.012,4.018)
57	100	10	2	100	3.893267	-4.345136e-05	0.002103496	(3.889,3.897)
58	100	10	4	20	3.965287	2.915844e-05	0.005786533	(3.954,3.977)
59	100	10	4	50	4.030215	8.268332e-05	0.001475268	(4.027,4.033)
60	100	10	4	100	3.94845	8.036184e-05	0.001854451	(3.945,3.952)
61	100	10	8	20	3.961845	5.765152e-05	0.006221765	(3.949,3.974)
62	100	10	8	50	3.975604	1.95224e-06	0.001889223	(3.972,3.979)
63	100	10	8	100	3.86289	1.60452e-05	0.002894523	(3.856,3.868)
64	100	50	2	20	4.112607	0.0002377269	0.003485589	(4.105,4.119)
65	100	50	2	50	4.16086	-8.78768e-06	0.0003462271	(4.160,4.162)
66	100	50	2	100	4.151189	1.224468e-05	0.0002909353	(4.151,4.152)
67	100	50	4	20	4.127254	-3.073824e-05	0.002544677	(4.122,4.132)
68	100	50	4	50	4.168678	-5.7012e-07	0.0002973808	(4.168,4.169)
69	100	50	4	100	4.16645	2.46324e-06	0.0001479322	(4.166,4.167)
70	100	50	8	20	4.131595	7.878016e-05	0.003240633	(4.125,4.137)
71	100	50	8	50	4.16818	-4.1142e-06	0.0001907996	(4.168,4.168)
72	100	50	8	100	4.162713	8.05008e-06	0.0001771665	(4.162,4.163)
73	100	100	2	20	4.116423	-7.9918e-06	0.001821539	(4.113,4.120)
74	100	100	2	50	4.168171	4.8732e-07	0.0001410302	(4.168,4.168)
75	100	100	2	100	4.160616	6.076e-07	0.0002286137	(4.160,4.161)
76	100	100	4	20	4.127228	9.720896e-05	0.002851542	(4.121,4.132)
77	100	100	4	50	4.169419	7.8332e-07	0.0002271663	(4.169,4.170)
78	100	100	4	100	4.171245	2.78368e-06	0.0001350034	(4.171,4.171)
79	100	100	8	20	4.137764	-6.338808e-05	0.002729424	(4.131,4.142)
80	100	100	8	50	4.169938	-4.0564e-07	0.0001925013	(4.169,4.170)
81	100	100	8	100	4.172338	-1.33004e-06	9.425381e-05	(4.172,4.172)

Tabla 1: Resultados experimentales del análisis de varianza para determinar la sensibilidad del algoritmo propuesto ante los parámetros de entrada, utilizando la función ZDT1 (indicada en el Apéndice 7) con 1000 variables de decisión. Los parámetros analizados son: el número de *Ciclos*, el número máximo de generaciones por ciclo ($GenMax$), el número de *Especies* y el tamaño de población (NP) de cada una de ellas. Se creó una distribución de *bootstrap* con 1000 remuestreos, a partir de 25 ejecuciones para cada combinación, de la cual se muestran: la media, el error estándar, el sesgo (*bias*) y los intervalos de confianza del 95 % para cada muestra.