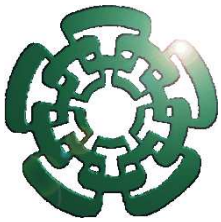
CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL IPN

Departamento de Ingeniería Eléctrica

Sección de Computación



Diseño de un algoritmo evolutivo multiobjetivo paralelo

Tesis que presenta
Antonio López Jaimes[*]

para obtener el grado de
Maestro en Ciencias

en la especialidad de
Ingeniería Eléctrica,
Opción Computación

Dirigida por el
Doctor Carlos A. Coello Coello

México, D.F. Febrero de 2005

[*]Quisiera agradecer por la beca terminal otorgada por medio del proyecto CONACyT 31892-A, cuyo responsable es el Dr. Arturo Díaz Pérez

LOS MENDIGOS
¡Había mendigos! Unos imploraban el amor. Otros, la estima. Otros la gloria. Despreciaban a los que pedían pan o dinero. Algunos pedían una idea por el amor de los dioses, o un verso bien hecho – o un estilo «original».

Paul VALÉRY

Resumen

La mayoría de los problemas del mundo real involucran la optimización simultánea de varios objetivos (generalmente en conflicto entre sí y expresados en distintas unidades de medida). Estos problemas son llamados *multiobjetivo* o *multicriterio* (POMs). La investigación de operaciones ha desarrollado un número considerable de técnicas para resolver estos problemas. Sin embargo, dada la complejidad de la mayoría de los problemas de optimización multiobjetivo del mundo real (p. ej., alta dimensionalidad, discontinuidad, multimodalidad), estas técnicas presentan varias limitantes o no resultan prácticos para resolverlos.

Uno de los enfoques alternativos más exitosos para resolver eficazmente estos problemas es el uso de los *algoritmos evolutivos multiobjetivo* (AEMO). No obstante, uno de los puntos débiles de estos algoritmos es su alto consumo de recursos computacionales, lo cual motiva, de manera natural, el uso del paralelismo. La paralelización de los AEMOs es un campo relativamente nuevo, y como consecuencia, existen pocas publicaciones al respecto. En la mayoría de estas publicaciones no se discuten detalladamente los aspectos relacionados con el desarrollo y la implementación de los AEMOs *paralelos* (pAEMOs). Asimismo, no se ha considerado una selección justificada de los problemas de prueba y de las métricas para evaluar tanto la *eficacia* como la *eficiencia* de los pAEMOs.

En esta tesis se presenta un nuevo AEMO paralelo competitivo con respecto a los AEMOs representativos del estado del arte desde el punto de vista de la efectividad y la eficiencia. El algoritmo propuesto, denominado *algoritmo genético multiobjetivo con múltiples resoluciones* (MRMOGA, '*Multiple Resolution Multi-Objective Genetic Algorithm*'), está basado en el *modelo de islas* con nodos heterogéneos y se construyó sobre un cúmulo de computadoras. El algoritmo se caracteriza por codificar las soluciones con una resolución distinta en cada isla. De esta manera se consigue dividir el espacio de búsqueda en regiones disjuntas del espacio de las variables de decisión.

Para evaluar la efectividad del algoritmo propuesto se utilizaron métricas conocidas para evaluar AEMOs secuenciales, a saber: el *conteo de aciertos*, la *distancia generacional invertida*, el *espaciamiento* y la *cobertura de conjuntos*. Para evaluar la eficiencia se utilizaron las siguientes métricas conocidas en la computación paralela: *aceleración*, *eficiencia* y *fracción serial*. Los resultados del algoritmo propuesto se compararon con los obtenidos por una versión paralela del NSGA-II.

Con base en los resultados obtenidos podemos afirmar que el esquema de MRMOGA para dividir el espacio de búsqueda mediante distintas resoluciones mejora considerablemente la eficacia y la eficiencia de un pAEMO. Además, el algoritmo propuesto tiene una gran capacidad exploratoria ya que, con respecto al algoritmo de referencia, en todas las funciones de prueba encontró soluciones que se extienden mejor a lo largo del frente de Pareto real. Asimismo, MRMOGA demostró una capacidad inusualmente buena para resolver POMs con restricciones, incluso para aquellos problemas reconocidos por su dificultad.

Abstract

Most real-world problems involve simultaneous optimization of several objectives (usually incommensurable and in conflict with each other). These problems are called multiobjective, or multicriteria optimization problems (MOPs). Operations Research has developed plenty of methods to solve these problems. However, due to the complexity of most real-world multiobjective optimization problems (e.g., high dimensionality, discontinuity, multimodality), these methods have several drawbacks or become impractical to solve them.

One of the most successful approaches for solving effectively these problems is the use of multiobjective evolutionary algorithms (MOEAs). Nevertheless, one of the drawback of these algorithms is their high consumption of computational resources. To overcome this problem, the parallel and distributed processing of the MOEAs is a natural solution.

Parallelization of MOEAs is a relatively new field, so there are few publications on that subject. In general, these publications do not discuss in detail parallel MOEA (pMOEA) development and implementation issues. Also, these publications lack a thorough rationale of the MOP test problem selection and pMOEA performance metrics to evaluate both *effectiveness* and *efficiency*.

This thesis presents a novel parallel MOEA with regard to the representative MOEAs of the state-of-the-art in terms of effectiveness and efficiency. The proposed algorithm, called Multi-Resolution Multi-Objective Genetic Algorithm (MRMOGA), is based on the *island model* with heterogeneous nodes and is built upon a cluster of computers. This algorithm is characterized by encoding the solutions using a different resolution for each island. In this way, the search space is divided into disjoint regions in the variable decision space.

In order to evaluate the effectiveness of the proposed approach we use well-known metrics used to evaluate serial MOEAs, namely: *success counting*, *generational distance*, *spacing*, and *two set coverage*. On the other hand, efficiency was evaluated using the following well-known metrics from parallel computing: *speedup*, *efficiency* and *serial fraction*. The results of the proposed algorithm were compared against those of a parallel version of the NSGA-II.

In the light of the obtained results, we can state that the MRMOGA scheme to divide the search space with different resolutions improves considerably the effectiveness and efficiency of a parallel MOEA. In addition, the proposed algorithm has a great exploratory ability, because in all the test functions it found solutions with a better spread along the real Pareto front than the algorithm with respect to which it was compared. Also, MRMOGA showed an unusual good ability to solve constrained MOPs, even for those problems recognized for their difficulty.

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

Lista de acrónimos

CE	Computación evolutiva
AE	Algoritmo evolutivo
PE	Programación evolutiva
EEs	Estrategias evolutivas
AG	Algoritmo genético
POM	Problema de optimización multiobjetivo
AEMO	Algoritmo evolutivo multiobjetivo
VEGA	*Vector Evaluated Genetic Algorithm*
MOGA	*Multiobjective Genetic Algorithm*
MRMOGA	Algoritmo genético multiobjetivo con múltiples resoluciones (*Multiple Resolution Multiobjective Genetic Algorithm*)
NPGA	*Niched Pareto Genetic Algorithm*
NSGA-II	*Nondominated Sorting Genetic Algorithm* II
SPEA	*Strength Pareto Evolutionary Algorithm*
PAES	*Pareto Archived Evolution Strategy*
pAEMO	*Algoritmo evolutivo multiobjetivo paralelo*
pAE	*Algoritmo evolutivo monoobjetivo paralelo*

Lista de símbolos

P_{real}	Conjunto de óptimos de Pareto real
$P_{actual}(t)$	Conjunto de óptimos de Pareto determinado por la población primaria del AEMO en la generación t
$P_{conocido}(t)$	Conjunto de óptimos de Pareto determinado por la población secundaria del AEMO a lo largo de t generaciones
$P_{conocido}$	Conjunto de óptimos de Pareto determinado por el AEMO al término de su ejecución
FP_{real}	Frente de Pareto real
$FP_{actual}(t)$	Frente de Pareto determinado por la población primaria del AEMO en la generación t
$FP_{conocido}(t)$	Frente de Pareto determinado por la población secundaria del AEMO a lo largo de t generaciones
$FP_{conocido}$	Frente de Pareto determinado por el AEMO al término de su ejecución
$P_{actual}\binom{p}{t}$	Conjunto de óptimos de Pareto determinado por el procesador p en la generación t
$P_{conocido}\binom{p}{t}$	Conjunto de óptimos de Pareto determinado por el procesador p a lo largo de t generaciones
$P_{conocido}\binom{p}{}$	Conjunto de óptimos de Pareto determinado por el procesador p al término de su ejecución
$FP_{actual}\binom{p}{t}$	Frente de Pareto determinado por el procesador p en la generación t
$FP_{conocido}\binom{p}{t}$	Frente de Pareto determinado por el procesador p a lo largo de t generaciones
$FP_{conocido}\binom{p}{}$	Frente de Pareto determinado por el procesador p al término de su ejecución

Capítulo 1

Introducción

Sumario

1.1. Antecedentes y motivación

La mayoría de los problemas del mundo real involucran la optimización de varios objetivos simultáneamente (los cuales se expresan en unidades de medida diferentes o se encuentran en conflicto entre sí). Estos problemas son llamados *multiobjetivo* o *multicriterio*. La naturaleza conflictiva de sus objetivos produce un conjunto de soluciones compromiso alternativas. Este conjunto es llamado *conjunto de óptimos de Pareto*, y la proyección de este conjunto en el espacio de los objetivos se le denomina *frente de Pareto*.

A pesar de que ha surgido una gran variedad de técnicas clásicas [Miettinen98] para resolver problemas multiobjetivo, la complejidad de estos problemas (p. ej., alta dimensionalidad, discontinuidad, multimodalidad) hace necesario recurrir a técnicas alternativas. Uno de los enfoques actuales más exitosos para resolver eficazmente estos problemas es la utilización de los *algoritmos evolutivos multiobjetivo* (AEMOs).

Los AEMOs han resuelto con efectividad varios *problemas de optimización multiobjetivo* (POMs) del mundo real. Sin embargo, una vez que se ha conseguido la efectividad el siguiente paso es resolver los problemas de manera eficiente. Una de las opciones naturales para reducir el tiempo de ejecución es el procesamiento paralelo o distribuido de los AEMOs.

La efectividad es la medida de la «calidad» de las soluciones que encuentra el algoritmo. Mientras que la eficiencia es la medida en la que se hace uso de los recursos computacionales, tales como el espacio de memoria, la utilización de CPU, el espacio de disco, o el tiempo de ejecución del algoritmo evolutivo. En la optimización multiobjetivo se acepta generalmente que la calidad de la solución se evalúa teniendo en cuenta tres objetivos [Zitzler99b]: 1) minimizar la distancia del frente no dominado generado al frente de Pareto real; 2) conseguir una buena distribución de las soluciones; 3) maximizar la amplitud del frente no dominado generado.

Hoy en día se cuenta con un amplio estudio [Alba99, Sawai99, Tomassini99, Cantu00] sobre los algoritmos evolutivos paralelos (pAEs) utilizados para la optimización con una sola función objetivo. Sin embargo, los pAEs no son efectivos para resolver la mayoría de los problemas de optimización multiobjetivo [Veldhuizen03]. Por ejemplo, los pAEs que plantean la utilización de funciones de agregación solamente encuentran una solución por corrida, por lo que se necesitan varias ejecuciones del algoritmo para obtener un conjunto de varias soluciones [Coello99]. Además, algunos pAEs tradicionales no pueden identificar el frente de Pareto completo [Veldhuizen03]. Si tomamos en cuenta que actualmente los AEMOs son utilizados para resolver aplicaciones de ingeniería (las cuales suelen tener funciones objetivo costosas), se muestra plenamente la necesidad de diseñar *algoritmos evolutivos multiobjetivo paralelos* (pAEMOs).

Tradicionalmente se ha distinguido entre paradigmas de paralelización de *grano grueso* y de *grano fino*, según que la relación computación/comunicación fuera alta o baja, respectivamente. La paralelización de los AEMOs, por otro lado, es un campo relativamente nuevo, y como consecuencia, existen pocos trabajos al respecto. En la mayoría de estos trabajos no se discuten detalladamente los aspectos relacionados con el desarrollo y la implementación de los pAEMOs. Entre la cuestiones más importantes que aún no se discuten a profundidad se encuentran las siguientes:

- Argumentos que expliquen por qué un paradigma es apropiado para un dominio de POMs específico.
- Una selección justificada de los POMs de prueba y las métricas de desempeño de los pAEMOs.
- Aspectos del desarrollo e implementación de pAEMOs. Es decir, las estructuras de datos utilizadas, la arquitectura paralela, o la topología de interconexión empleada.
- Un estudio práctico que compare la eficiencia y efectividad de los principales paradigmas de pAEMOs.

1.2. Propuesta

En esta tesis se presenta un nuevo AEMO paralelo competitivo con respecto a los AEMOs representativos del estado del arte desde el punto de vista de la efectividad y la eficiencia. Como producto de este análisis se propuso el algoritmo genético multiobjetivo con múltiples resoluciones (MRMOGA, *'Multiple Resolution Multi-Objective Genetic Algorithm'*) (descrito detalladamente en el capítulo 5). El algoritmo se basa en el paradigma de islas con nodos heterogéneos y se caracteriza por codificar las soluciones con una resolución distinta en cada isla. Con esta técnica se consigue dividir el espacio de búsqueda en regiones disjuntas del espacio de las variables de decisión.

Para analizar el desempeño del algoritmo se identificaron un conjunto de métricas y de problemas de prueba adecuados para evaluar un pAEMO. La efectividad se evaluó con las métricas que se utilizan para evaluar AEMOs secuenciales, a saber: el *conteo de aciertos* (*success count*), la *distancia generacional invertida* (*inverted generational distance*), el *espaciamiento* (*spacing*) y la *cobertura de conjuntos* (*two set coverage*). Para evaluar la eficiencia se utilizaron las siguientes métricas conocidas en la computación paralela: *aceleración*, *eficiencia* y *fracción serial*. Los resultados del algoritmo propuesto se compararon con los obtenidos por una versión paralela del NSGA-II (*Non-dominated Sorting Genetic Algorithm* II) [Deb00].

El algoritmo propuesto presenta un buen compromiso entre efectividad y el costo de comunicaciones que incide en la eficiencia. Con base en los resultados obtenidos podemos afirmar

que el esquema de MRMOGA para dividir el espacio de búsqueda mediante distintas resoluciones mejora considerablemente la eficacia y la eficiencia de un algoritmo paralelo. Además, el algoritmo propuesto tiene una gran capacidad exploratoria ya que, con respecto al algoritmo de referencia, en todas las funciones de prueba encontró soluciones que se extienden mejor a lo largo del frente de Pareto real. Asimismo, es importante mencionar que MRMOGA demostró una capacidad inusualmente buena para resolver problemas con restricciones, incluso para problemas reconocidos por su dificultad.

1.3. Objetivos generales y específicos del proyecto

1.3.1. Objetivo General

El objetivo general de este trabajo de tesis es contribuir al avance del estado del arte con respecto a los algoritmos evolutivos multiobjetivo paralelos (pAEMOs). Con el fin de cumplir este objetivo se presentan los siguientes objetivos particulares:

1. Desarrollar un nuevo algoritmo evolutivo multiobjetivo paralelo competitivo con respecto a los algoritmos representativos del estado del arte.

2. Obtener conocimientos sobre las aspectos clave del diseño y desarrollo de pAEMOs.

3. Identificar un conjunto de POMs de prueba y métricas de análisis específicos para evaluar pAEMOs.

4. Establecer un marco de trabajo para comparar el desempeño del algoritmos paralelos multiobjetivo.

1.4. Estructura de la tesis

La presente tesis está organizada en 7 capítulos y un apéndice. Los cuatro primeros capítulos describen los antecedentes y los conceptos básicos para comprender el trabajo de tesis. Los últimos tres capítulos se dedican a describir el trabajo de tesis y sus resultados.

El capítulo 2 presenta una breve introducción a la *computación evolutiva*. Al inicio del capítulo se proporcionan, de manera breve, los principios biológicos que sustentan a la computación evolutiva. Enseguida se describen los *algoritmos evolutivos* y sus componentes principales. El capítulo termina con una descripción de los paradigmas principales de la computación evolutiva: los *algoritmos genéticos*, las *estrategias evolutivas* y la *programación genética*.

En el capítulo 3 se trata la optimización multiobjetivo usando algoritmos evolutivos. El capítulo comienza dando una definición formal de los problemas de *optimización multiobjetivo*. Asimismo se presentan conceptos y terminología que se necesitan en los capítulos siguientes. Posteriormente se describe cómo se emplean los algoritmos evolutivos para resolver problemas de optimización multiobjetivo y varias de sus características más importantes. Finalmente, se muestra un panorama general de los *algoritmos evolutivos multiobjetivo* que existen actualmente.

El capítulo 4 contiene una introducción a los *algoritmos evolutivos multiobjetivo paralelos*. Inicialmente se presentan terminología y conceptos relacionados con la computación paralela y distribuida utilizados en el resto del capítulo. Asimismo se describen los tres modelos más importantes para paralelizar los algoritmos evolutivos multiobjetivo. También se discuten los elementos clave de un algoritmo multiobjetivo paralelo. Finalmente, se discuten varios de los pAEMOs actuales.

En el capítulo 5 se describe nuestra propuesta de un nuevo pAEMO. Se describen cuáles fueron los criterios que se consideraron para el diseño. Asimismo se detalla el funcionamiento general del algoritmo y se describen las técnicas utilizadas. Finalmente se describen los detalles de la implementación.

El capítulo 6 se dedica a presentar los resultados obtenidos por el algoritmo. Se comienza con una descripción y una justificación de los problemas de prueba y de las métricas utilizadas. También se describen brevemente los algoritmos evolutivos, cuyos resultados se utilizaron para comparar el desempeño de nuestro algoritmo. El capítulo finaliza con la evaluación de los resultados realizando un breve análisis del desempeño de los algoritmos comparados para cada problema de prueba.

En el capítulo 7 se resumen las conclusiones alcanzadas. Además, se analizan algunas de las decisiones de implementación y diseño, y se resaltan los puntos fuertes y débiles del algoritmo propuesto. Para terminar, se describen algunas líneas de trabajo futuro que tienen interés en virtud de los resultados obtenidos.

El apéndice A contiene un glosario de términos de biología que son utilizados a lo largo de este documento.

Capítulo 2

Computación Evolutiva

Sumario

2.1. Introducción

La teoría de la evolución de Darwin, el seleccionismo de Weismann, y la genética de Mendel, confluyen en un conjunto de argumentos que actualmente son conocidos como el paradigma del *neodarwinismo*. Este paradigma afirma que la vida en el planeta es el resultado de cuatro procesos estadísticos que operan sobre las especies, a saber: la *reproducción*, la *mutación*, la *compentencia* y la *selección*. El neodarwinismo produjo una revolución de enorme envergadura, que no solamente unificó el pensamiento moderno de la biología, sino que trascendió los límites de la biología.

La ciencia de la computación concibió la evolución como un proceso de optimización que podía ser simulado para resolver problemas de ingeniería que antes parecían intratables. La idea fundamental de este proceso consiste en evolucionar una población de posibles soluciones para un problema dado, usando operadores inspirados en la variación genética y en la selección natural. La evolución es, en efecto, un método de búsqueda entre una inmensa cantidad de «soluciones» posibles. En el ámbito biológico el enorme conjunto de posibilidades lo constituyen el conjunto de posibles secuencias genéticas, y las soluciones deseables son los organismos altamente aptos, es decir, los organismos mejor dotados para sobrevivir y reproducirse en su ambiente.

Actualmente existen tres vertientes que se inspiran en los principios del neodarwinismo: las *estrategias evolutivas*, la *programación evolutiva* y los *algoritmos genéticos*. Conjuntamente, a estas técnicas se les denomina *computación evolutiva*.

El resto de este capítulo está organizado de la siguiente manera: en la sección 2.2 se presentan los principios biológicos que fundamentan el neodarwinismo. Posteriormente, en la sección 2.3

se introducen los conceptos básicos referentes a la computación evolutiva. Por último, la sección 2.4 se dedica a describir los paradigmas principales de la computación evolutiva.

2.2. Antecedentes históricos

Si bien Charles Darwin (figura 2.1) no es considerado el padre del evolucionismo, sí fue quien logró que éste se implantara definitivamente. En 1859, Charles Robert Darwin publicó su revolucionaria obra *El origen de las especies por medio de la selección natural o la preservación de las razas favorecidas en su lucha por la vida* [Darwin59]. En su obra, Darwin argumenta que los individuos con características favorables tendrán más probabilidad de ser preservados dentro de su lucha por la vida. Además, estos individuos, por medio del *principio de herencia*, transmitirán estas características a su descendencia. A este principio de preservación Darwin lo llamó *selección natural*.

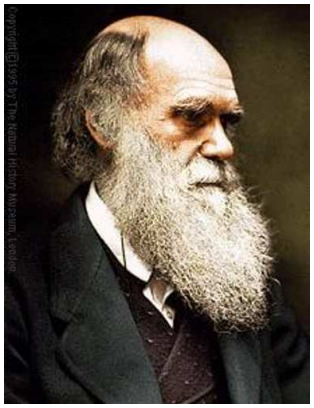


Figura 2.1: Charles Robert Darwin.

Los principios de la selección natural se pueden resumir en cuatro proposiciones:

1. En cualquier generación, no todos los individuos de una especie logran reproducirse.
2. Los miembros de la especies no son idénticos; tienen variaciones individuales.
3. La mayoría de las variaciones son heredadas y transmitidas de los padres a los descendientes.
4. El éxito reproductivo no es aleatorio. Está asociado con las características heredadas; algunas de ellas son más benéficas que otras en situaciones particulares.

Una de las principales deficiencias del argumento de Darwin es que, a pesar de que la herencia juega un papel preponderante en su teoría, no ofrece una explicación acerca de su funcionamiento. Más tarde, por fortuna, los mecanismos de la herencia serían explicados en los trabajos realizados por Weismann y Mendel.

Treinta años después de la publicación del *Origen de las especies*, el biólogo alemán August Friedrich Leopold Weismann (figura 2.2) establece la *teoría del germoplasma* [Weismann93], que afirma que los organismos multicelulares están constituidos por células germinales (germoplasma) que contienen la información de la herencia, y células somáticas (somatoplasma) que se encargan de la funciones corporales.

Figura 2.2: August Friedrich Leopold Weismann.

Weismann llegó a la conclusión de que el germoplasma no puede ser afectado por las habilidades que el organismo aprende (es decir, cambios en el somatoplasma) durante su vida, así que no puede transmitir esta información a la siguiente generación. De estas dos células, el somatoplasma muere, pero el germoplasma es prácticamente inmortal, ya que se transmite de una generación a otra.

Por su parte, el monje austríaco Johann Gregor Mendel (figura 2.3) publicó el trabajo *Experimentos de hibridación en plantas* [Mendel01]. En éste se resumían los experimentos que había llevado a cabo con la especie de guisante *Pisum sativum*. Como resultado de sus experimentos, Mendel estableció tres leyes fundamentales de la genética. La primera de ellas, la ley de segregación, sostiene que los genes recibidos de los padres se segregan (separan) durante la formación de gametos sin afectarse entre sí. La segunda ley, llamada de independencia, sostiene que los pares de alelos se independizan durante la formación de gametos. La tercera, la ley de la uniformidad, sostiene que cada característica heredada se determina mediante dos factores provenientes de ambos padres, lo cual decide si un gene determinado es dominante o recesivo.

Figura 2.3: Johan Gregor Mendel.

Actualmente, a la teoría evolutiva postulada por Darwin, complementada con el seleccionismo de Weismann, y la genética de Mendel se le denomina *neodarwinismo*. Este paradigma sostiene que toda la diversidad de vida del planeta se puede explicar mediante unos pocos procesos estadísticos que actúan sobre las especies. Estos procesos son la *reproducción*, la *mutación*, la *competencia* y la *selección*.

2.3. Conceptos de Computación Evolutiva

En las secciones siguientes se definen algunos conceptos básicos de la computación evolutiva. Los términos empleados en estas secciones equivalen de manera muy cercana a su contraparte genética. El lector interesado puede consultar el apéndice A para encontrar la acepción de estos términos en un contexto puramente genético.

2.3.1. Representación

Denominamos *cromosoma* a una estructura de datos que contiene una cadena de variables de diseño de algún problema (figura 2.4). Usualmente es una cadena binaria, pero también es posible usar una cadena con valores enteros o reales.

Figura 2.4: Ejemplo de un cromosoma binario (a) y otro real (b).

Llamamos *gene* a una subcadena del cromosoma que codifica, comúnmente, a un solo parámetro de diseño. Estos genes toman ciertos valores, llamados *alelos*, de algún alfabeto genético. De esta manera, si usamos una representación binaria, los alelos pueden tomar el valor de 0 o de 1. Un *locus* define la posición de un gene dentro del cromosoma (figura 2.5).

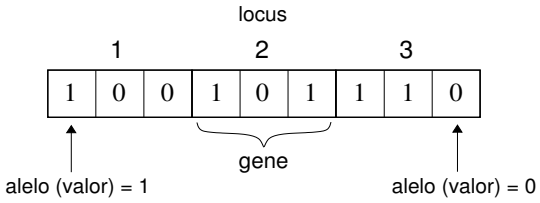


Figura 2.5: Cromosoma binario constituido por tres genes.

Se denomina *genotipo* a la codificación (por ejemplo, binaria, entera o real) de los parámetros de diseño. Mientras que *fenotipo* es la decodificación del genotipo, con el fin de obtener los valores de los parámetros usados como entrada en la función objetivo del problema que se trata (figura 2.6).

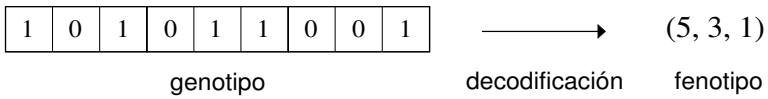


Figura 2.6: Decodificación del genotipo al fenotipo.

Un *individuo* es una solución potencial al problema que se trata. Cada individuo contiene un cromosoma. A un conjunto de individuos se le nombra *población*. La *aptitud* de un individuo es la evaluación de la función de aptitud e indica qué tan bueno es el individuo (es decir, la solución al problema) con respecto a los demás. Usualmente, la función de aptitud es igual a la función objetivo del problema. Por ejemplo, si $f(x) = x^2$, entonces $f(1010_2) = f(10) = 100$ es la aptitud del individuo.

2.3.2. Operadores evolutivos

Existe una amplia variedad de operadores evolutivos que se han empleado en los algoritmos evolutivos. Algunos de ellos están diseñados especialmente para una clase particular de problemas. En esta sección nos limitaremos a describir los operadores usados comúnmente en los algoritmos genéticos.

Selección. La selección determina la probabilidad de elegir un individuo para que produzca descendencia por medio de la recombinación y la mutación. El esquema de selección es una de las partes cruciales de un algoritmo evolutivo, puesto que sesga la búsqueda de manera que eventualmente se llegue a la solución óptima (o su proximidad). Dentro de las técnicas de selección empleadas en los algoritmos genéticos se distinguen 3 grupos principalmente:

- ***Selección proporcional***. Estos esquemas de selección, propuestos por Holland [Holland75], eligen individuos de acuerdo a su contribución de aptitud con respecto a la población total. Algunos ejemplos de este esquema de selección incluyen:
  1. *La ruleta*. Esta técnica, propuesta por De Jong [DeJong75], simula una ruleta cuyas casillas tienen un tamaño proporcional a la aptitud de cada individuo. La ruleta se gira una vez por cada individuo que se requiera seleccionar.
  2. *Muestreo determinístico*. De Jong [DeJong75] propone tomar la parte entera del valor esperado[1] de cada individuo para determinar su número de copias en el grupo de padres. Posteriormente, la población es ordenada de acuerdo a la partes fraccionarias. Los individuos que falten para completar el grupo de padres son tomados de la lista ordenada.
  3. *Sobrante estocástico*. Booker [Booker82] y Brindle [Brindle81] proponen una técnica similiar al muestreo determinístico. Aquí también se usa la parte entera de los valores esperados para determinar el número de copias de un individuo, pero los individuos que resten para completar el grupo de padres se eligen mediante un esquema de selección propocional utilizando la parte fraccionaria del valor esperado.
- ***Selección mediante torneo*** [Wetzel83]. Estos métodos se basan en la comparación directa entre dos o más individuos para decidir el ganador. En esta técnica se elige una muestra aleatoria de individuos y el mejor individuo de la muestra es seleccionado. Existen dos variantes de la selección por torneo:
  1. *Determinística*. Siempre se elige al individuo más apto de la muestra.
  2. *Probabilística*. Se aplica la operación $flip(p)$ [2], y si el resultado es $cierto$, se elige al individuo más apto; en caso contrario, se elige al menos apto.
- ***Selección de estado uniforme***. Esta técnica fue propuesta por Whitley [Whitley89] y se usa en algoritmos genéticos no generacionales, es decir, cuando solamente unos cuantos individuos son reemplazados por la descendencia en cada generación.

Recombinación. Los operadores de recombinación (cruza) tienen la finalidad de heredar la información (genes) de dos o más padres a la descendencia. En la computación evolutiva la

[1] El valor esperado, $e(x) \in \mathbb{R}$, denota el número de copias de un individuo x, que se esperan tener en el grupo de padres.

[2] $flip(p)$ devuelve $cierto$ con una probabilidad p.

cruza entre cromosomas se simula intercambiando segmentos de cadenas lineales de longitud fija. Lo usual es que la técnicas de cruza se apliquen sobre representaciones binarias, sin embargo, con la modificaciones adecuadas, se pueden generalizar a alfabetos de cardinalidad mayor. Entre las técnicas de cruza más usuales se encuentra la cruza de n puntos, propuesta por De Jong [DeJong75]. Para crear un nuevo cromosoma, los padres mezclan segmentos de su cadena cromosómica alternadamente según los puntos de cruza. En la figura 2.7 se muestra, a modo de ejemplo, una cruza de 2 puntos.

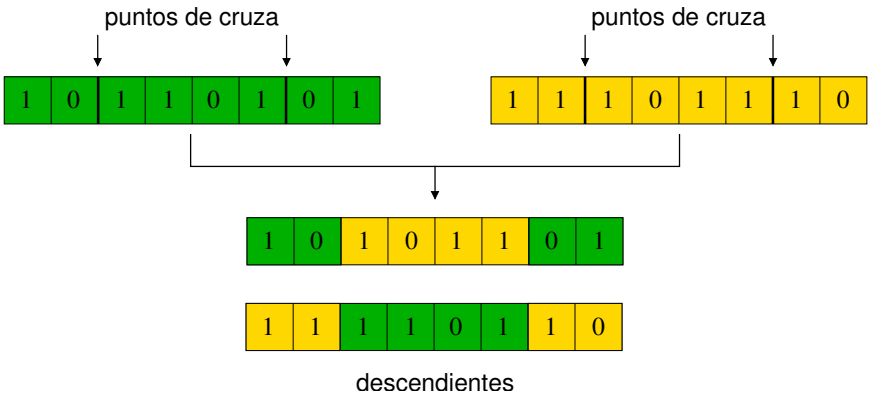


Figura 2.7: Cruza de $n = 2$ puntos.

Mutación. La mutación se consigue realizando pequeñas modificaciones al cromosoma de un individuo. En los algoritmos genéticos la mutación es considerada un operador secundario. Para las representaciones binarias, una de las más utilizadas es la mutación uniforme. En esta técnica cada valor (alelo) de la cadena es intercambiado por su contrario (es decir, de 0 a 1 o viceversa) con una cierta probabilidad (figura 2.8). La mutación uniforme se puede generalizar para cadenas de otros alfabetos. En este caso, cada alelo en la cadena es reemplazado, con cierta probabilidad, por un alelo distinto elegido aleatoriamente .

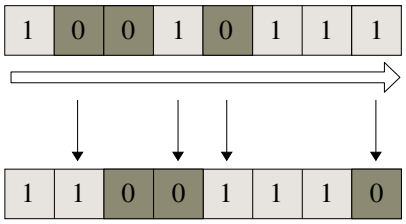


Figura 2.8: Mutación uniforme con probabilidad de 0.5 de intercambiar el alelo de cada posición del cromosoma binario.

2.4. Principales paradigmas

Actualmente existen, principalmente, tres paradigmas inspirados en los principios del neodarwinismo: los *algoritmos genéticos* (AG), la *programación evolutiva* (PE) y las *estrategias evolutivas* (EE). Estas técnicas se originaron en los años 60 y se desarrollaron de manera independiente durante casi 15 años. Sin embargo, a principios de los años 90 se concibieron como diferentes paradigmas de una sola tecnología denominada *computación evolutiva* (CE) [Eiben03]. Conjuntamente, a los algoritmos de la CE se les llama *algoritmos evolutivos* (AE). Estas tres técnicas tienen

en común la reproducción, la variación aleatoria, la competencia, y la selección de individuos contendientes dentro de una población. Estas técnicas difieren en los detalles técnicos para implementar los operadores evolutivos descritos en la sección anterior. Por ejemplo, la representación de las soluciones difiere en los tres paradigmas: los AG utilizan cadenas sobre un alfabeto finito (típicamente binario), las EE utilizan vectores con valores reales, y la PE usa autómatas de estado finito. En el algoritmo 2.1 se muestra el esquema general que describe un algoritmo evolutivo.

1: *Inicializar* la población con soluciones candidatas aleatorias
2: *Evaluar* cada solución candidata.
3: **mientras** no se satisfaga la la condición de terminación **haz**
4: *Seleccionar* padres
5: *Recombinar* pares de padres
6: *Mutar* la descendencia resultante.
7: *Evaluar* los nuevos candidatos
8: *Seleccionar* individuos para la siguiente generación.
9: **fin mientras**

Algoritmo 2.1: Pseudocódigo del esquema general de un algoritmo evolutivo.

2.4.1. Estrategias Evolutivas

Las estrategias evolutivas (EEs) tienen su origen en la Universidad Técnica de Berlín en 1964 con el trabajo de Peter Bienert, Ingo Rechenberg, y Hans-Paul Schwefel [Fogel99]. Este modelo enfatiza los nexos conductuales entre padres e hijos, en lugar del nexo genético. Las EEs fueron concebidas para construir sistemas capaces de resolver problemas de optimización complejos con parámetros con valores reales. Por ello, la representación natural fue un vector de genes con valores reales, el cual era manipulado, principalmente, por operadores de mutación que perturbaban los valores reales [Back97].

La primera versión de una estrategia evolutiva, nombrada $(1+1)$-EE, creaba un solo hijo a partir de un solo padre. Ambos competían para sobrevivir; el peor individuo era eliminado, mientras que el mejor se mantenía para la siguiente generación.

En la $(1+1)$-EE, a partir de un padre $\mathbf{x}^{(t)} = (x_1, x_2, \ldots, x_n)$, el hijo se generaba mediante la expresión:

$$x_i^{(t+1)} = x_i^{(t)} + N_i(0, \sigma_i^{(t)}) \tag{2.1}$$

donde t se refiere a la generación actual, $i = 1, \ldots, n$ es la i-ésima componente de los vectores $\mathbf{x}$, $\mathbf{N}$ y σ, y $N_i(0, \sigma_i^{(t)})$ es un número aleatorio gaussiano con media cero y desviación estándar σ_i. Los n números aleatorios son generados de manera independiente.

Más tarde, Rechenberg [Rechenberg73] propone la EE con una población de padres, $(\mu+1)$-EE, en la cual hay μ padres que generan un solo hijo que puede reemplazar al padre de la población.

Por su parte, Schwefel [Schwefel74] formula las estrategias $(\mu+\lambda)$-EE y (μ, λ)-EE, donde μ son los padres y λ son los hijos. La primera estrategia selecciona μ individuos de la unión de padres e hijos; la segunda selecciona μ individuos solamente de la población de hijos. Con base en los resultados de los experimentos con $(\mu+1)$-EE, Schwefel encontró la manera de que el programa adaptara $\sigma^{(t)}$ de manera automática, a la cual denominó *autoadaptación* [Schwefel77]. Para este propósito, cada individuo se representa con la tupla $(\mathbf{x}^{(t)}, \sigma^{(t)})$, y la nueva solución

$(\mathbf{x}^{(t+1)}, \sigma^{(t+1)})$ se crea mediante:

$$\sigma_i^{(t+1)} = \sigma_i^{(t)} \times \exp(\tau' N(0,1) + \tau N_i(0,1)) \quad (2.2)$$

$$x_i^{(t+1)} = x_i^{(t)} + N(0, \sigma_i^{(t+1)}) \quad (2.3)$$

donde $N(0,1)$ representa un número aleatorio gaussiano con media cero y desviación estándar uno. $N_i(0,1)$ denota un número aleatorio distinto para cada i, mientras que τ y τ' son constantes de proporcionalidad que están en función de n.

2.4.2. Programación Evolutiva

La programación evolutiva (PE) fue propuesta por Lawrence J. Fogel en los años 60 durante su estancia en la *National Science Foundation* [Fogel64]. Fogel consideraba que el comportamiento inteligente requiere de dos habilidades: 1) la habilidad de un organismo para hacer predicciones correctas dentro de su ambiente; 2) la capacidad de traducir estas predicciones en una respuesta adecuada para una meta dada. Los autómatas de estado finito fueron la representación ideal para modelar este comportamiento.

Originalmente, Fogel definió la PE de la siguiente manera: una población de autómatas de estado finito padres eran expuestos a una secuencia de símbolos (el ambiente). Conforme cada símbolo de entrada era presentado a los autómatas padre, se hacía una predicción para determinar el símbolo de salida, el cual era comparado con el siguiente símbolo de entrada. La idea era que el autómata, eventualmente, pudiera predecir la secuencia futura de símbolos de entrada. Se utilizaba una función de «recompensa» para medir la certeza de la predicciones.

La descendencia era creada mutando cada máquina padre, considerando cinco tipos de mutaciones: 1) cambiar el símbolo de salida; 2) cambiar la transición a un estado; 3) agregar un estado; 4) eliminar un estado; y 5) cambiar el estado inicial. Estos descendientes eran evaluados de la misma manera que los padres. Los mejores autómatas con respecto a la función de recompensa son retenidos para ser los padres de la siguiente generación.

Los PEs fueron aplicados exitosamente a problemas de predicción, identificación, control automático, y reconocimiento de patrones, entre otros [Back97].

2.4.3. Algoritmos Genéticos

Los algoritmos genéticos (AG) (originalmente llamados «planes reproductivos genéticos») fueron propuestos por John H. Holland [Holland75] en el contexto del aprendizaje de máquina. Hay tres características principales que distinguen a los AG de los demás algoritmos evolutivos: la representación de los individuos (tradicionalmente una cadena binaria); el método de selección (selección proporcional); y el operador principal para modificar al individuo. El operador principal es la cruza sexual, mientras que la mutación es un operador secundario.

Goldberg define a los algoritmos genéticos de la siguiente manera [Goldberg89]:

> Algoritmos de búsqueda basados en los mecanismos de la selección natural y de la genética natural. Combinan la supervivencia del más apto entre individuos representados por cadenas, con un intercambio de información aleatorio para formar un algoritmo de búsqueda que refleje algo del ingenio de la búsqueda humana.

El funcionamiento del algoritmo básico es el siguiente: se genera una población de individuos $P(0)$ y cada individuo es evaluado para asignarle una aptitud. Posteriormente se seleccionan algunos individuos para aparearse y copiarse al grupo de padres. La probabilidad de seleccionar un individuo es proporcional a su aptitud (presuponiendo un esquema de selección

proporcional). Así, los mejores individuos tienen más opurtunidad de tener descendientes. Luego, los operadores genéticos (cruza y mutación) se aplican al grupo de padres, produciendo la descendencia $C'(t)$. El pseudocódigo de este procedimiento se muestra en el algoritmo 2.2.

1: $t \leftarrow 0$
2: inicializar $P(t)$
3: evaluar los individuos en $P(t)$
4: **mientras** no se satisfaga la la condición de terminación **haz**
5: $t \leftarrow t + 1$
6: seleccionar padres $C(t)$ de $P(t-1)$
7: cruzar y mutar los individuos en $C(t)$ para formar $C'(t)$
8: evaluar los individuos en $C'(t)$
9: selecciona nueva población $P(t)$ de $C'(t)$ y $P(t-1)$
10: **fin mientras**

Algoritmo 2.2: Esquema básico de un algoritmo genético.

En opinión de [Michalewicz96], para aplicar un AG a un problema se requieren cinco componentes principales:

1. Un esquema de representación y codificación de las soluciones potenciales del problema.
2. Una función de evaluación para clasificar a las soluciones de acuerdo a su aptitud.
3. Operadores genéticos que alteren la composición de los hijos que se producirán para las siguientes generaciones.
4. Valores para los parámetros que utiliza el algoritmo genético (p. ej., tamaño de la población, porcentaje de cruza, porcentaje de mutación, número máximo de generaciones.)

Capítulo 3

Optimización multiobjetivo con algoritmos evolutivos

Sumario

3.1. Introducción

En la ingeniería, en las ciencias de la computación, y en otras áreas, la mayoría de los problemas que se presentan involucran la optimización simultánea de varios objetivos. En varios casos, los objetivos se expresan en unidades de medida no comparables, o se encuentran en conflicto mutuamente (es decir, que un objetivo no puede ser mejorado sin empeorar el valor de otro). Estos problemas son llamados *multiobjetivo* o *multicriterio*. En la optimización global la solución óptima está claramente definida. Por el contrario, en la optimización multiobjetivo, debido a la naturaleza conflictiva de sus objetivos, existe un conjunto de soluciones compromiso alternativas. Este conjunto es conocido como *conjunto de óptimos de Pareto*, y a su proyección en el espacio de los objetivos se le denomina *frente de Pareto*. Estas soluciones son óptimas en el sentido de que ninguna de las demás soluciones del espacio de búsqueda son superiores a ellas cuando se consideran todos los objetivos simultáneamente.

La investigación de operaciones ha desarrollado un número considerable de técnicas para resolver problemas multiobjetivo [Miettinen98]. Sin embargo, dada la complejidad (p. ej., alta dimensionalidad, discontinuidad, multimodalidad) de la mayoría de los problemas de optimización multiobjetivo del mundo real, estas técnicas tienen severas limitantes (p. ej., algunas son aplicables sólo cuando el frente de Pareto es convexo) o resultan incluso inaplicables para resolverlos. Uno de los enfoques más exitosos para resolver eficazmente estos problemas lo constituyen los algoritmos evolutivos. Estos algoritmos son especialmente apropiados para resolver problemas multiobjetivo ya que su naturaleza poblacional les permite encontrar varios

elementos del conjunto de soluciones compromiso en una sola ejecución. Además, son menos susceptibles a la forma y continuidad del frente de Pareto.

La organización de este capítulo es la siguiente: la sección 3.2 presenta un resumen de los conceptos básicos concernientes a la optimización de problemas multiobjetivo, necesarios para comprender el resto de esta tesis. La sección 3.3 se destina para describir cómo se emplean los algoritmos evolutivos para resolver problemas de optimización multiobjetivo, y varias de sus características más importantes. Finalmente, en la sección 3.5 se muestra un panorama de los algoritmos evolutivos multiobjetivo que existen actualmente.

3.2. Conceptos de optimización multiobjetivo

Informalmente, un problema de optimización multiobjetivo (POM[1]), se define como [Osyczka84]:

> El problema de encontrar un vector de variables de decisión que satisfaga un conjunto de restricciones y optimice una función vectorial cuyos elementos representan a las funciones objetivo. Estas funciones son una descripción matemática de criterios de desempeño que están en conflicto entre sí. En este sentido, el término «optimizar» significa encontrar aquella solución que proporcione valores para todas las funciones objetivo aceptables para el diseñador.

Concepto 1 (Variables de decisión). Las variables de decisión son el conjunto de parámetros para los cuales se buscan los valores que resuelvan el problema de optimización. La variables pueden ser enteras, reales, o una combinación de ambas. Estos parámetros se denotan mediante $x_j,\ j = 1, 2, \ldots, n$.

El vector de n variables de decisión $\vec{x}$ se representa mediante:

$$\vec{x} = [x_1, x_2, \ldots, x_n]^T \tag{3.1}$$

Concepto 2 (Funciones objetivo). Las funciones objetivo de un POM se denotan mediante $f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})$, donde k ($k \geq 2$) es el número de funciones objetivo del POM. De esta manera, las funciones objetivo conforman una función vectorial $\vec{f}(\vec{x})$ que se define mediante:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{3.2}$$

De los dos conceptos anteriores (1 y 2), es importante notar que intervienen dos espacios euclidianos diferentes.

1. El espacio n-dimensional de las variables de decisión (llamado también espacio de decisión, o espacio genotípico), en el cual cada eje coordenado corresponde con cada componente del vector $\vec{x}$.

2. El espacio k-dimensional de las funciones objetivo (llamado también espacio de criterios, o espacio fenotípico), en el cual cada eje coordenado corresponde con cada componente del vector $\vec{f}(\vec{x})$.

[1] El término en inglés es *Multiobjective Optimization Problem* (MOP).

Concepto 3 (Restricciones). En la mayoría de los problemas del mundo real existen restricciones dadas por las condiciones del ambiente o los recursos disponibles. Es preciso que las restricciones sean cumplidas para que las soluciones encontradas tengan validez. Las restricciones se modelan matemáticamente mediante desigualdades:

$$g_i(\vec{x}) \leq 0 \qquad i = 1, \ldots, m \tag{3.3}$$

o igualdades:

$$h_i(\vec{x}) = 0 \qquad i = 1, \ldots, p \tag{3.4}$$

Concepto 4 (Problema de optimización multiobjetivo). Formalmente, un POM se define como el problema de encontrar el vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ que satisfaga m restricciones de desigualdad

$$g_i(\vec{x}) \leq 0 \qquad i = 1, \ldots, m \tag{3.5}$$

p restricciones de igualdad

$$h_i(\vec{x}) = 0 \qquad i = 1, \ldots, p \tag{3.6}$$

y optimice la función vectorial

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{3.7}$$

En la definición anterior el vector $\vec{x}^*$ se destina para denotar a las soluciones óptimas (usualmente más de una). Las restricciones dadas por las ecuaciones (3.5) y (3.6) definen una región factible Ω dentro del espacio de las variables de decisión; cualquier solución $\vec{x}$ dentro de esta región es considerada una solución factible. La función vectorial $\vec{f}(\vec{x})$ proyecta el conjunto Ω al conjunto Λ el cual representa el conjunto de todos los valores posibles de las funciones objetivo.

En la figura 3.1 se muestra, a manera de ejemplo, una función vectorial $\vec{f}(\vec{x})$ para el caso de 2 variables de decisión y 3 funciones objetivo.

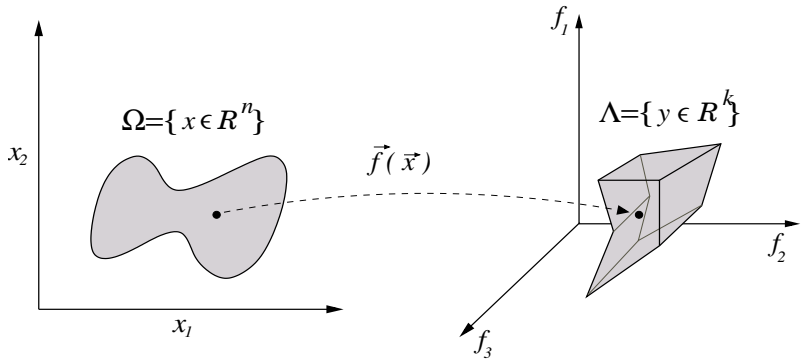


Figura 3.1: Función de proyección de un POM.

3.2.1. Optimalidad de Pareto

En los problemas de optimización de un solo objetivo, el algoritmo se concentra en encontrar el óptimo global (o su vecindad). Sin embargo, en los problemas multiobjetivo, tratamos con un valor de aptitud por cada función objetivo. Esto hace que la noción de «óptimo» se torne

más compleja en este contexto. Para comprender mejor esta situación, pongamos por caso, el problema de optimizar el diseño de un puente, en el cual se requiere minimizar el riesgo de accidentes y el costo del puente (figura 3.2). La solución A consigue un costo bastante bajo, pero a costa de un alto riesgo; por otro lado, la solución D implica un costo alto, pero minimiza el riesgo de accidentes; alternativamente, la solución C tiene valores promedio en ambos objetivos. En este ejemplo, ninguna de las soluciones puede ser considerada mejor que la otra con respecto a todos los objetivos. Por esta razón, en los POM, con objetivos en conflicto, buscamos un conjunto de soluciones óptimas, en lugar de una sola solución. La noción de optimalidad que se acepta comúnmente es aquella que propuso originalmente Francis Ysidro Edgeworth [Edgeworth81] y que más tarde generalizara Vilfredo Pareto [Pareto96].

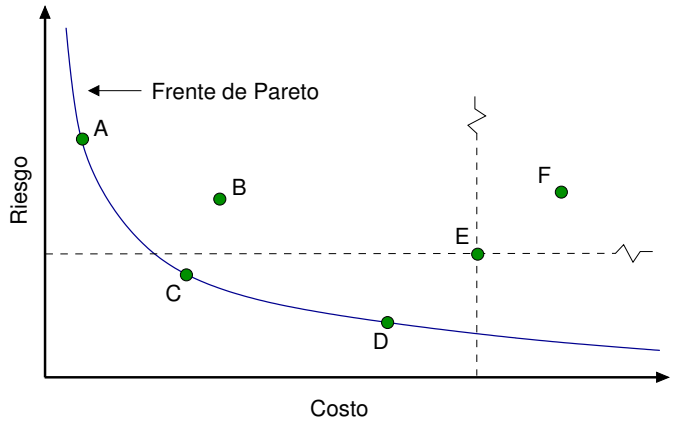


Figura 3.2: Ilustración de los conceptos de optimalidad de Pareto, dominancia de Pareto y frente de Pareto.

Concepto 5 (Optimalidad de Pareto). Un punto $\vec{x}^* \in \Omega$ es un óptimo de Pareto si para $I = 1, 2, \ldots, k$ no existe $\vec{x} \in \Omega$ tal que

$$\forall_{i \in I}(f_i(\vec{x}) \leq f_i(\vec{x}^*)) \tag{3.8}$$

y para al menos una $i \in I$

$$f_i(\vec{x}) < f_i(\vec{x}^*) \tag{3.9}$$

Concepto 6 (Dominancia de Pareto). Decimos que un vector $\vec{u} = (u_1, \ldots, u_k)$ domina a $\vec{v} = (v_1, \ldots, v_k)$ (denotado mediante $\vec{u} \preceq \vec{v}$) si y sólo si $\vec{u}$ es parcialmente menor que $\vec{v}$. Es decir si

$$u_i \leq v_i \quad \forall\, i \in \{1, \ldots, k\} \tag{3.10}$$

y

$$\exists\, i \in \{1, \ldots, k\} : u_i < v_i \tag{3.11}$$

En el ejemplo de la figura 3.2 observamos que la solución E domina a la solución F, y a su vez, la solución E es dominada por la solución D.

Concepto 7 (Conjunto de óptimos de Pareto). Para un POM $\vec{f}(\vec{x})$ dado, el conjunto de óptimos de Pareto ($\mathcal{P}^*$) se define como:

$$\mathcal{P}^* := \{\vec{x} \in \Omega \mid \neg\, \exists\, \vec{y} \in \Omega \,:\, \vec{f}(\vec{y}) \preceq \vec{f}(\vec{x})\}. \tag{3.12}$$

Concepto 8 (Frente de Pareto). Para un POM $\vec{f}(\vec{x})$ y un conjunto de óptimos de Pareto $\mathcal{P}^*$, el frente de Pareto ($\mathcal{FP}^*$) se define como:

$$\mathcal{FP}^* := \{\vec{u} = \vec{f} = (f_1(\vec{x}), \ldots, f_k(\vec{x})) \mid \vec{x} \in \mathcal{P}^*\}. \tag{3.13}$$

Volviendo al ejemplo de la figura 3.2, las soluciones A, C y D dominan a todas las demás, y por lo tanto, pertenecen al frente de Pareto.

3.3. Algoritmos evolutivos multiobjetivo (AEMO)

Hoy en día existen más de 30 técnicas de programación matemática para resolver problemas de optimización multiobjetivo [Miettinen98]. Con todo, la complejidad de muchos POMs del mundo real vuelve a estas técnicas inadecuadas o incluso inaplicables para resolverlos. La complejidad de estos problemas se debe, por ejemplo: a la multimodalidad, a la alta dimensionalidad del espacio de búsqueda, a la discontinuidad de sus funciones objetivo, a desconexiones tanto en el espacio de la variables de decisión como en el de la funciones objetivo, o a que son NP-completos.

Algunos investigadores [Coello99, Deb99, Fogel99, Michalewicz00] han identificado algunas dificultades que tienen las técnicas clásicas para resolver POMs. A continuación se listan algunas de ellas:

1. Los algoritmos se necesitan ejecutar varias veces para encontrar varias soluciones del conjunto de óptimos de Pareto.
2. La mayoría de los algoritmos requieren información sobre el dominio del problema que se trata.
3. Algunos algoritmos son sensibles a la forma o continuidad del frente de Pareto.
4. En los problemas que involucran incertidumbre o eventos estocásticos, los métodos clásicos son inadecuados.
5. La dispersión de las soluciones del frente de Pareto depende de la eficiencia del optimizador monoobjetivo.

La complejidad de los POMs del mundo real ha conducido a la búsqueda de enfoques alternativos para resolver este tipo de problemas. Uno de esos enfoques lo encabezan los algoritmos evolutivos.

A finales de 1960, Rosenberg [Rosenberg67] plantea utilizar un método genético de búsqueda para resolver problemas de optimización multiobjetivo. No obstante, no fue hasta 1984 cuando David Schaffer [Schaffer84] propone la primera implementación de lo que actualmente conocemos como algoritmo evolutivo multiobjetivo (AEMO[2]). A partir de ese momento, varios investigadores [Coello01, Srinivas94, Zitzler99, Horn93, Fonseca93, Knowles99] han desarrollado su propio AEMO. La publicación de los resultados de estos algoritmos mostró la superioridad de los AEMOs sobre la técnicas clásicas de programación matemática.

Los algoritmos evolutivos (AEs) son naturalmente adecuados para resolver POMs gracias a que trabajan simultáneamente con un conjunto de soluciones potenciales (es decir, la población). Esta característica les permite encontrar varias soluciones del conjunto óptimo de Pareto en una sola ejecución. Asimismo, son menos sensibles a la forma o continuidad del frente de Pareto.

En general, las características fundamentales de un AEMO, de acuerdo con [Zitzler99a] son las siguientes:

[2] El término en inglés es *Multi-Objective Evolutionary Algorithm* (MOEA).

- 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.

Los AEs y los AEMOs son estructuralmente similares. La diferencia fundamental es que un AEMO calcula k ($k \geq 2$) funciones de aptitud. Sin embargo, el operador de selección espera un solo valor de aptitud. La forma más sencilla de mantener la estructura de un AE simple al abordar problemas multiobjetivo consiste en transformar el vector de aptitudes en un valor escalar (en la sección 3.4.1 se muestran algunos procedimientos para tal fin). En el algoritmo 3.1 se describe formalmente la estructura básica de un algoritmo evolutivo multiobjetivo.

1: $t \leftarrow 0$
2: Generar aleatoriamente la población inicial $P(t)$
3: **mientras** no se cumpla la condición de terminación **haz**
4: Para cada individuo $x \in P(t)$ calcular el valor escalar de aptitud $F(x)$
5: Seleccionar de $P(t)$ un grupo de padres $P'(t)$ basándose en la aptitud $F(x)$
6: Recombinar los individuos de $P'(t)$ para obtener $P''(t)$
7: Mutar los individuos de $P'''(t)$
8: $P(t+1) \leftarrow P'''(t)$
9: $t \leftarrow t+1$
10: **fin mientras**

Algoritmo 3.1: Algoritmo evolutivo multiobjetivo

3.4. Elementos clave de los AEMOs

3.4.1. Asignación de aptitud

Una de las diferencias más importantes entre un AE y un AEMO estriba en el esquema para asignar la aptitud a los individuos de la población. Mientras que en un AE se calcula un solo valor de aptitud (generalmente idéntico al valor de la función objetivo), en un AEMO se calculan varias funciones de aptitud. Por consiguiente, en un AEMO se necesita un proceso adicional para convertir este vector de aptitudes en un escalar. En general se distinguen tres esquemas para llevar a cabo este proceso [Zitzler99]: basados en criterios, basados en agregación, y basados en el concepto de dominancia de Pareto.

Basados en criterios. Esta clase de AEMOs elige alternadamente una de las funciones objetivo durante la etapa de selección. Cada vez que se elige un individuo para reproducirse, un objetivo distinto decidirá cuál miembro de la población será seleccionado para formar el grupo de padres.

Basados en agregación. Este esquema está inspirado en las técnicas clásicas de optimización. De manera similar a aquellas técnicas, en este método las funciones objetivo son agregadas (o combinadas) en una sola función objetivo parametrizada. Los parámetros de esta función son sistemáticamente modificados durante el proceso de optimización para encontrar un conjunto de soluciones no dominadas.

Basados en jerarquización de Pareto. El esquema para asignar la aptitud con base en la dominancia de Pareto fue planteado por Goldberg [Goldberg89]. Sea x una solución y $\mathbf{x}_u$ su

vector de objetivos en la generación t. Entonces el algoritmo 3.2 describe el esquema de Goldberg.

```
1:  Parámetros:
2:      N: tamaño de la población
3:
4:  j_actual ← 1
5:  m ← N
6:  mientras N ≠ 0 haz
7:      para i ← 1 hasta m haz
8:          si x_u es no dominada entonces
9:              jerarquía(x, y) = j_actual
10:         fin si
11:     fin para
12:     para i ← 1 hasta m haz
13:         si jerarquía(x, y) = j_actual entonces
14:             Separar x, y almacenarlo aparte
15:             N ← N − 1
16:         fin si
17:     fin para
18:     j_actual ← j_actual + 1
19:     m = N
20: fin mientras
```

Algoritmo 3.2: Algoritmo de asignación de jerarquía de Goldberg

Hoy en día los esquemas basados en la jerarquización de Pareto son los más populares en el campo de la optimización multiobjetivo. Muchos investigadores [Fonseca93, Horn93, Srinivas94] han propuesto sus propios esquemas de asignación de aptitud basados en este esquema. Pese a esta popularidad, algunos investigadores [Fonseca95, Purshouse03] afirman que los AEs «puros» no tienen buen desempeño en problemas que involucran muchos objetivos en conflicto debido a la elevada dimensionalidad y el tamaño del conjunto de óptimos de Pareto.

3.4.2. Elitismo

Debido a los efectos de muestreo y a la disrupción de los operadores genéticos las buenas soluciones se pueden perder durante el proceso de optimización. Para resolver este problema, De Jong [DeJong75] propuso un mecanismo llamado elitismo, el cual retiene a las mejores soluciones de la población de una generación a otra sin alteración. A diferencia de la optimización global, la implementación del elitismo es más compleja en los AEMOs. En lugar de un solo mejor individuo, hay un conjunto de elite de un tamaño considerablemente mayor que la población. En la actualidad hay dos enfoques principales para llevar a cabo el mecanismo de elitismo. Uno de ellos es combinar la población vieja y la descendencia, y posteriormente aplicar una selección determinista [Deb00] en lugar de reemplazar la población vieja por la descendencia. Otro enfoque es mantener una población secundaria llamada archivo histórico[3], en el cual se retienen las soluciones no dominadas encontradas en el transcurso del proceso de búsqueda [Zitzler99]. En la figura 3.3 se muestran estas dos estrategias.

[3] Al archivo histórico también se le denomina *población secundaria*, *población elitista*, o simplemente *archivo*. A lo largo de este documento usaremos de manera indistinta cualquiera de estos términos.

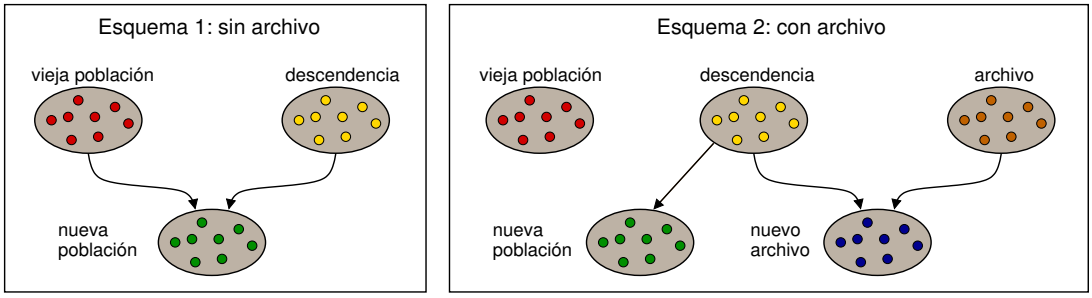


Figura 3.3: Dos esquemas para llevar a cabo el elitismo.

El archivo puede usarse simplemente como un almacén externo o puede ser integrado al AE incluyendo a los miembros del archivo en el proceso de selección. En muchos problemas de optimización, el conjunto de óptimos de Pareto es infinito (o al menos muy grande). Por ello, se debe aplicar una estrategia para decidir la entrada de la nuevas soluciones al archivo y evitar que el archivo crezca indefinidamente. Las estrategias más comunes consideran información relacionada con la densidad de las soluciones [Knowles99, Zitzler99] y con el tiempo que ha transcurrido desde que un individuo entró al archivo [Rudolph00].

3.4.3. Diversidad poblacional

En un AE simple, el mecanismo de selección replica las soluciones con aptitud alta y descarta la soluciones con aptitud baja, lo cual ocasiona que la población converja a una solución única [Mahfoud95]. Este es el comportamiento apropiado cuando se desea encontrar el óptimo global. Sin embargo, en los AEMOs la meta es encontrar, en una sola ejecución, un conjunto de soluciones bien distribuidas a lo largo del frente de Pareto. De Jong [DeJong75] afirma que la clave para resolver este problema es encontrar la manera de preservar la diversidad de la población. Enseguida se describen las técnicas más importantes para mantener diversidad.

Compartir aptitud. La técnica para compartir aptitud (*fitness sharing*) [Goldberg87] tiene la finalidad de formar y mantener subpoblaciones (nichos). Se basa en la idea de que la aptitud es un recurso que se debe compartir entre los individuos que compiten dentro de un mismo nicho. De esta manera, cuanto mayor sea el número de individuos dentro de un nicho, mayor será disminuida su aptitud. Formalmente, la aptitud compartida, f_{s_i}, de un individuo i es igual a su aptitud original dividida por el contador de su nicho:

$$f_{s_i} = \frac{f_i}{\sum_{j=1}^{N} \phi(d_{ij})} \tag{3.14}$$

El contador del nicho es la suma de los valores de la función para compartir obtenidos por los N individuos de la población. La función para compartir normalmente es

$$\phi(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{sh}}\right)^{\alpha} & , \; d_{ij} < \sigma_{share} \\ 0 & , \text{ en otro caso} \end{cases} \tag{3.15}$$

donde suelen adoptarse $\alpha = 1$, y σ_{share}, el radio del nicho, es el parámetro que define la distancia a la cual un individuo puede afectar la aptitud de otro. Dependiendo del espacio donde se calcule la distancia d_{ij}, se distinguen tres maneras de compartir:

1. Compartir la aptitud en el espacio de las cadenas codificadas, donde d_{ij} es una función de distancia apropiada (p. ej., la distancia de Hamming) entre dos cadenas binarias.
2. Compartir la aptitud en el espacio de las variables de decisión, donde d_{ij} puede ser la distancia Euclidiana.
3. Compartir la aptitud en el espacio de los objetivos, donde d_{ij} puede ser la distancia Euclidiana.

La mayoría de los AEMOs actuales utilizan una técnica para compartir aptitud [Hajela92, Fonseca93, Horn93, Srinivas94, Cunha97].

Apareamiento restringido. Deb y Goldberg [Deb89] sugieren emplear el apareamiento restringido con respecto a la distancia fenotípica (es decir, medida en el espacio de los objetivos). La idea es permitir el apareamiento solamente entre individuos similares, es decir, si la distancia es menor que un parámetro llamado σ_{mate}. La técnica tiene la intención de evitar la creación de individuos con baja aptitud (llamados «letales»). Al igual que en las técnicas para compartir aptitud, la distancia de dos individuos se puede definir en tres espacios: el espacio de las cadenas codificadas, el espacio de las variables de decisión, o el espacio de los objetivos.

Reemplazo restringido. En esta técnica propuesta por De Jong [DeJong75] los nuevos individuos reemplazan a los individuos más similares en la población vieja. El reemplazo restringido (*crowding*) se aplica en AEs llamados de estado uniforme (*steady-state*), es decir, donde solamente una fracción de la población se reproduce y muere en cada generación. El procedimiento de la técnica es el siguiente: se toma una porción de la población para reproducirse y generar descendientes. Cada descendiente encuentra al individuo que reemplazará al tomar una muestra de individuos de la población. La descendencia reemplaza a los individuos más similares de la muestra.

Reinicialización. El objetivo de esta técnica es evitar la convergencia prematura al reinicializar la población entera o partes de ella después de cierto período o si la búsqueda se ha estancado. Goldberg sugiere que cuando un AG esté cercano a la convergencia, se reinicie su población utilizando individuos generados aleatoriamente y los mejores individuos de la población que convergió. Una vez que el AE encuentra la solución, el tiempo que resta para llegar a la condición de paro lo consume oscilando alrededor de la solución [Mahfoud95]. La reinicialización permitiría que el AE ocupara el tiempo extra en una actividad productiva.

Hipermallas. Las hipermallas dividen el espacio de las funciones objetivo en regiones llamados hipercubos. Cada una de las soluciones no dominadas ocupa un hipercubo como se muestra en la figura 3.4. La idea es que solamente se acepten las soluciones no dominadas que pertenezcan a hipercubos menos poblados. Aunque el número de divisiones de la malla en cada dimensión es fijo, la localización y la extensión de la malla se puede adaptar con el tiempo (figura 3.4).

Agrupación. Un algoritmo de agrupación (*clustering*) tiene como objetivo particionar un conjunto de puntos en varios grupos (*clusters*), de tal manera que: *a*) cada grupo tenga puntos que sean muy similares entre ellos; y *b*) los puntos de cada grupo sean muy diferentes de los puntos de otro grupo [Freitas02]. En un AEMO la agrupación se emplea para mantener diversidad en el archivo externo y reducir su tamaño. El proceso de agrupación se resume de la siguiente manera (figura 3.5):

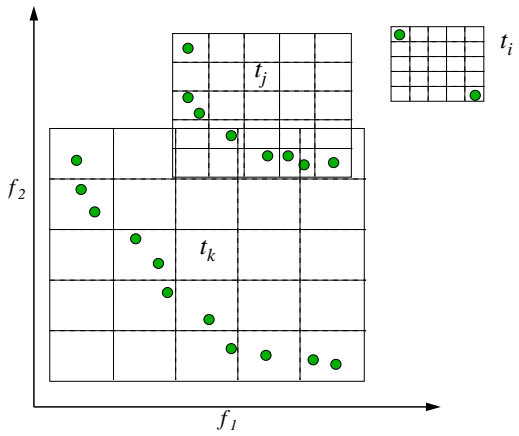


Figura 3.4: Hipermalla para mantener diversidad en el archivo externo. En la figura se muestra la adaptación de la hipermalla durante el proceso de búsqueda.

1. Particionar la población secundaria mediante un algoritmo de agrupación.
2. Seleccionar el individuo representativo de cada grupo, a saber: el centroide.
3. Eliminar el resto de los individuos de cada grupo.

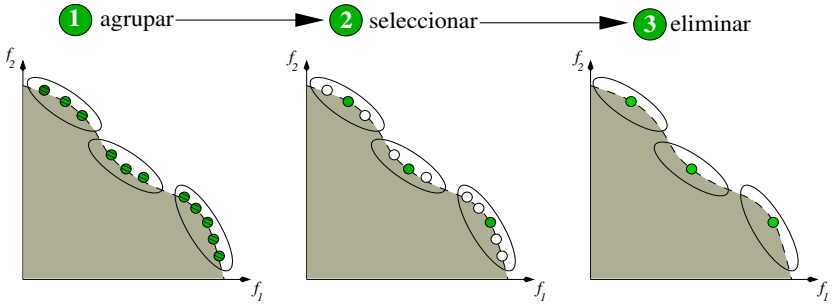


Figura 3.5: Estrategia de agrupación (*clustering*) para reducir y mantener diversidad en el archivo histórico.

3.5. Clasificación de los AEMOs

En [Coello02] se propone clasificar las técnicas evolutivas en tres categorías principales:

- Técnicas *a priori*
- Técnicas progresivas
- Técnicas *a posteriori*

Enseguida se describirán brevemente algunas técnicas específicas de cada categoría.

3.5.1. Técnicas *a priori*

Ordenación lexicográfica En esta técnica el diseñador tiene que ordenar las funciones objetivo de acuerdo con su importancia. Para encontrar la solución óptima se minimizan por separado cada una de las funciones objetivo comenzando por la más importante. Una vez que

se obtiene la solución de una función, ésta se agrega como una restricción para optimizar la solución de los objetivos restantes. La expresión general para optimizar la función $f_i(\vec{x})$ es la siguiente:

$$\text{Minimizar } f_i(\vec{x}) \tag{3.16}$$

sujeta a

$$g_j(\vec{x}) \leq 0 \qquad j = 1, 2, \ldots, m \tag{3.17}$$
$$f_l(\vec{x}) = f_l^* \qquad l = 1, 2, \ldots, i-1 \tag{3.18}$$

Funciones de suma lineal Esta técnica consiste en sumar las funciones objetivo para obtener la función de aptitud global siguiente:

$$\text{aptitud} = \sum_{i=1}^{k} w_i f_i(\vec{x})$$

donde $w_i \geq 0$ son los factores de peso para la función objetivo $f_i(\vec{x})$. Estos pesos se eligen de manera que $\sum_{i=1}^{k} w_i = 1$. En esta técnica los pesos no reflejan la importancia de los objetivos, sino que son factores que se varían para obtener distintos puntos del conjunto óptimo de Pareto. La desventaja principal de esta técnica es que, sin importar los pesos seleccionados, no se pueden obtener las porciones no convexas del frente de Pareto [Das97].

Funciones de suma no lineal Estas técnicas no son muy populares en la literatura. Esto se debe a la dificultad para determinar funciones de utilidad. Dicha dificultad no justifica la calidad de las soluciones obtenidas. Entre los enfoques más populares de esta técnica se encuentran híbridos con: la programación por metas [Deb99a], el alcance de metas [Zebulum98], la teoría de juegos [Sefrioui00], y el algoritmo min-max [Coello98].

3.5.2. Técnicas progresivas

Estas técnicas operan en tres etapas: 1) encontrar soluciones no dominadas, 2) obtener la opinión del tomador de decisiones con respecto a las soluciones, y 3) repetir los pasos anteriores hasta que el tomador de decisiones quede satisfecho con la solución. En el área de la optimización evolutiva multiobjetivo aún no se han reportado técnicas progresivas. No obstante, varios AEMOs [Fonseca93, Cvetkovic02, Rekiek00a] tienen la capacidad de incorporar preferencias del tomador de decisiones dentro del proceso de búsqueda.

3.5.3. Técnicas *a posteriori*

***Multi-Objective Genetic Algorithm* (MOGA).** Fonseca y Fleming [Fonseca93] propusieron el *Multi-Objective Genetic Algorithm* (MOGA) basados en el esquema propuesto por Goldberg [Goldberg89]. 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. La expresión que define esto es la siguiente:

$$\text{jerarquía}(x_i, t) = 1 + p_i^{(t)} \tag{3.19}$$

El procedimiento para asignar la aptitud consta de los tres pasos siguientes:

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 $n \leq M$). La interpolación se hace usualmente con una función lineal.
3. Promediar y compartir la aptitud de los individuos con la misma jerarquía.

Como se discute en [Goldberg91], este tipo de asignación produce una presión de selección alta, que puede producir convergencia prematura. Para contrarrestar este efecto, en MOGA se usa un método de nichos para distribuir la población a lo largo del frente de Pareto. Es importante notar que MOGA tradicionalmente comparte la aptitud en el espacio de las funciones objetivo, en lugar de hacerlo en el espacio de las variables de decisión. Este enfoque para compartir tiene dos inconvenientes: 1) no se mantiene la diversidad en espacio de la funciones objetivo, lo cual es una cuestión importante para el diseñador [Deb99]; y 2) MOGA no es capaz de encontrar soluciones que tengan los mismos valores en una de las funciones objetivo [Deb98].

***Non-dominated Sorting Genetic Algorithm* (NSGA).** Srinivas y Deb [Srinivas94] aplican la idea de Goldberg de manera más directa. Este método utiliza una selección basada en jerarquización de Pareto para favorecer a las soluciones no dominadas, y un método de nichos para mantener la diversidad. 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 «ficticio» (*dummy fitness value*). Para mantener la diversidad en la población, estas soluciones comparten entre ellas su aptitud ficticia. Posteriormente estos individuos son ignorados momentáneamente y se genera el segundo frente no dominado de individuos, a los cuales se les asigna una aptitud ficticia menor que la del primer frente. Este proceso continúa hasta que todos los individuos están clasificados. Puesto que los individuos del primer frente tienen la mayor aptitud, ellos obtienen más copias que el resto de la población.
Deb *et al.* [Deb00] propusieron una nueva versión de este algoritmo, llamado NSGA-II. El nuevo algoritmo es más eficiente en términos computacionales, incorpora elitismo y un operador de *crowding* para mantener la diversidad sin la necesidad de elegir un parámetro adicional. Aunque el NSGA-II es más eficiente que su antecesor, parece tener dificultades para generar soluciones no dominadas que se encuentren en ciertas regiones aisladas del espacio de búsqueda [Coello01].

***Niched Pareto Genetic Algorithm* (NPGA).** Horn y Nafpliotis [Horn93] combinan la selección por torneo y el concepto de dominancia de Pareto. En este método se eligen aleatoriamente dos individuos que serán comparados con un subconjunto de la población total. El tamaño (t_{dom}) de este subconjunto es un parámetro seleccionado por el usuario. Si uno de los individuos es dominado y el otro no, entonces el individuo no dominado es el ganador. Por otro lado, si ambos son dominados o no dominados, el ganador se decide mediante el resultado de una función para compartir aptitud (veáse § 3.4.3). NPGA comparte la aptitud en el espacio de las funciones objetivo. No obstante, los autores sugieren usar una métrica que combina el espacio de las variables y el espacio de los objetivos. A este esquema para compartir aptitud se le denomina «clase de equivalencia» (*equivalence class sharing*).
El desempeño de esta técnica depende fuertemente del valor del parámetro t_{dom}. Si valor del t_{dom} es pequeño, esto puede ocasionar que se encuentren pocas soluciones no do-

minadas. Por el contrario, un valor de t_{dom} grande puede conducir a una convergencia prematura.

***Strength Pareto Evolutionary Algorithm* (SPEA).** El *Strength Pareto Evolutionary Algorithm* fue concebido por Zitzler y Thiele [Zitzler99] como una manera de combinar las técnicas más exitosas de diferentes AEMOs. SPEA 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 (*strength*). Este valor es proporcional al número de soluciones que domina un cierto individuo del conjunto externo. La aptitud de los miembros de la población actual se calcula de acuerdo a la fortaleza de los individuos de la población externa que los dominan. En este esquema el objetivo es minimizar la aptitud (es decir, valores pequeños de aptitud corresponden a altas probabilidades de reproducción). El esquema para asignar la aptitud pretende conseguir que la búsqueda se dirija hacia el frente de Pareto y a la vez preservar la diversidad de las soluciones no dominadas.

Capítulo 4

Algoritmos Evolutivos Multi-Objetivo paralelos (pAEMO)

Sumario

4.1. Introducción

PESE al éxito con el cual se han aplicado los algoritmos evolutivos multiobjetivo (AEMOs) en varios problemas del mundo real [Coello02], consiguiendo efectividad hasta cierto grado, la eficiencia sigue siendo una meta por cumplir. Por lo tanto, para reducir los tiempos de ejecución y utilizar eficientemente los recursos de cómputo es natural considerar la paralelización de los AEMOs.

Durante más de tres décadas se ha hecho un extenso estudio acerca de los algoritmos evolutivos paralelos utilizados para la optimización con una sola función objetivo (pAEs) [Alba99][Cantu00][Sawai99][Tomassini99]. Con todo, los resultados obtenidos por los AEs para un solo objetivo carecen de efectividad para resolver la mayoría de los problemas de optimización multiobjetivo (POMs) [Veldhuizen03]. Por ejemplo, los pAEs que plantean la utilización de funciones de agregación solamente encuentran una solución por corrida, por lo que se necesitan varias ejecuciones del algoritmo para obtener un conjunto de varias soluciones [Coello99]. Además, algunos pAEs tradicionales no pueden identificar el frente de Pareto completo [Veldhuizen03].

En un sentido general, un pAEMO puede ser útil cuando nos enfrentamos a problemas en los cuales las funciones de aptitud son costosas desde el punto de vista computacional. En este caso, es posible utilizar técnicas para descomponer paralelamente estas funciones. Otra posibilidad

es distribuir la población entre varios procesadores. Dicho de manera breve, el objetivo de un pAEMO es encontrar soluciones de superior o igual calidad, en menor tiempo que su contraparte serial, a la vez que explora un territorio más amplio del espacio de soluciones.

Cuando estamos resolviendo un problema de optimización, la paralización de los algoritmos involucrados puede producir una implementación más eficiente (en cuanto a tiempo de ejecución), y, lo que es más interesante, se puede lograr una implementación más efectiva, es decir, podemos encontrar mejores soluciones. Esto se debe, principalmente, al incremento en el número de procesadores (y memorias locales) utilizadas en el problema, lo cual permite explorar un espacio de soluciones mayor en menor tiempo.

4.2. Conceptos de computación paralela y distribuida

Los sistemas paralelos y distribuidos son un elemento clave para satisfacer las crecientes demandas de alto rendimiento de las aplicaciones científicas y de propósito general. Ejemplos de estas aplicaciones incluyen: modelado de movimientos oceánicos, minería de datos, diseño de proteínas, diseño de farmacéuticos, modelado de la biósfera, y modelado biomolecular. En general, el espacio de búsqueda de los problemas de optimización crece exponencialmente con el número de variables, por lo que encontrar la solución a estos problemas está fuera del alcance de la computación secuencial convencional [Kumar94].

El procesamiento paralelo lo podemos definir como el procesamiento de información acentuando la manipulación concurrente de elementos de información que pertenecen a uno o más procesos que resuelven un solo problema [Quinn94]. Las computadoras capaces de realizar esta clase de procesamiento, llamadas paralelas, requieren por tanto de varios procesadores.

Actualmente existen varios esquemas para clasificar las computadoras paralelas [Flynn72, Bell92, Tanenbaum02], pero ninguno se ha constituido en un estándar. La diferencia entre estos esquemas radica en las características del sistema de cómputo que son tomadas en cuenta por cada esquema, a saber: la organización del espacio de direcciones, la red de interconexión, la granularidad de los procesadores, etc. Las computadoras paralelas se encuentran dentro de las categorías SIMD (*Single instruction stream, multiple data stream*) y MIMD (*Multiple instruction stream, multiple data stream*). Estas categorías forman parte de la conocida taxonomía propuesta por Flynn [Flynn72]. Actualmente, la tendencia se inclina del lado de la computadoras MIMD, mientras que la computadoras SIMD pierden popularidad [Hwang93]. Para los propósitos de este trabajo de tesis, partiremos de la categoría MIMD. Las computadoras MIMD tienen varios procesadores que trabajan juntos a través de una red de interconexión. Cada procesador es capaz de ejecutar un programa distinto independientemente de los demás (figura 4.1).

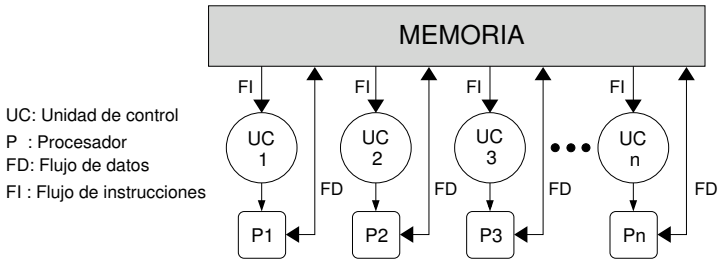


Figura 4.1: Arquitectura típica de una computadora MIMD.

Atendiendo a la organización del espacio de direcciones, las computadoras MIMD se subdividen en dos categorías: sistemas de memoria distribuida, llamados comúnmente *multiprocesa-*

dores; y sistemas de memoria compartida, usualmente llamados *multicomputadoras*.

Sistemas multiprocesador. En esta clase de sistemas todos los procesadores leen y escriben en un solo espacio de direcciones físico a través de una red de interconexión (figura 4.2). La manera en que se comunican los procesadores es escribiendo datos en la memoria global para que los demás procesadores los puedan leer. El problema de este esquema de comunicación es que compromete la integridad de los datos y degrada la eficiencia. Para evitar conflictos de escritura el programador debe usar los mecanismos clásicos de sincronización (p. ej., semáforos, candados, barreras, etc.). Para mejorar la eficiencia, a cada procesador se le agregan memorias caché para acelerar el tiempo de acceso a la información usada con más frecuencia. Sin embargo, la incorporación de estas memorias requiere protocolos para asegurar la coherencia entre la memoria global y la memoria caché de cada procesador.

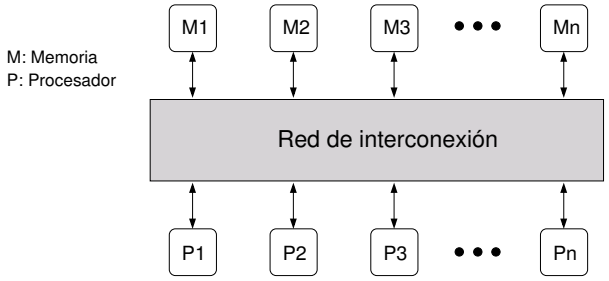


Figura 4.2: Esquema general de un sistema multiprocesador.

Sistemas multicomputadora. En este sistema cada procesador tiene su propia memoria, la cual sólo puede ser accedida por ese procesador. En este caso, se utiliza una red de interconexión para que los procesadores se comuniquen entre sí por medio de un mecanismo de envío de mensajes (*message passing*)(figura 4.3).

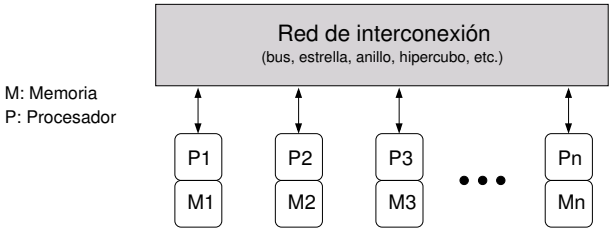


Figura 4.3: Esquema general de un sistema multicomputadora.

A esta categoría pertenecen las computadoras nCUBE 2, Paragon XP/S, Cosmic Cube, y CM-5 (sistemas fuertemente acoplados). Asimismo pertenecen los cúmulos de computadoras (*clusters*) y las redes LAN (sistemas débilmente acoplados). Los cúmulos son un conjunto de computadoras conectadas a través de una red de comunicación de alta velocidad. Estos sistemas usan tanto computadoras (estaciones de trabajo o computadoras personales) como redes de interconexión (Fast Ethernet, Myrinet) de propósito general. Otros sistemas que pertenecen a esta categoría son los sistemas de procesamiento paralelo masivo (MPP, *Massively parallel processing*). Estas computadoras están constituidas por cientos o miles

de procesadores[1]. Generalmente sus procesadores son iguales a aquellos que utilizan las computadoras personales o las estaciones de trabajo. La diferencia con otras multicomputadoras estriba en que utilizan redes de interconexión *ad hoc* de alto rendimiento fabricadas por los propietarios.

4.3. Paradigmas de paralelización de AEMOs

Actualmente, existe un amplio estudio relacionado con la paralelización de algoritmos evolutivos para un solo objetivo [Alba99][Cantu00][Sawai99][Tomassini99]. Sin embargo, existen pocos artículos sobre algoritmos evolutivos paralelos para resolver problemas multiobjetivo.

Como se menciona en [Coello02], la mayoría de los estudios para paralelizar un AEMO se concentran en tres paradigmas[2]: el modelo «amo-esclavo», el modelo de «isla», y el modelo de «difusión».

Las tres secciones siguientes están dedicadas a una breve descripción de estos tres modelos conceptuales. El lector interesado puede consultar [Cantu00] para una discusión más amplia de estos paradigmas, aunque desde un punto de vista de la optimización monoobjetivo.

4.3.1. Modelo «amo-esclavo»

El modelo «amo-esclavo» es uno de los más simples de implementar. Un proceso amo ejecuta el AEMO, y la evaluación de las funciones objetivo se distribuye entre varios procesos esclavo. El espacio de búsqueda que se explora con un pAEMO «amo-esclavo» es conceptualmente el mismo que exploraría un AEMO serial. En consecuencia, el número de procesadores empleados es independiente de las soluciones evaluadas; sin embargo, el tiempo de ejecución sí se reduce (idealmente P veces con P procesadores). En otras palabras: este paradigma obtiene las mismas soluciones que aquellas obtenidas con un AEMO serial. Este modelo se ilustra en la figura 4.4, donde el proceso amo distribuye los individuos de la población, controla dónde y cuándo se evalúan las funciones objetivo, y almacena los valores de las funciones objetivo que regresan los esclavos. El tiempo de ejecución de un pAEMO «amo-esclavo» se distribuye en dos elementos: el tiempo requerido para evaluar las funciones objetivo y el tiempo usado para la comunicación entre procesadores. El tiempo utilizado para aplicar los operadores evolutivos es ignorado.

Es importante notar que el cómputo de las funciones objetivo necesita ser suficientemente complejo y consumir un tiempo considerable a fin de obtener una aceleración significativa [Veldhuizen03].

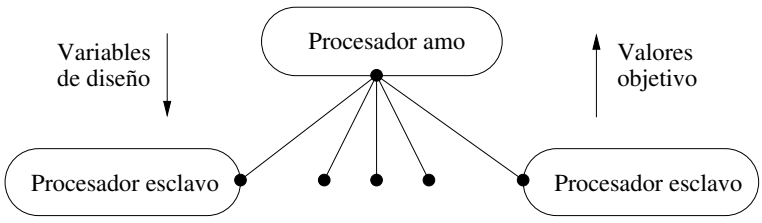


Figura 4.4: Vista esquemática del modelo amo-esclavo.

[1] Desde luego esta definición depende del tiempo, ya que a la par del avance tecnológico aumenta la demanda de alto rendimiento.

[2] Los términos paradigma y modelo se usan indistintamente en este documento.

4.3.2. Modelo de isla

Este modelo está inspirado en el fenómeno natural en el cual un conjunto de poblaciones separadas geográficamente evolucionan con un aislamiento relativo, tal como sucede en una cadena de islas oceánicas. La población global del pAEMO se divide en varias subpoblaciones o *demes*, separadas e independientes. Aunque cada isla evoluciona de manera aislada la mayor parte de la ejecución del pAEMO, ocasionalmente se migran individuos entre las subpoblaciones. Estos pAEMOs también son conocidos como «distribuidos», ya que usualmente se implementan en máquinas MIMD con memoria distribuida; también son llamados multi-población o multi-*deme*. Por último, puesto que la razón entre cómputo y comunicación es generalmente alta, también se les llama pAEMOs de «grano grueso».

Una representación típica del modelo se ilustra en la figura 4.5, en la cual se usa una topología de anillo. Se pueden observar canales de comunicación lógicos como parte de la estrategia de migración. Otras topologías de interconexión lógicas o físicas son posibles: anillos, mallas (2D y 3D), toros, triángulos, e hipercubos. En cada uno de los cuatro nodos de la figura 4.5 se ejecuta un AEMO serial.

El modelo de isla es muy popular; sin embargo, es el modelo que requiere de más parámetros y decisiones de diseño. Los principales aspectos a considerar incluyen: 1) determinar el tamaño y número de subpoblaciones, 2) la topología de interconexión de las subpoblaciones, 3) la frecuencia de migración, 4) el número de individuos a migrar, 5) la política de migración que determina cuáles individuos migrar y cuáles individuos reemplazar por los inmigrantes.

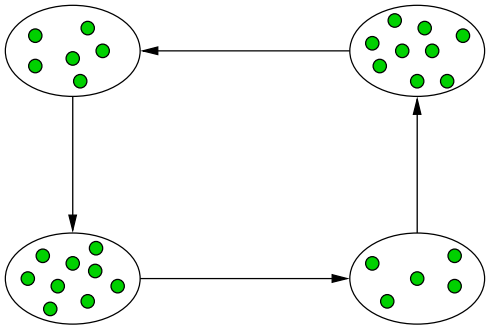


Figura 4.5: Vista esquemática del modelo de isla.

4.3.3. Modelo de difusión

Al igual que el modelo «amo-esclavo», en el paradigma de difusión se considera una sola población, pero en este caso distribuida espacialmente en una estructura de vecindario. Usualmente, la estructura es una malla rectangular bidimensional, en la que se coloca una población pequeña en cada vértice (figura 4.6). De hecho, idealmente se debería tener un individuo por cada procesador; de aquí, que a este modelo se le llame a veces de «grano fino» [Cantu00]. A los pAEMOs de esta clase también se les denomina «celulares» porque son una clase de autómata celular con reglas de transición estocásticas.

La selección y la cruza se restringe a un pequeño vecindario alrededor de cada individuo. Los vecindarios se traslapan (como se observa en la figura 4.6), o pueden ser dinámicos, de manera que las buenas soluciones se propaguen o se «difundan» lentamente a través de la población entera. No hay migración *per se*, y los costos de comunicación tienden a ser altos, ya que los individuos considerados para la selección están distribuidos en varios procesadores. Por esta

razón, este paradigma es más apropiado para arquitecturas MIMD de memoria compartida, o para máquinas SIMD de procesamiento paralelo masivo.

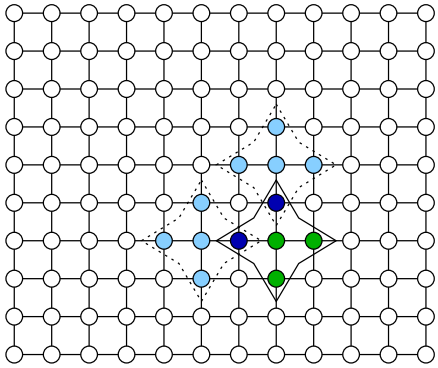


Figura 4.6: Vista esquemática del modelo de difusión.

4.4. Aspectos clave de un pAEMO

4.4.1. Distribución del espacio de búsqueda

Para utilizar eficientemente todos los procesadores es conveniente que cada procesador se dedique a resolver regiones disjuntas del espacio fenotípico o genotípico. En general es muy difícil realizar una asignación *a priori* de las regiones del espacio para un POM general [Veldhuizen03].

Existen dos enfoques generales para llevar a cabo tal asignación:

- Restringir cada procesador a una región particular, forzando al proceso a generar individuos hasta encontrar el número adecuado de individuos dentro de su región.
- Restringir cada procesador a una región particular, migrando individuos al proceso que tenga la región adecuada.

Algunas de las propuestas existentes son las siguientes:

- Sesgar la búsqueda de cada proceso hacia una región específica del frente de Pareto, aunque cada proceso busque en todo el espacio de búsqueda [Deb03] (véase § 4.6.3).
- Ordenar la población con respecto a una de las funciones objetivo y dividir la población en partes iguales, de manera que cada porción se asigne a un proceso distinto [Hiroyasu00] (véase § 4.6.1).

4.4.2. Estrategia para mantener la población secundaria

La población secundaria es un archivo externo que mantiene los «mejores» individuos encontrados por el pAEMO (véase § 3.4.2). Esta área tiene el potencial de incrementar la eficiencia de un pAEMO si se aborda inteligentemente.

Algunas propuestas de archivado posibles son las siguientes [Veldhuizen03]:

- Tener un proceso que mantenga su propio archivo y al final del proceso de búsqueda un procesador combina todos los archivos y presenta la solución final al usuario.
- Realizar migración de individuos de las poblaciones secundarias.

La elección de una de estas estrategias depende de la utilización del archivo externo. Para los AEMOs que utilicen activamente el archivo externo durante el proceso de búsqueda, la segunda opción puede beneficiar la eficacia del algoritmo.

4.4.3. Técnica para mantener diversidad

Como se observó en la sección 3.4.3, la técnicas para mantener diversidad son un elemento clave en la eficacia de un (p)AEMO. Sin embargo la implementación paralela de las técnicas para mantener diversidad (técnicas de nichos) es un problema abierto.

La técnicas de nichos son básicamente un proceso secuencial, así que paralelizarlos eficientemente es una tarea difícil. Solamente los pAEMOs «amo-esclavo» podrían realizar la técnica de nichos en la forma usual. Los demás paradigmas de paralelización ofrecen una gama más amplia de posibilidades para paralelizar el mecanismo de nichos. Por ejemplo, un pAEMO de isla podría ejecutar un mecanismo de nichos distinto en cada isla. Las técnicas de nichos originalmente no requieren comunicación adicional para llevarse a cabo. Sin embargo, se podría considerar intercambiar individuos de los frentes locales para ejecutar el mecanismo de nichos de manera global.

Independientemente de la estrategia elegida para paralelizar el mecanismo de nichos, debemos considerar el siguiente compromiso: una representación precisa del frente de Pareto y del conjunto de óptimos de Pareto, contra el tiempo requerido para ejecutar el algoritmo. Si bien la paralelización del mecanismo de nichos redunda en una mejor eficacia del pAEMO, también puede acarrear un costo computacional alto debido a las comunicaciones. También es importante hacer notar que las características del sistema paralelo (p. ej., memoria distribuida o compartida) con que contamos juegan un papel importante en la elección de la estrategia de paralelización de la técnica de nichos.

4.5. Aceleración de los pAEMO

La aceleración (*speedup*) de los AEs paralelos (pAEs) es un tema controversial puesto que, contrariamente a lo que ocurre con los algoritmos paralelos convencionales, en la comunidad de los pAEs se han reportado aceleraciones superlineales [Lin94] [Belding95] [Alba01]. El argumento que propicia tal controversia es el siguiente: «si todas las tareas de un programa paralelo son ejecutadas por varios hilos (*threads*) [o procesos] en un solo procesador, el tiempo de ejecución total no puede ser menor que el tiempo de ejecución de un programa serial que realice los mismos cálculos» [Cantu00]. No obstante, autores como [Alba02][Cantu00] afirman que, aún tomando la consideraciones debidas, es posible que los pAEs proporcionen aceleraciones superlineales. En esta sección se presentará inicialmente la definición de aceleración en el contexto de los pAEMOs. Posteriormente, se presentará una clasificación de aceleración con el fin de establecer medidas justas para calcular la aceleración. Por último, se describirán las fuentes que producen la aceleración superlineal.

La aceleración es una medida que representa el beneficio relativo al resolver un problema en paralelo. La definición tradicional de aceleración de un algoritmo es la razón del tiempo de ejecución del mejor algoritmo secuencial y el tiempo de ejecución del algoritmo paralelo sobre p procesadores:

$$S_p = \frac{T_1}{T_p} \tag{4.1}$$

Sin embargo, debido al carácter estocástico de los AEMO, una sola ejecución no es estadísticamente significativa. Por tal motivo, Alba [Alba02] recomienda que la aceleración debe relacionar los tiempos promedio de ejecución, es decir:

$$S_p = \frac{\overline{T}_1}{\overline{T}_p} \tag{4.2}$$

A partir de esta definición se distinguen tres tipos de aceleración:

- Aceleración sublineal, $S_p < p$.
- Aceleración lineal, $S_p = p$.
- Aceleración superlineal, $S_p > p$.

4.5.1. Taxonomía de la aceleración

Con el fin de hacer una comparación justa entre cualquier algoritmo secuencial y un pAEMO, Alba [Alba02] propone una serie de medidas para calcular la aceleración en el contexto de los pAE(MO)s. Estas recomendaciones dan pie a la clasificación de la aceleración mostrada en la tabla 4.1.

La aceleración tipo I se refiere a la comparación de los tiempos de ejecución del pAEMO y del mejor algoritmo secuencial existente (evolutivo o no) que resuelva el problema que se trata. Esta aceleración es poco realista por cuanto presupone que se conoce el mejor algoritmo secuencial. La dificultad para evaluar este tipo de aceleración se agrava si tomamos en cuenta que en muchas ocasiones el AEMO es la única opción para resolver el problema. En este sentido, la aceleración tipo II es más práctica, ya que compara el pAEMO con su propia versión secuencial. Algunos experimentos aparentan ser justos cuando comparan los algoritmos secuencial y paralelo fijando a priori un criterio de paro (p. ej. el número de evaluaciones) o manteniendo el tamaño de la población. Con todo, tanto [Cantu00] como [Alba02] coinciden en que este tipo de aceleración (tipo II.B) es inadecuado porque no tiene sentido comparar algoritmos que devuelven soluciones de calidades distintas. De este modo, el criterio más justo es detener a los algoritmos comparados cuando alcancen una solución de la misma calidad (aceleración tipo II.A). Por otro lado, debemos tener presente que los AEs secuenciales, con una población única sin estructura (AEs panmíticos), son distintos de los pAEMOs, que separan la población en partes. La aceleración tipo II.A.1 compara el pAEMO ejecutado en p procesadores contra un AE secuencial panmítico. Está aceleración es inadecuada para estudiar la aceleración ya que compara dos algoritmos distintos. Para medir la aceleración es más adecuado comparar el mismo pAEMO ejecutado en 1 y p procesadores (tipo II.A.2). Esta aceleración supone que el código ejecutado en 1 y p procesadores es el mismo.

Retomemos la aceleración tipo II.A para definir cómo evaluar la calidad de dos soluciones con el fin de aplicar este tipo de aceleración en el contexto multiobjetivo. En la optimización monoobjetivo es claro determinar si dos soluciones tienen la misma calidad (es decir, la misma aptitud). Sin embargo, en la optimización multiobjetivo tratamos con un conjunto de soluciones alternativas (conjunto de óptimos de Pareto), por lo que es difícil, incluso establecer un criterio universal para determinar la calidad del conjunto de óptimos de Pareto obtenido. En el contexto de la optimización multiobjetivo se considera que la calidad de una solución viene dada mediante lo siguiente [Zitzler99b]: 1) la distancia del conjunto no dominado generado al frente de Pareto debe ser mínima, 2) una buena distribución de las soluciones, 3) la amplitud del frente no dominado debe ser máxima. Para comparar cuantitativamente estás características entre

I. Aceleración fuerte.

II. Aceleración débil.

- **A.** Aceleración con parada según la solución.
  - **1.** Frente a panmixia.
  - **2.** Ortodoxa.
- **B.** Aceleración con esfuerzo predefinido.

Tabla 4.1: Clasificación de la aceleración

dos soluciones se han propuesto varias métricas (véase por ejemplo [Coello02]). Por lo tanto, podríamos usar un conjunto de tales métricas para decidir si dos soluciones tienen la misma calidad. Por ejemplo, podríamos utilizar la *distancia generacional* [Veldhuizen99] para evaluar 1), la métrica de *espaciamiento* (*spacing*) [Schott95] para evaluar 2) y la métrica de *amplitud máxima* ($\mathcal{M}_3^*$) [Zitzler99] para evaluar 3). Otra alternativa, más práctica, es utilizar la métrica de *hipervolumen* [Zitzler99a] que calcula el tamaño de la región dominada por el $FP_{conocido}$ para evaluar su calidad global.

4.5.2. Fuentes que producen aceleración superlineal

Aún considerando las medidas de la sección anterior, se han reportado aceleraciones superlineales. Afortunadamente, algunos autores [Alba02] [Shonkwiler93] han ofrecido razones teóricas que demuestran que la aceleración superlineal en los pAEs es posible.

Las fuentes que originan la aceleración superlineal se pueden separar en aquellas relacionadas con el carácter estocástico de los AEs y aquellas relacionadas con las características del *hardware*. Enseguida se describen cada una de estas fuentes.

Carácter estocástico. Teóricamente, la aceleración nunca puede exceder el número de procesadores, p. No obstante, la premisa clave de esta afirmación es que el AE paralelo y el secuencial ejecutan exactamente las mismas tareas. De acuerdo con Cantú [Cantu00], la única explicación posible de la aceleración superlineal es que el pAE de alguna manera ejecute menos trabajo que el AE secuencial. Esta reducción de trabajo se debe a que la política de migración afecta la presión de selección. Como se muestra teóricamente en [Cantu00], el incremento en la presión de selección acelera la convergencia, lo cual puede producir la aceleración superlineal. Otra de las explicaciones de la aceleración superlineal es que un AE secuencial tiene que buscar una gran porción del espacio de búsqueda antes de encontrar la solución. En cambio, la versión paralela puede encontrar la solución más rápido debido al orden en el cual se explora el espacio de búsqueda [Alba02].

Características del *hardware*. Un pAE puede producir una aceleración superlineal si aprovecha la memoria adicional de una computadora paralela. La población de un AE secuencial puede ser lo suficientemente grande para no caber en la memoria principal de un solo procesador, y por lo tanto, degradaría su desempeño al utilizar memoria secundaria. En cambio, las subpoblaciones de un pAE pueden caber completamente en la memoria principal, o incluso en la memoria caché del procesador.

4.6. Actualidad de los pAEMOs

A continuación se revisarán algunos de los pAEMOs más representativos que se han propuesto en la literatura especializada en los últimos años.

4.6.1. Divide Range Multi-Objetive Genetic Algorithm (DRMOGA)

El DRMOGA (*Divide Range Multi-Objetive Genetic Algorithm*) es un pAEMO de isla propuesto por Hiroyasu et al. [Hiroyasu00] para resolver problemas de optimización numérica. La aportación principal de este algoritmo es el esquema propuesto para dividir la población global en subpoblaciones. Aquí la población global es ordenada con respecto a la función objetivo f_i. Posteriormente, la población se divide en m subpoblaciones con N/m individuos cada una (donde N es la población total) (figura 4.7).

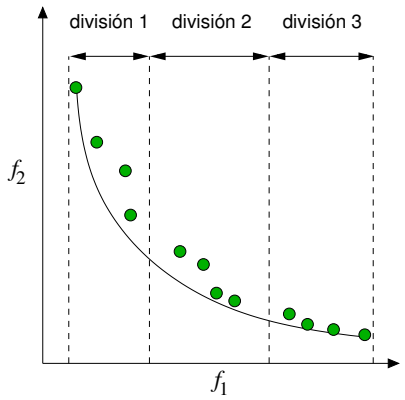


Figura 4.7: División de la población en DRMOGA.

En cada subpoblación se ejecuta un AEMO serial y al término de k generaciones se reúnen todos los individuos y se vuelve a dividir la población tomando como criterio de ordenación otra función objetivo. En la ejecución del AEMO se emplean tres tipos de selección:

- Elegir todos los individuos con jerarquía 1. Cuando la población excede un cierto tamaño, los individuos se eligen mediante selección por ruleta.
- Selección por ruleta, en la cual los valores de aptitud para construir la ruleta son determinados únicamente por la jerarquía.
- Selección por ruleta, en la cual los valores de aptitud son determinados con la jerarquía y una función para compartir aptitud.

El algoritmo se ejecutó en un cúmulo de computadoras personales con 5 procesadores PII a 500 MHz, con Linux como sistema operativo, una red Fast Ethernet, y MPICH como biblioteca de comunicaciones. Los autores reportan resultados favorables al comparar DRMOGA contra algoritmos basados en el «modelo simple de isla» y en modelos de una población. De acuerdo a los autores, DRMOGA tiene una notable habilidad para encontrar soluciones óptimas de Pareto en los problemas difíciles.

4.6.2. Metapopulation evolutionary algorithm (MEA)

Kirley [Kirley01] propone un pAEMO de isla que utiliza un esquema de metapoblación (MEA), es decir, una población formada de poblaciones, el cual se inspira en la dinámica poblacional descrita en modelos ecológicos. El propósito de MEA es combinar las características de los modelos de «grano fino» y de los modelos de «grano grueso». En este planteamiento la población se distribuye sobre un retículo (*lattice*) bidimensional, pero dejando algunos sitios vacíos. La característica principal de este planteamiento es la incorporación de un esquema de «extinción/colonización». Un evento de extinción elimina de manera aleatoria individuos que se encuentren en alguna posición del retículo. Los sitios que quedan disponibles por la extinción pueden ser «colonizados» (ocupados) por un individuo tomado aleatoriamente de la vecindad del sitio disponible. Debido a la acción de estos eventos se comienzan a formar aglomeraciones de sitios contiguos ocupados, los cuales conforman subpoblaciones. Debido a este comportamiento se producen subpoblaciones de tamaño variable. El autor señala que este esquema de «extinción/colonización» es un control de migración autoadaptable. Asimismo, mantiene la diversidad de la población en el retículo.

La selección se lleva a cabo por medio de una forma de torneo, y está restringida a un vecindario local. Por otra parte, la asignación de la aptitud se realiza mediante una técnica de agregación de funciones llamada «método de agregación de gradiente ambiental». En este esquema la ubicación geográfica (posición a lo largo del eje x) de un individuo es utilizada para determinar el peso de cada objetivo. En la figura 4.8 se ilustra un ejemplo con dos objetivos.

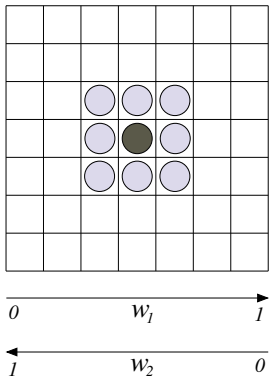


Figura 4.8: La situación geográfica de un individuo (coordenada en x) determina el peso de cada objetivo.

En los resultados obtenidos, el autor menciona que la selección por medio del «método de agregación de gradiente ambiental» tiene mejor desempeño que el método de jerarquización de Pareto. La razón de esta situación, de acuerdo al autor, es que la pequeña población en el vecindario no provee suficientes individuos para que la jerarquización de Pareto sea efectiva, y en cambio, la jerarquización con el «gradiente ambiental» considera la importancia relativa de cada objetivo.

4.6.3. Distributed Computing of Pareto-Optimal Solutions

Deb et al. [Deb03] proponen un pAEMO de isla basado en el NSGA-II [Deb00] (*Nondominated Sorting Genetic Algorithm* II). Este algoritmo se distingue por el esquema para asignar a cada procesador una región del frente de Pareto. Para concentrar la búsqueda en una región determinada del frente proponen el concepto de «dominación guiada» que usa la siguiente transformación

que pondera las funciones objetivo:

$$\Omega_i(f(\mathbf{x})) = f_i(\mathbf{x}) + \sum_{j=1, j\neq i}^{M} a_{ij} f_j(\mathbf{x}), \quad i = 1, 2, ..., M.$$

El nuevo concepto de denominación es el siguiente:

> Una solución $\mathbf{x}^{(1)}$ domina a otra solución $\mathbf{x}^{(2)}$, si $\Omega_i(\mathbf{f}(\mathbf{x}^{(1)})) \leq \Omega_i(\mathbf{f}(\mathbf{x}^{(2)})) \quad \forall\, i = 1, 2, \ldots, M$, y $\exists\, i : \Omega_i(\mathbf{f}(\mathbf{x}^{(1)})) < \Omega_i(\mathbf{f}(\mathbf{x}^{(2)}))$.

Este nuevo concepto de dominación produce una región dominada más amplia como se muestra en la figura 4.9(b), lo cual conduce a que solamente se descubra una región del frente de Pareto 4.9(c). Con esta definición se permite que los dominios del frente de Pareto asignados a cada procesador se traslapen. Es interesante notar que, ya que se domina una región más amplia, el frente de Pareto real (según la definición original de dominancia) puede no ser totalmente no dominado de acuerdo con la nueva definición de dominancia. Por ello, cada procesador encuentra solamente una región del frente de Pareto real.

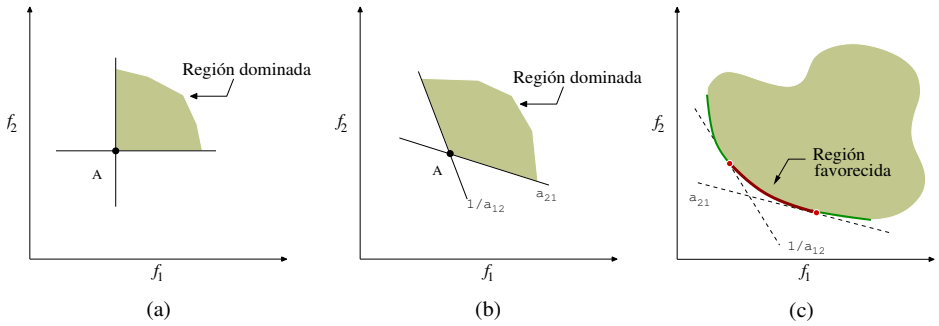


Figura 4.9: Región dominada con la definición usual (a); región dominada con el concepto de dominación guiada (b); región no dominada del frente de Pareto.

Es importante notar que a pesar de que cada procesador se concentra en una región del frente de Pareto, cualquiera de ellos busca en todo el espacio de soluciones. Además, para asignar los pesos adecuados a la transformación se necesitan conocer la forma del frente de Pareto real y algunos vectores del frente de Pareto conocido.

4.6.4. Parallel diffusion GA for Multiobjective Functions

Un pAEMO de «difusión» es propuesto por Jon Rowe et al. [Rowe96] para resolver un problema de análisis de sensibilidad. En este enfoque se genera una población aleatoria que se distribuye espacialmente en cada uno de los vértices de un retículo rectangular. Para generar un nuevo individuo cada miembro elige aleatoriamente un vecino contiguo para producir un descendiente. El descendiente reemplaza al padre si resulta más apto. El pseudocódigo para el proceso de recombinación y selección es el siguiente:

1. Generar una población aleatoria $x = (x_1, \ldots, x_M)$
2. Para cada individuo x_i:
   a) Elegir un vecino aleatorio x_j de x_i
   b) Cruzar x_j y x_i para producir un descendiente y_i

3. Para cada individuo x_i:

   a) Si y_i es mejor que x_i entonces $x_i = y_i$

   b) Aplicar mutación a x_i

4. Ir al paso 2.

Para decidir si un individuo es mejor que otro se utiliza dominancia de Pareto. La característica importante de este planteamiento es que el pAEMO no calcula la jerarquización de Pareto para toda la población sino que hace una comparación local entre dos individuos de la población.

Los autores mencionan que debido a que la información se intercambia únicamente de manera local, la diversidad de los alelos se mantiene en un nivel mayor que en los AEMOs ordinarios, lo cual a su vez provoca la creación de nichos.

4.6.5. Virtual Subpopulation Genetic Algorithm (VSGA)

VSGA se trata de un pAEMO de «isla» propuesto por Quagliarella y Vicini [Quagliarella98]. En el VSGA la población se distribuye en una malla toroidal. Las subpoblaciones se crean al definir fronteras lógicas. La selección se lleva a cabo mediante una «caminata aleatoria» (*random walk*). Desde un punto de la malla se recorren dos caminos aleatorios de cierta longitud. Los individuos no dominados de cada camino se eligen como padres, y el individuo encontrado en el punto de inicio de los caminos es reemplazado por el descendiente de los padres. Para la estrategia de migración se recorre un camino aleatorio que cruce las fronteras de las islas de acuerdo a una función de probabilidad. Los autores emplearon la función de $flip$, donde el valor de 1 significa que se puede cruzar la frontera, en tanto que el valor de 0 significa que la frontera no se puede cruzar.

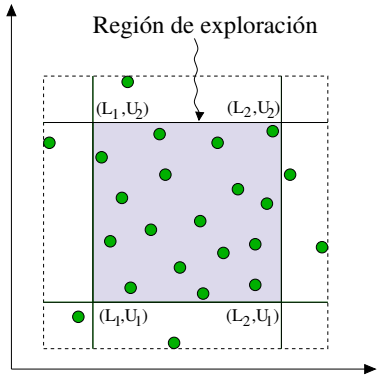
El frente de Pareto de la generación actual se obtiene extrayendo los individuos no dominados con respecto a toda la población.

El pAEMO se ejecutó en una SGI Power Challenge con 16 procesadores R-10000 y se implementó con procesos de nativos de UNIX. Uno de los nodos se designa como el proceso amo, y la población es dividida y asignada a varios procesos esclavos (las islas). El proceso amo envía a los esclavos la información necesaria para que realicen la evaluación de las funciones objetivo, y espera a que todas las evaluaciones terminen. Los autores advierten que esta implementación es eficiente sólo si los esclavos tienen igual carga de trabajo. De otra manera, la eficiencia es pobre ya que el proceso amo tiene que esperar a que termine el proceso más lento. Finalmente, los autores hacen notar que el beneficio del modelo de subpoblaciones (el modelo de «isla») depende fuertemente del problema en cuestión. En sus casos de prueba, por ejemplo, observaron que para un problema monoobjetivo el modelo de subpoblaciones mejoró el desempeño general del algoritmo, mientras que para un problema multiobjetivo no se observaron ventajas con respecto al modelo que emplea una población global.

4.6.6. Asynchronous Self-Adjustable Island Genetic Algorithm (aSAIGA)

Zhu y Leung [Zhu02] proponen un algoritmo genético de isla (aSAIGA) que se caracteriza por usar un esquema asíncrono de migración y un esquema de autoajuste para distribuir de manera precisa el esfuerzo de búsqueda entre las islas. En cada una de las islas de aSAIGA se ejecuta una versión adaptada del *Strength Pareto Evolutionary Algorithm* (SPEA) [Zitzler99]. A SPEA se le agregaron las dos operaciones siguientes: la *operación de comunicación asíncrona* (CA) y la *operación de autoajuste* (AA).

La operación CA se encarga de intercambiar la información de manera asíncrona[3] entre las islas. En aSAIGA en lugar de migrar un grupo de individuos, como ocurre normalmente en los AEMOs de isla, la operación CA intercambia información asociada con la región que explora cada isla. La *región de exploración* (RE) es un hipercubo que contiene a la mayoría de los individuos del archivo externo que mantiene SPEA (figura 4.10). El hipercubo tiene la finalidad de estimar la región de búsqueda actual de cada isla. La operación CA difunde a las demás islas un mensaje que consiste de la RE actual, el centro de la RE y un número que representa el orden de la isla en la secuencia de todas las islas. La CA también tiene la función de recibir y almacenar los mensajes de las otras islas, los cuales serán usados por la operación AA.



La región de exploración (RE) se define mediante el hipercubo $[L_1, U_1] \times \ldots \times [L_N, U_N]$. Sea $\{f_i^1, f_i^2, \ldots, f_i^{n'}\}$ la permutación con los valores de la i-ésima ($i = 1, \ldots, N$) función objetivo ordenados crecientemente, donde n' es el tamaño del archivo histórico y N el número de objetivos. Entonces: $L_i = f_i^{\lceil \frac{1}{4} n \rceil}$, $U_i = f_i^{\lceil \frac{3}{4} n \rceil}$ para $1 \leq i \leq N$.

Figura 4.10: Región de exploración de las islas.

La operación AA tiene como fin guiar la búsqueda de cada isla de manera que exploren regiones disjuntas del espacio de búsqueda. Para conseguir guiar la búsqueda, la operación AA ajusta el paisaje de aptitud de cada isla (es decir, los valores de aptitud de los individuos), de manera que los individuos de regiones apartadas tengan mayor aptitud que aquellos de regiones traslapadas. La aptitud $f(d)$ de un individuo d se modifica con la expresión $(\alpha_p(d) + \alpha_g(d)) \cdot f(d)$, donde $\alpha_p(d)$ y $\alpha_g(d)$ son los coeficientes de ajuste fenotípico y genotípico, respectivamente. Ambos coeficientes son calculados utilizando la información de las RE recibidas de todas las islas. El coeficiente α_p evita que dos islas exploten la misma región de búsqueda en el espacio de las funciones objetivo; mientras que el coeficiente α_g evita que las islas evolucionen a la misma región de búsqueda del espacio de las variables de decisión.

4.6.7. Local Cultivation Genetic Algorithm (LCGA)

Watanabe et al. [Watanabe02] proponen un algoritmo genético basado en el modelo «amo-esclavo» llamado *Local Cultivation Genetic Algorithm* (LCGA). El algoritmo se concentra en reforzar la capacidad exploratoria de la cruza y, además, reúne los mecanismos más exitosos de los AEMOs actuales, a saber: elitismo (véase § 3.4.2), utilización de la población secundaria en el mecanismo de búsqueda, y reducción de la población secundaria.

El LCGA utiliza dos poblaciones: la población de búsqueda P de tamaño N y un archivo histórico A del mismo tamaño. Inicialmente, la población global P se debe ordenar con respecto a una de las funciones objetivo. En cada generación se elige otra función objetivo para realizar la ordenación. Posteriormente, la población se agrupa en pares tomando consecutivamente individuos de la población ordenada. Un proceso «amo» se encarga de enviar cada uno de estos

[3] En la comunicación asíncrona la isla que envía la información continúa su ejecución sin esperar a que la isla receptora la reciba.

grupos a distintos procesos «esclavo» para que realicen la cruza, la mutación y la evaluación de los dos hijos resultantes. Enseguida los esclavos envían su par de individuos al proceso amo para reunirlos en la población global P. Esta población, a su vez se combina con el archivo histórico A, y se seleccionan N individuos entre los $2N$ totales. Para reducir el número de individuos se realiza el mismo procedimiento que emplea la selección ambiental (*environmental selection*) del SPEA2 [Zitzler01].

Para evaluar su desempeño, LCGA fue comparado con los AEMOs seriales, NSGA-II, MOGA y SPEA2. Los autores consideran que LCGA es un método robusto ya que, a pesar de no tener los mejores resultados en todos los problemas de prueba, obtuvo resultados favorables en todos ellos.

Capítulo 5

El algoritmo genético multiobjetivo con múltiples resoluciones

Sumario

5.1. Motivación

En el campo de los algoritmos evolutivos paralelos para un solo objetivo se han demostrado teórica y empíricamente varios resultados importantes. Con el reciente surgimiento de los algoritmos evolutivos multiobjetivo paralelos (pAEMOs) es necesario validar aquellos resultados en el contexto multiobjetivo y resolver las nuevas problemáticas propias de la optimización multiobjetivo (véase §4.4). En este sentido han surgido diversos pAEMOs, sin embargo, en la mayoría de ellos no se trata el problema de asignar una región distinta del espacio de búsqueda a cada procesador. En todos los algoritmos que atienden esta problemática, la asignación se hace con respecto al espacio de las funciones objetivo [Hiroyasu00][Deb03][Zhu02][Watanabe02]. Con todo, estos esquemas tienen el problema de no garantizar que los individuos generados en cada procesador pertenezcan a la región que les fue asignada. Por otra parte, ninguno de los trabajos incluye una estrategia adecuada para manejar la población secundaria.

Partiendo de las observaciones anteriores, diseñamos el algoritmo llamado *algoritmo genético multiobjetivo con múltiples resoluciones* (MRMOGA, *'Multiple Resolution Multi-Objective Genetic Algorithm'*), el cual constituye la contribución principal de este trabajo de tesis.

5.2. El algoritmo genético multiobjetivo con múltiples resoluciones

El modelo propuesto está basado en el paradigma de isla con nodos heterogéneos. Se caracteriza por lo siguiente:

- Integra un esquema para manejar la división del espacio de búsqueda en el espacio de las variables de decisión
- Cada isla codifica la solución con una resolución distinta (es decir, con distintos dígitos de precisión después del punto decimal).
- Utiliza un esquema de migración asíncrono en donde interviene tanto la población primaria como el archivo histórico (o población secundaria).
- Incorpora una estrategia para detectar la convergencia del AEMO serial con el fin de incrementar la resolución inicial.

En el contexto monoobjetivo Lin et al. [Lin94] propusieron un modelo de isla llamado iiGA (*Injection Island* GA). Este esquema tiene varias subpoblaciones que codifican el problema usando diferente precisión. El iiGA está diseñado para optimización combinatoria, y por consiguiente las cadenas binarias representan por sí solas la solución al problema. En contraste, el MRMOGA está diseñado para optimización numérica y, por tanto, incorpora una estrategia apropiada para adaptar las cadenas binarias de una resolución a otra.

5.2.1. El principio de múltiples resoluciones

El funcionamiento de MRMOGA se fundamenta en la siguiente suposición: las soluciones óptimas de Pareto se encuentran en menos iteraciones utilizando representaciones con resoluciones bajas que usando representaciones con resoluciones altas. En efecto, el espacio de búsqueda es menor conforme se disminuye la resolución, por lo que se necesitan explorar menos soluciones para determinar el frente de Pareto conocido (estas soluciones no pertenecen necesariamente al frente de Pareto real).

Aprovechando esta idea podemos diseñar un modelo paralelo con islas con resoluciones distintas. Las islas con baja resolución tendrán la finalidad de aproximarse rápidamente al frente de Pareto encontrando individuos altamente aptos. Estos individuos posteriormente se introducen en las islas con resoluciones altas para que refinen las soluciones.

Este esquema puede ser considerado como una división del espacio de las variables de decisión. En esta división hay puntos que se traslapan, de manera que el espacio de búsqueda de una isla de baja resolución está contenido en el espacio de una isla con más alta resolución. En la figura 5.1 se ilustra la contención de los subespacios de búsqueda. En la figura se muestra que el espacio de búsqueda $\mathcal{S}_{r1}$ está contenido en el espacio de búsqueda de $\mathcal{S}_{r2}$ para las resoluciones $r1$ y $r2$ ($r1 < r2$).

5.2.2. El esquema general de MRMOGA

En el algoritmo 5.1 se muestra el esquema de MRMOGA. El proceso de búsqueda se divide en varios subprocesos distribuidos en cada una de las islas.

En la figura 5.2 se muestra una visión esquemática del funcionamiento de MRMOGA.

Los componentes principales son el AEMO secuencial que se ejecuta en cada isla, y el esquema de migración y reemplazo. En las secciones siguientes se describen de manera detallada cada uno de estos componentes.

1: **Parámetros:**
2: $\mathcal{P}_i$: población del i-ésimo procesador
3: E : número máximo de épocas
4: G : número de generaciones por época
5: $FP_{conocido}\binom{i}{t}$: frente de Pareto conocido del procesador i en la generación t
6: $FP_{conocido}\binom{i}{}$: frente de Pareto conocido del procesador i en la última generación
7: $FP_{conocido}$: frente de Pareto conocido considerando la población global
8:

9: **procedimiento** MRMOGA
10: Generar aleatoriamente la población $\mathcal{P}_i$ en cada procesador

11: **para** $e \leftarrow 1$ hasta E **haz**
12: **para** $g \leftarrow 1$ hasta G **haz**
13: AEMOSERIAL($\mathcal{P}_i$)
14: **fin para**
15: **si** $e \neq E$ **entonces** ▷ No migrar en la última época
16: **para todo** vecino j del procesador i **haz** ▷ De acuerdo con la topología
17: MIGRAR($FP_{conocido}\binom{i}{g}$, $\mathcal{P}_j$)
18: **fin para**
19: **para todo** vecino j del procesador i **haz**
20: REEMPLAZAR($\mathcal{P}_i$, $FP_{conocido}\binom{j}{g}$)
21: **fin para**
22: **fin si**
23: **fin para**

24: Combinar todos los $FP_{conocido}\binom{i}{}$ en el procesador 0 y presentar $FP_{conocido}$
25: **fin procedimiento**

Algoritmo 5.1: Pseudocódigo del algoritmo MRMOGA.

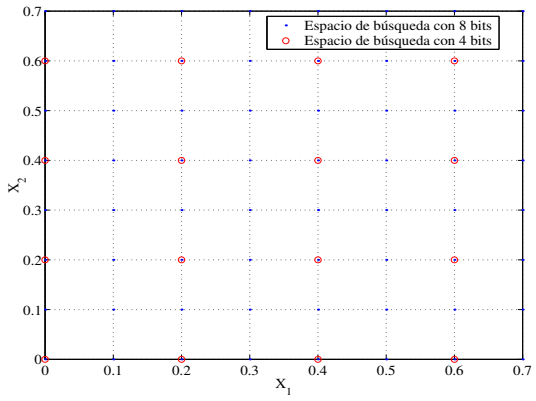


Figura 5.1: Espacios de búsqueda para las resoluciones r_1 y r_2 de 4 y 8 bits respectivamente.

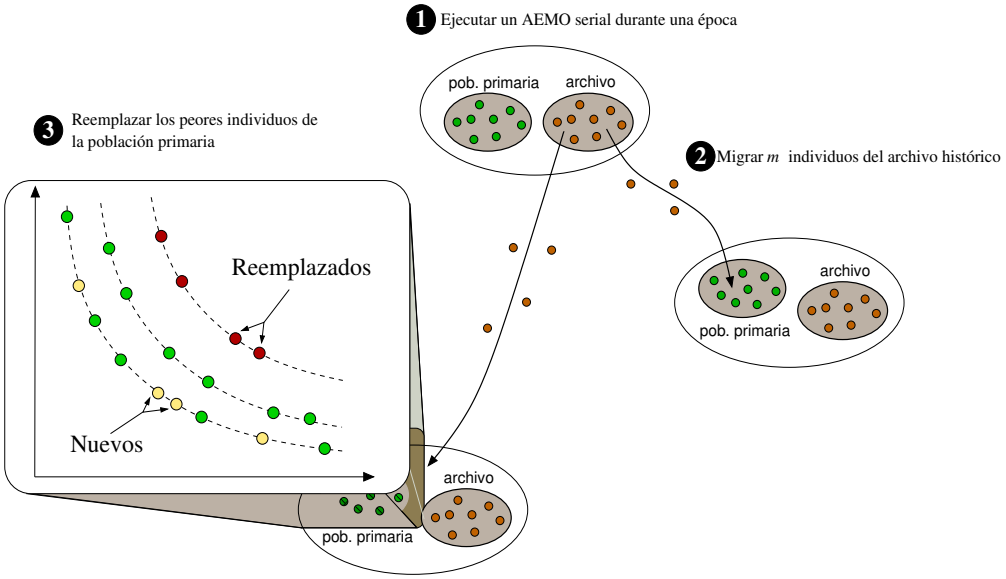


Figura 5.2: Vista esquemática de MRMOGA.

5.3. Conversión en la migración

La resolución de las soluciones está en función de la longitud de las cadenas binarias de los cromosomas: cuanto mayor sea la longitud de las cadenas, mayor será la resolución de las soluciones. La longitud de la cadena binaria s_i que codifica a la variable $\mathbf{x}^{(i)}$ que pertenece al intervalo real $[x_i^{min},\ x_i^{max}]$ se obtiene mediante la expresión:

$$\ell_i = \left\lceil log_2 \left((x_i^{max} - x_i^{min}) \times 10^{p_i} + 1 \right) \right\rceil, \tag{5.1}$$

donde p_i (las cifras de precisión) es el número de cifras significativas después del punto decimal de la variable i. La longitud ℓ de la cadena binaria s (cromosoma) que codifica todas las variables es pues, $\ell = \sum_{i=1}^{n} \ell_i$, donde n es el número de variables.

Al migrar entre islas con individuos con cromosomas de diferente longitud, ℓ y ℓ' ($\ell < \ell'$), es preciso adaptar la cadena de menor longitud, ℓ, a una de mayor longitud, ℓ'. La adaptación debe cumplir con las siguientes características:

- Debe ser posible aplicar los operadores evolutivos a la cadena recodificada con la longitud ℓ'.
- La cadena s' (longitud ℓ') debe representar el mismo fenotipo que la cadena s (longitud ℓ).

La expresión para obtener un valor real, $x_i \in [x_i^{min},\ x_i^{max}]$, a partir de una cadena binaria, s_i, es la siguiente:

$$x_i = x_i^{min} + \frac{d_i}{2^{\ell_i} - 1}(x_i^{max} - x_i^{min}), \tag{5.2}$$

donde ℓ_i es la longitud de la cadena que codifica a la i-ésima variable y d_i es el valor entero decodificado de la cadena s_i.

En el modelo propuesto solamente se migra el fenotipo de los individuos, es decir, el vector con las variables decodificadas, $\vec{x} = [x_1, x_2, \ldots, x_n]$. Para realizar la conversión de longitud, necesitamos encontrar las cadenas s_i' (con la nueva longitud ℓ_i') que producen los valores x_i. Para esto, despejamos d_i de la ecuación (5.2) para obtener:

$$d_i^* = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}(2^{\ell_i'} - 1), \tag{5.3}$$

donde d_i^* es el valor real más cercano a d_i obtenido con una precisión de ℓ_i' bits. Este valor lo redondeamos al entero más cercano. Finalmente obtenemos la cadena binaria correspondiente con longitud ℓ'. En el algoritmo 5.2 se muestra el pseudocódigo para convertir un cromosoma de longitud ℓ a otro de longitud ℓ'. Cabe destacar que el vector $\vec{x}^*$ generado con la cadena s' (por medio de la ecuación (5.2)) no es exactamente igual al vector original $\vec{x}$. Sin embargo, la transformación garantiza mantener una precisión de al menos p_i' ($p_i < p_i'$) cifras decimales después del punto decimal.

Parámetros:
$\vec{x}$: vector de las variables decodificadas.
x_i^{min}: límite inferior del intervalo.
x_i^{max}: límite superior del intervalo.
s': cadena binaria resultante de longitud ℓ'.
ℓ': vector con las nuevas longitudes de las cadenas s_i'.

procedimiento CONVERSIÓN($\vec{x}, x_i^{min}, x_i^{max}, \ell'$)
  $s' \leftarrow \varepsilon$
  para cada variable i codificada en el cromosoma s' **haz**
    $d_i^* \leftarrow$ REDONDEAR$\Big((x_i - x_i^{min}) * (2^{\ell_i'} - 1)/(x_i^{max} - x_i^{min})\Big)$
    $s_i' \leftarrow$ DECIMALABINARIO($d_i^*,\ \ell_i'$)
    s′ ← s′ + s_i' ▷ Concatenación de cadenas
  fin para
fin procedimiento

Algoritmo 5.2: Transformación de una cadena binaria s de longitud ℓ a una cadena s' de longitud ℓ' ($\ell < \ell'$).

5.4. Topología y políticas de migración

Puesto que el espacio de búsqueda de una isla está contenido en el espacio de búsqueda de cualquier isla con mayor resolución, pero no lo contrario, el flujo de migración solamente

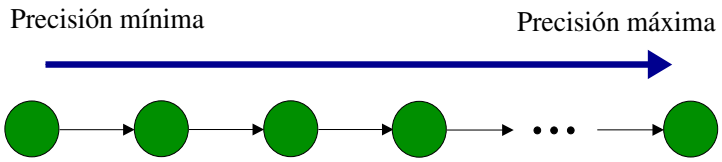


Figura 5.3: Topología simple

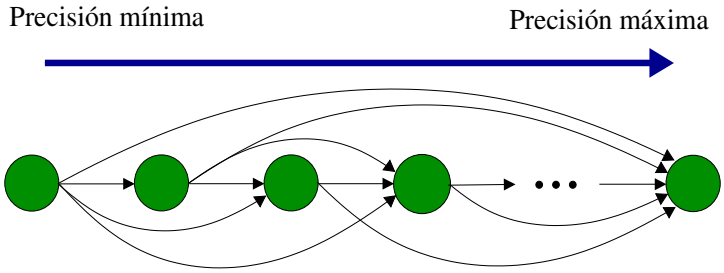


Figura 5.4: Topología completa

puede ir de las islas de menor resolución a las de mayor resolución. Esta condición establece una topología de interconexión jerárquica. Es decir una isla padre tiene como hijos a todas las islas con mayor resolución que ella. La raíz de la topología es la isla con las cadenas binarias con menor longitud. Atendiendo a estas consideraciones se definieron las dos topologías siguientes:

- **Topología simple**. Cada isla solamente migra individuos a su hijo directo (véase figura 5.3). El grado de esta topología (es decir, el número de vecinos de cada nodo) es $\delta(i) = 1$ para cada nodo $i = 0, \ldots, n$.

- **Topología completa**. Cada isla migra individuos a todos sus hijos en la jerarquía (véase figura 5.4). Esta topología tiene grado $\delta(i) = i$ para cada nodo $i = 0, \ldots, n$.

En el esquema de migración y reemplazo intervienen tanto la población primaria como la población secundaria. En el algoritmo 5.3 se muestra el pseudocódigo del esquema de migración y reemplazo.

- **Esquema de migración**. En la terminología utilizada en [Veldhuizen03], este esquema se clasifica como *elitista aleatorio*. En nuestro esquema se migran aleatoriamente x individuos del archivo histórico (es decir, el frente de Pareto conocido). Si el tamaño del archivo histórico es menor que x, entonces de la población primaria se eligen al azar los individuos necesarios para completar los x individuos.

- **Esquema de reemplazo**. Inicialmente, los individuos de la población primaria se jerarquizan por capas utilizando el algoritmo 3.2 (utilizado también por el NSGA-II). Seleccionamos x individuos del último frente en la jerarquía (los peores individuos) para ser reemplazados por los individuos que arribaron. Si el tamaño del frente es menor que x, tomamos sucesivamente individuos de los siguientes frentes hasta completar los x. A los nuevos individuos que llegaron se les aplica el procedimiento del algoritmo 5.2 para transformar sus cromosomas a la nueva longitud.

```
1:  Parámetros:
2:      m: número de individuos que se migran.
3:      P_i   : población del i-ésimo procesador.
4:      FP_conocido(i, t)   : frente de Pareto conocido del procesador i en la generación t.
5:
6:  procedimiento MIGRAR(m)
7:      si el tamaño de FP_conocido(i, g) ≥ m entonces
8:          migrantes ← muestra aleatoria de tamaño m del FP_conocido(i, g)
9:          Enviar migrantes
10:     otro
11:         migrantes ← FP_conocido(i, g)
12:         migrantes ← migrantes+ selección aleatoria de los
                individuos de P_i necesarios para completar m.
13:         Enviar migrantes
14:     fin si
15: fin procedimiento

16: procedimiento REEMPLAZAR(m)
17:     JERARQUIZARFRENTES(P_i)
18:     Elegir m individuos partiendo del último frente
19:     Recibir los m individuos provenientes de los procesadores vecinos
20:     Reemplazar los m individuos
21:     para todo cromosoma x de los individuos que arribaron haz
22:         CONVERSIÓN(x⃗, x_i^min, x_i^max, ℓ')
23:     fin para
24: fin procedimiento
```

Algoritmo 5.3: Pseudocódigo de la migración y reemplazo.

5.4.1. Carácter asíncrono del MRMOGA

Ya que las islas con resoluciones bajas tienen cadenas binarias pequeñas, los operadores evolutivos toman menos tiempo en aplicarse que en cadenas grandes. De esta manera, las islas con resoluciones bajas completan una época en menos tiempo y están listas para migrar antes de que las islas con resoluciones altas estén listas para recibir a los individuos. Con el fin de que las islas lentas no retrasen a las más rápidas, la comunicación entre las islas se realiza de manera asíncrona. Es decir, las islas continúan su ejecución antes de que las islas receptoras estén listas para recibir los individuos.

5.5. Características del AEMO serial

En cada procesador se ejecuta el AEMO serial descrito en el algoritmo 5.4.

```
1:  procedimiento MOEASERIAL(resolucionMaxima, P_i)
2:      EVALUARVALORESDEOBJETIVOS()
3:      ASIGNARJERARQUÍA()
4:      ASIGNARAPTITUDLINEAL()
5:      ESCALARAPTITUD()

6:      FILTROREJILLA(P_i)
7:      NUEVAGENERACIÓN()

8:      si CONVERGIDO(FP_conocido_i) y resolucion_i < resolucionMaxima entonces
9:          AUMENTARRESOLUCIÓN(P_i)
10:     fin si
11: fin procedimiento
```

Algoritmo 5.4: Pseudocódigo del algoritmo secuencial de cada procesador.

En las secciones siguientes se describen los elementos que conforman el AEMO secuencial.

5.5.1. Asignación de las jerarquías

La asignación de jerarquías está basada en la estrategia que utiliza MOGA [Fonseca93]. La jerarquía de un individuo está dada por:

$$\text{jerarquía}(x_i, t) = 1 + p_i^{(t)}$$

donde x_i es un individuo en la generación t, el cual es dominado por $p_i^{(t)}$ individuos en la generación t.

La aptitud se asigna mediante los pasos siguientes:

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 $n \leq M$). La interpolación se hace usualmente con una función lineal.

3. Promediar la aptitud de los individuos con la misma jerarquía.

Es importante destacar que a diferencia de MOGA, en MRMOGA no se utiliza ningún mecanismo de nichos (véase §3.4.3) para mantener diversidad. En varias publicaciones se conjetura que el modelo de islas provee un mecanismo natural para mantener diversidad debido a la evolución independiente de varias subpoblaciones [Alba99][Cantu00][Lin94][Tomassini99]. Con la intención de comprobar esta hipótesis se decidió no utilizar un mecanismo de nichos tanto en la localidad de las islas como de manera global.

5.5.2. Operadores genéticos

Los operadores genéticos utilizados en el algoritmo se describen a continuación:

Selección. La técnica de selección que se utilizó fue la selección universal estocástica propuesta por Baker [Baker87]. La técnica se muestra en el algoritmo 5.5.

$ptr \leftarrow$ aleatorio() ▷ genera un número aleatorio entre 0.0 y 1.0
$sum \leftarrow 0$
para $i \leftarrow 1$ hasta n **haz**
  para $sum \leftarrow sum + ValEsp(i,t)$ y mientras $sum > ptr$ **haz**
    seleccionar(i)
    $ptr \leftarrow ptr + 1$
  fin para
fin para

Algoritmo 5.5: Selección universal estocástica.

Cruza. La cruza de los individuos se llevó a cabo mediante la cruza de dos puntos, la cual, de acuerdo a resultados empíricos, minimiza los efectos disruptivos de la cruza (para detalles sobre su funcionamiento véase § 2.3.2).

Mutación. MRMOGA emplea la mutación uniforme, la cual mantiene un porcentaje de mutación constante a lo largo de todo el proceso evolutivo (para detalles sobre su funcionamiento véase § 2.3.2).

5.5.3. Malla adaptativa

Para retener las soluciones no dominadas y distribuirlas uniformemente en el espacio de las funciones objetivo se utilizó una malla como la descrita en [Toscano01]. El funcionamiento de la malla es similar al mostrado en la sección 3.4.3, pero tiene la particularidad de adaptarse cuando un individuo se ubica fuera del rango de la malla.

5.5.4. Manejo de restricciones

El MRMOGA utiliza una estrategia muy simple para manejar problemas con restricciones. Para determinar la dominancia entre dos individuos se compara el número de restricciones que violan. El individuo dominante será aquél que viole menos restricciones. En caso de que los individuos violen el mismo número de restricciones se llevará a cabo la comparación de acuerdo con la dominancia de Pareto. Esta estrategia se lleva a cabo al momento de asignar la jerarquía de Pareto a los individuos de la población primaria, y al momento de decidir la entrada de un individuo a la población secundaria.

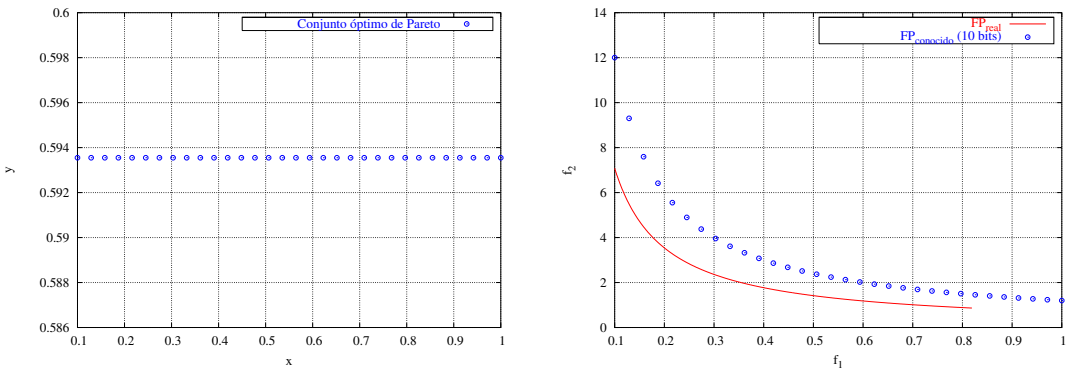
5.5.5. Aumento de la resolución en las islas

En general, en los esquemas para asignar regiones disjuntas del espacio de búsqueda a los procesadores no se puede garantizar que las islas generen únicamente individuos dentro la región que tienen asignada. Sin embargo, hay esquemas en los cuales se garantiza que todos los individuos generados estén dentro de la región asignada. Un ejemplo de estos esquemas es el que utiliza MRMOGA. En efecto, las islas con una cierta resolución solamente generan individuos dentro del subespacio de búsqueda determinado por su resolución. No obstante, este tipo de esquemas tiene algunos inconvenientes cuando las regiones se asignan de manera estática. El conjunto de óptimos de Pareto puede estar constituido por soluciones concentradas en una pequeña región del espacio de búsqueda de manera que algunas regiones contendrían unas pocas o ninguna solución óptima. Esto implica un desequilibrio en la contribución de cada isla para encontrar el frente de Pareto. Desde el punto de vista de la eficiencia, se desaprovecha la potencia de cómputo de algunos procesadores, y desde el punto de vista de la eficacia, disminuye la probabilidad de encontrar el frente de Pareto real. Se ha demostrado que en un AE la probabilidad de encontrar un óptimo global tras un número cualquiera de pasos depende del número de procesadores que se están usando [Alba99a].

En el caso de MRMOGA nos encontramos con las siguientes dificultades:

- El espacio de búsqueda de las islas con baja resolución es proporcionalmente más pequeño, y por tanto, el frente no dominado es encontrado en muchas menos iteraciones que en las islas con resoluciones altas. En consecuencia, cuando las islas con baja resolución convergen, consumen el tiempo que resta para completar la búsqueda oscilando alrededor del frente sin contribuir significativamente a encontrar otras regiones del frente.

- Para ciertas resoluciones (particularmente las bajas) el frente de Pareto conocido no pertenece al frente de Pareto real por lo que las islas con estas resoluciones no contribuyen a encontrar soluciones del frente. En la figura 5.5 se ilustra esta situación. En esta figura se observa que el frente de Pareto conocido obtenido con una resolución de 10 bits no pertenece al frente de Pareto real del problema bimodal de Deb (definido en la sección 6.3.1).



(a) (b)

Figura 5.5: Conjunto óptimo de pareto conocido (a) utilizando cadenas de 10 bits (precisión de 1 decimal), y su frente de Pareto conocido (b).

Para resolver esta problemática, la estrategia que propone MRMOGA consiste en aumentar la resolución de cada isla una vez que alcance la convergencia nominal. De esta manera, en todo

momento se aprovechará la potencia de todos los procesadores y las islas que ya convergieron seguirán contribuyendo a encontrar otras regiones del frente de Pareto real.

5.5.6. Criterio para determinar la convergencia nominal

La detección de la convergencia nominal en las islas se consigue monitoreando los movimientos del archivo histórico. En el proceso para determinar si un individuo pertenece al frente de Pareto conocido ($FP_{conocido}(t)$) ocurre alguna de las situaciones siguientes:

1. Sin importar si el individuo pertenece o no al $FP_{conocido}(t)$, se eliminan del archivo a los individuos que dominó. En este caso podemos interpretar que el $FP_{conocido}(t)$ se está acercando al FP_{real}.

2. El individuo entra al $FP_{conocido}(t)$ sin eliminar individuos. El archivo crece, lo cual significa que se están encontrado nuevas regiones del FP_{real}.

3. El individuo es no dominado, pero reemplaza al individuo que está en una región más poblada de la rejilla adaptativa. La búsqueda llegó a una etapa en la que, principalmente, se concentra en obtener una mejor distribución del $FP_{conocido}(t)$.

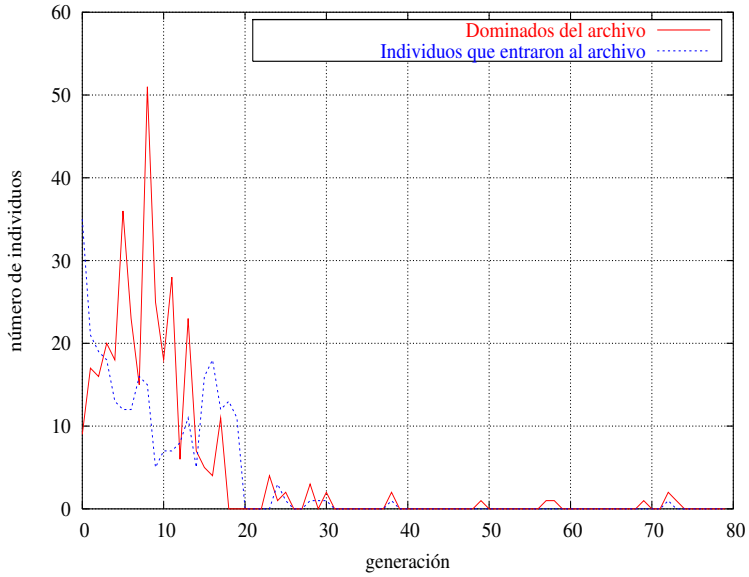
Para ilustrar los tres puntos anteriores, tomemos como ejemplo la ejecución de una sola isla al resolver el problema bimodal de Deb (v. § 6.3.1)) . En la figura 5.6(a) se ilustra el número de dominados del archivo histórico y el número de individuos que entran al archivo en cada generación. En la figura 5.6(b) se muestra el $FP_{conocido}(t)$ en dos etapas del proceso de búsqueda. En las primeras 30 generaciones del proceso de búsqueda hay un gran número de individuos dominados en $FP_{conocido}(t = 30)$ (5.6(a)) por lo que podemos inferir que la población sigue avanzando hacia el FP_{real}. Esto se refleja en el frente conocido encontrado en la generación 30 en cual se muestran unos pocos individuos que ya están próximos al FP_{real}. En cambio, después de la generación 30 el número de individuos dominados de $FP_{conocido}(t = 80)$ comienzan a disminuir hasta hacerse casi cero. En este momento el $FP_{conocido}(t = 80)$ está sobre el FP_{real} (5.6(b)).

La población de una isla se encuentra cerca de la convergencia nominal cuando en cada generación la mayoría de los movimientos del archivo histórico son del tercer tipo. Para detectar la convergencia, en cada generación se registra el número de individuos dominados del $FP_{conocido}(i)$ (tipo 1) en la generación i ($dominados_i$).

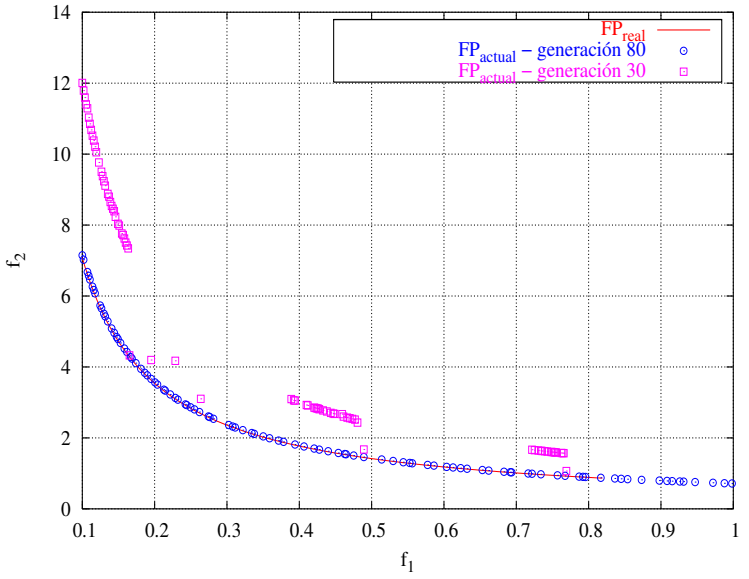
En función de $dominados_i$ se determina si la isla ha convergido lo suficiente para aumentar la resolución de la población. En cada generación se determina si en las k generaciones más recientes se satisface la condición:

$$\sum_{i=1}^{k} \frac{dominados_i}{k} \leq \epsilon, \tag{5.4}$$

donde ϵ es el promedio de individuos dominados del $FP_{conocido}$ en las últimas k generaciones. Experimentalmente se determinó que $\epsilon = 0.05$ funciona bien para todos los problemas de prueba que se consideraron. Si la condición 5.4 se cumple, entonces la isla modifica su representación a la siguiente resolución. El cambio de resolución se realiza de la siguiente manera: la mitad de la población se reinicia de manera aleatoria con la nueva resolución. La otra mitad se toma de una muestra aleatoria del archivo histórico. La conversión de la resolución de estos últimos individuos se realiza con el mismo procedimiento que se utiliza en la migración.



(a)



(b)

Figura 5.6: En (a) se muestra el número de individuos que entran al archivo histórico y el número de individuos dominados del $FP_{conocido}(t)$. En (b) se muestra el $FP_{conocido}(t)$ en la generación $t = 30$ y $t = 80$.

Capítulo 6

Evaluación del algoritmo propuesto y resultados

Sumario

6.1. Introducción

En este capítulo se presenta la evaluación experimental de la eficacia y la eficiencia de MRMOGA. La evaluación se realizó utilizando 9 problemas multiobjetivo que reúnen diferentes características que causan dificultades a un AEMO. El grupo de problemas más importante fue elegido por tener un espacio de búsqueda grande que los hace apropiados para resolverse mediante un pAEMO. Otros problemas fueron seleccionados por tener un frente de Pareto real desconectado que pone a prueba la capacidad que tiene un pAEMO para mantener una población diversa. Asimismo, se eligieron tres problemas con restricciones. Como punto de referencia, los resultados de MRMOGA se contrastan con aquellos obtenidos por una versión paralela de NSGA-II.

El estudio experimental descrito en este capítulo está dividido en tres partes. La primera estudia la influencia de la topología en el desempeño de MRMOGA. Se estudian la topología simple y la compuesta descritas en la sección 5.4. La segunda parte de este estudio se concentra

en la eficacia de MRMOGA con la topología simple. Para tal fin, se realiza una comparación sistemática de los algoritmos sobre todos los problemas de prueba. En la tercera parte se analiza la eficiencia de MRMOGA. En este último estudio se utilizan tres métricas bien conocidas en la computación paralela: la aceleración, la eficiencia, y la fracción serial.

Al término de cada uno de estos estudios se recogen las conclusiones obtenidas de los resultados experimentales.

6.2. Métricas para evaluar la eficacia del MRMOGA

Una sola métrica difícilmente puede reflejar el desempeño global de un AEMO. Algunas métricas miden la convergencia del algoritmo al frente de Pareto real y otras la distribución y diversidad de las soluciones. En este caso necesitamos evaluar un AMEO considerando varias métricas simultáneamente. Sin embargo, la métricas evalúan dos objetivos en conflicto (la convergencia y la diversidad). Si los valores de las métricas de un algoritmo dominan a las de otro, entonces podemos afirmar con seguridad que el primero es mejor que el segundo. En otro caso, no podemos concluir nada acerca de los dos algoritmos.

Las métricas que se utilizarán para evaluar la eficacia de MRMOGA son el *espaciamiento* para evaluar la distribución; en tanto que para evaluar la convergencia se usaron la *tasa de error*, el *conteo de aciertos*, la *distancia generacional*, la *distancia generacional invertida*, y la *cobertura de conjuntos*. Adicionalmente, se usarán las comparaciones visuales de los conjuntos $FP_{conocido}$ para confirmar los resultados obtenidos con las métricas. En esta sección también se define la métrica de hipervolumen, la cual será utilizada en el análisis de eficiencia (en la sección 6.10 se describirá la manera en que se utilizó esta métrica).

6.2.1. Tasa de error (TE)

La tasa de error [Veldhuizen99] mide el porcentaje de vectores del frente de Pareto conocido ($FP_{conocido}$) que no pertenecen al frente de Pareto real (FP_{real}). Matemáticamente se describe mediante:

$$TE = \frac{\sum_{i=1}^{n} e_i}{n}, \tag{6.1}$$

donde n es el número de vectores en $FP_{conocido}$ y

$$e_i = \begin{cases} 0 & \text{si el vector } \mathbf{u}^{(i)} \in FP_{real} \ (i = 1, \ldots, n), \\ 1 & \text{en otro caso.} \end{cases}$$

$TE = 0$ indica que todo vector reportado por el AEMO en $FP_{conocido}$ pertenece[1] a FP_{real}; mientras que $ER = 1$ indica que ningún vector reportado es miembro de FP_{real}.

6.2.2. Conteo de aciertos (CA)

Esta métrica cuenta el número de vectores de $FP_{conocido}$ que pertenecen a FP_{real}. Matemáticamente se describe mediante:

$$CA = \frac{\sum_{i=1}^{n} s_i}{n}, \tag{6.2}$$

donde n es el número de vectores en el conjunto $FP_{conocido}$ y

$$s_i = \begin{cases} 1 & \text{si el vector } \mathbf{u}^{(i)} \in FP_{real} \ (i = 1, \ldots, n), \\ 0 & \text{en otro caso.} \end{cases}$$

[1] Es útil recordar que un vector, $\mathbf{u}$, pertenece a FP_{real} si no existe otro vector, $\mathbf{v}$, en FP_{real} tal que $\mathbf{v} \preceq \mathbf{u}$

$CA = n$ indica que todo vector reportado por el AEMO en el conjunto $FP_{conocido}$ pertenece a FP_{real}; mientras que $CA = 0$ indica que ningún vector reportado es miembro de FP_{real}.

6.2.3. Cobertura de conjuntos ($\mathcal{C}$)

Esta métrica fue propuesta en [Zitzler99a] para comparar la calidad global de dos conjuntos no dominados de manera directa. Si A y B son dos conjuntos no dominados, la métrica de cobertura $\mathcal{C}(A, B)$ calcula la fracción de vectores en B que son dominados débilmente por los vectores de A:

$$\mathcal{C}(A, B) = \frac{|\{b \in B \mid \exists\, a \in A : a \preceq b\}|}{|B|} \tag{6.3}$$

El valor $\mathcal{C}(A, B) = 1$ significa que todos los miembros de B son dominados o iguales a los miembros de A (es decir, B es cubierto por A). El resultado opuesto, $\mathcal{C}(A, B) = 0$, representa la situación cuando ninguna de las soluciones en B es cubierta por el conjunto A. Es preciso calcular tanto $\mathcal{C}(A, B)$ como $\mathcal{C}(B, A)$, ya que, en general, $\mathcal{C}(A, B)$ no es igual a $1 - \mathcal{C}(B, A)$.

6.2.4. Distancia generacional (DG)

La distancia generacional [Veldhuizen99] representa el promedio de la distancia de los vectores de $FP_{conocido}$ a FP_{real}. Se define mediante:

$$DG = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \tag{6.4}$$

donde n es el número de vectores en $FP_{conocido}$, y d_i es la distancia euclidiana entre cada vector de $FP_{conocido}$ y el miembro más cercano de FP_{real}. $DG = 0$ indica que $FP_{conocido} \subseteq FP_{real}$; cualquier otro valor indica que $FP_{conocido}$ se desvía de FP_{real}.

6.2.5. Distancia generacional invertida (DGI)

La distancia generacional invertida [Veldhuizen99] representa el promedio de la distancia de los vectores del conjunto FP_{real} al conjunto $FP_{conocido}$. Se define mediante:

$$DGI = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \tag{6.5}$$

donde n es el número de vectores en FP_{real}, y d_i es la distancia euclidiana entre cada vector de FP_{real} y el miembro más cercano del conjunto $FP_{conocido}$. El propósito de esta métrica es medir el progreso global hacia el frente de Pareto real y, además, la extensión del conjunto $FP_{conocido}$ sobre FP_{real}. Un conjunto no dominado que se sitúe en una sola región del frente de Pareto real será penalizado en el valor de esta métrica aunque sus miembros pertenezcan (o estén próximos) a FP_{real}. $DG = 0$ indica que $FP_{conocido} \subseteq FP_{real}$ y sus elementos se extienden a lo largo de todo el conjunto FP_{real}; cualquier otro valor indica que $FP_{conocido}$ se desvía de FP_{real}.

6.2.6. Espaciamiento (SP)

La métrica de espaciamiento [Schott95] indica qué tan bien está distribuido $FP_{conocido}$ sobre el espacio que ocupa. Es decir, si los vectores de $FP_{conocido}$ guardan la misma distancia unos con

otros a lo largo de las regiones que cubren. La métrica se define mediante:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}, \tag{6.6}$$

donde $d_i = min_j(|f_1^i(\mathbf{x}) - f_1^j(\mathbf{x})| + |f_2^i(\mathbf{x}) - f_2^j(\mathbf{x})|)$, $i, j = 1, \ldots, n$, $\bar{d}$ es la media de todas las d_i y n es el número de vectores en $FP_{conocido}$. Un valor de cero indica que todos los vectores de $FP_{conocido}$ están dispuestos equidistantemente. Cabe mencionar que esta métrica no muestra si todas la regiones de FP_{real} son cubiertas, sino solamente la equidistancia entre los vectores encontrados. La distancia d_i no está normalizada, lo cual puede acarrear resultados poco confiables para determinar la distribución de un conjunto no dominado. Si un conjunto de soluciones no dominadas se concentra en una región reducida del frente de Pareto, las distancias entre ellas será mucho menor que si se encuentran distribuidas a lo largo de todo el frente de Pareto real. Esta desproporción en las distancias producirá un valor menor para la métrica SP en el caso de la soluciones concentradas en una sola región. Por esta razón tenemos que apoyarnos de la comparación visual de los conjuntos $FP_{conocido}$ para confirmar el resultado de esta métrica.

6.2.7. Hipervolumen (HV)

La métrica de hipervolumen [Zitzler99a] evalúa la calidad global de un conjunto $FP_{conocido}$. Para este fin calcula el volumen en el espacio de las funciones objetivo que cubren los miembros del conjunto $FP_{conocido}$. Este volumen representa el tamaño de la región que dominan los miembros de $FP_{conocido}$.

Para un problema con k objetivos, dado un vector $\mathbf{u}^1 \in FP_{conocido}$, y un vector de referencia $\mathbf{u}^{ref}$ dominado por $\mathbf{u}^1$, se define la región dominada por $\mathbf{u}^1$ y acotada por $\mathbf{u}^{ref}$:

$$R(\mathbf{u}^1, \mathbf{u}^{ref}) = \{\mathbf{v} \,|\, \mathbf{v} < \mathbf{u}^{ref} \text{ y } \mathbf{u}^1 < \mathbf{v}, \mathbf{v} \in \mathbb{R}^k\}. \tag{6.7}$$

La región que domina $FP_{conocido}$ se define mediante:

$$HV = \bigcup_{i=1}^{n} R(\mathbf{u}^i, \mathbf{u}^{ref}), \tag{6.8}$$

donde n es el número de vectores en $FP_{conocido}$. En un problema de minimización (maximización), el vector de referencia $\mathbf{z}^{ref}$ es el vector formado con los valores máximos (mínimos) de cada objetivo.

6.3. Problemas sin restricciones

6.3.1. Problema DEB

Deb [Deb99b] propone el siguiente problema bimodal (es decir, con un frente de Pareto local y otro global) con dos objetivos y dos variables:

Minimizar $F = (f_1(x, y), f_2(x, y))$, donde

$$f_1(x, y) = x_1, \tag{6.9}$$

$$f_2(x, y) = \frac{g(x_2)}{x_1}, \tag{6.10}$$

donde ambas variables están en el intervalo $[0.1, 1]$ y $g(x_2)$ es la siguiente función bimodal:

$$g(x_2) = 2.0 - \exp\left\{-\left(\frac{x_2 - 0.2}{0.004}\right)^2\right\} - 0.8\exp\left\{-\left(\frac{x_2 - 0.6}{0.4}\right)^2\right\}. \tag{6.11}$$

El frente de Pareto local ocurre en $x_2 \approx 0.6$ y el frente de Pareto global en $x_2 \approx 0.2$. En la figura 6.1 se muestran los dos frentes del problema DEB.

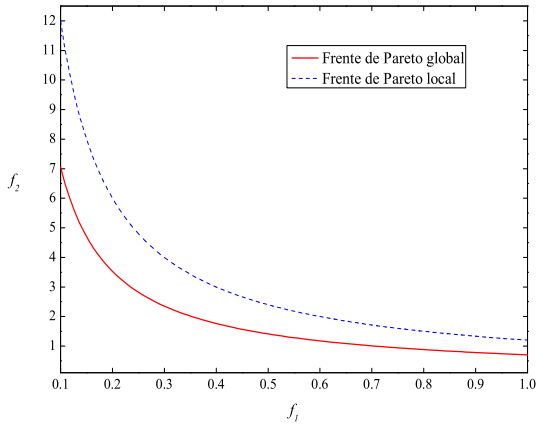


Figura 6.1: Los frentes de Pareto global y local del problema DEB.

6.3.2. La serie de problemas ZDT

Los tres problemas siguientes son parte de la serie de problemas propuestos por Zitzler et al. [Zitzler99b]. Estos problemas están diseñados poniendo énfasis en algunas de las características que causan dificultades para un AEMO. De este modo, el problema ZDT1 presenta un frente convexo, el problema ZDT2 tiene un frente no convexo, y el problema ZDT3 un frente discontinuo.

Cada uno de los tres problemas de prueba descritos a continuación tienen la misma estructura y consisten de tres funciones, f_1, g, h:

$$\begin{array}{lrcl} \text{Minimizar} & F & = & (f_1(x_1), f_2(\mathbf{x})), \\ \text{sujeto a} & f_2(\mathbf{x}) & = & g(x_2, \ldots, x_m)h(f_1(x_1), g(x_2, \ldots, x_m)), \\ \text{donde} & \mathbf{x} & = & (x_1, \ldots, x_m). \end{array} \tag{6.12}$$

La función f_1 depende solamente de la primer variable de decisión, g es una función de las $m-1$ variables restantes, y los parámetros de h son los valores de las funciones f_1 y g.

- La función de prueba ZDT1 (figura 6.2(a)) tiene un frente de Pareto convexo:

$$f_1(x_1) = x_1, \tag{6.13}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \times \sum_{i=2}^{m} \frac{x_i}{m-1}, \tag{6.14}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}, \tag{6.15}$$

donde $m = 30$ y $x_i \in [0,\ 1]$. El frente de Pareto real se forma con $g(\mathbf{x}) = 1$.

- La función de prueba ZDT2 (figura 6.2(b)) tiene un frente de Pareto no convexo:

$$f_1(x_1) = x_1, \tag{6.16}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \times \sum_{i=2}^{m} \frac{x_i}{m-1}, \tag{6.17}$$

$$h(f_1, g) = 1 - (f_1/g)^2, \tag{6.18}$$

donde $m = 30$ y $x_i \in [0, 1]$. El frente de Pareto real se forma con $g(\mathbf{x}) = 1$.

- La función de prueba ZDT3 (figura 6.2(c)) tiene un frente de Pareto que consiste de varias regiones covexas no contiguas:

$$f_1(x_1) = x_1, \tag{6.19}$$

$$g(x_2, \ldots, x_m) = 1 + 9 \times \sum_{i=2}^{m} \frac{x_i}{m-1}, \tag{6.20}$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\text{sen}(10\pi f_1), \tag{6.21}$$

donde $m = 30$ y $x_i \in [0,\ 1]$. El frente de Pareto real se forma con $g(x) = 1$. La función seno en h produce una discontinuidad en el frente de Pareto. Sin embargo, no hay discontinuidad en el espacio de las variables de decisión.

6.3.3. Problema KUR

Este problema de dos objetivos fue definido por Kursawe [Kursawe91]:
Minimizar $F = (f_1(\vec{x}), f_2(\vec{x}))$, donde

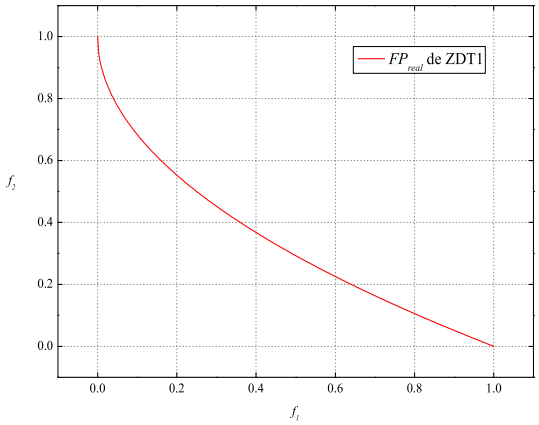
$$f_1(\vec{x}) = \sum_{i=1}^{2} \left(-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2} \right) \right), \tag{6.22}$$

$$f_2(\vec{x}) = \sum_{i=1}^{3} \left(|x_i|^{0.8} + 5\,\text{sen}(x_i^3) \right), \tag{6.23}$$

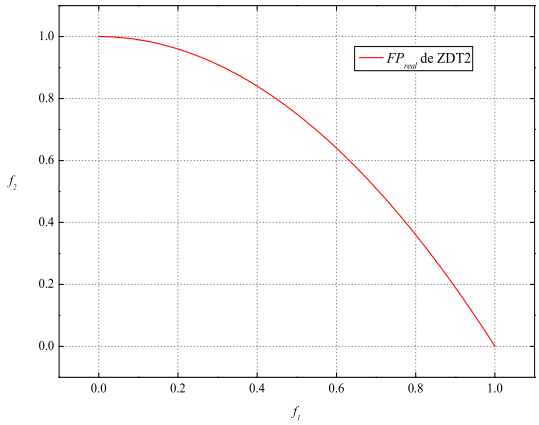
sujeto a

$$-5 \leq x_i \leq 5, \qquad i = 1, 2, 3$$

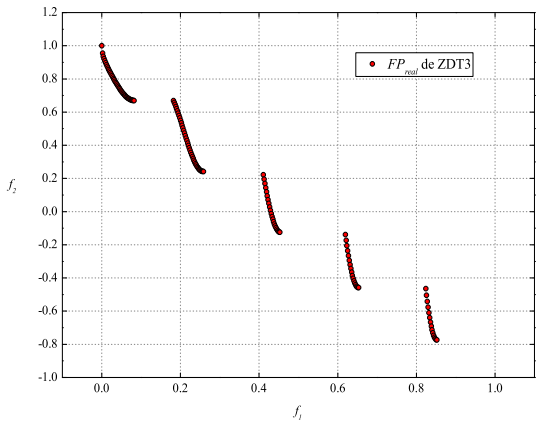
Este problema es difícil por presentar un frente de Pareto no convexo y discontinuo. En la figura 6.3 se muestran las cuatro regiones desconectadas que componen el frente de Pareto. A diferencia del problema ZDT3, este problema es discontinuo tanto en el espacio de la funciones objetivo como en el de las variables de decisión.



(a) Frente de Pareto de ZDT1.



(b) Frente de Pareto de ZDT2.



(c) Frente de Pareto de ZDT3.

Figura 6.2: Frentes de Pareto reales de la serie de problemas ZDT

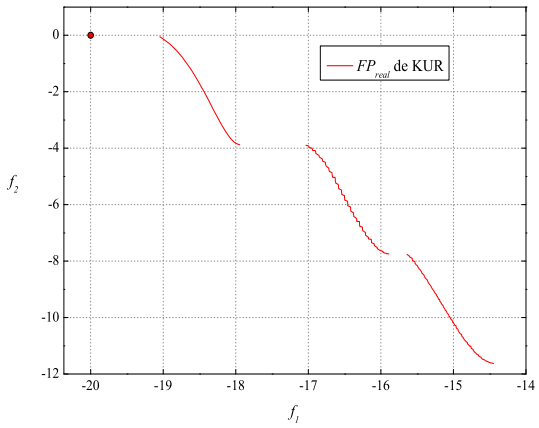


Figura 6.3: El frente de Pareto de KUR está compuesto por cuatro regiones.

6.3.4. Problema DTLZ7

Este problema con M objetivos fue propuesto en [Deb02]. Tiene un conjunto de regiones desconectadas que forman el frente de Pareto:

$$\begin{array}{rl} \text{Minimizar} & f_1(\mathbf{x}) = x_1, \\ \text{Minimizar} & f_2(\mathbf{x}) = x_2, \\ \vdots & \vdots \\ \text{Minimizar} & f_{M-1}(\mathbf{x}) = x_{M-1}, \\ \text{Minimizar} & f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M))h(f_1, f_2, \ldots, f_{M-1}, g), \\ \text{donde} & g(\mathbf{x}_M) = 1 + \frac{9}{|\mathbf{x}_M|}\sum_{x_i \in \mathbf{x}_M} x_i, \\ & h(f_1, f_2, \ldots, f_{M-1}, g) = M - \sum_{i=1}^{M-1}\left[\frac{f_i}{1+g}(1 + \sin(3\pi f_i))\right], \\ \text{sujeto a} & 0 \leq x_i \leq 1, \quad \text{para } i = 1, 2, \ldots, n. \end{array} \tag{6.24}$$

Los autores recomiendan usar $M = 3$ objetivos y $n = 22$ variables. El problema tiene 2^{M-1} regiones desconectadas del frente de Pareto. Las soluciones óptimas de Pareto son $x_M = 0$. En la figura 6.4 se muestra el frente de Pareto del problema DTLZ7.

6.4. Problemas con restricciones

6.4.1. Problema KIT

El problema KIT fue propuesto en [Kita96]:
Maximizar $F = (f_1(x, y), f_2(x, y))$, donde

$$f_1(x, y) = -x^2 + y, \tag{6.25}$$

$$f_2(x, y) = \frac{1}{2}x + y + 1. \tag{6.26}$$

Sujeto a

$$x, y \geq 0,$$

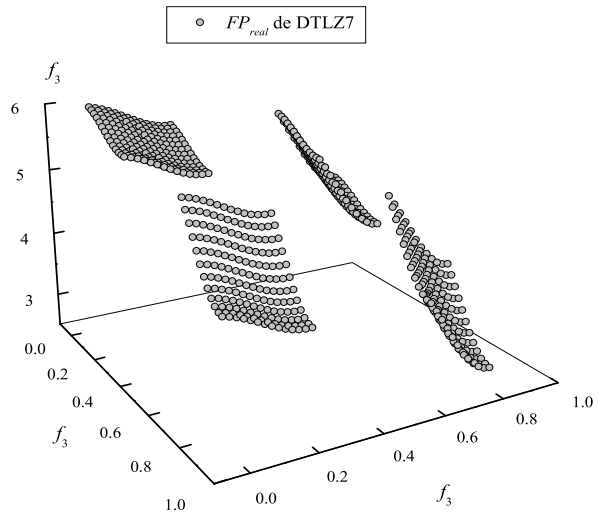


Figura 6.4: El frente de Pareto de DTLZ7 tiene cuatro regiones desconectadas para $M = 3$.

$$0 \geq \frac{1}{6}x + y - \frac{13}{2}, \tag{6.27}$$

$$0 \geq \frac{1}{2}x + y - \frac{15}{2}, \tag{6.28}$$

$$0 \geq 5x + y - 30. \tag{6.29}$$

$$\tag{6.30}$$

Este problema se caracteriza por ser cóncavo y estar desconectado tanto en el conjunto P_{real} como en el conjunto FP_{real} (figura 6.5). La mayor dificultad que presenta KIT es que los miembros de P_{real} se sitúan en la frontera de la zona factible con la no factible.

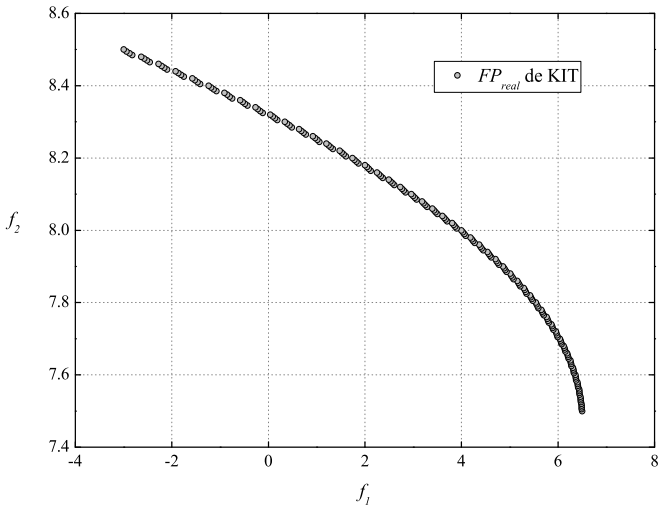


Figura 6.5: Frente de Pareto del problema de maximización KIT.

6.4.2. Problema OSY

Osyczka y Kundu [Osyczka95] propusieron el siguiente problema con seis variables y seis restricciones:

Minimizar $F = (f_1(\vec{x}), f_2(\vec{x}))$, donde

$$f_1(\vec{x}) = -(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2), \quad (6.31)$$
$$f_2(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2. \quad (6.32)$$

Sujeto a

$$0 \leq x_1, x_2, x_6 \leq 10, \quad (6.33)$$
$$1 \leq x_3, x_5 \leq 5, \quad (6.34)$$
$$0 \leq x_4 \leq 6, \quad (6.35)$$

y

$$0 \leq x_1 + x_2 - 2, \quad (6.36)$$
$$0 \leq 6 - x_1 - x_2, \quad (6.37)$$
$$0 \leq 2 - x_2 + x_1, \quad (6.38)$$
$$0 \leq 2 - x_1 + 3x_2, \quad (6.39)$$
$$0 \leq 4 - (x_3 - 3)^2 - x_4, \quad (6.40)$$
$$0 \leq (x_5 - 3)^2 + x_6 - 4 \quad (6.41)$$

Tanto P_{real} como FP_{real} están desconectados en el problema OSY. Su frente de Pareto está compuesto por 5 segmentos (AB, BC, CD, DE, EF), los cuales se sitúan en alguna de las fronteras impuestas por las restricciones (figura 6.6). Este problema es difícil porque requiere que un AEMO mantenga subpoblaciones en las diferentes intersecciones con las fronteras de la zona no factible.

6.4.3. Problema TMK

El siguiente problema con tres objetivos fue propuesto por Tamaki [Tamaki96]:

Maximizar $F = (f_1(x,y,z), f_2(x,y,z), f_3(x,y,z))$, donde

$$f_1(x,y,z) = x, \quad (6.42)$$
$$f_2(x,y,z) = y, \quad (6.43)$$
$$f_3(x,y,z) = z. \quad (6.44)$$
$$(6.45)$$

Sujeto a

$$0 \leq x, y, z \leq 1,$$
$$x^2 + y^2 + z^2 \leq 1.$$

Los elementos del frente de Pareto se localizan en el octante positivo de una esfera de radio uno como se muestra en la figura 6.7.

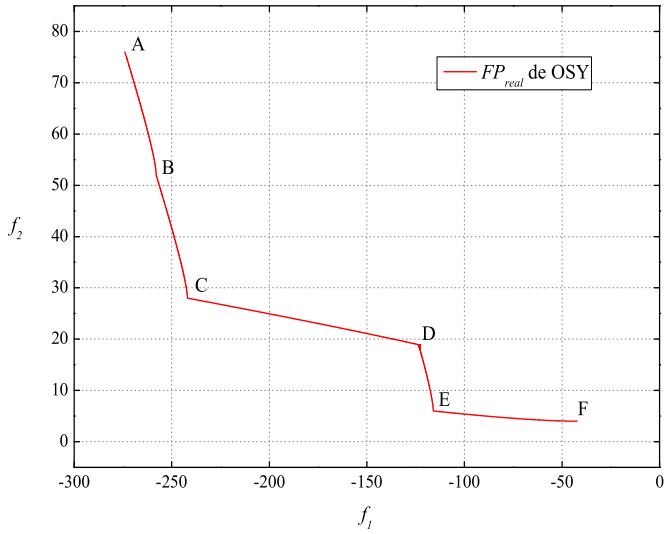


Figura 6.6: Las cinco regiones que componen el frente de Pareto de OSY.

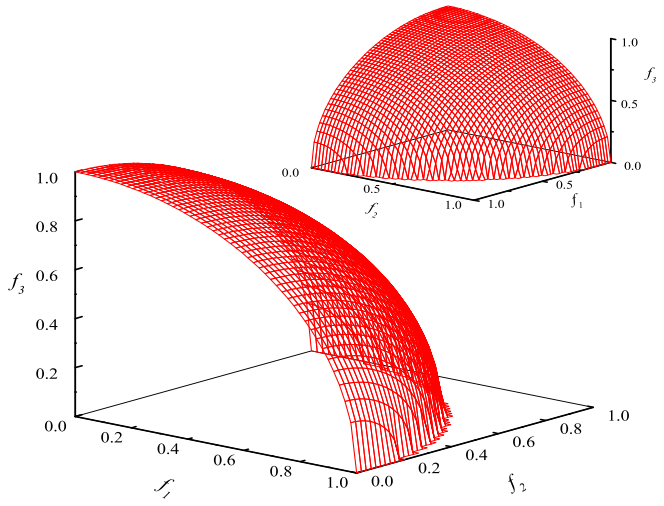


Figura 6.7: El frente de Pareto del problema TMK.

6.5. El pAEMO de referencia

Como un punto de referencia para evaluar el desempeño de MRMOGA se desarrolló una versión paralela del NSGA-II (PNSGA-II). La versión serial es bien conocida, mostrando muy buenos resultados en otras publicaciones. En adición, en [Lucken03] se realizó la paralelización de varios algoritmos de segunda generación, entre ellos NSGA-II, y se llegó a la conclusión de que el más beneficiado por la paralelización es el NSGA-II.

La paralelización de NSGA-II se realizó apegándose al modelo de islas estándar con el fin de observar las ventajas del esquema propuesto en MRMOGA sobre el esquema tradicional de islas. En los párrafos siguientes se describen las características principales de PNSGA-II.

El AEMO que se ejecuta en cada isla es el NSGA-II con representación binaria. Las únicas extensiones que se le hicieron son las operaciones para intercambiar individuos, y el mecanismo para construir el $FP_{conocido}$ a partir de los $FP_{conocido}^{(p)}$ de cada procesador.

En cuanto a la política de migración, se optó por migrar un número, x, de individuos tomados del primer frente de la población ordenada por no dominancia. En la isla receptora se eligen aleatoriamente x individuos que dejarán su lugar a los individuos provenientes de otras islas.

Teniendo en cuenta que las topologías usadas por MRMOGA tienen grado $\delta(i) = 1$ y $\delta(i) = i$ para cada nodo i, se decidió utilizar en PNSGA-II una topología de anillo unidireccional con grado $\delta(i) = 1$, ya que es la más semejante a la topología simple de MRMOGA. En el pseudocódigo mostrado en el algoritmo 6.1 se muestra el esquema básico de PNSGA-II.

```
1:  Parámetros:
        P_i   : población del i-ésimo procesador
        E : número máximo de épocas
        G : número de generaciones por época
        FP_conocido^(i) : frente de Pareto conocido del procesador i
        FP_conocido : frente de Pareto conocido considerando la población global
2:
3:  procedimiento PNSGA-II
4:      Generar aleatoriamente la población inicial P_i en cada procesador i
5:      para e ← 1 hasta E haz
6:          para g ← 1 hasta G haz
7:              NSGA-II(P_i)
8:          fin para
9:          si e ≠ E entonces        ▷ No migrar en la última época
10:             MIGRAR(P_i, P_{i+1})
11:             ASIMILAR(P_i)
12:         fin si
13:     fin para

14:     Combinar todos los conjuntos FP_conocido^(i) en el procesador 0 y presentar FP_conocido
15: fin procedimiento
```

Algoritmo 6.1: Pseudocódigo del algoritmo PNSGA-II.

Para presentar el $FP_{conocido}$ de tamaño N se realiza un procedimiento similar al que usa NSGA-II para elegir una nueva población a partir de la combinación de la población actual y la descendencia. En el caso del PNSGA-II, en el procesador raíz se combinan los $FP_{conocido}^{(p)}$ de cada procesador p. Esta población combinada es ordenada por capas de acuerdo a la no dominancia. Para formar el conjunto $FP_{conocido}$, comenzando por el mejor frente, elegimos con-

Parámetro	valor
Épocas	50
Generaciones por época	10
Tamaño isla	30
Número de migrantes	20
Evaluaciones por isla	15000
Tamaño pobl. final	100
Decimales de precisión	5
Ptje. de cruza	0.9
Ptje. de mutación	$1/\ell_i$

Tabla 6.1: Parámetros utilizados para evaluar las topologías.

secutivamente cada frente hasta que ninguno de ellos pueda ser acomodado completamente sin exceder el tamaño N. Si aún no se completa la población total, se ordena descendentemente el siguiente frente usando el operador $\prec_n$ (descrito en [Deb00]), y se eligen las mejores soluciones para completar $FP_{conocido}$.

6.6. Influencia de la topología

Este primer experimento tiene la finalidad de estudiar la influencia de las topologías simple y completa (véase § 5.4) en el desempeño de MRMOGA. Para tal fin, con cada topología se realizaron 30 ejecuciones del algoritmo con los problemas ZDT1 (sin restricciones) y KIT (con restricciones). Los valores de los parámetros utilizados para ambos problemas se resumen en la tabla 6.1. El porcentaje de mutación, $p_m = 1/\ell_i$, es distinto en cada isla. Aquí ℓ_i es la longitud de las cadenas binarias de acuerdo a la resolución de la isla i. Estos experimentos se realizaron en un cúmulo de computadoras de 16 nodos duales cuyas características se listan en la tabla 6.5.

6.6.1. Evaluación de la topología usando KIT

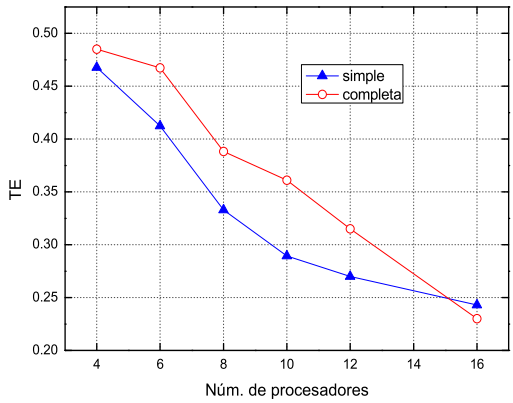
En la tabla 6.2 se presentan los resultados estadísticos de la métricas para MRMOGA usando la topología simple y la compuesta. Al nombre del algoritmo se le pospone *-s* para denotar a la versión con la topología simple, y *-c* a la versión con la topología completa. El número que se pospone significa el número de procesadores utilizados.

En la figura 6.8 se presentan las gráficas de los valores promedio para cada topología en cada una de las métricas. Los conjuntos $FP_{conocido}$ para las topologías utilizando 10 procesadores se muestran en la figura 6.9.

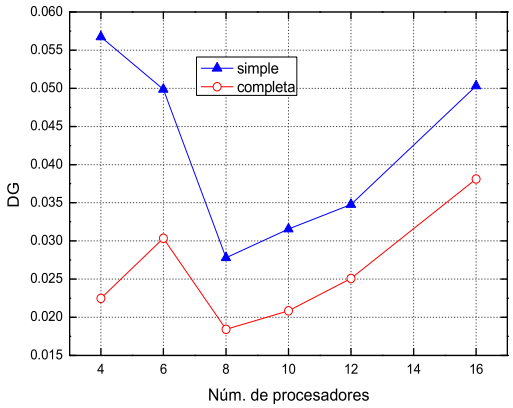
Conforme a lo que podríamos esperar, para ambas topologías la tasa de error disminuye cuando aumentamos el número de procesadores. Sin embargo, contrario a nuestras suposiciones, la TE es mejor con la topología simple para la mayor parte del número de procesadores que con la topología completa. Una observación más detenida nos revela que la TE de la topología simple disminuye (mejora) a un ritmo cada vez menor conforme aumentamos el número de procesadores. En cambio, para la topología completa la TE parece disminuir de manera lineal hasta el punto que llega a ser mejor que la TE de la topología simple cuando usan 16 procesadores (v. figura 6.8(a)). Una posible razón para este comportamiento es que las soluciones óptimas encontradas en la isla con menor resolución tardan más en propagarse hasta llegar a la isla con resolución máxima cuando hay más islas intermedias. De este modo, la labor que tienen las islas de menor resolución de aproximarse rápidamente al frente de Pareto pierde efectividad. En

Algoritmo	TE			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	0.467667	0.28	0.63	0.066317
MRMOGA-s-6	0.412333	0.26	0.67	0.098613
MRMOGA-s-8	0.333000	0.21	0.51	0.064039
MRMOGA-s-10	0.289333	0.18	0.46	0.073163
MRMOGA-s-12	0.270000	0.17	0.50	0.082179
MRMOGA-s-16	0.243000	0.12	0.36	0.063095
MRMOGA-c-4	0.485000	0.29	0.69	0.094154
MRMOGA-c-6	0.467333	0.34	0.62	0.080122
MRMOGA-c-8	0.388333	0.27	0.50	0.063984
MRMOGA-c-10	0.361000	0.19	0.56	0.074760
MRMOGA-c-12	0.315000	0.21	0.47	0.065815
MRMOGA-c-16	0.230000	0.14	0.35	0.056451
Algoritmo	**DG**			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	0.056741	0.002613	0.190318	0.056031
MRMOGA-s-6	0.049850	0.002631	0.336114	0.083525
MRMOGA-s-8	0.027803	0.002444	0.155830	0.036308
MRMOGA-s-10	0.031564	0.002382	0.252013	0.055568
MRMOGA-s-12	0.034773	0.002492	0.168147	0.042364
MRMOGA-s-16	0.050304	0.002609	0.217912	0.057703
MRMOGA-c-4	0.022481	0.002469	0.150895	0.034747
MRMOGA-c-6	0.030355	0.002612	0.082520	0.025450
MRMOGA-c-8	0.018414	0.002515	0.115899	0.026291
MRMOGA-c-10	0.020847	0.002631	0.120682	0.033819
MRMOGA-c-12	0.025077	0.002602	0.185551	0.039377
MRMOGA-c-16	0.038116	0.002444	0.218438	0.048202
Algoritmo	**SP**			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	0.273610	0.042776	1.274420	0.343587
MRMOGA-s-6	0.158841	0.040552	1.181640	0.219713
MRMOGA-s-8	0.215520	0.041184	1.537760	0.334829
MRMOGA-s-10	0.083690	0.037629	0.486861	0.084503
MRMOGA-s-12	0.253390	0.039805	1.631710	0.391623
MRMOGA-s-16	0.246672	0.042608	1.550230	0.341890
MRMOGA-c-4	0.135181	0.040399	0.765176	0.196537
MRMOGA-c-6	0.124127	0.043269	0.524229	0.136289
MRMOGA-c-8	0.087367	0.044427	0.339432	0.066553
MRMOGA-c-10	0.079978	0.036793	0.473766	0.082484
MRMOGA-c-12	0.097862	0.039950	0.795969	0.148329
MRMOGA-c-16	0.205071	0.043722	1.123980	0.281781
Algoritmo	**Tiempo**			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	0.748579	0.637463	0.837690	0.050105
MRMOGA-s-6	0.825384	0.734822	0.890035	0.038405
MRMOGA-s-8	0.682234	0.635146	0.793570	0.043579
MRMOGA-s-10	0.869378	0.798618	0.953635	0.039689
MRMOGA-s-12	0.913902	0.834965	1.055060	0.058888
MRMOGA-s-16	0.992931	0.901707	1.077470	0.036760
MRMOGA-c-4	1.011545	0.760614	1.573550	0.150400
MRMOGA-c-6	1.038223	0.909668	1.146320	0.055281
MRMOGA-c-8	1.222576	1.130540	1.385100	0.064623
MRMOGA-c-10	1.532228	1.425370	1.764480	0.072946
MRMOGA-c-12	1.796343	1.637290	2.484540	0.135356
MRMOGA-c-16	1.835220	1.75895	2.28357	0.091631058

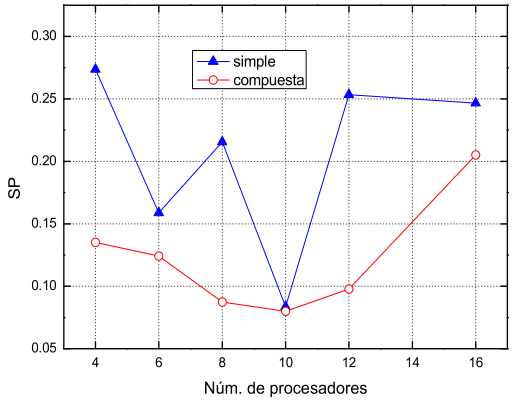
Tabla 6.2: Influencia de la topología para el problema KIT.



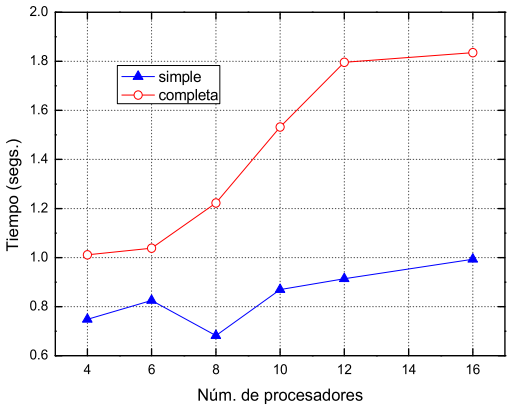
(a) Impacto de la topología en la TE.



(b) Impacto de la topología en la DG.

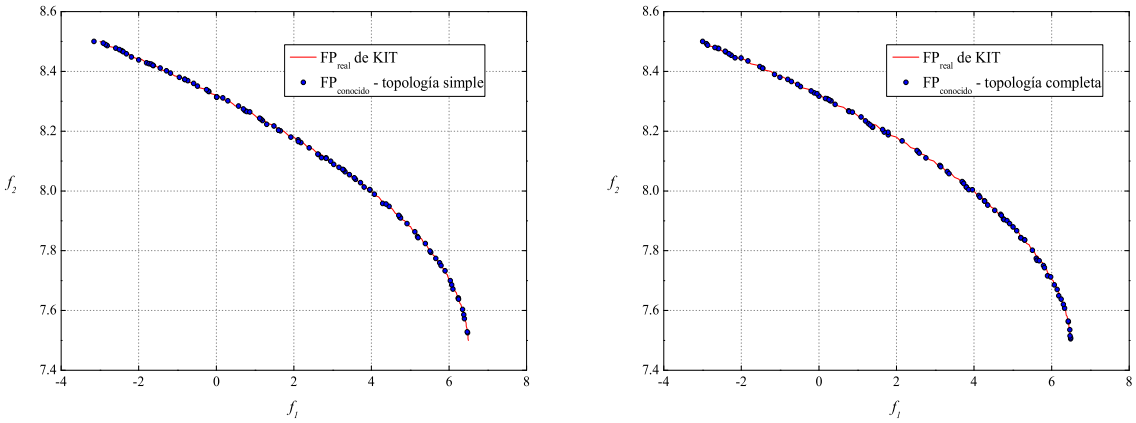


(c) Impacto de la topología en el SP.



(d) Impacto de la topología en el tiempo.

Figura 6.8: Impacto de la topología en la eficacia de MRMOGA (POM KIT).



(a) $FP_{conocido}$ con la topología simple.

(b) $FP_{conocido}$ con la topología completa.

Figura 6.9: Frentes de Pareto encontrados por la topología simple y por la topología completa usando 10 procesadores.

cambio, MRMOGA con la topología completa no es afectado por el número de procesadores ya que la isla con mayor resolución tiene comunicación directa las islas de menor resolución.

Con respecto a la distancia generacional, la topología completa parece tener mejor convergencia que la topología simple, lo cual contradice el resultado obtenido con la TE. Esta inconsistencia generalmente se explica porque a pesar de que varias soluciones no pertenecen a FP_{real} (produciendo un valor grande en TE), están muy cercanos al frente, lo cual produce un valor pequeño en DG. Cabe destacar que ambas topologías tienen un comportamiento similar en sus resultados de DG: el valor de DG disminuye hasta llegar a un mínimo cuando se usan 8 procesadores, y a partir de este punto comienza aumentar.

Por lo que toca a la S, la topología completa genera una mejor distribución de las soluciones sobre el frente de Pareto. Finalmente, puesto que en la topología completa hay más comunicaciones entre procesos, ésta consume más tiempo que la topología simple.

6.6.2. Evaluación de la topología usando ZDT1

En la tabla 6.3 se presentan los resultados de las métricas usando la topología simple y la compuesta. En la figura 6.10 se presentan las gráficas de los valores promedio para cada topología en cada una de las métricas. Los conjuntos $FP_{conocido}$ para las topologías utilizando 10 procesadores se muestran la figura 6.11.

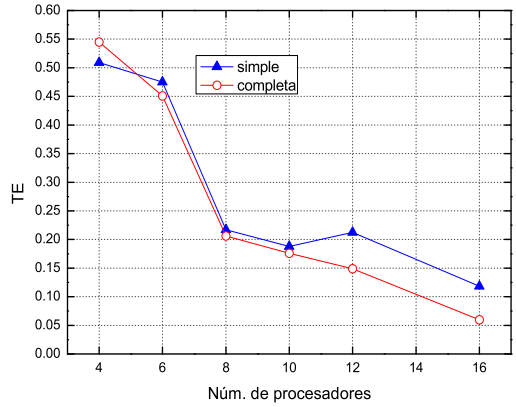
A diferencia de lo que ocurrió en el problema KIT, en el problema ZDT1 la topología completa obtuvo los mejores valores de TE para la mayor parte del número de procesadores, aunque de manera marginal. Además, juzgando los valores de TE (véase figura 6.10(a)), se confirma que la topología completa tiene mejor convergencia que la topología simple cuando se usan más de 12 procesadores.

La convergencia, de acuerdo con la DG, es muy semejante para ambas topologías. Es importante resaltar que el comportamiento de los resultados de DG es similar al que se observó en el problema KIT. Es decir, el valor de DG desciende hasta alcanzar un mínimo, y posteriormente comienza a aumentar.

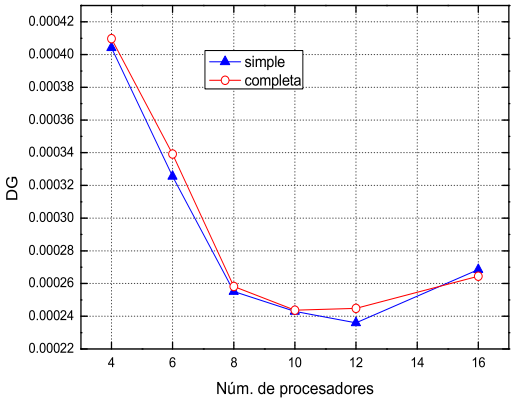
De los resultados de la figura 6.10 se confirma que la topología completa produce una mejor

Algoritmo	TE			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	0.509000	0.27	0.86	0.174363
MRMOGA-s-6	0.475000	0.19	0.93	0.203400
MRMOGA-s-8	0.217000	0.09	0.45	0.073944
MRMOGA-s-10	0.188000	0.07	0.46	0.083403
MRMOGA-s-12	0.212333	0.06	0.37	0.079611
MRMOGA-s-16	0.118333	0.01	0.30	0.068366
MRMOGA-c-4	0.545000	0.28	0.95	0.180458
MRMOGA-c-6	0.450667	0.18	0.80	0.167351
MRMOGA-c-8	0.206000	0.10	0.37	0.063277
MRMOGA-c-10	0.176000	0.08	0.46	0.082648
MRMOGA-c-12	0.149000	0.08	0.25	0.040029
MRMOGA-c-16	0.059667	0.01	0.12	0.026011
Algoritmo	**DG**			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	0.000404	0.000183	0.000722	0.000119
MRMOGA-s-6	0.000326	0.000126	0.000572	0.000099
MRMOGA-s-8	0.000255	0.000180	0.000352	0.000044
MRMOGA-s-10	0.000243	0.000149	0.000335	0.000048
MRMOGA-s-12	0.000236	0.000147	0.000312	0.000040
MRMOGA-s-16	0.000269	0.000184	0.000391	0.000052
MRMOGA-c-4	0.000410	0.000235	0.000687	0.000131
MRMOGA-c-6	0.000339	0.000169	0.000460	0.000079
MRMOGA-c-8	0.000258	0.000156	0.000329	0.000045
MRMOGA-c-10	0.000244	0.000155	0.000378	0.000056
MRMOGA-c-12	0.000245	0.000147	0.000325	0.000046
MRMOGA-c-16	0.000265	0.000172	0.000350	0.000035
Algoritmo	**SP**			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	0.008741	0.007219	0.010427	0.000709
MRMOGA-s-6	0.009004	0.007685	0.010275	0.000644
MRMOGA-s-8	0.009626	0.008136	0.011222	0.000703
MRMOGA-s-10	0.009225	0.007744	0.011344	0.000855
MRMOGA-s-12	0.008978	0.007829	0.010786	0.000663
MRMOGA-s-16	0.009536	0.007945	0.011806	0.000883
MRMOGA-c-4	0.009095	0.008067	0.010837	0.000627
MRMOGA-c-6	0.008617	0.007278	0.009682	0.000550
MRMOGA-c-8	0.008938	0.007652	0.010652	0.000809
MRMOGA-c-10	0.008748	0.007045	0.010645	0.000869
MRMOGA-c-12	0.008777	0.007431	0.009968	0.000737
MRMOGA-c-16	0.009822	0.007158	0.010919	0.000843
Algoritmo	**Tiempo**			
	Promedio	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-4	1.774925	1.702620	1.882390	0.042230
MRMOGA-s-6	1.296269	1.246950	1.372530	0.028332
MRMOGA-s-8	1.308544	1.266860	1.342180	0.015867
MRMOGA-s-10	1.349987	1.307670	1.489270	0.033440
MRMOGA-s-12	1.862288	1.780930	1.937840	0.045766
MRMOGA-s-16				
MRMOGA-c-4	1.770902	1.699370	1.827980	0.041816
MRMOGA-c-6	1.500502	1.425140	1.952690	0.094247
MRMOGA-c-8	1.593870	1.549010	1.798020	0.043320
MRMOGA-c-10	1.804069	1.716390	2.526000	0.137739
MRMOGA-c-12	2.958858	2.868370	3.066040	0.045936
MRMOGA-c-16				

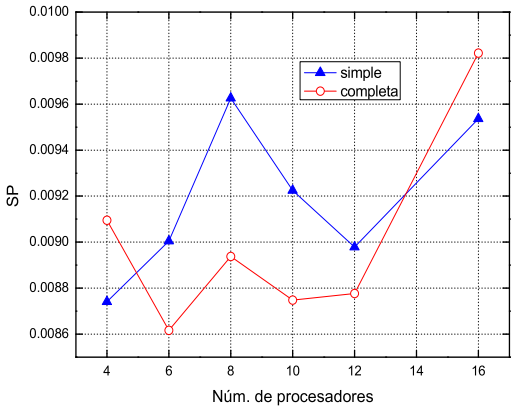
Tabla 6.3: Influencia de la topología para el problema ZDT1.



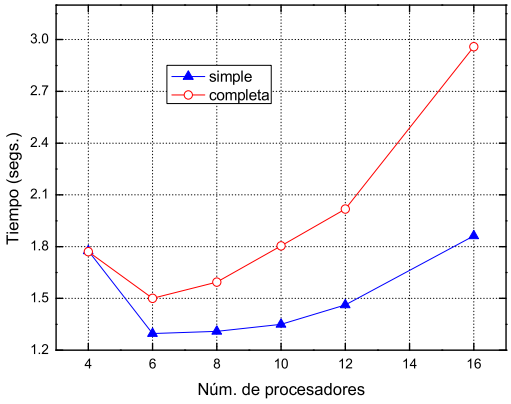
(a) Impacto de la topología en la TE.



(b) Impacto de la topología en la DG.



(c) Impacto de la topología en el SP.



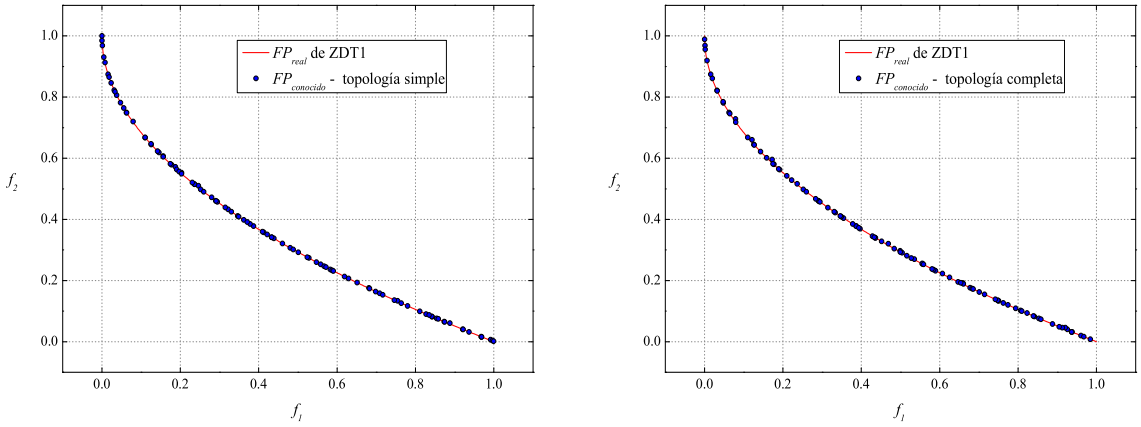
(d) Impacto de la topología en el tiempo.

Figura 6.10: Impacto de la topología en la eficacia de MRMOGA (POM ZDT1).

distribución de la soluciones sobre las regiones que ocupa. También podemos observar que hay un punto donde se alcanza una mejor distribución y después comienza a deteriorarse cuando aumentamos el número de procesadores.

6.6.3. Conclusiones sobre la topología

Con los resultados obtenidos en los dos problemas, no es claro determinar un ganador en la convergencia, ya que ninguna topología fue superior en la métrica TE en ambos problemas. Sin embargo, con base en la distancia generacional podemos decir que la topología completa se acerca más rápido al frente de Pareto ya que ambos problemas la topología completa fue superior en esta métrica. No obstante, es importante notar que la TE de la topología simple, aunque siempre mejora, la ganancia es cada vez menor, mientras que con la topología completa la TE parece mejorar linealmente (figuras 6.10(a) y 6.8(a)). Esto se debe, posiblemente, a que en la topología simple, las soluciones óptimas (encontradas rápidamente) de la isla de menor resolución tardan



(a) $FP_{conocido}$ con la topología simple.

(b) $FP_{conocido}$ con la topología completa.

Figura 6.11: Frentes de pareto encontrados por la topología simple y por la topología completa usando 10 procesadores (POM ZDT1).

más generaciones en llegar a la isla de mayor resolución. Respecto a la distribución, la topología completa tiene mucho mejor desempeño que la topología simple.

El desempeño de topología completa se puede mejorar si se logra mejorar la convergencia cuando se usa un número pequeño de procesadores. Por otro lado, el desempeño de la topología simple se mejoraría si se aumenta su capacidad para distribuir las soluciones.

6.7. Resultados experimentales con relación a la eficacia

6.7.1. Metodología para evaluar la eficacia

Con el fin de estudiar la ganancia en la eficacia que se obtiene con el paralelismo, para cada uno de los problemas de prueba se realizaron 30 ejecuciones de los algoritmos con diferente número de procesadores (1, 4, 8, 12 y 16). De la misma manera que se evalúa la eficacia de los AEMOs, para cada una de las ejecuciones se mantuvo fijo el número de evaluaciones de cada isla. Para cada número de procesadores se evaluará el desempeño de cada algoritmo en dos aspectos: la convergencia (conteo de aciertos, distancia generacional invertida) y la distribución de las soluciones (espaciamiento). Para los resultados correspondientes a las 30 ejecuciones se calcularon los siguientes valores estadísticos: promedio, máximo, mínimo, y desviación estándar. La métrica de cobertura $\mathcal{C}$ se calculó tomando como conjunto no dominado a la unión de los conjuntos $FP_{conocido}$ que reportó cada algoritmo en todas sus ejecuciones. Además de utilizar estas métricas nos apoyaremos de comparaciones visuales para verificar la confiabilidad de las métricas.

Con el fin de realizar una comparación justa, independientemente del problema de prueba se utilizaron los mismos parámetros en los algoritmos. En virtud de que el desempeño de los algoritmos es muy sensible al tamaño de la isla, la longitud de la época y al número de migrantes, estos parámetros se determinaron experimentalmente para cada algoritmo. Para ambos algoritmos se realizaron 10 de corridas para el problema ZDT1 variando estos parámetros de manera que el número de evaluaciones fuera constante. Al final, se eligieron los parámetros que pro-

	MRMOGA	PNSGA-II
Parámetro	**valor**	**valor**
Épocas	60	24
Generaciones por época	10	15
Tamaño isla	30	50
Número de migrantes	20	25
Evaluaciones por isla	18000	18000
Tamaño pobl. final	100	100
Decimales de precisión	5	5
Ptje. de cruza	0.9	0.9
Ptje. de mutación	$1/\ell_i$	$1/k$

Tabla 6.4: Parámetros utilizados en las ejecuciones de los algoritmos

Característica	**Descripción**
CPU	Intel Xeon; 2.45MHz
Núm. de nodos	16 (2 procesadores c/u)
Memoria	2 GB por nodo
Sistema operativo	Red Hat Linux 3.2.2-5
Red de comunicaciones	FastEthernet
Biblioteca de comunicaciones	MPICH 1.12

Tabla 6.5: Características del sistema de cúmulo utilizado.

porcionaron mejores resultados para la tasa de error. En la tabla 6.4 se resumen los parámetros utilizados para todos los problemas.

Para NSGA-II, k es la longitud de los cromosomas necesaria para representar las soluciones con la precisión deseada. Para MRMOGA, ℓ_i es la longitud de los cromosomas de acuerdo con la resolución de cada isla. Para la isla p, con mayor resolución tenemos que $\ell_p = k$.

La implementación de PNSGA-II se realizó a partir del código proporcionado en C por Deb[2]. La implementación de MRMGOA está escrita en lenguaje C++. Para la comunicación entre procesos ambos algoritmos utilizan la biblioteca MPICH. Desde luego, MRMOGA utiliza la versión para C++ y PNSGA-II la versión para C. La ejecución de ambos algoritmos se realizó en un cúmulo de 16 nodos duales con las características de la tabla 6.5.

6.7.2. Resultados y discusión para el problema DEB

En la tabla 6.6 se presentan los resultados estadísticos de las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo[3]. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.7. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para DEB se muestran en las figuras 6.13 y 6.14, respectivamente.

En las figuras 6.13(a) y 6.14(a) observamos que ambos algoritmos quedan lejos del frente de Pareto real usando un solo procesador. Sin embargo, PNSGA-II tiene mucho mejor convergencia que MRMOGA si consideramos los valores del conteo de aciertos que obtuvieron ambos algo-

[2]El código del NSGA-II está disponible en http://www.iitk.ac.in/kangal/soft.htm.

[3]Por cuestiones de espacio los resultados del tiempo se incluyen en esta tabla, pero no serán discutidos hasta la sección 6.10.2

ritmos (véase figura 6.12(a)). Es importante notar que cuando se utiliza más de un procesador, MRMOGA mejora notablemente su convergencia hasta el punto de que, en promedio, más del 98 % de $FP_{conocido}$ pertenece al conjunto FP_{real}, y en sus mejores resultados todo miembro de $FP_{conocido}$ pertenece al FP_{real} (valores promedio y mejor de métrica CA mostrados en la tabla 6.6). PNSGA-II muestra una convergencia similar a la de MRMOGA cuando se usa más de un procesador. A pesar de que los valores de SP muestran una ventaja de PNSGA-II sobre MRMOGA con respecto a la distribución, observando los conjuntos $PF_{conocido}$ que consiguió MRMOGA con 8 y 16 procesadores (figuras 6.13(b) y 6.13(c)) podemos afirmar que su distribución es competitiva.

La tabla 6.7 muestra que PNSGA-II-n ($n = 1, \ldots, 16$) domina a lo sumo 15 % de los frentes generados por MRMOGA en su mejor resultado (obtenido con 16 procesadores). En cambio, MRMOGA-n ($n = 1, \ldots, 16$) cubre a lo más el 4 % de los frentes de PNSGA-II en su mejor resultado (obtenido con 16 procesadores).

6.7.3. Resultados y discusión para el problema KUR

En la tabla 6.8 se presentan los resultados estadísticos correspondientes a las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.9. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para KUR se muestran en las figuras 6.16 y 6.17, respectivamente.

Con menos de 12 procesadores, PNSGA-II obtiene los mejores resultados para la métrica CA, pero no es suficiente para afirmar que tiene mejor convergencia que MRMOGA. Para cualquier número de procesadores MRMOGA tuvo ampliamente los mejores valores para la métrica DGI. Esto nos revela que MRMOGA se aproxima a FP_{real} con mayor rapidez que PNSGA-II (en términos del número generaciones). Esta situación se aprecia en las figuras 6.16(a) y 6.17(a). En estas figuras se observa que mientras que todas las soluciones de MRMOGA están próximas al frente de Pareto, algunas de la soluciones de PNSGA-II aún se encuentran alejadas. Además, cabe mencionar que la convergencia de MRMOGA mejoró siempre que se aumentaba el número de procesadores como se aprecia en la figura 6.15(a). Por el contrario, con base en la métrica AC, la convergencia de NSGA-II alcanzó su máximo alrededor de 4 procesadores y después comenzó a disminuir. Los resultados de la métrica $\mathcal{C}$ concuerdan con los resultados de la métrica CA, en el sentido de que la fracción de soluciones no dominadas cubiertas de PNSGA-II aumenta a partir de 4 procesadores, y el porcentaje de vectores cubiertos de MRMOGA disminuye conforme se utilizan más procesadores. Sin embargo, PNSGA-II-n ($n = 1, .., 16$) cubre un porcentaje mayor de soluciones no dominadas de MRMOGA-S-n ($n = 1, .., 16$) que este de aquel. En esta caso, no tenemos una clara evidencia de que un algoritmo supere al otro en función de la convergencia, aunque MRMOGA parece mejorar su convergencia consistentemente.

Con referencia a la distribución es importante notar que MRMOGA cubre todas las regiones que componen el frente de Pareto real de KUR. En cambio, independientemente del número de procesadores, PNSGA-II no es capaz de cubrir, totalmente, la región inferior derecha de FP_{real} ni tampoco el vector $(-20, 0)$. La métrica DGI refleja esta situación. Los vectores de las regiones del FP_{real} que no cubre PNSGA-II están más alejados de los vectores más cercanos de su $FP_{conocido}$, lo cual produce peores valores de DGI que los obtenidos por MRMOGA, que sí cubre todas las regiones del frente de Pareto real.

Desde el punto de vista de la métrica SP, PNSGA-II tiene una mejor distribución, sin embargo, debido a la falta de normalización de la distancia ocupada en al métrica SP (desventaja descrita en § 6.2.6), los valores que arroja no son confiables. En este caso, los elementos del con-

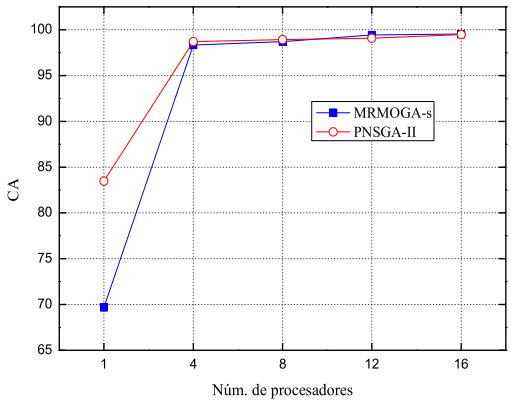
	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	69.666667	**100**	5	37.310707
MRMOGA-s-4	98.333333	100	95	1.468181
MRMOGA-s-8	98.700000	100	92	1.530795
MRMOGA-s-12	99.433333	100	**97**	0.919541
MRMOGA-s-16	**99.533333**	100	96	0.884433
PNSGA-II-1	83.466667	**100**	0	16.223303
PNSGA-II-4	98.700000	100	96	1.294862
PNSGA-II-8	98.933333	100	**97**	0.853750
PNSGA-II-12	99.066667	100	96	0.891939
PNSGA-II-16	99.466667	100	97	**0.805536**

	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.002702	0.001038	0.007647	0.002112
MRMOGA-s-4	0.001200	0.000938	0.001473	0.000116
MRMOGA-s-8	0.001204	0.001034	0.001492	0.000114
MRMOGA-s-12	0.001240	0.001045	0.001456	0.000113
MRMOGA-s-16	0.001258	0.001068	0.001519	0.000129
PNSGA-II-1	0.002683	0.001619	0.009882	0.002437
PNSGA-II-4	**0.000811**	**0.000715**	**0.000890**	**0.000038**
PNSGA-II-8	0.000937	0.000775	0.001199	0.000092
PNSGA-II-12	0.001142	0.000987	0.001641	0.000153
PNSGA-II-16	0.001219	0.000931	0.001562	0.000171

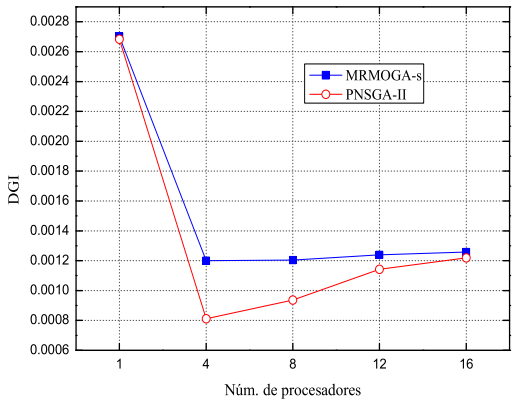
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.216070	0.030220	0.704670	0.198410
MRMOGA-s-4	0.118723	0.034288	1.053960	0.216955
MRMOGA-s-8	0.113698	0.034040	1.073050	0.212136
MRMOGA-s-12	0.111172	0.038678	0.511914	0.156562
MRMOGA-s-16	0.098691	0.030546	0.735538	0.163700
PNSGA-II-1	0.085147	0.067447	0.152537	0.021692
PNSGA-II-4	0.036619	0.031239	0.042630	**0.002918**
PNSGA-II-8	0.035358	0.028991	**0.041680**	0.003072
PNSGA-II-12	0.032763	**0.024848**	0.043073	0.004543
PNSGA-II-16	**0.032523**	0.025394	0.043632	0.004238

	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	**0.608534**	**0.285076**	1.075550	0.242217
MRMOGA-s-4	0.753840	0.725686	0.778144	0.014382
MRMOGA-s-8	0.781777	0.750952	0.835837	0.016974
MRMOGA-s-12	0.853921	0.808907	0.966912	0.032806
MRMOGA-s-16	0.882182	0.844137	0.931682	0.021062
PNSGA-II-1	0.637567	0.635499	**0.639555**	**0.000972**
PNSGA-II-4	0.656929	0.649466	0.693031	0.011566
PNSGA-II-8	0.669747	0.658574	0.700496	0.010847
PNSGA-II-12	0.679599	0.670747	0.712364	0.009279
PNSGA-II-16	0.708773	0.686080	0.756914	0.018847

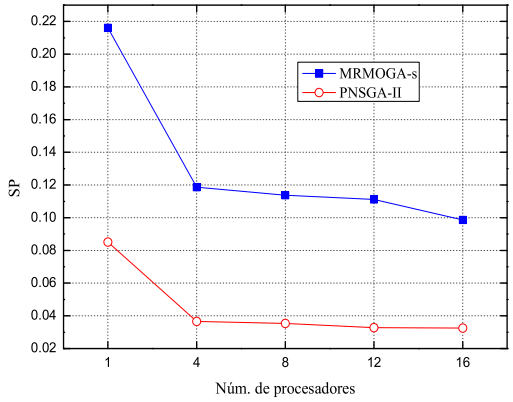
Tabla 6.6: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema DEB (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.

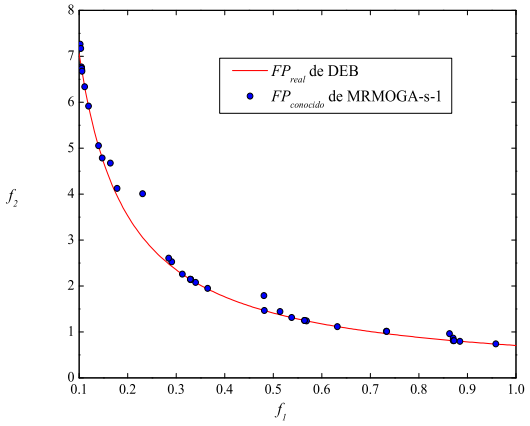


(c) Comparación con relación a SP.

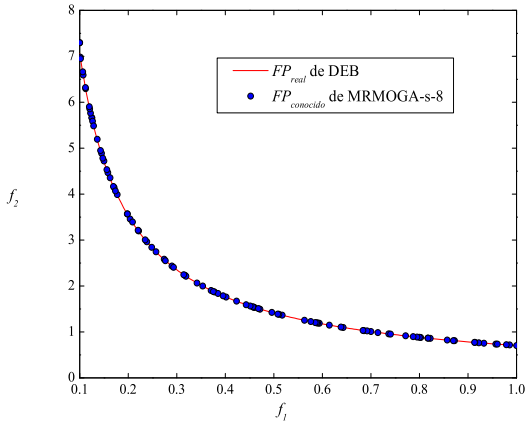
Figura 6.12: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema DEB).

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.270667	0.030333	0.034000	0.027000	0.029333
MRMOGA-S-4	0.288667	0.040333	0.036667	0.033000	0.034000
MRMOGA-S-8	0.284667	0.037333	0.036333	0.032000	0.032000
MRMOGA-S-12	0.286667	0.046333	0.037667	0.034333	0.038667
MRMOGA-S-16	0.293333	0.043333	0.039667	0.038000	0.037000
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	0.455124	0.181667	0.220667	0.086000	0.081000
PNSGA-II-4	0.589399	0.292000	0.352333	0.161333	0.144000
PNSGA-II-8	0.607067	0.295333	0.351333	0.163000	0.144000
PNSGA-II-12	0.591873	0.293667	0.357667	0.155333	0.135333
PNSGA-II-16	0.597527	0.293667	0.343000	0.157667	0.149000

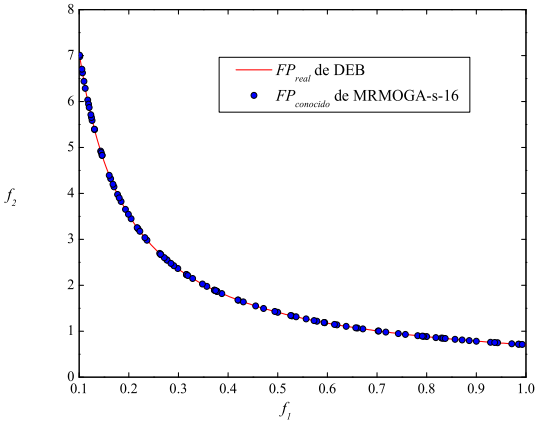
Tabla 6.7: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema DEB).



(a) $FP_{conocido}$ obtenido con un procesador.

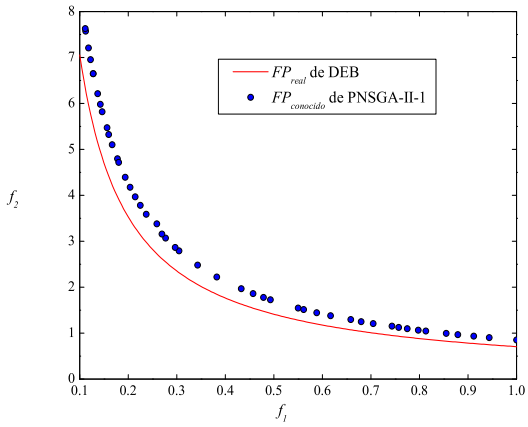


(b) $FP_{conocido}$ obtenido con 8 procesadores.

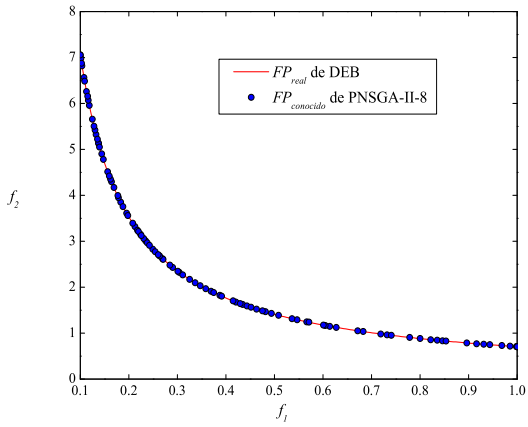


(c) $FP_{conocido}$ obtenido con 16 procesadores.

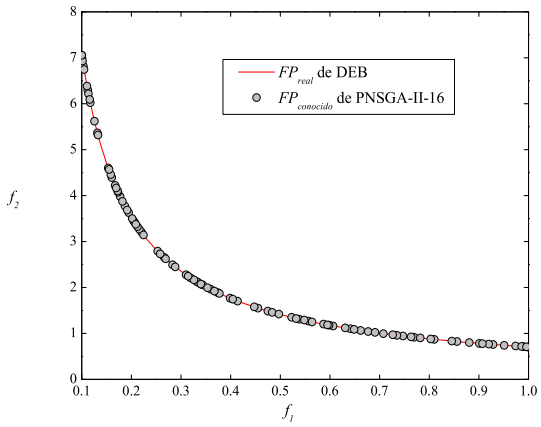
Figura 6.13: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema DEB.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.14: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema DEB.

junto $FP_{conocido}$ generado por PNSGA-II se localizan principalmente en dos regiones del conjunto FP_{real}, provocando que las distancias entre ellos sean más pequeñas que en una situación donde las soluciones estuvieran esparcidas a lo largo de todo el frente. Aunado a esto, la distancia del vector $(-20, 0)$ a su punto más cercano es mucho mayor que la distancia promedio entre los demás puntos del frente de Pareto real. Puesto que el PNSGA-II no encuentra este vector, el valor de SP no se ve afectado por esta distancia.

6.7.4. Resultados y discusión para el problema ZDT1

En la tabla 6.10 se presentan los resultados estadísticos correspondientes a las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.11. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para ZDT1 se muestran en las figuras 6.19 y 6.20, respectivamente.

A partir de los resultados obtenidos para CA mostrados en la tabla 6.10 y de la comparación gráfica de la figura 6.18(a) es claro que MRMOGA tiene mucho mejor convergencia que PNSGA-II cuando se utiliza más de un procesador. Asimismo, se observa claramente que MRMOGA supera a PNSGA-II en la métrica $\mathcal{C}$. PNSGA-II-n ($n = 1, \ldots, 16$) cubre, a lo sumo, el 42 % de los frentes no dominados de MRMGOA-s-4, y, en contraste, MRMOGA-s-n ($n = 1, \ldots, 16$) cubre a lo más el 50 % de los frentes no dominados de PNSGA-II-16 (su mejor resultado). Una diferencia mayor se observa cuando MRMOGA usa 16 procesadores, puesto que PNSGA-II-n ($n = 1, \ldots, 16$) domina menos del 6 % de sus frentes no dominados. La diferencia de la convergencia de los algoritmos es imperceptible observando simplemente los conjuntos $FP_{conocidos}$ de cada algoritmo. Sin embargo, la ampliación de la región del frente mostrada en las figuras 6.19(b) y 6.20(b) facilita la percepción de esta diferencia.

En cuanto a la distribución, ambos algoritmos tiene una distribución con un comportamiento similar, aunque PNSGA-II es superior.

6.7.5. Resultados y discusión para el problema ZDT2

En la tabla 6.12 se presentan los resultados estadísticos de las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.13. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para ZDT2 se muestran en las figuras 6.22 y 6.23, respectivamente.

ZDT2 es un problema ligeramente más difícil y, al igual que ZDT1, tiene 30 variables, pero su FP_{real} es cóncavo. El desempeño que mostraron ambos algoritmos en este problema es semejante al anterior. No obstante, debemos notar que de acuerdo con el conteo de aciertos (tabla 6.12), PNSGA-II empeoró su convergencia con respecto a la que obtuvo en el problema ZDT1. MRMOGA, por su parte, consiguió mejores resultados en este problema que en ZDT1. Asimismo, los resultados de la métrica $\mathcal{C}$ son similares a los del problema anterior, pero en este caso, MRMOGA supera a PNSGA-II con mayor margen.

6.7.6. Resultados y discusión para el problema ZDT3

En la tabla 6.14 se presentan los resultados estadísticos de las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$

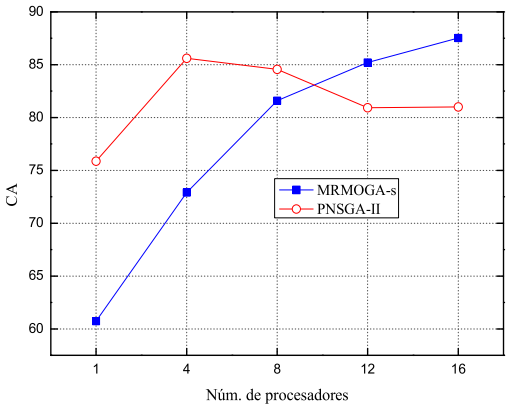
	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	60.733333	75	10	11.212889
MRMOGA-s-4	72.900000	84	61	5.497575
MRMOGA-s-8	81.600000	90	75	3.647830
MRMOGA-s-12	85.200000	94	78	3.290390
MRMOGA-s-16	**87.533333**	94	**82**	**3.073905**
PNSGA-II-1	75.866667	92	0	8.872930
PNSGA-II-4	85.600000	**98**	76	6.194621
PNSGA-II-8	84.566667	96	72	6.453337
PNSGA-II-12	80.933333	96	67	7.192280
PNSGA-II-16	81.000000	94	69	5.573748

	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.007594	0.005086	0.044635	0.006925
MRMOGA-s-4	**0.004672**	0.004089	**0.005472**	**0.000369**
MRMOGA-s-8	0.004984	0.004109	0.006113	0.000549
MRMOGA-s-12	0.005018	0.004331	0.006199	0.000502
MRMOGA-s-16	0.004907	**0.004009**	0.006023	0.000496
PNSGA-II-1	0.096375	0.039585	0.153309	0.035397
PNSGA-II-4	0.079186	0.022501	0.127043	0.042256
PNSGA-II-8	0.062758	0.016109	0.123329	0.039157
PNSGA-II-12	0.044521	0.019092	0.119843	0.027858
PNSGA-II-16	0.040100	0.019168	0.121190	0.022646

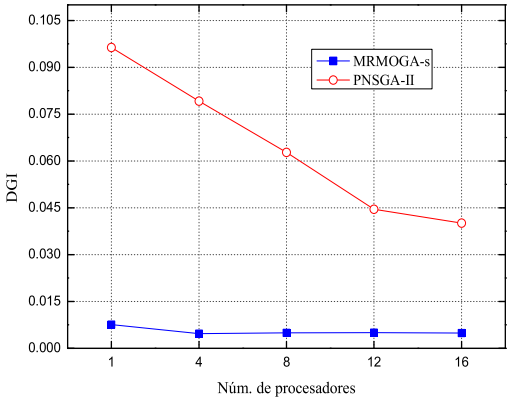
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.122167	0.075497	0.317227	0.043239
MRMOGA-s-4	0.107519	0.075138	0.132337	0.014776
MRMOGA-s-8	0.107217	0.065641	0.136359	0.019409
MRMOGA-s-12	0.107053	0.068414	0.128528	0.018248
MRMOGA-s-16	0.113396	0.076215	0.130725	**0.013317**
PNSGA-II-1	0.088634	0.049994	0.341968	0.049280
PNSGA-II-4	**0.047684**	0.031225	**0.082654**	0.015841
PNSGA-II-8	0.050488	**0.028081**	0.098338	0.015847
PNSGA-II-12	0.058623	0.030809	0.100757	0.017907
PNSGA-II-16	0.068903	0.035352	0.143970	0.023094

	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	**0.757855**	**0.658861**	**0.873471**	0.048196
MRMOGA-s-4	1.006982	0.946815	1.064270	0.032315
MRMOGA-s-8	1.100211	1.043650	1.168340	0.028814
MRMOGA-s-12	1.147425	1.103210	1.210340	0.025694
MRMOGA-s-16	1.215093	1.168990	1.280460	0.029147
PNSGA-II-1	1.000883	0.998688	1.004186	**0.001215**
PNSGA-II-4	1.034225	1.011798	1.092176	0.019828
PNSGA-II-8	1.042738	1.020811	1.119928	0.023136
PNSGA-II-12	1.050297	1.035069	1.069320	0.009607
PNSGA-II-16	1.077966	1.050244	1.229250	0.044066

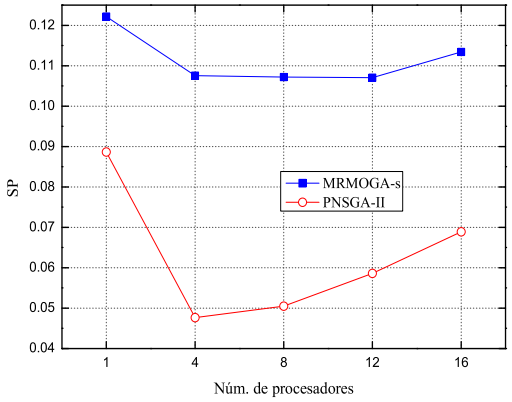
Tabla 6.8: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema KUR (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.

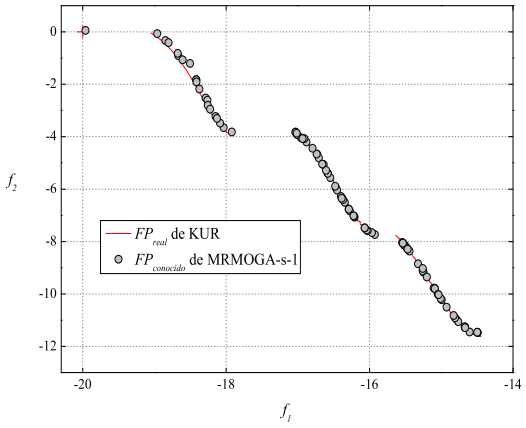


(c) Comparación con relación a SP.

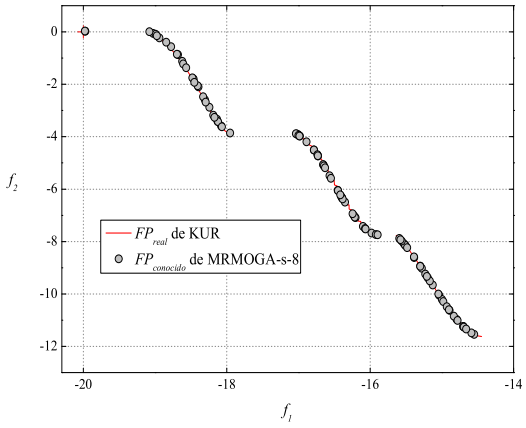
Figura 6.15: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema KUR).

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.394667	0.270000	0.277667	0.309667	0.306667
MRMOGA-S-4	0.416000	0.280000	0.286000	0.318667	0.313000
MRMOGA-S-8	0.454667	0.357000	0.357000	0.385333	0.384000
MRMOGA-S-12	0.463333	0.350000	0.369000	0.379000	0.385667
MRMOGA-S-16	0.474667	0.374333	0.389000	0.404000	0.405333
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	0.505333	0.453333	0.373000	0.356667	0.321667
PNSGA-II-4	0.587000	0.542667	0.482667	0.470667	0.444333
PNSGA-II-8	0.610667	0.557667	0.499000	0.492333	0.448333
PNSGA-II-12	0.625333	0.573000	0.506000	0.501000	0.456000
PNSGA-II-16	0.644667	0.594333	0.527333	0.513667	0.474667

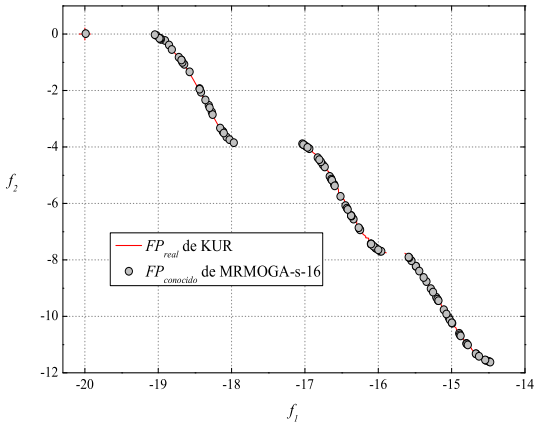
Tabla 6.9: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema KUR).



(a) $FP_{conocido}$ obtenido con un procesador.

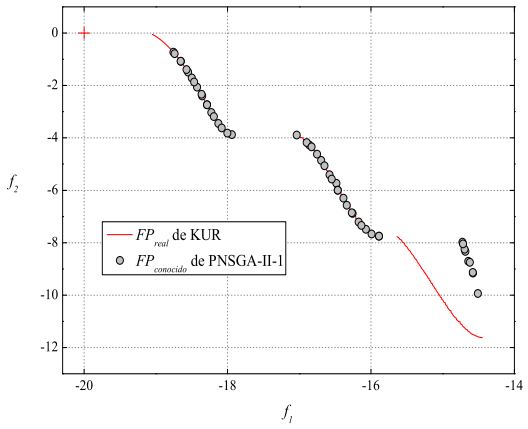


(b) $FP_{conocido}$ obtenido con 8 procesadores.

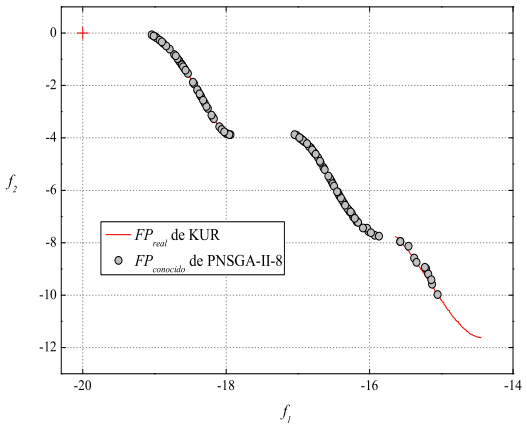


(c) $FP_{conocido}$ obtenido con 16 procesadores.

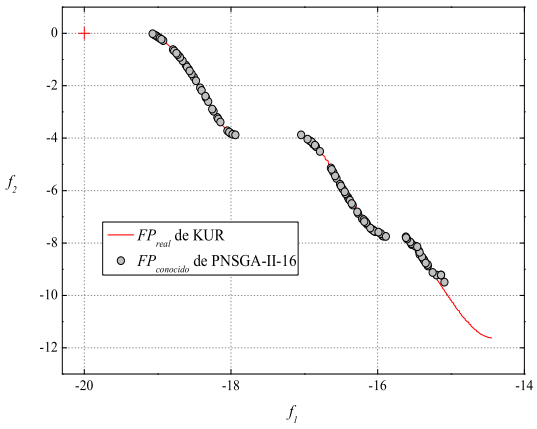
Figura 6.16: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema KUR.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.17: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema KUR.

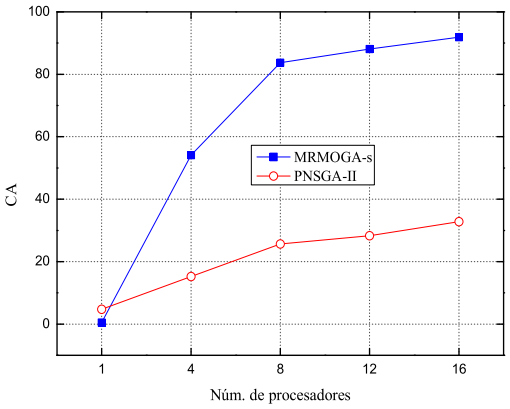
	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.466667	2	0	**0.718022**
MRMOGA-s-4	54.100000	79	19	14.358157
MRMOGA-s-8	83.733333	95	70	6.622856
MRMOGA-s-12	88.100000	**98**	75	5.387331
MRMOGA-s-16	**91.866667**	97	**85**	3.148898
PNSGA-II-1	4.742853	6	0	1.642830
PNSGA-II-4	15.200000	39	6	6.710191
PNSGA-II-8	25.633333	43	12	8.352578
PNSGA-II-12	28.266667	39	16	6.217895
PNSGA-II-16	32.833333	45	19	7.220726

	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.001377	0.001016	0.002197	0.000265
MRMOGA-s-4	0.000373	0.000338	0.000430	0.000020
MRMOGA-s-8	0.000353	0.000309	0.000418	0.000024
MRMOGA-s-12	0.000351	0.000310	0.000383	**0.000019**
MRMOGA-s-16	0.000359	0.000312	0.000418	0.000030
PNSGA-II-1	0.001070	0.000725	0.001963	0.000281
PNSGA-II-4	0.000301	0.000263	**0.000344**	0.000021
PNSGA-II-8	**0.000280**	**0.000254**	0.000335	0.000017
PNSGA-II-12	0.000339	0.000264	0.000529	0.000049
PNSGA-II-16	0.000378	0.000312	0.000650	0.000065

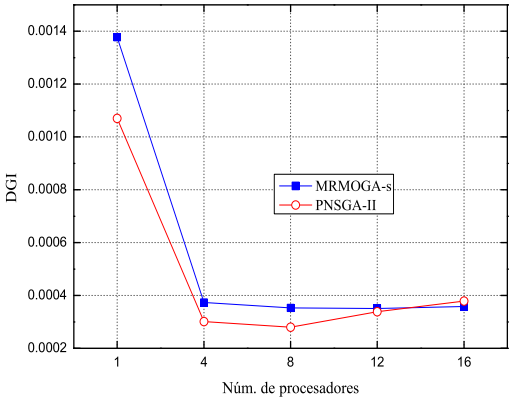
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.016651	0.010335	0.030767	0.004412
MRMOGA-s-4	0.008838	0.006948	0.010378	0.000762
MRMOGA-s-8	0.009282	0.007783	0.012053	0.000857
MRMOGA-s-12	0.009049	0.007843	0.010079	0.000641
MRMOGA-s-16	0.008924	0.007382	0.010112	**0.000617**
PNSGA-II-1	0.018009	0.011202	0.034348	0.005100
PNSGA-II-4	0.008343	0.006399	0.020431	0.002368
PNSGA-II-8	0.007067	0.006140	**0.008924**	0.000678
PNSGA-II-12	**0.007064**	**0.004750**	0.009009	0.001002
PNSGA-II-16	0.007729	0.005243	0.014014	0.001717

	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	**1.458993**	**1.445180**	**1.493190**	**0.009474**
MRMOGA-s-4	1.499159	1.449620	1.591670	0.038714
MRMOGA-s-8	1.615469	1.563670	1.723200	0.034978
MRMOGA-s-12	1.630996	1.598150	1.691210	0.024261
MRMOGA-s-16	1.675115	1.622490	1.747600	0.025015
PNSGA-II-1	6.144273	6.122504	6.181463	0.014633
PNSGA-II-4	6.327773	6.290441	6.436452	0.028545
PNSGA-II-8	6.370632	6.304255	6.528832	0.045248
PNSGA-II-12	6.442961	6.389245	6.599747	0.040474
PNSGA-II-16	6.475304	6.436281	6.531440	0.023945

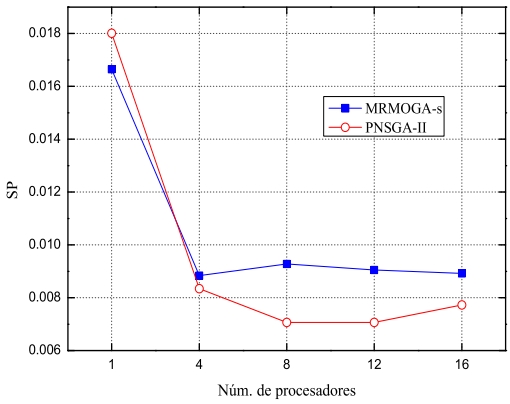
Tabla 6.10: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema ZDT1 (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.

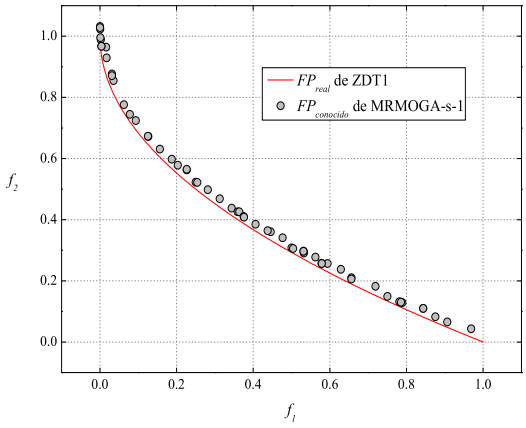


(c) Comparación con relación a SP.

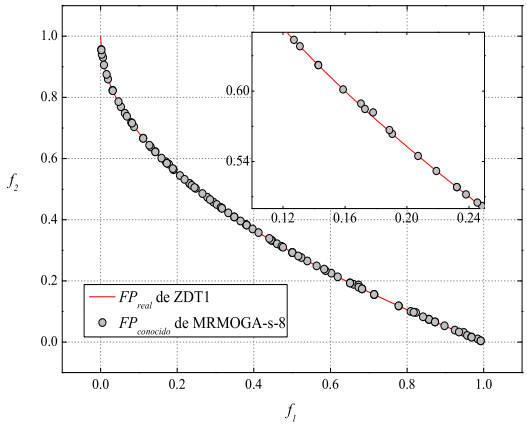
Figura 6.18: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema ZDT1).

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.000668	0.000667	0.000000	0.000000	0.000000
MRMOGA-S-4	0.923848	0.703667	0.540000	0.517667	0.506000
MRMOGA-S-8	0.939880	0.735333	0.601000	0.577333	0.562000
MRMOGA-S-12	0.949900	0.750000	0.636333	0.591000	0.577667
MRMOGA-S-16	0.965932	0.794667	0.705333	0.659333	0.650667
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	1.000000	0.221333	0.019667	0.006667	0.005333
PNSGA-II-4	1.000000	0.363000	0.077000	0.050333	0.033333
PNSGA-II-8	1.000000	0.399000	0.101000	0.069333	0.044000
PNSGA-II-12	1.000000	0.406667	0.099000	0.073667	0.051000
PNSGA-II-16	1.000000	0.416667	0.111000	0.080000	0.056000

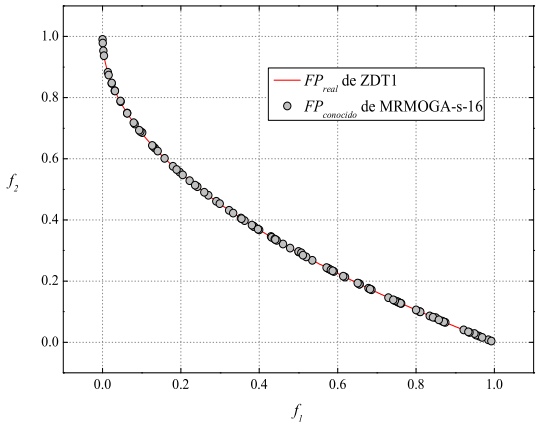
Tabla 6.11: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema ZDT1).



(a) $FP_{conocido}$ obtenido con un procesador.

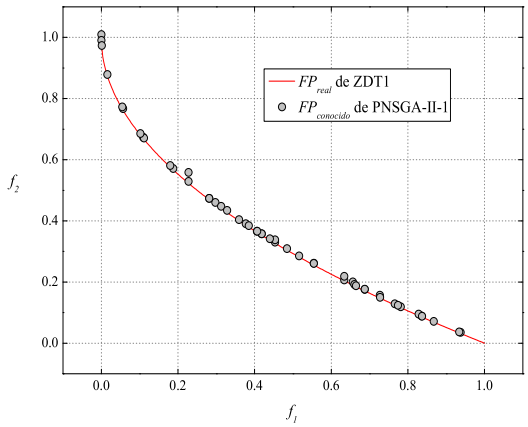


(b) $FP_{conocido}$ obtenido con 8 procesadores.

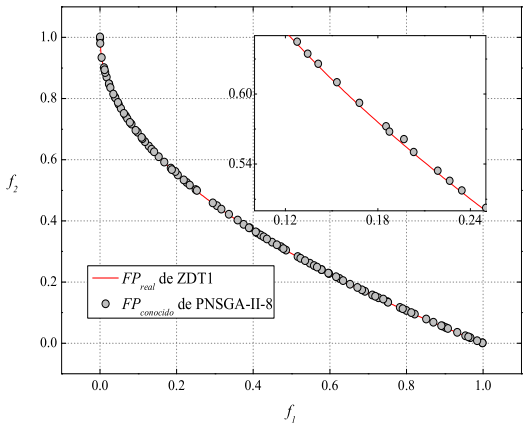


(c) $FP_{conocido}$ obtenido con 16 procesadores.

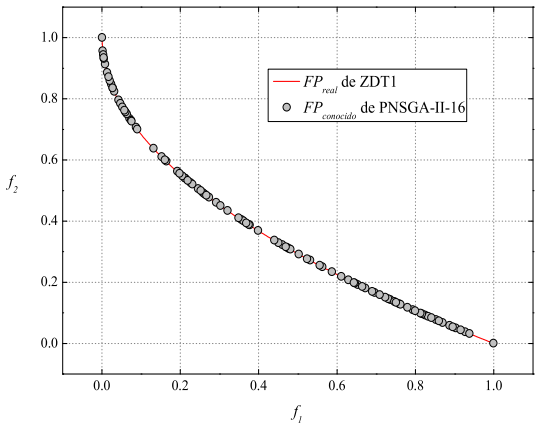
Figura 6.19: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema ZDT1.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.20: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema ZDT1.

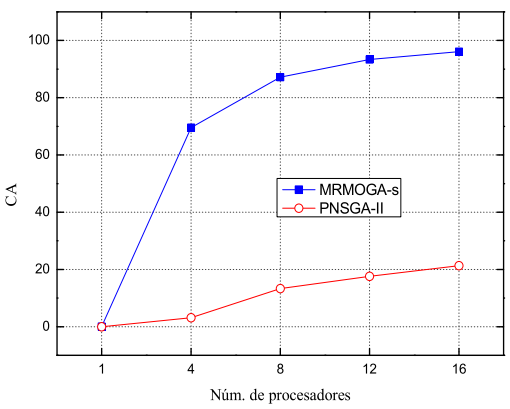
	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.06666667	1	0	0.24944383
MRMOGA-s-4	69.566667	89	18	16.626652
MRMOGA-s-8	87.200000	96	76	4.679031
MRMOGA-s-12	93.333333	98	86	2.784880
MRMOGA-s-16	**96.066667**	**100**	**91**	1.896195
PNSGA-II-1	0.000000	0	0	**0.000000**
PNSGA-II-4	3.133333	16	0	3.593822
PNSGA-II-8	13.333333	36	0	9.665517
PNSGA-II-12	17.566667	36	0	10.219861
PNSGA-II-16	21.300000	32	9	5.780138

	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.001488	0.001129	0.002086	0.000231
MRMOGA-s-4	0.000443	0.000358	0.000528	0.000041
MRMOGA-s-8	0.000416	0.000344	0.000485	0.000032
MRMOGA-s-12	0.000426	0.000369	0.000543	0.000046
MRMOGA-s-16	0.000430	0.000364	0.000516	0.000037
PNSGA-II-1	0.001297	0.000856	0.001997	0.000320
PNSGA-II-4	0.000337	**0.000278**	0.000454	0.000048
PNSGA-II-8	**0.000296**	0.000269	**0.000373**	**0.000024**
PNSGA-II-12	0.000352	0.000293	0.000439	0.000036
PNSGA-II-16	0.000350	0.000281	0.000437	0.000037

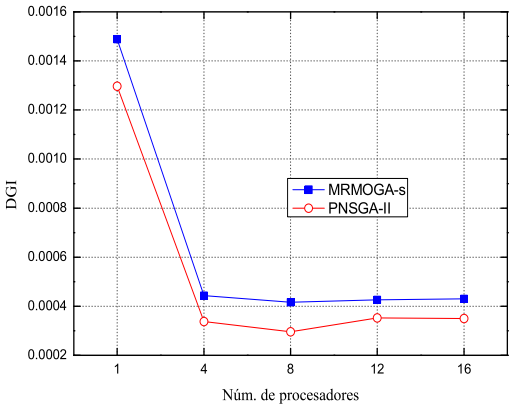
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.020222	0.009831	0.037153	0.006381
MRMOGA-s-4	0.008813	0.007483	0.010780	**0.000649**
MRMOGA-s-8	0.009232	0.007779	0.010770	0.000800
MRMOGA-s-12	0.008722	0.006603	0.010076	0.000876
MRMOGA-s-16	0.008973	0.007441	0.010286	0.000690
PNSGA-II-1	0.008416	0.006733	0.019851	0.002239
PNSGA-II-4	**0.007147**	0.005232	0.009616	0.000842
PNSGA-II-8	0.007147	0.005232	0.009616	0.000842
PNSGA-II-12	0.007192	0.005833	**0.009448**	0.000917
PNSGA-II-16	0.007165	**0.005228**	0.010113	0.001212

	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	**1.436509**	**1.421720**	**1.456200**	**0.007837**
MRMOGA-s-4	1.515458	1.440460	1.675630	0.044107
MRMOGA-s-8	1.608533	1.558410	1.692100	0.032393
MRMOGA-s-12	1.636861	1.606060	1.695500	0.023128
MRMOGA-s-16	1.665152	1.625560	1.731260	0.023031
PNSGA-II-1	5.994191	5.949310	6.034001	0.024157
PNSGA-II-4	6.170927	6.143469	6.196933	0.013879
PNSGA-II-8	6.251327	6.205869	6.356052	0.041353
PNSGA-II-12	6.322473	6.266451	6.460659	0.043628
PNSGA-II-16	6.334883	6.310818	6.373239	0.018779

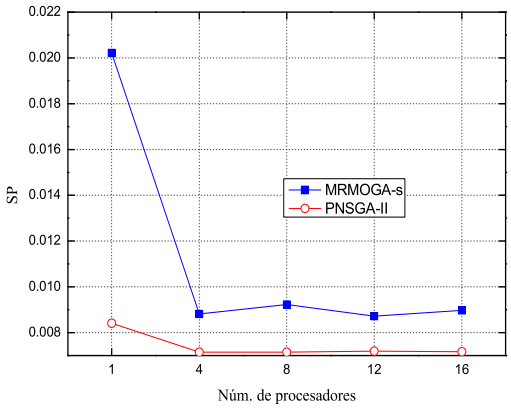
Tabla 6.12: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema ZDT2 (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.

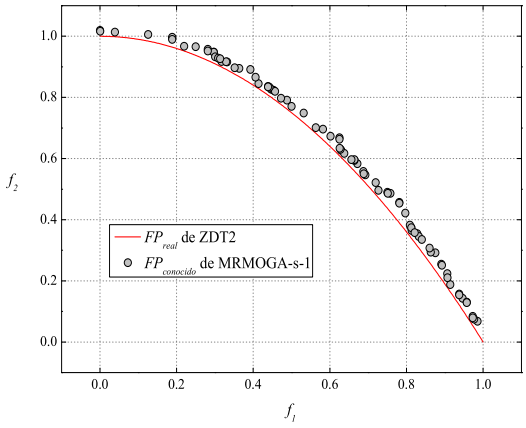


(c) Comparación con relación a SP.

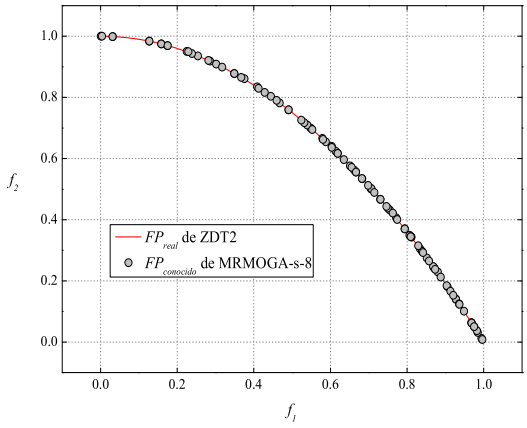
Figura 6.21: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema ZDT2).

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.005394	0.000667	0.000000	0.000000	0.000000
MRMOGA-S-4	0.997303	0.885000	0.806667	0.788000	0.753333
MRMOGA-S-8	0.992583	0.890333	0.789667	0.767667	0.731667
MRMOGA-S-12	0.987862	0.867333	0.766667	0.744667	0.733667
MRMOGA-S-16	0.991234	0.891333	0.790000	0.760000	0.736000
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	1.000000	0.069667	0.004333	0.001000	0.000333
PNSGA-II-4	1.000000	0.159000	0.024000	0.011000	0.005667
PNSGA-II-8	1.000000	0.196667	0.041667	0.020667	0.011000
PNSGA-II-12	1.000000	0.197000	0.043333	0.021333	0.011333
PNSGA-II-16	1.000000	0.206000	0.048333	0.028333	0.010667

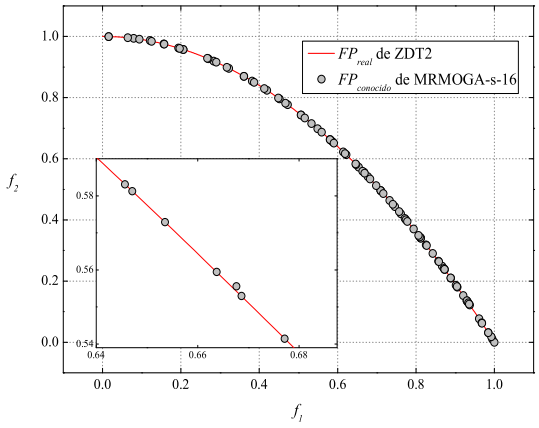
Tabla 6.13: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema ZDT2).



(a) $FP_{conocido}$ obtenido con un procesador.

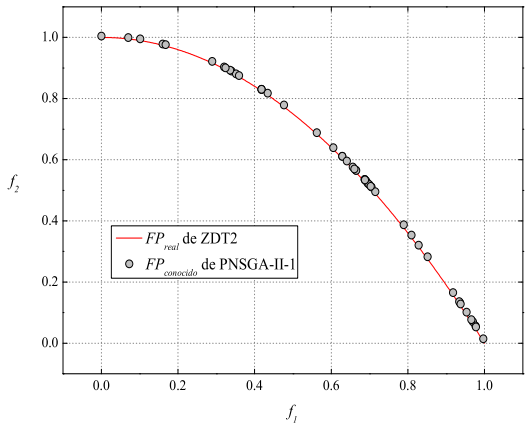


(b) $FP_{conocido}$ obtenido con 8 procesadores.

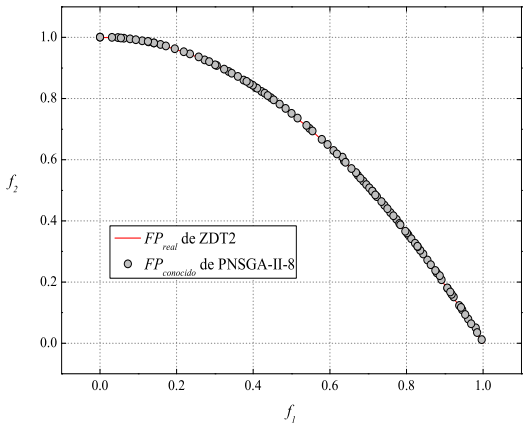


(c) $FP_{conocido}$ obtenido con 16 procesadores.

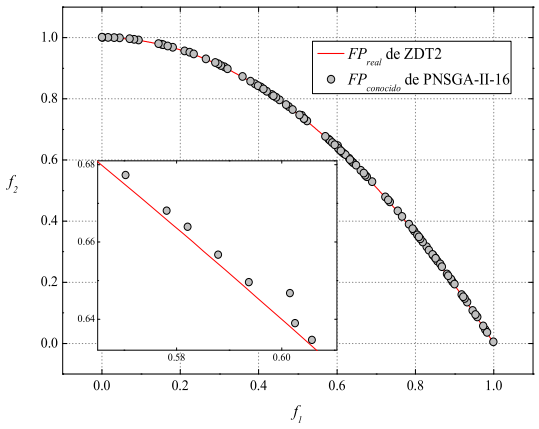
Figura 6.22: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema ZDT2.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.23: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema ZDT2.

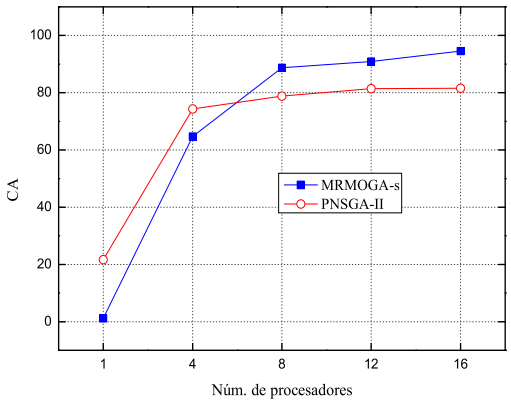
se muestran en la tabla 6.15. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para ZDT3 se muestran en las figuras 6.25 y 6.26, respectivamente.

De manera semejante a lo que ocurrió en el problema KUR, en este problema, MRMOGA demostró tener mejor convergencia que PNSGA-II a partir de 8 procesadores (figura 6.24(a)) (curiosamente ambos problemas tienen un conjutno FP_{real} desconectado). En la comparación con la métrica $\mathcal{C}$ observamos que solamente cuando MRMOGA utiliza 1 y 4 procesadores, PNSGA-II-n ($n = 1, \ldots, 16$) se desempeña mejor. A partir de que ambos algoritmos utilizan 8 procesadores, MRMOGA cubre una fracción mayor de los frentes generados por PNSGA-II.

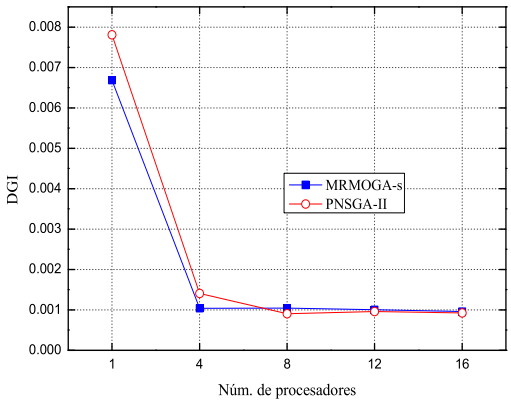
De nueva cuenta, es interesante la notable mejoría de la convergencia y la distribución de MRMOGA cuando usa más de dos procesadores (figuras 6.24(a)–(c)). Con respecto a la distribución, PNSGA-II se desempeña mejor, pero MRMOGA tiene resultados bastante competitivos como se muestra en la figura 6.25.

	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	1.166667	4	0	1.067187
MRMOGA-s-4	64.700000	86	39	14.116539
MRMOGA-s-8	88.733333	95	79	3.776536
MRMOGA-s-12	90.866667	96	84	2.872088
MRMOGA-s-16	**94.566667**	**100**	**86**	3.353439
PNSGA-II-1	21.666667	32	4	6.436010
PNSGA-II-4	74.300000	86	57	5.235456
PNSGA-II-8	78.833333	86	70	3.715583
PNSGA-II-12	81.433333	88	74	3.489826
PNSGA-II-16	81.533333	90	75	3.947432
	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.006684	0.003244	0.022052	0.003228
MRMOGA-s-4	0.001037	**0.000763**	0.001247	0.000132
MRMOGA-s-8	0.001045	0.000866	0.001366	0.000116
MRMOGA-s-12	0.001005	0.000877	0.001232	0.000089
MRMOGA-s-16	0.000962	0.000715	0.001314	0.000140
PNSGA-II-1	0.007811	0.003042	0.017052	0.003071
PNSGA-II-4	**0.001407**	0.000761	0.004023	0.000717
PNSGA-II-8	0.000903	0.000615	0.001138	0.000181
PNSGA-II-12	0.000962	0.000620	0.001228	**0.000198**
PNSGA-II-16	0.000924	0.000709	**0.001295**	0.000152
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.019450	0.006137	0.066027	0.012841
MRMOGA-s-4	0.010028	0.008140	0.013066	0.001166
MRMOGA-s-8	0.010416	0.008049	0.013201	0.001094
MRMOGA-s-12	0.010681	0.008977	0.012510	0.000921
MRMOGA-s-16	0.011056	0.008645	0.012562	**0.000877**
PNSGA-II-1	0.026705	0.008191	0.077446	0.018186
PNSGA-II-4	0.008804	0.006219	0.020118	0.002599
PNSGA-II-8	**0.008129**	**0.005891**	**0.010650**	0.001162
PNSGA-II-12	0.008626	0.006034	0.013685	0.001520
PNSGA-II-16	0.009202	0.005904	0.014529	0.002149
	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	1.454533	1.437190	1.474640	0.009701
MRMOGA-s-4	1.475320	1.447530	1.582580	0.024976
MRMOGA-s-8	1.575250	1.530020	1.684730	0.032867
MRMOGA-s-12	1.602632	1.570830	1.648170	0.021832
MRMOGA-s-16	1.634419	1.577470	1.712980	0.033979
PNSGA-II-1	6.137891	6.127769	6.153882	0.006459
PNSGA-II-4	6.315023	6.281371	6.360734	0.015006
PNSGA-II-8	6.347413	6.305002	6.383994	0.017548
PNSGA-II-12	6.429337	6.391350	6.470492	0.018641
PNSGA-II-16	6.483275	6.449328	6.530492	0.022633

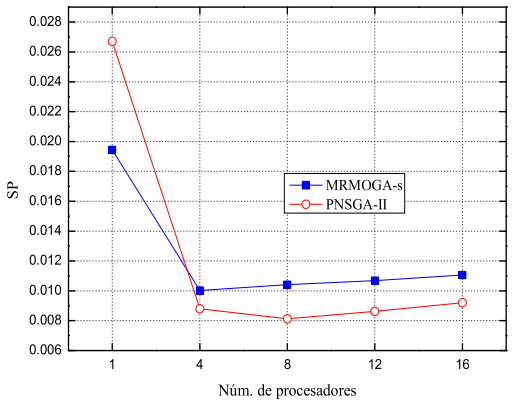
Tabla 6.14: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema ZDT3 (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.

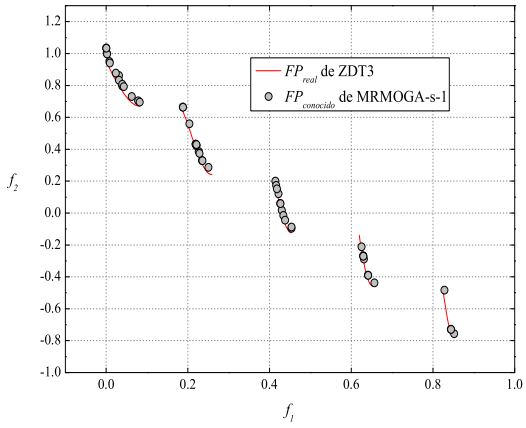


(c) Comparación con relación a SP.

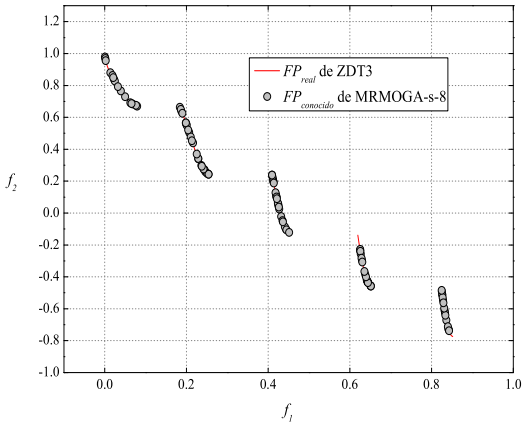
Figura 6.24: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DG) y el espaciamiento (SP) (problema ZDT3).

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.029074	0.005667	0.000333	0.000000	0.000000
MRMOGA-S-4	0.669371	0.339333	0.275667	0.229000	0.237333
MRMOGA-S-8	0.826910	0.500333	0.401333	0.346000	0.341667
MRMOGA-S-12	0.818796	0.525333	0.442000	0.409667	0.391333
MRMOGA-S-16	0.818120	0.530333	0.435667	0.390000	0.384667
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	0.999466	0.367000	0.080000	0.044000	0.018667
PNSGA-II-4	1.000000	0.543667	0.229000	0.164333	0.073333
PNSGA-II-8	1.000000	0.561667	0.259667	0.188333	0.095667
PNSGA-II-12	1.000000	0.578667	0.280333	0.208333	0.098667
PNSGA-II-16	1.000000	0.580000	0.274667	0.208000	0.103333

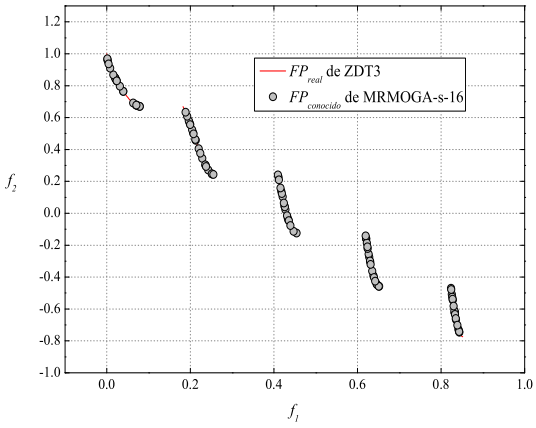
Tabla 6.15: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema ZDT3).



(a) $FP_{conocido}$ obtenido con un procesador.

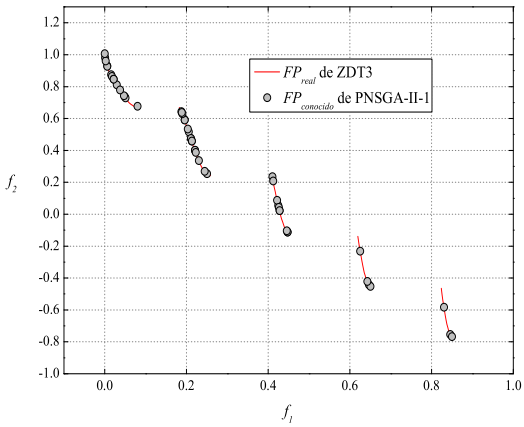


(b) $FP_{conocido}$ obtenido con 8 procesadores.

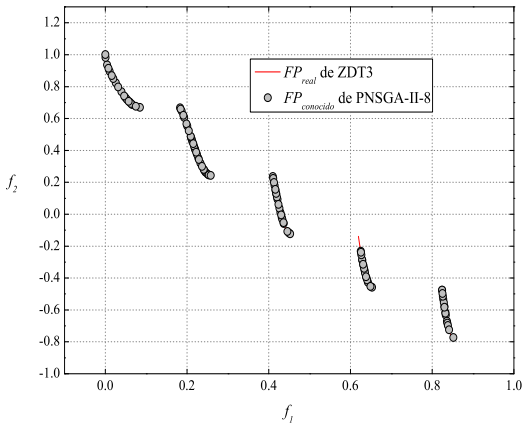


(c) $FP_{conocido}$ obtenido con 16 procesadores.

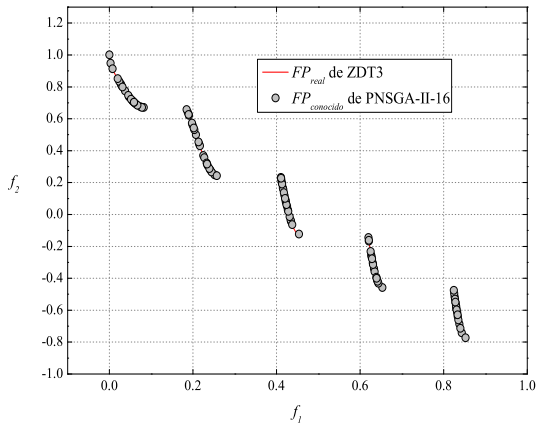
Figura 6.25: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema ZDT3.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.26: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema ZDT3.

6.7.7. Resultados y discusión para el problema DTLZ7

En la tabla 6.16 se presentan los resultados estadísticos de las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.17. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para DTLZ7 se muestran en las figuras 6.28 y 6.29, respectivamente.

Para DTLZ7, un problema con 22 variables, PNSG-II demostró tener la mejor convergencia con cualquier número de procesadores como lo muestra la comparación del conteo de aciertos de la figura 6.27(a). Los resultados de la métrica $\mathcal{C}$ concuerdan con este resultado ya que PNSGA-II domina una fracción mayor de los frentes de MRMOGA que este a los de aquel cuando usan el mismo número de procesadores. No obstante, con base en los valores de la métrica GDI (tabla 6.16), MRMOGA se acerca, en promedio, más al FP_{real} de DTLZ7 que PNSGA-II.

Para cualquier número de procesadores, MRMOGA consiguió una mejor distribución de sus soluciones como se puede comprobar cuantitativamente en la gráfica de la figura 6.27(c), y visualmente en la vista del plano f_1-f_2 de las figuras 6.28(b)–(c).

6.7.8. Resultados y discusión para el problema TMK

En la tabla 6.18 se presentan los resultados estadísticos de las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.17. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para TMK se muestran en las figuras 6.31 y 6.32, respectivamente.

El problema TMK es el primero de los tres problemas de prueba con restricciones utilizados en esta comparación. El comportamiento de PNSGA-II en este problema es un ejemplo claro donde las métricas proporcionan resultados confusos. De acuerdo con las métricas, independientemente del número de procesadores, PNSGA-II superó ampliamente a MRMOGA tanto en convergencia como en distribución. Sin embargo, como se observa en las figuras 6.32(a)–(c), las soluciones que encontró PNSGA-II están concentradas en una región muy reducida de la superficie del frente de Pareto real, y además, varias de ellas representan soluciones iguales (considerando 5 decimales de precisión). En el caso extremo mostrado en la figura 6.32(a), las 100 soluciones son el mismo vector en el espacio de los objetivos. En consecuencia, el espaciamiento será cero, y, puesto que el vector pertenece a FP_{real}, el conteo de aciertos será 100 (véase tabla 6.18). Por otro lado, las soluciones que encuentra MRMOGA se extienden sobre toda la superficie del frente de Pareto real (figuras 6.31(a)–(c)). La métrica DGI da cuenta de esta situación. Puesto que los vectores de las regiones no cubiertas del FP_{real} están muy alejados de los vectores del $FP_{conocido}$ de PNSGA-II, se producen valores muy pobres en DGI en comparación con los que obtiene MRMOGA. Claramente, el conjunto $PF_{conocido}$ que obtiene MRMOGA es mucho mejor que el obtenido por PNSGA-II. A pesar de que más del 93 % de los vectores no dominados de PNSGA-II-n pertenecen al frente de Pareto real, no dominan una región muy amplia del espacio de las funciones objetivo. Por esta razón, aunque PNSGA-II-n obtuvo buenos resultados en CA, cuando más, domina solamente al 3 % de los frentes de MRMOGA-s-16. El porcentaje de vectores que cubre MRMOGA-s-n de los frentes no dominados de PNSGA-II va aumentando con el número de procesadores debido a que cuando los frentes de este se diversifican MRMOGA los comienza a dominar.

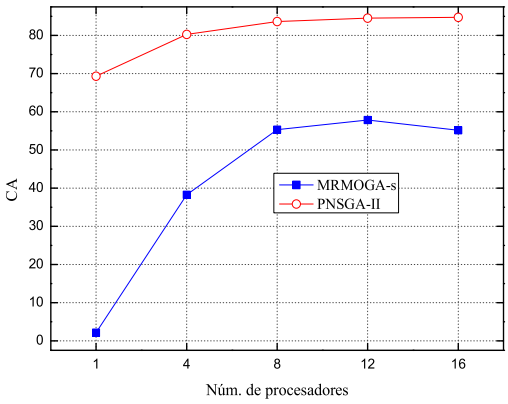
	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	2.133333	8	0	**2.028683**
MRMOGA-s-4	38.266667	60	14	12.425601
MRMOGA-s-8	55.300000	71	39	8.339265
MRMOGA-s-12	57.866667	76	44	8.535156
MRMOGA-s-16	55.166667	75	35	10.086570
PNSGA-II-1	69.333333	88	46	4.894441
PNSGA-II-4	80.266667	89	72	4.296769
PNSGA-II-8	83.633333	88	77	3.082027
PNSGA-II-12	84.566667	**92**	**78**	3.537262
PNSGA-II-16	**84.766667**	90	75	4.144742

	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.008110	0.006637	0.009969	0.000788
MRMOGA-s-4	0.005084	0.004736	0.005641	0.000246
MRMOGA-s-8	**0.004917**	**0.004665**	0.005481	0.000175
MRMOGA-s-12	0.004950	0.004678	**0.005270**	**0.000146**
MRMOGA-s-16	0.005000	0.004717	0.005369	0.000150
PNSGA-II-1	0.006552	0.005649	0.008136	0.000528
PNSGA-II-4	0.005064	0.004728	0.005649	0.000212
PNSGA-II-8	0.005108	0.004756	0.005605	0.000213
PNSGA-II-12	0.005173	0.004882	0.005651	0.000182
PNSGA-II-16	0.005155	0.004681	0.005649	0.000208

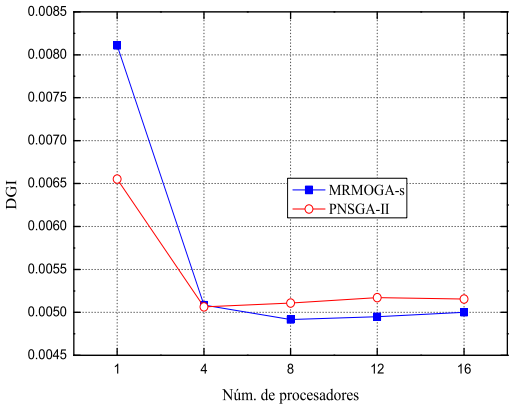
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.082011	**0.053766**	0.191083	0.025911
MRMOGA-s-4	0.071406	0.052718	**0.086113**	0.007660
MRMOGA-s-8	0.070000	0.057254	0.082389	0.006158
MRMOGA-s-12	0.070467	0.057397	0.084439	**0.006510**
MRMOGA-s-16	**0.071140**	0.060883	0.092923	0.007389
PNSGA-II-1	0.115015	0.076069	0.170825	0.022744
PNSGA-II-4	0.075400	0.061055	0.086900	0.006075
PNSGA-II-8	0.071730	0.047132	0.086999	0.009867
PNSGA-II-12	0.073832	0.056041	0.087577	0.007557
PNSGA-II-16	0.072942	0.054940	0.092067	0.009380

	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	**1.300837**	**1.222790**	**1.398330**	**0.034568**
MRMOGA-s-4	1.458312	1.352160	1.650320	0.059694
MRMOGA-s-8	1.735813	1.627070	1.937330	0.059696
MRMOGA-s-12	1.806236	1.735110	1.890860	0.040052
MRMOGA-s-16	1.796675	1.707060	1.896750	0.045238
PNSGA-II-1	5.422716	5.417163	5.433843	0.004089
PNSGA-II-4	5.566548	5.526003	5.629921	0.021289
PNSGA-II-8	5.612733	5.559212	5.664357	0.018270
PNSGA-II-12	5.680802	5.630037	5.742343	0.022039
PNSGA-II-16	5.719746	5.691631	5.766699	0.019893

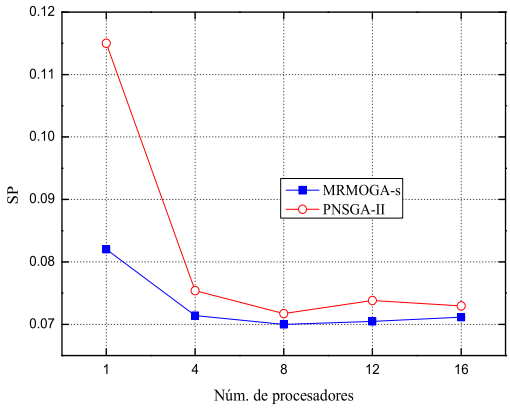
Tabla 6.16: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema DTLZ7 (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.



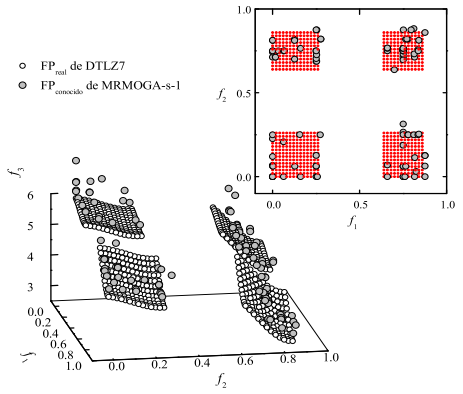
(c) Comparación con relación a SP.

Figura 6.27: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema DTLZ7).

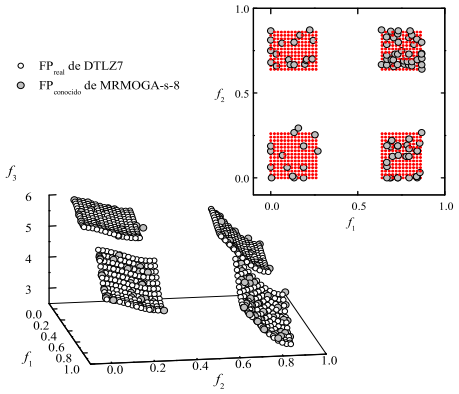
$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.023333	0.004000	0.004333	0.003000	0.000667
MRMOGA-S-4	0.219333	0.167000	0.120000	0.117667	0.103333
MRMOGA-S-8	0.250667	0.166000	0.133667	0.133667	0.107667
MRMOGA-S-12	0.310667	0.197667	0.160000	0.173333	0.148333
MRMOGA-S-16	0.276000	0.199333	0.155667	0.161333	0.146000

$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	0.964333	0.532667	0.338333	0.302333	0.333667
PNSGA-II-4	0.989333	0.689000	0.532333	0.475333	0.519333
PNSGA-II-8	0.968333	0.657333	0.511333	0.475667	0.516333
PNSGA-II-12	0.987333	0.684000	0.553333	0.491000	0.531000
PNSGA-II-16	0.983333	0.691333	0.543667	0.475667	0.524667

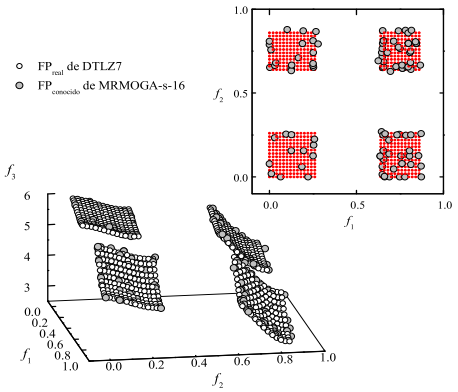
Tabla 6.17: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema DTLZ7).



(a) $FP_{conocido}$ obtenido con un procesador.

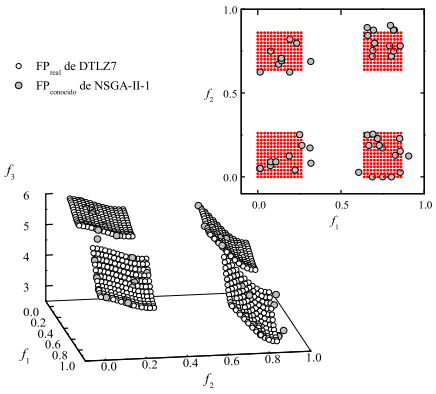


(b) $FP_{conocido}$ obtenido con 8 procesadores.

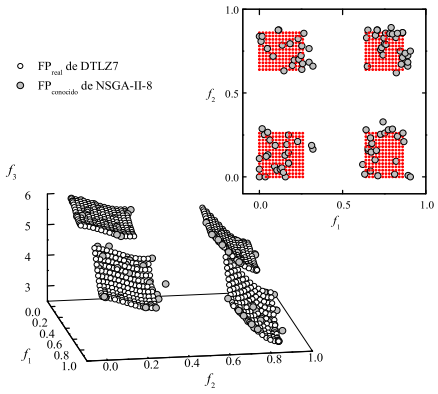


(c) $FP_{conocido}$ obtenido con 16 procesadores.

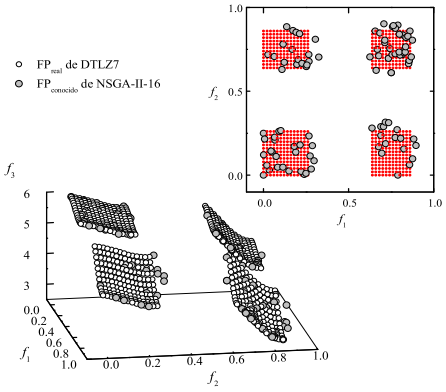
Figura 6.28: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema DTLZ7.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.

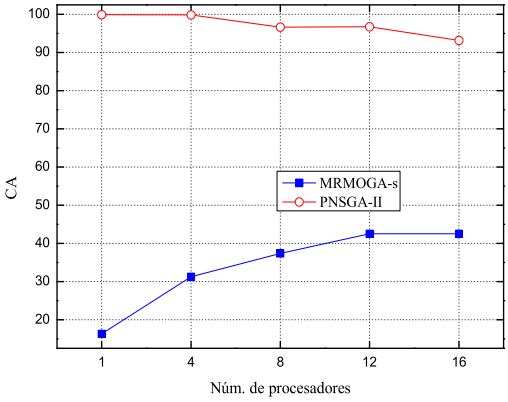


(c) $FP_{conocido}$ obtenido con 16 procesadores.

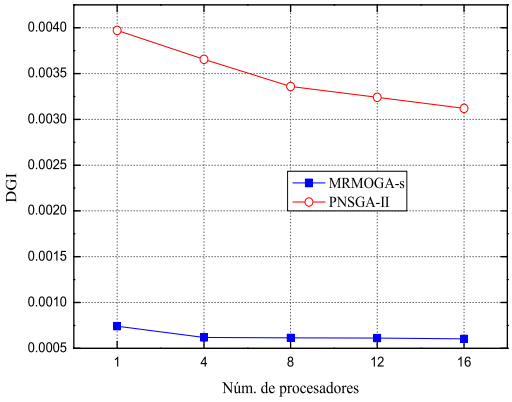
Figura 6.29: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema DTLZ7.

	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	16.300000	23	11	3.195309
MRMOGA-s-4	31.233333	45	23	5.037746
MRMOGA-s-8	37.333333	46	22	4.812022
MRMOGA-s-12	42.533333	57	32	6.338945
MRMOGA-s-16	42.500000	55	34	5.194548
PNSGA-II-1	**99.933333**	**100**	**99**	**0.179505**
PNSGA-II-4	99.866667	100	99	0.339935
PNSGA-II-8	96.633333	100	91	2.971905
PNSGA-II-12	96.766667	100	87	3.084189
PNSGA-II-16	93.166667	100	78	6.094168
	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.000741	0.000626	0.001206	0.000111
MRMOGA-s-4	0.000618	0.000571	0.000680	0.000024
MRMOGA-s-8	0.000614	0.000563	0.000681	0.000025
MRMOGA-s-12	0.000612	0.000573	0.000668	0.000023
MRMOGA-s-16	**0.000602**	**0.000570**	**0.000637**	**0.000018**
PNSGA-II-1	0.003972	0.003567	0.005245	0.000366
PNSGA-II-4	0.003654	0.003476	0.003862	0.000093
PNSGA-II-8	0.003359	0.002888	0.003637	0.000161
PNSGA-II-12	0.003241	0.002876	0.003503	0.000181
PNSGA-II-16	0.003119	0.002788	0.003436	0.000153
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.059353	0.045921	0.074465	0.006337
MRMOGA-s-4	0.050791	0.042681	0.061038	0.004241
MRMOGA-s-8	0.048813	0.040170	0.055887	0.003743
MRMOGA-s-12	0.047797	0.040510	0.054910	0.003170
MRMOGA-s-16	0.049233	0.039696	0.055308	0.004086
PNSGA-II-1	0.000782	**0.000000**	0.021044	0.003770
PNSGA-II-4	**0.000515**	0.000000	**0.006250**	**0.001258**
PNSGA-II-8	0.004067	0.000332	0.017341	0.003672
PNSGA-II-12	0.005544	0.000386	0.048723	0.008469
PNSGA-II-16	0.005056	0.001504	0.009972	0.002200
	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	2.223165	**0.906539**	2.623060	0.281674
MRMOGA-s-4	2.693477	2.529120	2.849750	0.063435
MRMOGA-s-8	2.843907	2.734750	3.897460	0.282883
MRMOGA-s-12	3.318608	2.758070	3.720390	0.311858
MRMOGA-s-16	5.975379	5.671390	6.248260	0.132277
PNSGA-II-1	**0.930808**	0.910414	**0.958129**	**0.009929**
PNSGA-II-4	1.085493	1.074837	1.125584	0.011795
PNSGA-II-8	1.408480	1.317570	1.589002	0.067819
PNSGA-II-12	1.142649	1.104610	1.199773	0.026127
PNSGA-II-16	1.002638	0.977353	1.055313	0.020794

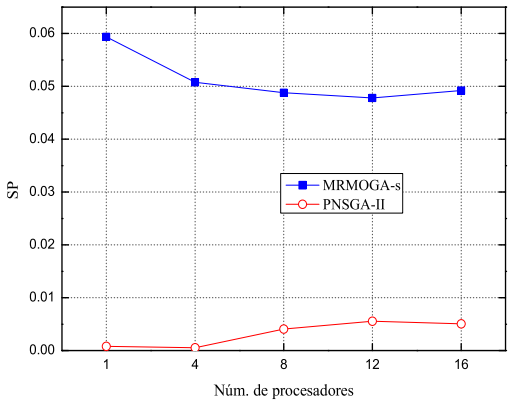
Tabla 6.18: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema TMK (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.

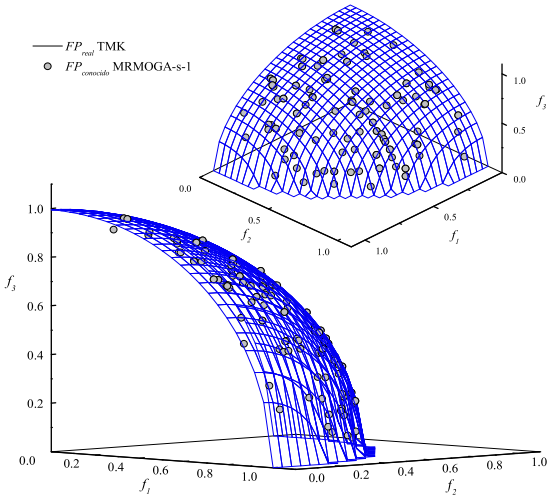


(b) Comparación con relación a DGI.

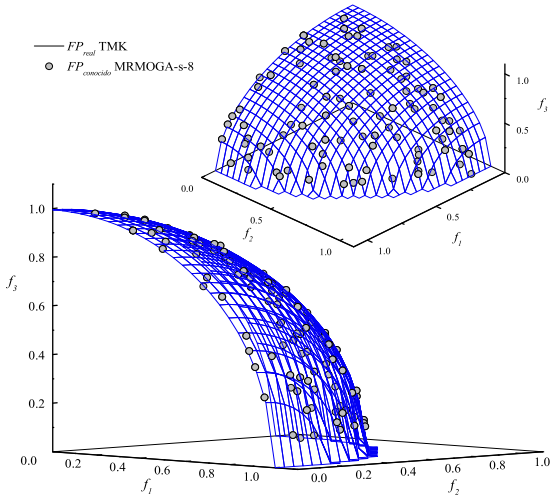


(c) Comparación con relación a SP.

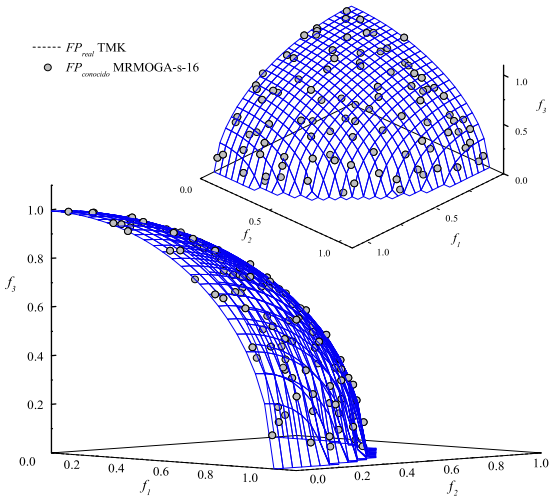
Figura 6.30: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema TMK).



(a) $FP_{conocido}$ obtenido con un procesador.

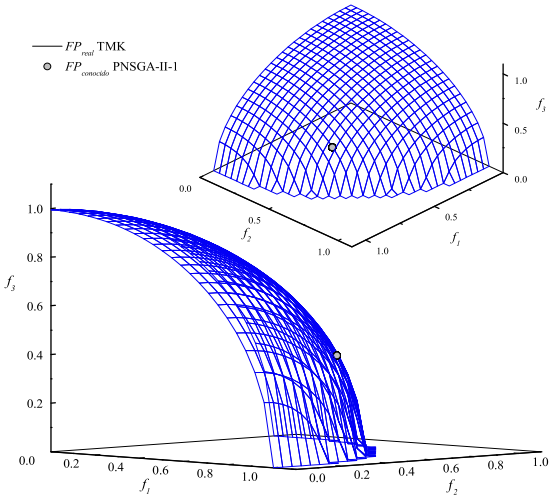


(b) $FP_{conocido}$ obtenido con 8 procesadores.

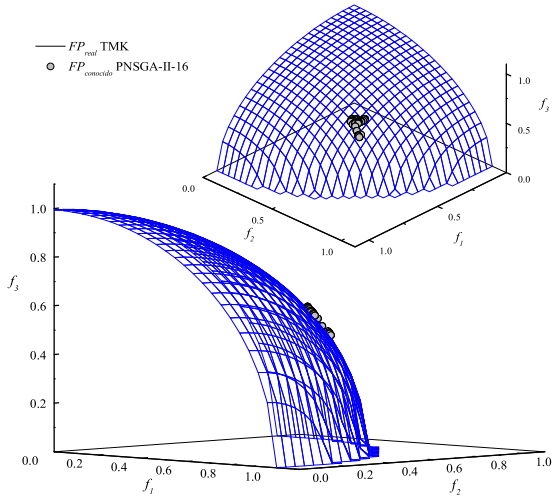


(c) $FP_{conocido}$ obtenido con 16 procesadores.

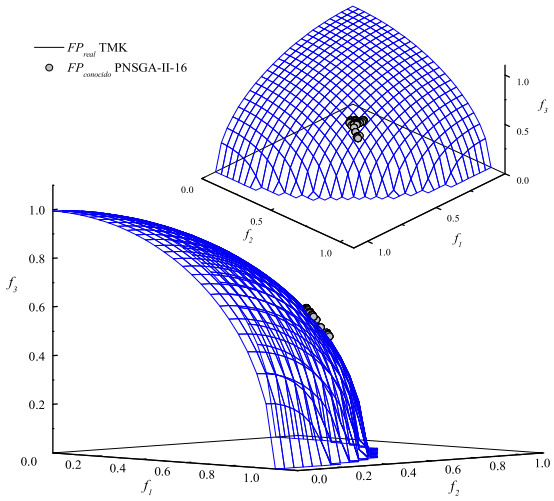
Figura 6.31: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema TMK.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.32: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema TMK.

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.000667	0.000333	0.006333	0.005333	0.009667
MRMOGA-S-4	0.000667	0.000333	0.009000	0.008000	0.012333
MRMOGA-S-8	0.000667	0.000667	0.007333	0.006000	0.011000
MRMOGA-S-12	0.000667	0.000667	0.007000	0.009333	0.011000
MRMOGA-S-16	0.000667	0.000667	0.007667	0.007667	0.015000
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	0.056667	0.009333	0.008000	0.006000	0.004000
PNSGA-II-4	0.027667	0.007000	0.004000	0.003333	0.002667
PNSGA-II-8	0.068000	0.022333	0.018333	0.017333	0.015000
PNSGA-II-12	0.052333	0.021333	0.019667	0.020333	0.022333
PNSGA-II-16	0.079000	0.027333	0.026667	0.023667	0.020667

Tabla 6.19: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema TMK).

6.7.9. Resultados y discusión para el problema OSY

En la tabla 6.20 se presentan los resultados estadísticos correspondientes a las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.17. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para OSY se muestran en las figuras 6.34 y 6.35, respectivamente.

En las gráficas de la figura 6.34 se demuestra la notable capacidad que tiene MRMOGA para mantener soluciones próximas a todas las regiones que conforman el frente de Pareto real de OSY. En cambio PNSGA-II, para algunas corridas no logra converger a ciertas regiones del frente, especialmente el segmento DE del frente de Pareto (véase figura 6.6). Además, de acuerdo con el conteo de aciertos, MRMOGA supera a PNSGA-II en convergencia, ya que consigue un mayor número de soluciones que pertenecen a FP_{real}. Por otro lado, es interesante notar que PNSGA-II superó a MRMOGA en la comparación mediante la métrica $\mathcal{C}$. Por ejemplo, MRMOGA-s-n solamente consiguió cubrir, cuando más, al 50 % de los frentes de PNSGA-II-16 (su mejor resultado), mientras que PNSGA-II-n, a lo sumo, cubrió el 84 % de los frentes de MRMOGA-s-16 (su mejor resultado). Esto posiblemente se debe a que los vectores del conjunto $FP_{conocido}$ que reporta MRMOGA que no pertenecen a FP_{real} están ligeramente más alejados que los vectores del $FP_{conocido}$ de PNSGA-II al frente de Pareto real. Esta suposición se sustenta en el hecho de que los valores que obtiene PNSGA-II en DGI son marginalmente mejores que los que obtiene MRMOGA.

Con relación a la distribución, PNSGA-II consigue una mejor distribución que MRMOGA, aunque este último se mantiene competitivo como se observa en las gráficas de la figura 6.34.

6.7.10. Resultados y discusión para el problema KIT

En la tabla 6.20 se presentan los resultados estadísticos de las 30 ejecuciones de MRMOGA y PNSGA-II para el conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP) y el tiempo. La comparación directa de los dos algoritmos con base en la métrica $\mathcal{C}$ se muestran en la tabla 6.17. Las gráficas de los conjuntos $FP_{conocido}$ obtenidos por MRMOGA y PNSGA-II para KIT se muestran en las figuras 6.37 y 6.38, respectivamente.

En este problema MRMOGA demostró una mucho mejor convergencia que PNSGA-II. De acuerdo a las métricas CA y DGI, MRMOGA demostró tener mejor convergencia que PNSGA-II. A partir de un procesador MRMOGA se aproximó rápidamente al frente de Pareto real como

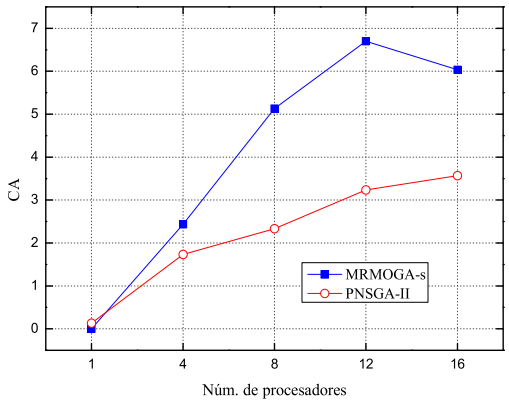
	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.000000	0	**0**	**0.000000**
MRMOGA-s-4	2.433333	10	0	2.871508
MRMOGA-s-8	5.133333	10	0	3.471151
MRMOGA-s-12	**6.700000**	11	0	2.853653
MRMOGA-s-16	6.033333	**13**	0	3.038457
PNSGA-II-1	0.133333	2	**0**	0.426875
PNSGA-II-4	1.733333	7	0	1.590248
PNSGA-II-8	2.333333	6	0	1.639783
PNSGA-II-12	3.233333	7	0	1.782944
PNSGA-II-16	3.566667	7	0	1.667000

	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.751967	0.099587	1.282490	0.312839
MRMOGA-s-4	0.143080	0.028575	0.321246	0.089191
MRMOGA-s-8	0.058722	0.022337	0.168895	0.033704
MRMOGA-s-12	0.050371	0.020231	0.092914	0.021939
MRMOGA-s-16	0.037787	**0.016565**	0.077440	0.018128
PNSGA-II-1	0.596249	0.066509	1.627550	0.538269
PNSGA-II-4	0.193988	0.035237	1.551870	0.361359
PNSGA-II-8	0.053831	0.019895	0.107441	0.022831
PNSGA-II-12	0.041481	0.018256	0.081996	**0.015982**
PNSGA-II-16	**0.031880**	0.017488	**0.069398**	0.013659

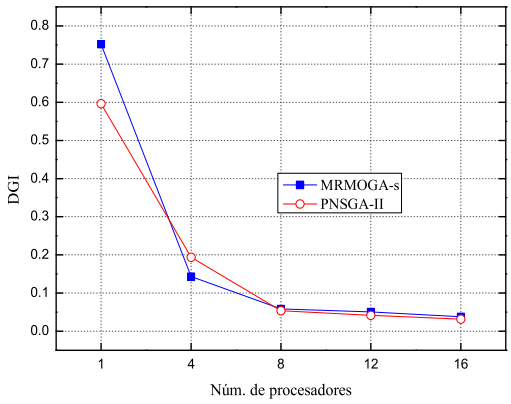
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	1.515959	0.206544	7.549620	1.635786
MRMOGA-s-4	1.618448	0.971453	5.455890	1.057467
MRMOGA-s-8	1.553995	0.962754	4.050550	0.571279
MRMOGA-s-12	1.523740	1.164540	3.981180	0.506852
MRMOGA-s-16	1.593621	1.075150	5.126060	0.766801
PNSGA-II-1	1.630102	0.108029	4.983180	1.055128
PNSGA-II-4	**1.191448**	**0.070370**	5.288670	0.825427
PNSGA-II-8	1.221750	0.942005	2.638720	0.294290
PNSGA-II-12	1.245077	0.971422	**1.963670**	**0.221008**
PNSGA-II-16	1.348825	0.992740	3.932880	0.552257

	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	**1.596234**	**1.130270**	2.244570	0.267065
MRMOGA-s-4	1.712171	1.196810	2.283740	0.245793
MRMOGA-s-8	1.969600	1.725700	2.259120	0.129287
MRMOGA-s-12	2.194705	2.062630	2.310920	0.058033
MRMOGA-s-16	2.283675	2.204660	2.366380	0.035750
PNSGA-II-1	1.682991	1.675787	**1.691528**	**0.003385**
PNSGA-II-4	1.735206	1.701858	1.769317	0.016853
PNSGA-II-8	1.757266	1.719655	1.800547	0.020578
PNSGA-II-12	1.777594	1.740285	1.823302	0.022638
PNSGA-II-16	1.809913	1.783324	1.835698	0.017122

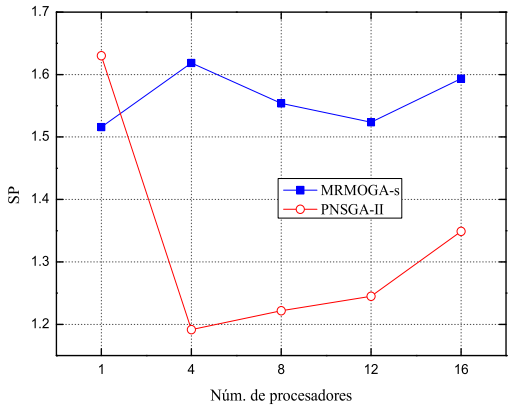
Tabla 6.20: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema OSY (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.

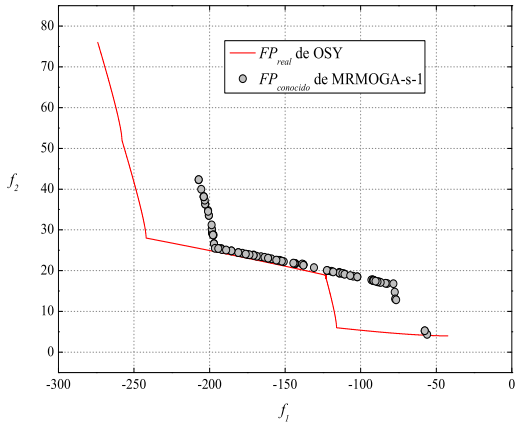


(c) Comparación con relación a SP.

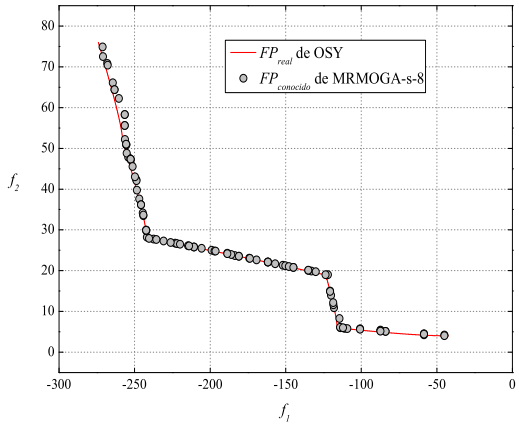
Figura 6.33: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema OSY).

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.767845	0.398333	0.168333	0.146000	0.120000
MRMOGA-S-4	0.869246	0.525333	0.312333	0.271000	0.251667
MRMOGA-S-8	0.934623	0.676333	0.549333	0.436000	0.439000
MRMOGA-S-12	0.931955	0.673667	0.548667	0.443000	0.441667
MRMOGA-S-16	0.947298	0.713000	0.612333	0.479000	0.490667
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	0.977097	0.702000	0.510333	0.393333	0.335000
PNSGA-II-4	0.991896	0.876333	0.821333	0.752333	0.733333
PNSGA-II-8	0.995419	0.916667	0.853333	0.810000	0.785667
PNSGA-II-12	0.994715	0.902000	0.872667	0.826667	0.817667
PNSGA-II-16	0.996124	0.931667	0.879000	0.837333	0.837000

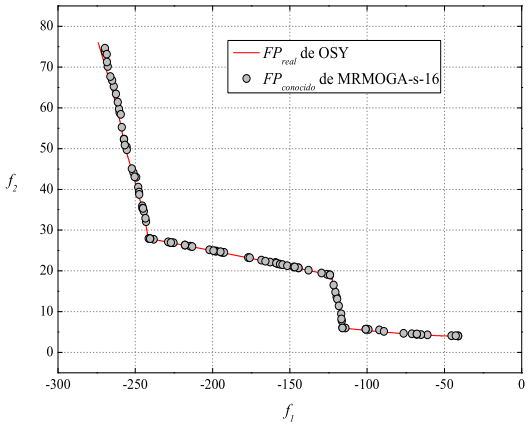
Tabla 6.21: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema OSY).



(a) $FP_{conocido}$ obtenido con un procesador.

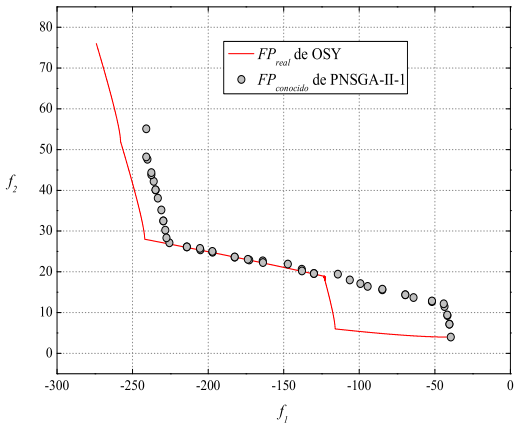


(b) $FP_{conocido}$ obtenido con 8 procesadores.

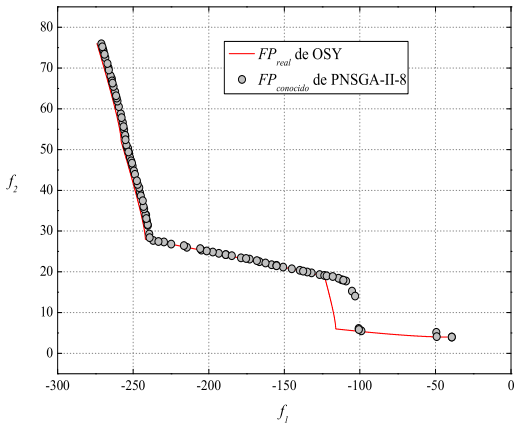


(c) $FP_{conocido}$ obtenido con 16 procesadores.

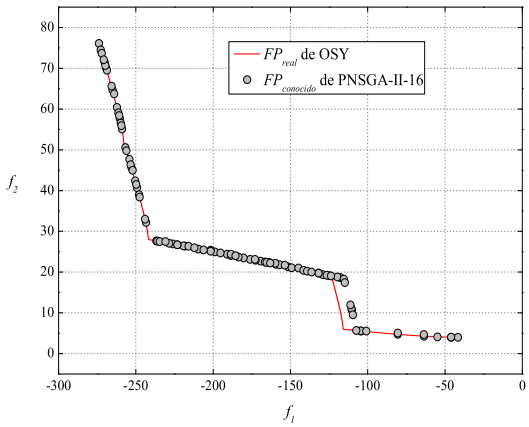
Figura 6.34: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema OSY.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.35: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema OSY.

se revela comparando las figuras 6.37(a) y 6.38(a) correspondientes a los conjuntos $FP_{conocido}$ generados por cada algoritmo. Esto demuestra que a MRMOGA no le afectó demasiado que los elementos del conjunto P_{real} de KIT estuvieran situados en el límite con la región no factible. En la comparación directa con la métrica $\mathcal{C}$, MRMOGA también superó claramente a PNSGA-II.

Apoyándonos en la comparación visual de las soluciones obtenidas por ambos algoritmos y en los resultados de la figura 6.36(c), podemos afirmar que MRMOGA logra una mejor distribución de sus soluciones.

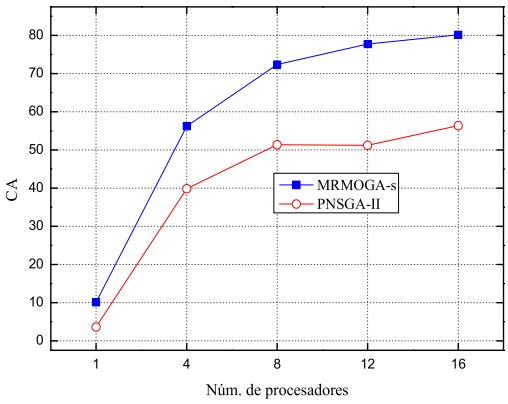
	CA			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	10.100000	18	4	4.002916
MRMOGA-s-4	56.233333	69	29	8.823013
MRMOGA-s-8	72.400000	85	58	6.364485
MRMOGA-s-12	77.766667	87	**72**	4.310324
MRMOGA-s-16	**80.200000**	**89**	69	5.764258
PNSGA-II-1	3.600000	9	0	**2.844878**
PNSGA-II-4	39.866667	69	8	11.580827
PNSGA-II-8	51.366667	65	33	6.843407
PNSGA-II-12	51.233333	63	38	6.453337
PNSGA-II-16	56.400000	67	45	6.058603

	DGI			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.018268	0.008484	0.042165	0.005501
MRMOGA-s-4	0.004367	0.003416	0.006567	0.000797
MRMOGA-s-8	**0.003676**	**0.002997**	**0.004534**	**0.000360**
MRMOGA-s-12	0.003920	0.003337	0.005107	0.000360
MRMOGA-s-16	0.003780	0.003089	0.004753	0.000379
PNSGA-II-1	0.165508	0.012975	0.267905	0.070293
PNSGA-II-4	0.038507	0.003702	0.195305	0.054447
PNSGA-II-8	0.011068	0.003289	0.063836	0.014280
PNSGA-II-12	0.008100	0.003092	0.034995	0.006276
PNSGA-II-16	0.008560	0.003443	0.036432	0.007590

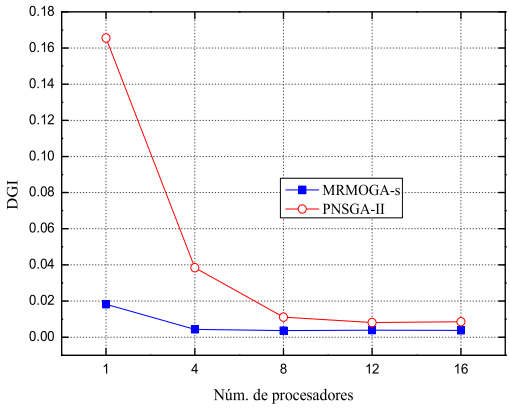
	SP			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	0.167766	0.034242	0.816884	0.222153
MRMOGA-s-4	0.142422	0.045828	**0.474379**	0.136175
MRMOGA-s-8	**0.076459**	0.040256	0.601189	**0.099727**
MRMOGA-s-12	0.118393	0.039773	0.815570	0.154066
MRMOGA-s-16	0.133477	0.044016	0.685588	0.154285
PNSGA-II-1	0.086525	**0.008277**	0.508285	0.126439
PNSGA-II-4	0.154568	0.019894	1.493240	0.284289
PNSGA-II-8	0.123254	0.036109	0.898479	0.190843
PNSGA-II-12	0.201868	0.041808	1.113750	0.254003
PNSGA-II-16	0.139653	0.037061	0.494960	0.111811

	Tiempo			
Algoritmo	**Promedio**	**Mejor**	**Peor**	**Desviación**
MRMOGA-s-1	**0.754548**	**0.426883**	0.994845	0.164746
MRMOGA-s-4	0.778117	0.647839	0.980664	0.055558
MRMOGA-s-8	0.804326	0.750169	0.872368	0.028927
MRMOGA-s-12	0.865746	0.814206	0.974591	0.031961
MRMOGA-s-16	0.944335	0.843562	1.100730	0.046768
PNSGA-II-1	0.757530	0.724777	0.901772	0.033748
PNSGA-II-4	0.755707	0.734520	**0.805547**	**0.014868**
PNSGA-II-8	0.782954	0.743539	0.817598	0.019817
PNSGA-II-12	0.800489	0.761043	0.933122	0.041751
PNSGA-II-16	0.819711	0.783056	0.837839	0.015730

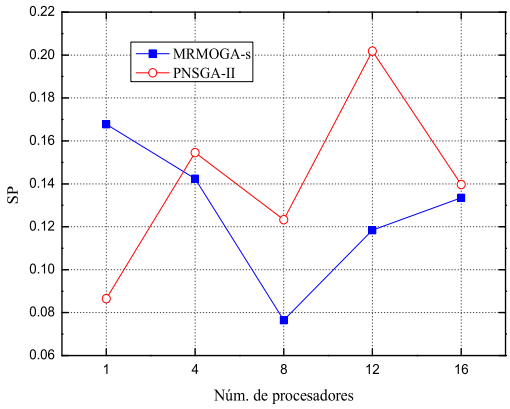
Tabla 6.22: Valores estadísticos del conteo de aciertos (CA), la distancia generacional invertida (DGI), el espaciamiento (SP), y el tiempo para el problema KIT (con negritas se destaca el mejor valor para cada estadística).



(a) Comparación con relación a CA.



(b) Comparación con relación a DGI.

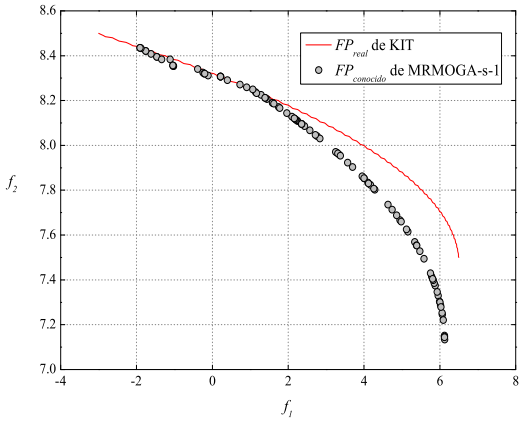


(c) Comparación con relación a SP.

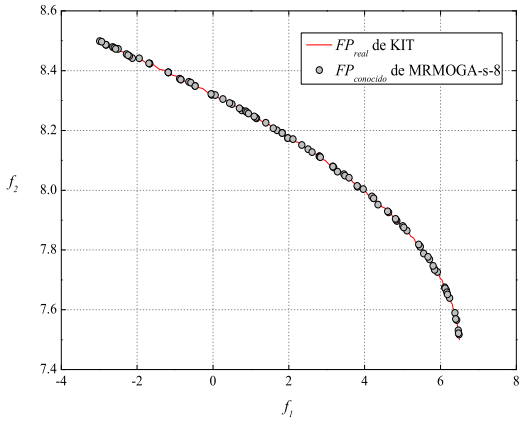
Figura 6.36: Comparación del desempeño de MRMOGA y PNSGA-II en función del número de procesadores de acuerdo con los valores promedio del conteo de aciertos (CA), la distancia generacional invertida (DGI) y el espaciamiento (SP) (problema KIT).

$\mathcal{C}(A,B)$	PNSGA-II-1	PNSGA-II-4	PNSGA-II-8	PNSGA-II-12	PNSGA-II-16
MRMOGA-S-1	0.880667	0.315667	0.222000	0.245667	0.211333
MRMOGA-S-4	0.964000	0.667333	0.557000	0.564333	0.537667
MRMOGA-S-8	0.967333	0.711667	0.620667	0.633333	0.599333
MRMOGA-S-12	0.977333	0.745333	0.668667	0.674667	0.649333
MRMOGA-S-16	0.980667	0.779000	0.703000	0.711000	0.690000
$\mathcal{C}(B,A)$	MRMOGA-S-1	MRMOGA-S-4	MRMOGA-S-8	MRMOGA-S-12	MRMOGA-S-16
PNSGA-II-1	0.744000	0.105667	0.037667	0.023000	0.015000
PNSGA-II-4	0.928333	0.592333	0.446667	0.327667	0.290667
PNSGA-II-8	0.946000	0.678000	0.523000	0.418000	0.382667
PNSGA-II-12	0.942667	0.675667	0.532667	0.406667	0.377333
PNSGA-II-16	0.955667	0.711667	0.553333	0.432000	0.395000

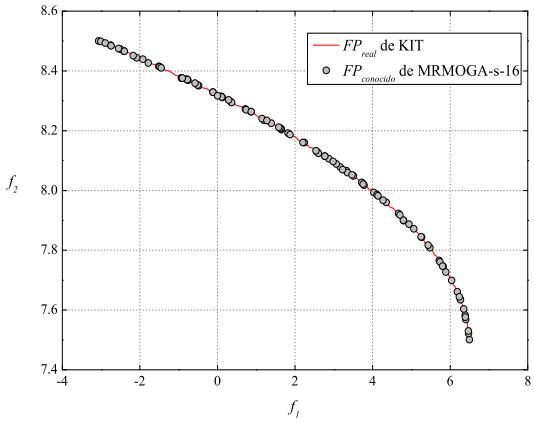
Tabla 6.23: Cada celda de la tabla superior denota la fracción del conjunto $FP_{conocido}$ del algoritmo B (asociado con las columnas) que es dominado débilmente por el conjunto $FP_{conocido}$ del algoritmo A (asociado con los renglones), es decir $\mathcal{C}(A,B)$. De modo análogo, la tabla inferior presenta los resultados para $\mathcal{C}(B,A)$ (problema KIT).



(a) $FP_{conocido}$ obtenido con un procesador.

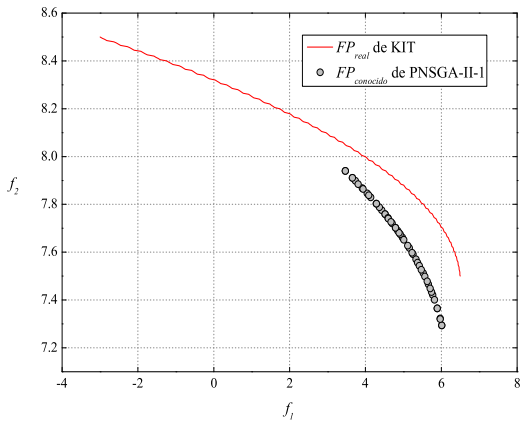


(b) $FP_{conocido}$ obtenido con 8 procesadores.

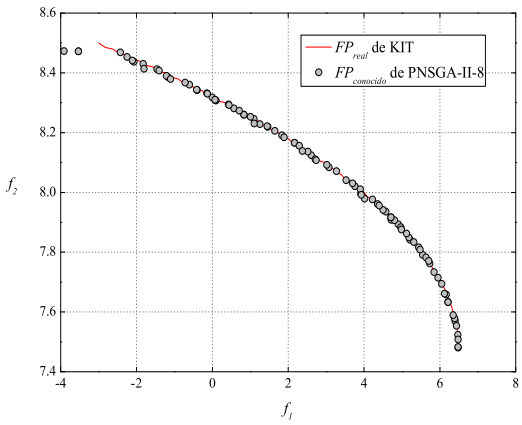


(c) $FP_{conocido}$ obtenido con 16 procesadores.

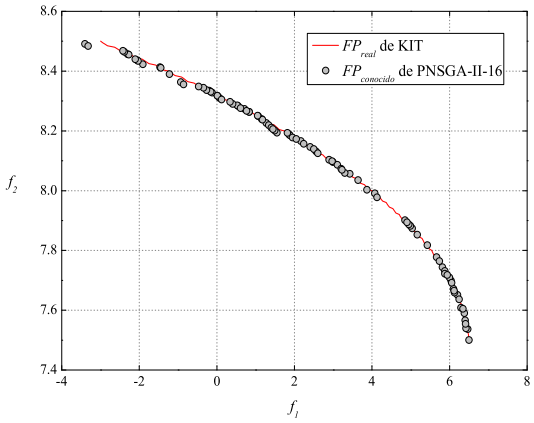
Figura 6.37: $FP_{conocido}$ generado por MRMOGA con 1, 8 y 16 procesadores para el problema KIT.



(a) $FP_{conocido}$ obtenido con un procesador.



(b) $FP_{conocido}$ obtenido con 8 procesadores.



(c) $FP_{conocido}$ obtenido con 16 procesadores.

Figura 6.38: $FP_{conocido}$ generado por PNSGA-II con 1, 8 y 16 procesadores para el problema KIT.

6.8. Conclusión de la comparación con relación a la eficacia

La comparación para evaluar MRMOGA incluyó 9 problemas de prueba con diferentes características que causan dificultades para los AEMOs. Las características estudiadas fueron, principalmente: conjuntos P_{real} y FP_{real} desconectados, FP_{real} desconectado y espacios de búsqueda grandes.

Las conclusiones más destacadas que se desprenden de estos experimentos se resumen en los siguientes puntos:

- *La convergencia no aumenta linealmente con el número de procesadores*. En los experimentos descritos anteriormente quedó claro que la convergencia de ambos algoritmos (de acuerdo con las métricas CA y DGI) no aumenta linealmente con el número de procesadores, sino que el incremento al usar más procesadores es cada vez menor (p. ej. en las figuras 6.15(a), 6.18(a) o 6.27(a)). En las gráficas de la CA de todos los problemas anteriores se puede observar que la curva de la CA crece cada vez menos aproximándose a un máximo, e incluso en algunos problemas comienza a descender.

- *La eficacia de* MRMOGA *es la más beneficiada por el paralelismo*. En comparación con PNSGA-II, la convergencia y la distribución de MRMOGA mejora notablemente cuando se utiliza más de un procesador. Por un lado, la convergencia de MRMOGA mejora siempre que aumenta el número de procesadores (al menos hasta 16 procesadores). En cambio, para algunos problemas (OSY, KUR, ZDT3, TMK), la convergencia de PNSGA-II alcanzó un máximo con un determinado número de procesadores y luego disminuyó. Por otro lado, en varios problemas (DEB, KUR, ZDT2, ZDT3, OSY) MRMOGA logró una convergencia pobre con un procesador (siendo superado por PNSGA-II), pero con más de un procesador mejoró su convergencia notablemente hasta el punto que superó a PNSGA-II. Esta ganancia en la convergencia nos revela el beneficio de la estrategia que usa MRMOGA para dividir el espacio de búsqueda asignando distintas resoluciones en cada isla.

- MRMOGA *mantiene diversidad en su conjunto* $FP_{conocido}$. En todos los problemas MRMOGA cubrió todas las regiones del frente, incluso en problemas difíciles como KUR, TMK y OSY, en los cuales PNSGA-II no logró cubrir todas sus regiones.

- MRMOGA *tiene facilidad para resolver problemas con restricciones*. Considerando la sencillez de la estrategia de MRMOGA para manejar restricciones, es inesperadamente buena la capacidad que tiene MRMOGA para resolver problemas con restricciones, incluso para el problema OSY, reconocido por su dificultad. En los tres problemas con restricciones que se consideraron, MRMOGA superó ampliamente a PNSGA-II en dos de los problemas (TMK, KIT), y fue competitivo en otro (OSY).

- MRMOGA *obtiene una distribución aceptable de su conjunto* $FP_{conocido}$. MRMOGA solamente consiguió superar en distribución a PNSGA-II en 3 de los 9 problemas de este estudio (TMK, KIT, DTLZ7). No obstante, la comparación visual de los conjuntos $FP_{conocido}$ de MRMOGA deja ver que su distribución es competitiva con respecto a la que obtuvo PNSGA-II.

6.9. Métricas para evaluar la eficiencia del MRMOGA

Con el fin de evaluar la eficiencia de MRMOGA se usarán tres métricas conocidas en la literatura de la computación paralela: la *aceleración*, la *eficiencia* y la *fracción serial*.

6.9.1. Aceleración (S_p)

La aceleración utilizada en este estudio será la aceleración ortodoxa definida la sección 4.5.1 (aceleración tipo II.A.2). La aceleración se calcula mediante la ecuación 4.2.

Para la aceleración II.A.2, el tiempo $\overline{T}_1$ de la ecuación 4.2 es el tiempo de ejecución promedio del pAEMO con p islas ejecutado en un procesador. Mientras que $\overline{T}_p$ es el tiempo de ejecución promedio del pAEMO con p islas ejecutado en p procesadores.

6.9.2. Eficiencia (E)

La eficiencia [Kumar94] es la medida de la fracción de tiempo que un procesador es empleado de manera útil. Se define mediante la razón entre la aceleración y el número de procesadores. Matemáticamente se define mediante:

$$E_p = \frac{S_p}{p}. \tag{6.46}$$

En un sistema ideal la eficiencia es igual a 1. En la práctica, la eficiencia fluctúa entre 0 y 1, dependiendo del grado de eficiencia con que se utilicen los procesadores.

6.9.3. Fracción serial (F)

Esta métrica fue definida por Karp y Flatt [Karp90] para estimar la fracción del tiempo de ejecución serial en un algoritmo paralelo. Matemáticamente se define mediante:

$$F_p = \frac{1/S_p - 1/p}{1 - 1/p}. \tag{6.47}$$

Idealmente, la fracción serial debe permanecer constante para un algoritmo. Si la aceleración de un algoritmo es pequeña, aún podemos decir que es eficiente si F_p permanece constante para diferentes valores de p. Si el valor de F_p aumenta con el número de procesadores, entonces se considera un indicador de escalabilidad pobre. Cuando un algoritmo tiene aceleración superlineal, entonces F puede tomar valores negativos.

6.10. Resultados experimentales con relación a la eficiencia

6.10.1. Metodología para evaluar la eficiencia

En los experimentos presentados en esta sección se calcularán las métricas con base en la aceleración tipo II.A.2. Esta aceleración nos brinda una idea más realista de la eficiencia del algoritmo evaluado y se ajusta más al campo de la computación paralela debido a que comparamos el mismo algoritmo.

Con el fin de definir el criterio de parada necesario para calcular la aceleración débil es preciso contar con un criterio para determinar la «calidad» del conjunto $FP_{conocido}$. En este estudio proponemos utilizar la métrica HV (§ 6.2.7) para determinar la calidad del frente de Pareto generado por un (p)AEMO. La ejecución del pAEMO se detendrá cuando el valor de la métrica HV

sea igual o mayor a un valor previamente establecido. De esta manera nos aseguramos que en diferentes ejecuciones, el algoritmo siempre converge a una solución de la misma calidad.

Con el fin de determinar la calidad del $FP_{conocido}(t)$ de MRMOGA se calculará la métrica HV al final de cada generación t. Si deseamos obtener de manera precisa el valor HV para el conjunto $FP_{conocido}(t)$, es preciso reunir los conjuntos $P_{conocido}\binom{p}{t}$ de cada procesador p en cada generación t . Sin embargo, este procedimiento incurre en un costo adicional debido a las comunicaciones, el cual no está presente en la ejecución con un procesador, lo cual resultaría en una comparación injusta de su eficiencia. Para resolver este problema, en nuestros experimentos consideramos que el conjunto $FP_{conocido}\binom{p}{t}$ de la isla con mayor resolución refleja de manera muy cercana al conjunto $FP_{conocido}(t)$ producto de todas las islas. De esta manera, el costo de calcular la métrica HV, se compensa al calcular la razón del tiempo del algoritmo serial y el paralelo, y evitamos el costo de comunicaciones adicionales. Para PNSGA-II, por desgracia, no podemos realizar un procedimiento similar al descrito ya que todas las islas contribuyen de igual manera a formar el frente de Pareto conocido. En consecuencia, no es posible calcular la aceleración de PNSGA-II ni las demás métricas derivadas de la aceleración. La comparación entre MRMOGA y PNSGA-II solamente tomará en cuenta el tiempo de ejecución promedio obtenido en las 30 pruebas con cada uno de los problemas de este estudio.

En los experimentos descritos en las siguientes secciones solamente se utilizó la topología simple. Asimismo, las métricas se calcularon considerando únicamente el problema ZDT1.

6.10.2. Comparación utilizando el tiempo de ejecución

Los tiempos de ejecución analizados en esta sección corresponden a los experimentos descritos en la sección 6.7, donde se estudió la eficacia de MRMOGA. Es importante mencionar que en aquellos experimentos el tamaño de la población global (tamaño del problema) no se mantuvo fijo al variar el número de procesadores, sino que la población global aumentó ya que a cada procesador se le asignó una subpoblación de igual tamaño para cualquier número de procesadores. De tal manera que los tiempos de ejecución de MRMOGA y PNSGA-II aumentaron con el número de procesadores[4] (véase figura 6.39). Este aumento de tiempo se debe, principalmente, al tiempo de comunicaciones y a la sincronización entre procesos.

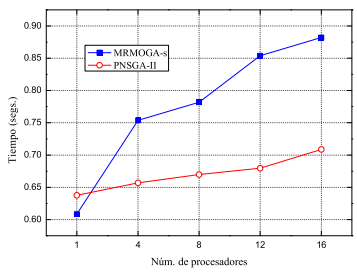
En los problemas con espacios de búsqueda pequeños que van de $2^{34} \approx 1.7 \times 10^{10}$ (POM DEB) a $2^{60} \approx 1.15 \times 10^{18}$ (KUR) soluciones, PNSGA-II tiene mejor desempeño. Por ejemplo, en los problemas DEB, KIT y TMK, PNSGA-II obtuvo mejores tiempos que MRMOGA para cualquier número de procesadores (v. figuras 6.39(a), 6.39(i) y 6.39(g)). En OSY y KUR, MRMOGA fue más rápido con pocos procesadores (menos de 8) y después PNSGA-II lo superó (v. figuras 6.39(h) y 6.39(b)).

Por otro lado, en los problemas con espacios de búsqueda tan grandes como $2^{510} \approx 3.352 \times 10^{153}$ (p. ej. los de los problemas ZDT), es notable la gran diferencia en el tiempo de ejecución de los algoritmos como se observa en las figuras 6.39(c)–(f). En estos problemas, MRMOGA es aproximadamente 6 veces más rápido que PNSGA-II cuando se utilizan 16 procesadores.

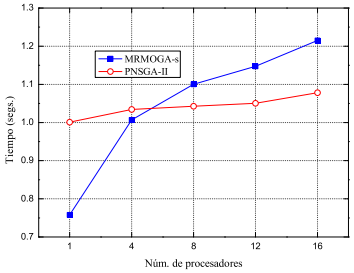
6.10.3. Evaluación utilizando la aceleración tipo II.A.2

Para calcular la aceleración II.A.2, en todas las pruebas se utilizó MRMOGA con 16 islas. Aquí consideramos que el algoritmo secuencial es la ejecución de 16 islas en un solo procesador. El algoritmo paralelo es la ejecución de MRMOGA-s-16 sobre un número creciente de procesadores (desde 2 hasta 16). La población de cada isla se mantuvo en 25 individuos por isla

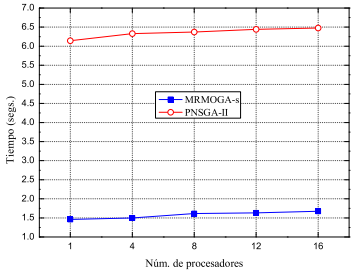
[4]De mantener fijo el tamaño del problema, el tiempo de ejecución disminuiría con el número de procesadores.



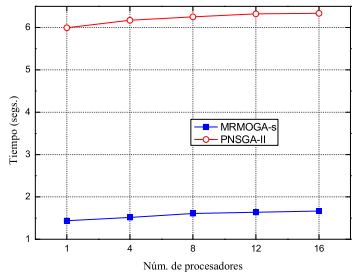
(a) Tiempos para el problema DEB.



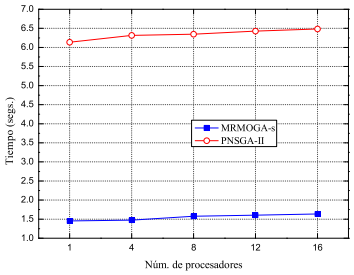
(b) Tiempos para el problema KUR.



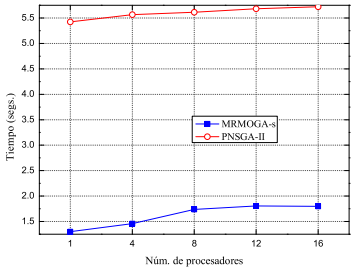
(c) Tiempos para el problema ZDT1.



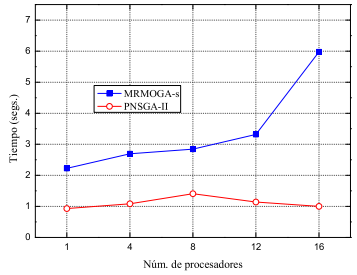
(d) Tiempos para el problema ZDT2.



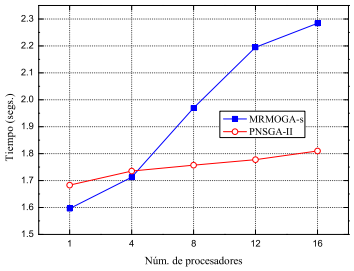
(e) Tiempos para el problema ZDT3.



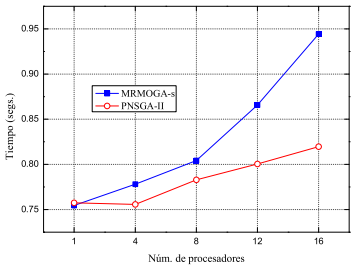
(f) Tiempos para el problema DTLZ7.



(g) Tiempos para el problema TMK.



(h) Tiempos para el problema OSY.



(i) Tiempos para el problema KIT.

Figura 6.39: Comparación del tiempo de ejecución promedio de MRMOGA y PNSGA-II en función del número de procesadores para todos los problemas de prueba.

Procesadores	1	2	4	6	8	10	12	14	16
$\mathbf{S_p}$ II.A.2	1	1.45768	3.03961	4.46722	6.53107	6.90449	7.64773	8.30983	13.55167
$\mathbf{E}$ II.A.2	1	0.72884	0.75990	0.74454	0.81638	0.69045	0.63731	0.59356	0.84698
$\mathbf{F_p}$ II.A.2	–	0.37204	0.10532	0.06862	0.03213	0.04981	0.05174	0.05267	0.01204

Tabla 6.24: Resultados de la eficiencia (E) y la fracción serial (F_p) con relación a la aceleración (S_p) tipo II.A.2.

($16 \times 25 = 400$ individuos en total) para todas las pruebas.

Como se aprecia en la figura 6.40(a), con cualquier número de procesadores MRMOGA obtuvo una aceleración sublineal, pero muy cercana a la lineal. De 2 a 8 procesadores la aceleración tiene una tendencia a ser superlineal como lo revela la métrica F_p, ya que disminuye con el número de procesadores (v. figura 6.40(c)). Entre 10 y 14 procesadores la aceleración comienza a decaer, como ocurre con algoritmos paralelos tradicionales, sin embargo, para 16 procesadores se acerca mucho a la aceleración lineal. En consecuencia, con 16 procesadores también se alcanza la mejor eficiencia y el valor más pequeño para la fracción serial (véase tabla 6.24).

Con excepción de 10-14 procesadores, F_p disminuye, lo cual es un indicio de buena escalabilidad. Es decir que el costo extra debido a las comunicaciones y sincronización no crece demasiado al aumentar el número de procesadores.

6.11. Conclusión de la eficiencia

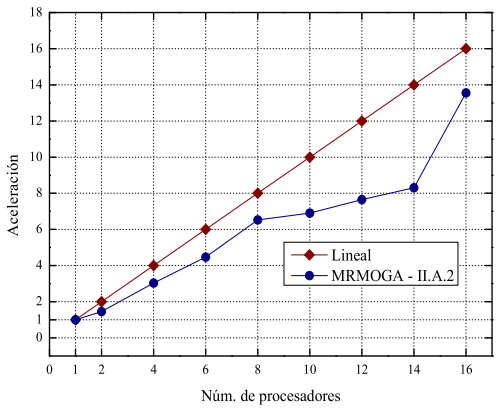
De la comparación de los tiempos de ejecución se desprende que MRMOGA es más eficiente respecto al tiempo de ejecución que PNSGA-II en espacios de búsqueda grandes. En todos los problemas con espacios de búsqueda grandes (ZDT1, ZDT2, ZDT3, DTLZ7), los tiempos de ejecución de MRMOGA fueron mucho menores que los obtenidos por PNSGA-II. Si consideramos que los pMOEAs están encaminados, principalmente, para resolver problemas con espacios de búsqueda grandes en un tiempo razonable, MRMOGA es una mejor opción para este tipo de problemas.

Los 9 problemas utilizados en este estudio no tienen funciones objetivo costosas y, por tanto, los tiempos de ejecución son pequeños para ambos algoritmos. No obstante, podemos establecer que MRMOGA, además de haber mostrado gran eficacia, se mantiene dentro de los ordenes de tiempo de PNSGA-II, e incluso en varios problemas es mejor que éste. Un análisis de la eficiencia utilizando problemas de prueba con funciones objetivo costosas quedan dentro de las perspectivas de este trabajo de tesis.

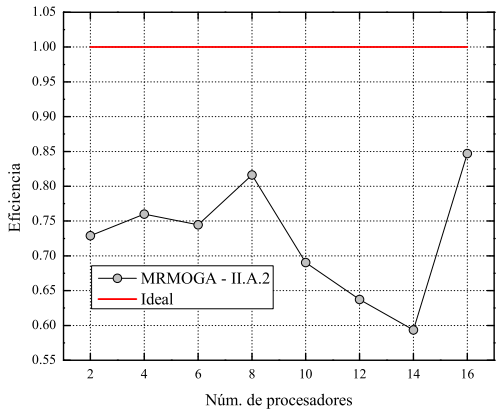
De acuerdo con los resultados obtenidos a partir de la aceleración tipo II.A.2, se desprende que MRMOGA tiene una buena escalabilidad. A pesar de que MRMOGA no obtuvo una aceleración superlineal, la métrica E muestra que su eficiencia disminuye a un ritmo muy lento, y la métrica F_p revela que la aceleración tiende a ser superlineal y que tiene buena escalabilidad.

Con el fin de analizar el compromiso desempeño/precio que se obtiene mediante MRMOGA debemos tener en cuenta que el modelo de islas está pensado para un cúmulo de computadoras (*cluster of computers*).

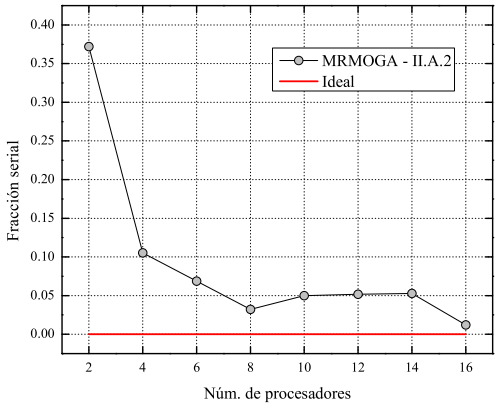
Si suponemos que el costo de un cúmulo es una función lineal del número de nodos, entonces la curva del compromiso desempeño/precio (suponiendo que el desempeño viene dado por la aceleración) es similar a la que describe la eficiencia en la figura 6.40(b). En la gráfica de la figura observamos que con 8 y 16 procesadores se obtienen los mejores valores para la razón desempeño/precio. Esto significa que después de 8 procesadores aún se puede obtener un desempeño (16 procesadores) a cambio del costo del cúmulo. Aunque para 10-14 procesadores la



(a) Aceleración



(b) Eficiencia



(c) Fracción serial

Figura 6.40: Resultados de la eficiencia de MRMOGA de acuerdo con la aceleración II.A.2, la eficiencia y la fracción serial (problema ZDT1).

razón desempeño/precio disminuye, no debemos olvidar que su aceleración se mantiene cerca de la lineal. Por lo tanto, como se verá a continuación, pueden existir problemas en los que convenga usar un cúmulo con ese número de procesadores a pesar de no tener una buena razón desempeño/precio.

La cuestión de qué cúmulo comprar no depende únicamente de la razón desempeño/precio, sino también de la prioridad o de la frecuencia del problema que tengamos en manos. Pongamos por caso la optimización del diseño de alas de avión. Si en una industria se realiza de manera intensa el diseño de alas, entonces la adquisición de un cúmulo con 10-16 nodos es una buena inversión. Por otro lado, si solamente se diseñara una ala y se puede esperar más tiempo por ella, entonces puede bastar con un cúmulo con menos de 8 nodos.

Finalmente, podemos decir que para 1-8 y 16 procesadores se obtiene un buen compromiso desempeño/precio que puede redituar la utilización de MRMOGA en un cúmulo de este tamaño para varios problemas del mundo real. Por otro lado, puesto que la aceleración para 10-14 procesadores no se aleja de la lineal, aún hay un amplia gama de problemas para los cuales puede ser conveniente usar ese número de procesadores, a pesar de no tener un buen compromiso desempeño/precio.

Capítulo 7

Conclusiones y trabajo futuro

Sumario

7.1. Resumen del trabajo realizado

El objetivo principal de este trabajo de tesis fue proponer un nuevo algoritmo evolutivo multiobjetivo paralelo (pAEMO) que demostrara ser competitivo con respecto a los algoritmos evolutivos multiobjetivo (AEMOs) representativos del estado del arte.

Para llevar a cabo este objetivo se realizó un estudio de los pAEMOs que existen actualmente. En el capítulo 4 se presentan algunos de estos algoritmos que fueron seleccionados con base en la técnicas innovadoras que presentaban.

Como producto de este análisis se propuso el algoritmo genético multiobjetivo con múltiples resoluciones (MRMOGA) (descrito detalladamente en el capítulo 5). Esta técnica se basa en el paradigma de isla con nodos heterogéneos y se caracteriza por codificar las soluciones con una resolución distinta en cada isla. Este esquema consigue dividir el espacio de búsqueda en el espacio de las variables de decisión.

A lo largo del estudio de los trabajos de otros pAEMOs se identificaron algunas carencias. En la mayoría de estos trabajos no se discuten detalladamente los aspectos relacionados con el desarrollo y la implementación del pAEMO. Asimismo, en varios de estos trabajos previos no se realiza un estudio del algoritmo que evalúe tanto la eficacia como la eficiencia. En algunos trabajos previos, por ejemplo, el desempeño del algoritmo se coteja solamente contra los resultados de AEMOs secuenciales, y en otros no se evalúa la eficiencia. En vista de esta situación, en el presente trabajo de tesis se realizó una evaluación del desempeño de la técnica propuesta que contempló tanto la eficacia como la eficiencia.

Para evaluar el desempeño de MRMOGA se compararon sus resultados con los obtenidos por una versión paralela del NSGA-II. La evaluación se realizó utilizando 9 problemas multiobjetivo que reúnen diferentes características que causan dificultades a un AEMO. El grupo de problemas más importante fue elegido por tener un espacio de búsqueda grande que los hace apropiados

para resolverse mediante un pAEMO. Otros problemas fueron seleccionados por tener un frente de Pareto real desconectado que pone a prueba la capacidad que tiene un pAEMO para mantener una población diversa. Asimismo la evaluación contempló tres problemas con restricciones.

7.2. Conclusiones

A partir de los experimentos descritos en el capítulo 6 se desprenden los siguientes resultados:

- La estrategia que usa MRMOGA para dividir el espacio de búsqueda mediante distintas resoluciones mejora considerablemente la convergencia de un algoritmo paralelo. En la mayoría de los problemas, MRMOGA fue ampliamente superado por el PNSGA-II en convergencia y distribución cuando ambos utilizaban un procesador[1]. Sin embargo, con más de un procesador, MRMOGA mejoró su convergencia notablemente hasta el punto que superó a PNSGA-II. Esta ganancia en la convergencia nos revela el beneficio de la técnica propuesta.

- Cuando la técnica propuesta utiliza la topología simple tiene un mejor desempeño en cuanto a convergencia y distribución cuando se usan pocos procesadores (1 a 8). Por otro lado, con la topología completa se obtienen mejores resultados cuando se usan varios procesadores (8 o más).

- La técnica propuesta tiene más valía cuando el espacio de búsqueda es considerablemente grande. En los problemas que tienen más de 20 variables, MRMOGA obtuvo mejores tiempos que PNSGA-II.

- La división del espacio de búsqueda mediante la variación de la resolución produce conjuntos $FP_{conocido}$ que cubren toda la extensión del frente de Pareto real. Sin embargo, como se observó en el capítulo 6, la técnica propuesta no consigue tan buenos resultados en la distribución de las soluciones.

- La exploración del espacio de búsqueda con distintas resoluciones no solamente acelera la convergencia, sino que encuentra soluciones que de otra manera sería difícil encontrar. Esto se pone de manifiesto en los problemas con restricciones. Por ejemplo, cuando no se utilizó la técnica (MRMOGA con un procesador) en el problema OSY, no se encontró ninguna solución que fuera parte del frente de Pareto real. En cambio con la técnica se encontraron varias soluciones que pertenecen al frente de Pareto.

7.3. Trabajo futuro

En vista de que uno de sus puntos débiles es la distribución de las soluciones, es necesario incorporar un nuevo mecanismo para conseguir una mejor distribución. La rejilla adaptativa funcionó bien al colocar las soluciones en todas las regiones del espacio de búsqueda, pero en el interior de esas regiones las soluciones no estaban uniformemente espaciadas. Una de las opciones para solucionar este problema sería incorporar la dominancia épsilon. Esta técnica se podría incorporar al optimizador de cada isla o aplicarse solamente al final de la búsqueda

[1] Al utilizar un procesador se aísla cada optimizador, de manera que MRMOGA no divide el espacio de búsqueda y PNSGA-II se reduce al NSGA-II original.

cuando el procesador raíz tiene que combinar los frentes no dominados provenientes de todos los procesadores.

Debido a la notable capacidad que mostró MRMOGA para resolver problemas con restricciones sería interesante realizar un estudio que contenga más problemas con restricciones para observar con mayor detalle el desempeño del algoritmo con estos problemas.

Con la topología completa no se consiguió tan buena convergencia como con la topología simple al usar pocos procesadores. No obstante, su capacidad de convergencia para más procesadores y su capacidad de distribución, nos hacen pensar que una nueva estrategia para reemplazar los individuos por los migrantes, resultaría en un excelente algoritmo. También sería conveniente experimentar con otras topologías para comunicar a los procesadores.

El conjunto de problemas elegidos para evaluar MRMOGA no tienen funciones objetivo costosas respecto al tiempo. Sería interesante realizar un estudio que contenga problemas del mundo real con funciones objetivo costosas que permitan observar de una manera más práctica la eficiencia de MRMOGA.

En este trabajo de tesis se aplicó la estrategia de división del espacio de búsqueda a un AG generacional (es decir, la población actual es reemplazada completamente cada generación). Sin embargo, sería interesante utilizar un AG estacionario (*steady-state*) (solamente una fracción de la población se reproduce y se reemplaza en cada generación), ya que en [Rogers99] se afirma que los AG estacionarios pueden reproducir el comportamiento de un AG generacional con la mitad del costo computacional.

El campo de la paralelización de AEMOs es aún joven y faltan muchas cosas por hacer. Una de las más importantes es definir una metodología más o menos estándar para comparar el desempeño de los pAEMOs que facilite la comparación y reproducción de los resultados para estudios posteriores. Esta metodología debería contemplar un conjunto de métricas específicas para evaluar la eficacia y la eficiencia de un pAEMO. En la mayoría de las publicaciones actuales no se reportan resultados de la eficacia (aceleración, eficiencia, etc.). También se necesita establecer una serie de problemas de prueba apropiados para comparar los AEMOs paralelos.

Bibliografía

[Alba99a] ALBA TORRES, Enrique. «¿Puede un algoritmo evolutivo paralelo proporcionar ganancia superlineal?» En *Actas de la VIII Conferencia de la Asociación Española para la Inteligencia Artificial. III Jornadas de Transferencia Tecnológica de Inteligencia Artificial (CAEPIA-TTIA 99)*. Murcia, España, 1999, tomo 2, 89–97.

[Alba02] —. «Parallel evolutionary algorithms can achieve super-linear performance.» *Information Processing Letters, Elsevier*, 82, nº 1, (2002), 7–13.

[Alba99] ALBA TORRES, Enrique, y José M.ª TROYA LINERO. «A Survey of Parallel Distributed Genetic Algorithms.» *Complexity*, 4, nº 4, (1999), 31–51.

[Alba01] —. «Analyzing Synchronous and Asynchronous Parallel Distributed Genetic Algorithms.» *Future Generation Computer Systems*, 17, nº 4, (2001), 451–465.

[Back97] BÄCK, Thomas, David B. FOGEL, y Zbigniew MICHALEWICZ (eds.) *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, Nueva York, EE. UU., 1997.

[Baker87] BAKER, James Edward. «Reducing Bias and Inefficiency in the Selection Algorithm.» En *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (John J. Grefenstette, ed.). Lawrence Erlbaum, Hillsdale, Nueva Jersey, 1987, 14–22.

[Belding95] BELDING, Theodore C. «The Distributed Genetic Algorithm Revisited.» En *Proceedings of the Sixth International Conference on Genetic Algorithms* (Larry Eshelman, ed.). Morgan Kaufmann, San Francisco, CA, 1995, 114–121.

[Bell92] BELL, Gordon. «Ultracomputers: A Teraflop Before Its Time.» *Communications of the ACM*, 35, nº 8, (1992), 27–47.

[Booker82] BOOKER, Lashon B. *Intelligent Behavior as an Adaptation to the Task Environment*. Tesis Doctoral, Logic of Computers Group, University of Michigan, Ann Arbor, Michigan, EE. UU., 1982.

[Brindle81] BRINDLE, A. *Genetic Algorithms for Function Optimization*. Tesis Doctoral, Department of Computer Science, University of Alberta, Edmonton, Alberta, Canadá, 1981.

[Cantu00] CANTÚ PAZ, Erick. *Efficient and Accurate Parallel Genetic Algorithms*. Boston: Kluwer Academic Publishers, 2002.

[Coello99] COELLO COELLO, Carlos A. «A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques.» *Knowledge and Information Systems. An International Journal*, 1, nº 3, (1999), 269–308.

[Coello98] COELLO COELLO, Carlos A., y Alan D. CHRISTIANSEN. «Two New GA-based methods for multiobjective optimization.» *Civil Engineering Systems*, 15, nº 3, (1998), 207–243.

[Coello01] COELLO COELLO, Carlos A., y Gregorio TOSCANO PULIDO. «Multiobjective Optimization using a Micro-Genetic Algorithm.» En *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)* (Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, y Edmund Burke, eds.). Morgan Kaufmann Publishers, San Francisco, California, 2001, 274–282.

[Coello02] COELLO COELLO, Carlos A., David A. Van VELDHUIZEN, y Gary B. LAMONT. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Nueva York: Kluwer Academic Publishers, 2002.

[Cvetkovic02] CVETKOVIĆ, Dragan, y Ian C. PARMEE. «Preferences and their Application in Evolutionary Multiobjective Optimisation.» *IEEE Transactions on Evolutionary Computation*, 6, nº 1, (2002), 42–57.

[Darwin59] DARWIN, Charles Robert. *The Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life*. Penguin Books Ltd., Nueva York, EE. UU., 1959. Publicado originalmente en 1859.

[Das97] DAS, Indraneel, y John DENNIS. «A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems.» *Structural Optimization*, 14, nº 1, (1997), 63–69.

[Deb98] DEB, Kalyanmoy. «Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems.» *Inf. Téc. CI-49/98*, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.

[Deb99] —. «Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design.» En *Evolutionary Algorithms in Engineering and Computer Science* (Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, y Jacques Periaux, eds.), John Wiley & Sons, Ltd, Chichester, Reino Unido, 1999. 135–161.

[Deb99b] —. «Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems.» *Evolutionary Computation*, 7, nº 3, (1999), 205–230.

[Deb99a] —. «Solving Goal Programming Problems Using Multi-Objective Genetic Algorithms.» En *1999 Congress on Evolutionary Computation*. IEEE Service Center, Washington, D.C., 1999, 77–84.

[Deb00] DEB, Kalyanmoy, Samir AGRAWAL, Amrit PRATAB, y T. MEYARIVAN. «A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II.» En *Proceedings of the Parallel Problem Solving from Nature VI Conference* (Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, y Hans-Paul Schwefel, eds.). Springer. Lecture Notes in Computer Science No. 1917, Paris, Francia, 2000, 849–858.

[Deb89] DEB, Kalyanmoy, y David Edward GOLDBERG. «An investigation of niche and species formation in genetic function optimization.» En *Proceedings of the Third International Conference on Genetic Algorithms* (J. David Schaffer, ed.). Georges Mason University, Morgan Kaufmann Publishers, San Mateo, California, 1989, 42–50.

[Deb02] DEB, Kalyanmoy, Lothar THIELE, Marco LAUMANNS, y Eckart ZITZLER. «Scalable Multi-Objective Optimization Test Problems.» En *Congress on Evolutionary Computation (CEC'2002)*. IEEE Service Center, Piscataway, New Jersey, 2002, tomo 1, 825–830.

[Deb03] DEB, Kalyanmoy, Pawan ZOPE, y Abhishek JAIN. «Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms.» En *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003* (Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, y Lothar Thiele, eds.). Springer. Lecture Notes in Computer Science. Volume 2632, Faro, Portugal, 2003, 534–549.

[DeJong75] DEJONG, K. A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Tesis Doctoral, University of Michigan, Ann Arbor, Michigan, EE. UU., 1975.

[Edgeworth81] EDGEWORTH, Francis Ysidro. *Mathematical Physics*. P. Keagan, 1881.

[Eiben03] EIBEN, Agoston E., y James E. SMITH. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[Flynn72] FLYNN, Michael J. «Some computer organizations and their effectiveness.» *IEEE Transactions on Computers*, 21, nº 9, (1972), 948–960.

[Fogel64] FOGEL, Lawrence J. *On the Organization of Intellect*. Tesis Doctoral, University of California, Los Angeles, California, EE. UU., 1964.

[Fogel99] —. *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., Nueva York, 1999.

[Fonseca93] FONSECA, Carlos M., y Peter J. FLEMING. «Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization.» En *Proceedings of the Fifth International Conference on Genetic Algorithms* (Stephanie Forrest, ed.). University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, San Mateo, California, 1993, 416–423.

[Fonseca95] —. «An Overview of Evolutionary Algorithms in Multiobjective Optimization.» *Evolutionary Computation*, 3, nº 1, (1995), 1–16.

[Freitas02] FREITAS, Alex. A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer Verlag, 2002, 1.ª edón.

[Cunha97] GASPAR CUNHA, A., Pedro OLIVEIRA, y José A. COVAS. «Use of Genetic Algorithms in Multicriteria Optimization to Solve Industrial Problems.» En *Proceedings of the Seventh International Conference on Genetic Algorithms* (Thomas Bäck, ed.). Michigan State University, Morgan Kaufmann Publishers, San Mateo, California, 1997, 682–688.

[Goldberg89] GOLDBERG, David Edward. *Genetic Algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[Goldberg91] GOLDBERG, David Edward, y Kalyanmoy DEB. *A comparison of selection schemes used in genetic algorithms*, Morgan Kaufmann, San Mateo, California, EE. UU., 1991 69–93.

[Goldberg87] GOLDBERG, David Edward, y J. RICHARDSON. «Genetic algorithms with sharing for multimodal function optimization.» En *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (John. J. Grefenstette, ed.). Lawrence Erlbaum, Hillsdale, Nueva Jersey, EE. UU., 1987, 41–49.

[Hajela92] HAJELA, P., y C. Y. LIN. «Genetic search strategies in multicriterion optimal design.» *Structural Optimization*, 4, (1992), 99–107.

[Hiroyasu00] HIROYASU, Tomoyuki, Mitsunori MIKI, y Shinya WATANABE. «The New Model of Parallel Genetic Algorithm in Multi-Objective Optimization Problems—Divided Range Multi-Objective Genetic Algorithm—.» En *2000 Congress on Evolutionary Computation*. IEEE Service Center, Piscataway, New Jersey, 2000, tomo 1, 333–340.

[Holland75] HOLLAND, John H. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University Press, 1975.

[Horn93] HORN, Jeffrey, y Nicholas NAFPLIOTIS. «Multiobjective Optimization using the Niched Pareto Genetic Algorithm.» *Inf. Téc. IlliGAl Report 93005*, University of Illinois at Urbana-Champaign, Urbana, Illinois, EE. UU., 1993.

[Hwang93] HWANG, Kai. *Advanced Computer Architecture: Parallelism, Scalability, Programmability*. McGraw-Hill Book Co., Nueva York, EE. UU., 1993.

[Karp90] KARP, Alan H., y Horace P. FLATT. «Measuring parallel processor performance.» *Communications of the ACM*, 33, nº 5, (1990), 539–543.

[Kirley01] KIRLEY, Michael. «MEA: A metapopulation evolutionary algorithm for multiobjective optimisation problems.» En *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*. IEEE Service Center, Piscataway, Nueva Jersey, EE. UU., 2001, tomo 2, 949–956.

[Kita96] KITA, Hajime, Yasuyuki YABUMOTO, Naoki MORI, y Yoshikazu NISHIKAWA. «Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm.» En *Parallel Problem Solving from Nature—PPSN IV* (Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, y Hans-Paul Schwefel, eds.). Springer-Verlag, Berlín, Alemania, 1996, Lecture Notes in Computer Science, 504–512.

[Knowles99] KNOWLES, Joshua D., y David W. CORNE. «The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation.» En *1999 Congress on Evolutionary Computation*. IEEE Service Center, Washington, D.C., 1999, 98–105.

[Kumar94] KUMAR, Vipin, y George Karypis ANANTH GRAMA, Anshul Gupta. *Introduction to Parallel Computing: design and analysis of parallel algorithms*. Benjamin Cummings Publishing Company, Redwood City, California, EE. UU., 1994.

[Kursawe91] KURSAWE, Frank. «A Variant of Evolution Strategies for Vector Optimization.» En *Parallel Problem Solving from Nature. 1st Workshop, PPSN I* (H. P. Schwefel, y R. Männer, eds.). Springer-Verlag, Berlín, Alemania, 1991, tomo 496 de *Lecture Notes in Computer Science Vol. 496*, 193–197.

[Lin94] LIN, Shyh-Chang, William F. PUNCH III, y Erik D. GOODMAN. «Coarse-grain genetic algorithms, categorization and new approaches.» En *Sixth IEEE Symposium on Parallel and Distributed Processing*. IEEE Computer Society Press, Dallas, Texas, EE. UU., 1994, 28–37.

[Mahfoud95] MAHFOUD, Samir W. *Niching Methods for Genetic Algorithms*. Tesis Doctoral, University of Illinois at Urbana-Champaign, Urbana, Illinois, EE. UU., Mayo 1995.

[Mendel01] MENDEL, Gregor Johann. «Experiments in plant hybridisation.» *Journal of the Royal Horticultural Society*, , nº 26, (1901), 1–32.

[Michalewicz96] MICHALEWICZ, Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Nueva York, EE. UU., 1996.

[Michalewicz00] MICHALEWICZ, Zbigniew, y David B. FOGEL. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[Miettinen98] MIETTINEN, Kaisa M. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, EE. UU., 1998.

[Osyczka84] OSYCZKA, Andrzej. *Multicreterion Optimization in engineering with FORTRAN programs*. Ellis Horwood Limited, 1984.

[Osyczka95] OSYCZKA, Andrzej, y Sourav KUNDU. «A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm.» *Structural Optimization*, 10, (1995), 94–99.

[Pareto96] PARETO, Vilfredo. *Cours D'Economie Politique*. F. Rouge, 1896.

[Purshouse03] PURSHOUSE, Robin Charles. *On the Evolutionary Optimisation of Many Objectives*. Tesis Doctoral, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, Reino Unido, Septiembre 2003.

[Quagliarella98] QUAGLIARELLA, Domenico, y Alessandro VICINI. «Sub-population Policies for a Parallel Multiobjective Genetic Algorithm with Applications to Wing Design.» En *1998 IEEE International Conference On Systems, Man, And Cybernetics* (Jarmo T. Alander, ed.). Institute of Electrical and Electronic Engineers (IEEE), San Diego, California, EE. UU., 1998, 3142–3147.

[Quinn94] QUINN, Michael Jay. *Parallel Computing: Theory and Practice*. McGraw-Hill, Nueva York, EE. UU., 1994.

[Rechenberg73] RECHENBERG, Ingo. «Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.» *Inf. téc.*, Technical University of Berlin, Frommann-Holzboog, Sttutgart, Alemania, 1973.

[Rekiek00a] REKIEK, Brahim. *Assembly Line Design (multiple objective grouping genetic algorithm and the balancing of mixed-model hybrid assembly line)*. Tesis Doctoral, Free University of Brussels, CAD/CAM Department, Bruselas, Bélgica, Diciembre 2000.

[Rogers99] ROGERS, A., y A. PRÜGEL-BENNETT. «Genetic Drift in Genetic Algorithm Selection Schemes.» *IEEE Transactions on Evolutionary Computation*, 3, (1999), 298–303.

[Rosenberg67] ROSENBERG, R. S. *Simulation of genetic populations with biochemical properties*. Tesis Doctoral, University of Michigan, Ann Harbor, Michigan, EE. UU., 1967.

[Rowe96] ROWE, Jon, Kevin VINSEN, y Nick MARVIN. «Parallel GAs for Multiobjective Functions.» En *Proceedings of the Second Nordic Workshop on Genetic Algorithms and Their Applications (2NWGA)* (Jarmo T. Alander, ed.). University of Vaasa, Vaasa, Finlandia, 1996, 61–70.

[Rudolph00] RUDOLPH, Günter, y Alexandru AGAPIE. «Convergence Properties of Some Multi-Objective Evolutionary Algorithms.» En *Proceedings of the 2000 Conference on Evolutionary Computation*. IEEE Press, Piscataway, Nueva Jersey, 2000, tomo 2, 1010–1016.

[Sawai99] SAWAI, H., y S. ADACHI. «Parallel distributed processing of a parameter-free ga by using hierarchical migration methods.» En *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)* (W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, y R. E. Smith, eds.). Morgan Kaufmann, San Francisco, California, EE. UU., 1999, tomo 1, 579–586.

[Schaffer84] SCHAFFER, J. David. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. Tesis Doctoral, Vanderbilt University, 1984.

[Schott95] SCHOTT, Jason R. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. Proyecto Fin de Carrera, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, Mayo 1995.

[Schwefel74] SCHWEFEL, Hans-Paul. «Adaptive Mechanismen in der biologischen Evolution un ihr Einfluß auf die Evolutionsgeschwindigkeit.» *Inf. téc. del Working Group of Bionics and Evolution Techniques at the Institute for Measurement and Control Technology Re 215/3*, Technical University of Berlin, Julio 1974.

[Schwefel77] —. *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, tomo 26 de *Interdisciplinary Systems Research*. Birkhäeuser Verlag, Basel, Alemania, 1977.

[Sefrioui00] SEFRIOUI, M., y J. PERIAUX. «Nash Genetic Algorithms: examples and applications.» En *2000 Congress on Evolutionary Computation*. IEEE Service Center, San Diego, California, 2000, tomo 1, 509–516.

[Shonkwiler93] SHONKWILER, Ron. «Parallel Genetic Algorithms.» En *Proceedings of the 5th International Conference on Genetic Algorithms* (Stephanie Forrest, ed.). Morgan Kaufmann, Urbana-Champaign, Illinois, EE. UU., 1993, 199–205.

[Srinivas94] SRINIVAS, N., y Kalyanmoy DEB. «Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms.» *Evolutionary Computation*, 2, nº 3, (1994), 221–248.

[Tamaki96] TAMAKI, Hisashi, Hajime KITA, y Shigenobu KOBAYASHI. «Multi-Objective Optimization by Genetic Algorithms : A Review.» En *Proceedings of the 1996 International Conference on Evolutionary Computation (ICEC'96)* (Toshio Fukuda, y Takeshi Furuhashi, eds.). IEEE, Nagoya, Japón, 1996, 517–522.

[Tanenbaum02] TANENBAUM, Andrew S., y Maarten VAN STEEN. *Distributed Systems: Principles and Paradigms*. Prentice Hall, Upper Saddle River, New Jersey, EE. UU., 2002.

[Tomassini99] TOMASSINI, Marco. «Parallel and distributed evolutionary algorithms: A review.» En *Evolutionary Algorithms in Engineering and Computer Science* (K. Miettinen, M. Mäkelä, P. Neittaanmäki, y J. Periaux, eds.). John Wiley and Sons, 1999, 113–133.

[Toscano01] TOSCANO PULIDO, Gregorio. *Optimización multiobjetivo usando un micro algoritmo genético*. Proyecto Fin de Carrera, Universidad Veracruzana - LANIA, Xalapa, Veracruz, México, Septiembre 2001.

[Veldhuizen99] VELDHUIZEN, David A. Van. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Tesis Doctoral, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Mayo 1999.

[Veldhuizen03] VELDHUIZEN, David A. Van, Jesse B. ZYDALLIS, y Gary B. LAMONT. «Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms.» *IEEE Transactions on Evolutionary Computation*, 7, nº 3, (2003), 144–173.

[Lucken03] VON LÜCKEN, Christian. *Algoritmos evolutivos para optimización multi-objetivo: un estudio comparativo en un ambiente paralelo asíncrono*. Proyecto Fin de Carrera, Universidad Nacional de Asunción, San Lorenzo, Paraguay, Diciembre 2003.

[Watanabe02] WATANABE, Shinya, Tomoyuki HIROYASU, y Mitsunori MIKI. «LCGA: Local Cultivation Genetic Algorithm for Multi-Objective Optimization Problems.» En *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)* (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, y N. Jonoska, eds.). Morgan Kaufmann Publishers, San Francisco, California, 2002, 702.

[Weismann93] WEISMANN, August Friedrich Leopold (ed.) *The Germ Plasm: A Theory of Heredity*. Scott, 1893.

[Wetzel83] WETZEL, A. «Evaluation of the effectiveness of genetic algorithms in combinatorial optimization.», 1983. University of Pittsburgh.

[Whitley89] WHITLEY, Darrell. «The GENITOR Algorithm and Selection Pressure: Why Rank-Based of Reproductive Trials is Best.» En *Proceedings of the Third International Conference on Genetic Algorithms* (J. David Schaffer, ed.). Morgan Kaufmann Publishers, San Mateo, California, 1989, 116–121.

[Zebulum98] ZEBULUM, R. S., M. A. PACHECO, y M. VELLASCO. «A multi-objective optimisation methodology applied to the synthesis of low-power operational amplifiers.» En *Proceedings of the XIII International Conference in Microelectronics and Packaging* (Ivan Jorge Cheuri, y Carlos Alberto dos Reis Filho, eds.). Curitiba, Brazil, 1998, tomo 1, 264–271.

[Zhu02] ZHU, Zhong-Yao, y Kwong-Sak LEUNG. «Asynchronous Self-Adjustable Island Genetic Algorithm for Multi-Objective Optimization Problems.» En *Congress on Evolutionary Computation (CEC'2002)*. IEEE Service Center, Piscataway, Nueva Jersey, 2002, tomo 1, 837–842.

[Zitzler99a] ZITZLER, Eckart. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Tesis Doctoral, Swiss Federal Institute of Technology (ETH), Zurich, Suiza, Noviembre 1999.

[Zitzler99b] ZITZLER, Eckart, Kalyanmoy DEB, y Lothar THIELE. «Comparison of Multiobjective Evolutionary Algorithms: Empirical Results.» *Inf. Téc. 70*, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Suiza, Diciembre 1999.

[Zitzler01] ZITZLER, Eckart, Marco LAUMANNS, y Lothar THIELE. «SPEA2: Improving the Strength Pareto Evolutionary Algorithm.» En *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems* (K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, y T. Fogarty, eds.). Atenas, Grecia, 2002, 95–100.

[Zitzler99] ZITZLER, Eckart, y Lothar THIELE. «Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach.» *IEEE Transactions on Evolutionary Computation*, 3, nº 4, (1999), 257–271.

Apéndice A

Glosario de términos genéticos

ADN. El ácido desoxirribonucleico (ADN) es el material genético fundamental de todos los organismos. Es portador de las instrucciones genéticas para la elaboración de los organismos vivientes. El ADN es una macromolécula doblemente trenzada que tiene una estructura helicoidal.

Alelo. Una de las formas alternativas de un gen que ocupa un locus (posición) en un cromosoma. Diferentes alelos de un gen producen variaciones en las características hereditarias tales como el color del cabello o el tipo de sangre.

Aptitud. La adaptación de una organismo juzgado en cuanto a su fertilidad.

Cromosoma. Una de las cadenas de ADN que se encuentra en el núcleo de las células. Los cromosomas son el medio para transportar información genética.

Deme. Parte de una población que representa un grupo parcialmente aislado de individuos reproductivos.

Desvío genético. Variaciones de la frecuencia de genes en una pequeña población debido al azar de la selección.

Diploide. El término diploide describe el número completo de copias del genoma en una célula determinada. Di significa dos y ploide se refiere al número de copias. Así, la inmensa mayoría de las células normales contienen dos copias del genoma, cada una proveniente de cada progenitor y se conocen como células diploides.

Dominante. Es la condición por la cual un miembro de un par alélico se manifiesta en el fenotipo de un individuo excluyendo la expresión del otro alelo.

Fenotipo. Las características de un organismo que resulta de la reacción de un genotipo dado con un ambiente particular.

Gameto. Son células que llevan información genética de los padres con el propósito de efectuar reproducción sexual.

Haploide. Célula que contiene un solo cromosoma o conjunto de cromosomas, cada uno de los cuales consiste de una sola secuencia de genes.

Gen. Es la unidad física y funcional de la herencia que se hereda de padres a hijos. Los genes están compuestos por ADN y la mayoría de ellos contiene la información para elaborar una proteína específica.

Genotipo. Es la composición genética que un organismo hereda de sus padres y no se muestra como características externas. El genotipo está dado por los alelos localizados en un locus particular de un cromosoma de un organismo.

Locus. Es el lugar del cromosoma donde está localizado un gen específico.

Migración. Es la transferencia de los genes de un individuo de una subpoblación a otra.

Mutación. Un cambio en uno o más de las bases del ADN que produce la formación de una proteína anormal. En ocasiones una mutación puede mejorar la probabilidad de supervivencia de un organismo y pasar el cambio positivo a sus descendientes. Las mutaciones solamente son heredadas si ocurren en las células germinales destinadas para generar los gametos.

Población. Una población es un conjunto de individuos de la misma especie que viven en un lugar geográfico determinado (nicho ecológico) y que real o potencialmente son capaces de cruzarse entre sí, compartiendo un acervo común de genes.

Selección. Un proceso en el cual aquellos organismo mejor adaptados a su ambiente tienden a sobrevivir y reproducirse a expensas de los menos aptos.