PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO DE SISTEMAS Y COMPUTACIÓN

"DESARROLLO DE UNA GUÍA METODOLÓGICA SOBRE COMPUTACIÓN EVOLUTIVA Y ALGORITMOS GENÉTICOS, PARA LA OPTIMIZACIÓN EVOLUTIVA MULTIOBJETIVO"

CLAUDIA MABEL SUBÍA PICO

DIRECTOR: ING. JAVIER CÓNDOR

QUITO, 2014

Ing. Javier Cóndor

Director de Tesis

Ing. Damián Nicolalde

Corrector de Tesis

Fis. Jorge Aguilar

Corrector de Tesis

Dedicatoria

Dedico este trabajo a mi familia, por tener siempre su apoyo en todos los proyectos que he emprendido en mi vida. A mi padre por escucharme y permitirme desarrollar mi imaginación. A mi madre por su presencia incondicional. A mi “nano”, Yosho, por ser siempre una de las razones para superarme y demostrarle que soy un buen ejemplo a seguir. A mis amigos que siempre han estado ahí apoyándome. Y finalmente como dijo alguna vez un gran mago “Draco Dormiens Nunquam Titillandus”.

“No importa las condiciones en las que uno nace, sino lo que llega a ser cuando crece”

Albus Dumbledore, Harry Potter y el Cáliz de Fuego

Claudia Subía

Contenido

PRÓLOGO

En el presente trabajo se pretende generar una guía la cual se enfocará en los principios básicos de la Computación Evolutiva y de la optimización multiobjetivo. Se revisará la parte teórica así como los conceptos y técnicas desarrolladas para el diseño de algoritmos que permiten la solución de problemas multiobjetivo.

En el primer capítulo se revisarán los conceptos básicos que llevaron a la Computación Evolutiva. Se estudiará sobre los antecedentes en la biología y genética que sustentan los conceptos de evolución de los que se deriva la Computación Evolutiva. Se establecerán las definiciones y conceptos básicos relacionados. También se describe brevemente sobre el origen de los conceptos de evolución y la Teoría de la Evolución de las Especies.

El segundo capítulo describe los conceptos de Computación Evolutiva. Comenzando por la historia de la Inteligencia artificial, dando un especial énfasis en la historia de la Computación Evolutiva. Adicionalmente se revisará la teoría de los algoritmos evolutivos tanto su estructura como las clases de algoritmos evolutivos existentes.

En el tercer capítulo se revisará los conceptos sobre los algoritmos evolutivos multiobjetivo. Enfocándose primeramente en el Óptimo de Edgeworth-Pareto, para a continuación revisar las medidas de rendimiento de los algoritmos.

En el cuarto capítulo se aplicará el algoritmo NSGA-II para la optimización de problemas con múltiples objetivos. Inicialmente se hará una introducción al algoritmo NSGA-II. Se resumirá el funcionamiento del algoritmo en forma de diagrama de bloques. Finalmente se pasará a la aplicación y análisis de los resultados obtenidos.

En el quinto capítulo se establecerá conclusiones y recomendaciones que se dará a través del análisis del rendimiento del algoritmo utilizado y los conceptos básicos estudiados en toda la guía,

con la finalidad de determinar las ventajas y desventajas del uso de algoritmos evolutivos multiobjetivos para la optimización de procesos.

INTRODUCCIÓN

El campo de la Computación Evolutiva es un conjunto de personas, ideas y aplicaciones cuyos inicios pueden ser trazados desde la década de 1930, aunque su mayor desarrollo se dio con el surgimiento de un mayor desarrollo de la tecnología computacional relativamente no tan cara en la década de 1960. Al tener ésta tecnología disponible se hizo posible el uso de la simulación por computadora para el análisis de sistemas mucho más complejos que aquellos que podían ser analizados matemáticamente.

El campo de la Computación Evolutiva ha experimentado un enorme crecimiento en los últimos 25 años, dando lugar a una amplia variedad de algoritmos y aplicaciones evolutivas. El resultado plantea un dilema interesante para muchos profesionales en el sentido de que, con una amplia variedad de algoritmos y métodos tales, a menudo es difícil en sí las relaciones entre ellos, evaluar las fortalezas y debilidades, y tomar buenas decisiones para nuevas áreas de aplicación.

En la vida real se pueden encontrar varios problemas que son difíciles de resolver debido a que entran en conflicto los múltiples objetivos que se desea cumplir para resolverlos, especialmente cuando se desea optimizar. La Computación Evolutiva nos permite modelar escenarios de la vida real por medio de algoritmos evolutivos. Conceptos como selección natural y evolución son utilizados de manera más formal al momento de tratar de simular la realidad.

En problemas multiobjetivo los métodos de modelamiento matemáticos no son los más adecuados debido a que no se puede encontrar múltiples soluciones en una sola ejecución, no son eficientes al momento de buscar soluciones óptimas.

Los algoritmos genéticos sin embargo poseen muchas características deseables para este tipo de problemas ya que usan conceptos de genética para representar un problema y luego simular la evolución mediante dos principios básicos, selección y variación, para determinar la mejor solución.

Éste tipo de algoritmos son utilizados para la optimización multiobjetivo ya que han probado ser un mecanismo de búsqueda general, robusto y poderoso.

La optimización evolutiva multiobjetivo permite resolver problemas complejos de escenarios de la vida real de una forma más eficiente y en la cual se pueden obtener varias soluciones óptimas. Para ello es necesario tener una guía que permita comprender los conceptos básicos de la Computación Evolutiva y también de las metodologías aplicadas al momento de diseñar algoritmos para la solución de problemas multiobjetivo.

1. ANTECEDENTES FUNDAMENTOS TEÓRICOS

> *"Hay grandeza en esta concepción de la vida,... que mientras este planeta ha ido girando según la constante ley de la gravitación, se han desarrollado y se están desarrollando, a partir de un comienzo tan sencillo, infinidad de formas cada vez más bellas y maravillosas"*
>
> *Charles Darwin* (Darwin, 1859, pág. 280)

En la actualidad se usan técnicas computacionales para la solución de una amplia variedad de problemas, ya sean cotidianos hasta más específicos dentro de las industrias y áreas científicas.

Los algoritmos evolutivos, se basan en comportamientos que se estudian en la biología, principalmente los principios de evolución, tales como selección natural, mutación y herencia.

La rama de la computación que se enfoca en el estudio de éstas técnicas se la conoce como Computación Evolutiva.

1.1. Biología y Biología Molecular

La biología molecular es la rama de la biología que estudia los organismos a nivel molecular es decir, se enfoca principalmente en los sub elementos de la célula, tales como el ADN[1], ARN[2] y demás enzimas que la conforman.

[1] Ácido desoxirribonucleico…
[2] Ácido ribonucleico…

La biología molecular, al ser parte de la biología, tiene gran relación con las demás ciencias naturales, con las cuales interactúa constantemente para así llegar a una mejor comprensión de la estructura y origen de los individuos que existen en la naturaleza.

Como un buen ejemplo se puede tomar la relación que se genera entre la biología molecular y la bioquímica. La bioquímica estudia las reacciones químicas que se producen en los organismos biológicos. Gracias a ello, la biología molecular puede comprender las reacciones que se generan al sintetizar las diferentes enzimas. (Lacadena, 1999)

BIOLOGÍA	Biología Molecular	Molécula
	Biofísica	
	Bioquímica	
	Citología	Célula
	Histología	
	Morfología	
	Fisiología	Organismo
	Etología	
	Embriología	
	Neurofisiología	
	Biología del Desarrollo	
	Genética	Población
	Socio biología	
	Ecología	

Cuadro 1.1. Relación entre la biología y las demás ciencias

1.2. Genética

Genética es la ciencia que permite el estudio del proceso que se da al momento en el que el individuo transmite ciertas características propias a la siguiente generación de individuos similares y los efectos que ésta transmisión produce en las siguientes generaciones. A este proceso se lo conoce como herencia.

Las características que se transmiten desde una generación de individuos a otra nueva (padres a hijos) están contenidas dentro del material genético de cada individuo. Ese material genético contiene la información sobre todas las características de un individuo y está formado por una cantidad de elementos conocidos como proteínas. (Aula Virtual de Biología, 2002)

1.2.1. Proteínas

Las proteínas son los materiales que conforman la estructura básica de toda materia viva y que desempeñan las funciones indispensables en todos los tejidos (metabólicas y reguladoras).

Al ser el elemento fundamental en la materia viva, contienen los elementos que definen y diferencian a cada ser vivo, pues son la base para el ADN.

La estructura química de las proteínas está conformada por C (carbono), H (hidrógeno), O (oxígeno), y N (nitrógeno) y en pequeñas cantidades por S (azufre)y P (fósforo) y casi en cantidades despreciables por Fe (hierro), Cu (cobre), Mg (magnesio) y I (yodo), que al unirse conforman moléculas conocidas como Aminoácidos. (Aula Virtual de Biología, 2002)

1.2.2. Ácidos Nucleicos

Son compuestos cual composición química está formada por C (carbono), H (oxígeno) y N (nitrógeno). Reaccionan de manera ácida al contacto con el agua, de lo cual se deriva su nombre.

Sus principales funciones son la transmisión de las características genéticas de una generación a otra y la síntesis de proteínas. (Aula Virtual de Biología, 2002)

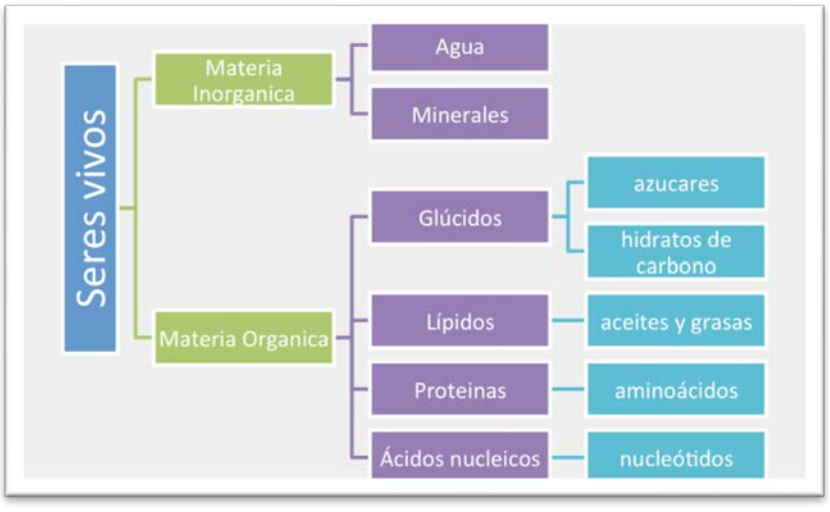
1.2.2.1. Ácido desoxirribonucleico (ADN)

La molécula de ADN contiene la información genética propia de todo ser vivo. Ésta información varía de especie en especie y determina los rasgos únicos de cada una.

1.2.2.2. Ácido ribonucleico (ARN)

El ARN ayuda en la decodificación de la información que contiene el ADN. Lleva ésta información desde el núcleo de la célula, donde se encuentra el ADN, hasta el citoplasma (donde se sintetizan las proteínas).

En las células eucariotas, el ADN sólo se encuentra en el núcleo mientras que el ARN puede ser encontrado en toda la estructura celular.



Cuadro 1.2. Composición de los seres vivos

1.2.3. Cromosoma

Los cromosomas son los elementos de la célula que contienen la información genética (ADN) de una especie, ésta se divide en pequeñas unidades conocidas como genes.

Cada especie tiene su número constante de cromosomas, el equilibrio de la cantidad de cromosomas existente es vital, puesto que hasta la mínima variación en su número puede ocasionar mutaciones en el individuo.

En los organismos más evolucionados, como las plantas y animales, al tener una mayor cantidad de cromosomas, éstos se organizan en pares lo que facilita la repetición del código genético al momento de la reproducción ya que al nuevo ser se le otorga la mitad de los cromosomas de cada uno de sus progenitores. (C.A, 1966)

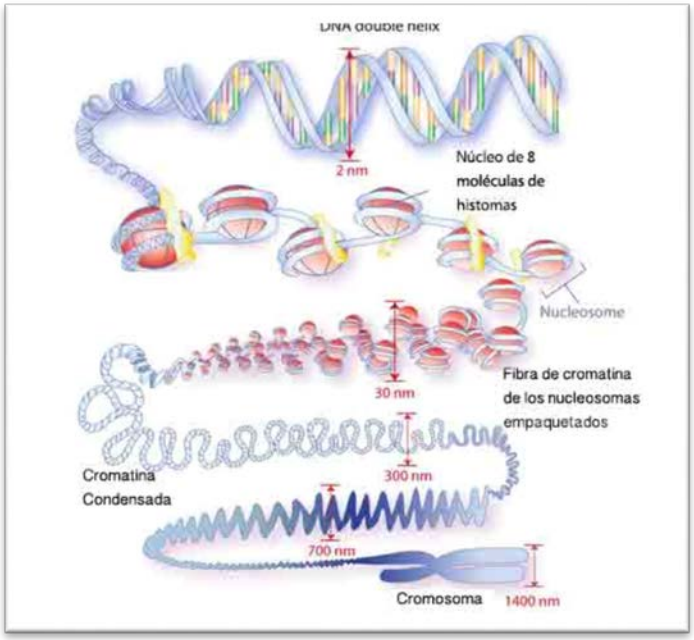


Figura 1.1 Estructura del ADN (Tecnología, 2013)

1.2.4. Gen

Los genes son las unidades más pequeñas de información de todo ser vivo y son los encargados de transmitirla de generación en generación. La información transmitida se conoce como información genética y es la que determina las características específicas de cada individuo, como su color y tamaño; en el caso de las plantas determina incluso la cantidad de flores y frutos, mientras que en los animales influye en la determinación de su comportamiento. (C.A, 1966)

1.2.5. Fenotipo

El fenotipo de una especie es lo que determina los rasgos visibles (físicos) de cada una.

Suelen ser características heredadas, pasadas de padres a hijos, como son los rasgos físicos, el tipo de sangre, entre otros, incluso varias características del comportamiento de las especies son heredadas. (Lacadena, 1999)

1.2.6. Genotipo

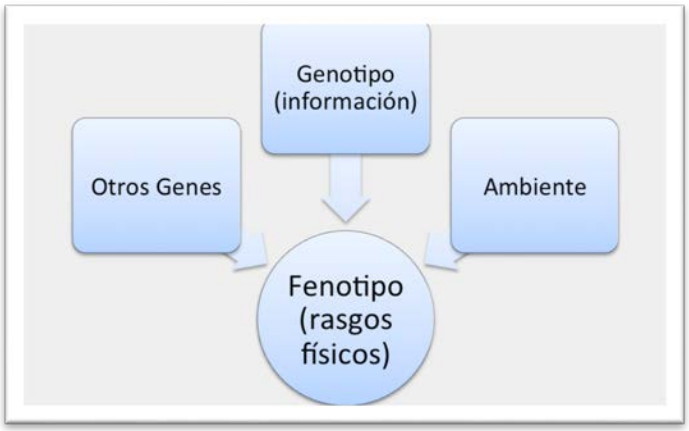
La información que determina las características distintivas de todo ser vivo están organizadas en pequeños grupos de información conocidos como genes. El conjunto de éstos se conoce como genotipo o genoma.

El genotipo es el que determina las características únicas de los individuos y que permite distinguir a los individuos de una especie. Se podría decir que el genotipo es la información que producirá como resultado la determinación del fenotipo (parte visible de todo ser vivo).

Para mayor comprensión de ésta información se la ha representado en lenguaje común, siendo representada por cuatro letras del alfabeto latino. A (Adenina), T (Timina), G (Guanina), C (Citosina).

Ésta representación ha permitido codificar la información contenida en el ADN y así facilitar su estudio.

La adenina se empareja con la timina y la citosina con la guanina. Al emparejarse entre sí, forman una especie de escalera circular de ADN. (Lacadena, 1999)



Cuadro 1.3. Relación entre el genotipo y fenotipo

1.2.6.1. Genoma Humano

El ser humano está compuesto por aproximadamente diez billones de células, cada una de éstas contiene 23 pares de cromosomas y a esto se lo conoce como genoma humano. (Lacadena, 1999)

El genoma del ser humano al igual que el de las demás especies puede ser representado por las cuatro letras ya antes mencionadas (A, T, C y G). Por ejemplo, la secuencia del gen de la hormona concentradora de melanina es:

Atggcaaagatgaatctctcttcctatatattaatactaacttttttctttgttttctcaaggtattttactttcagcatccaagtccataagaaatttaga
tgatgacatggtatttaatacattcaggttggggaaaggctttcagaaggaagacactgcagaaaaatcagttattgctccttccctggaacaat
ataaaaatgatgagagcagtttcatgaacgaagaggaaaataaagtttcaaagaacacaggctccaaacataatttcttaaatcatggtctgcc
actcaatctggctataaaaggatatcaagcactaaaaggatctgtagatttcccagctgagaatggagttcagaatactgaatcaacacaaga

aaagagagaaattggggatgaagaaaactcagctaaatttcctataggaaggagagattttgacatgctcagatgtatgctcggaagagtcta ccgaccctgttggcaagtctga (Dr. Tiessen Favier, 2004)

La codificación del genoma humano está compuesta aproximadamente por más de 3 billones de estas letras. (Igl, 2008)

No todos los segmentos del genoma humano contienen información, es más sólo el 3% de los genes que lo conforman contienen la información útil para la creación de un ser humano, y el resto simplemente es material desechable.

1.2.7. Replicación y Mutación del ADN

Para mantener la constante la información genética de un individuo, el ADN que se encuentra dentro de cada célula se copia a otra célula hija. Este proceso se llama replicación y ocurre cuando se sintetiza el ADN completo de una célula y se lo copia en su totalidad dentro de una célula hija.

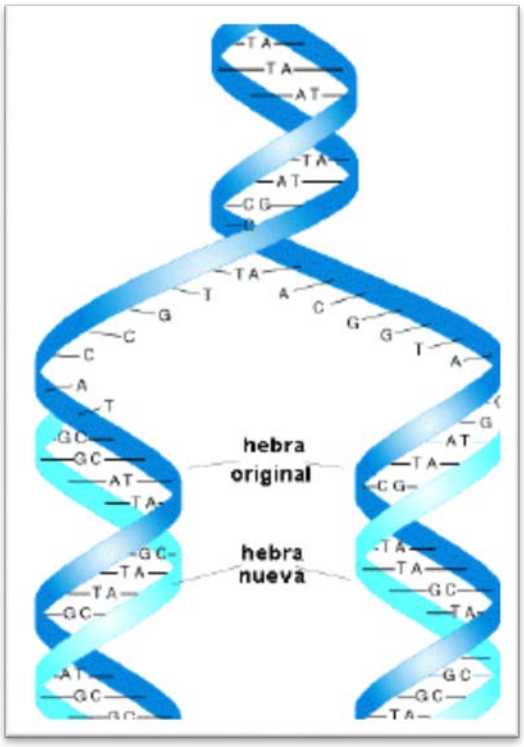


Figura 1.2 Replicación del ADN (Dr. Raisman & Dr. Gonzalez, 2013)

Es necesario duplicar el ADN en cada división celular para así poder mantener la calidad de la información genética contenida dentro de cada célula.

Matthew Meselson y Franklin W. Stahl durante sus estudios con la bacteria Escherichia coli, en el año de 1958, de la genética y de cómo se pasa la información contenida en el ADN de una célula a otra se propusieron tres teorías principales respecto a la replicación del ADN. (Dr. Raisman & Dr. Gonzalez, 2013)

1. Modelo conservador: este modelo propone que al momento de la replicación celular se generaba una nueva cadena de ADN completamente idéntica a la original y la original permanecía intacta.
2. Modelo semi-conservador: este modelo propone que al momento de la división celular la cadena de ADN divide a la mitad sus dos hebras y cada una genera una nueva hebra

dando como resultado dos nuevas cadenas las cuales están compuestas la mitad por la cadena original y la otra mitad por una nueva.

3. Modelo dispersivo: este modelo propone que al momento de la división celular se generan dos nueva cadenas, las cuales, parte de sus tramos están formados por segmentos de la cadena original y tramos nuevos.

Meselson y Stahl en 1957 demostraron que en la replicación del ADN se ajusta al modelo semi-conservador, puesto que demostraron que cada una de las células hijas contenía una hebra del ADN de la célula original y la otra era nueva.

La duplicación del ADN en cada célula hija permite que la cantidad y cualidad de la información transferida se mantenga en todas las células hijas.

1.2.7.1. Mutación del ADN

La evolución de las especies se ha dado en el transcurso de miles de años. Esta evolución se ha dado gracias a las variaciones genéticas que se van generando entre generaciones de la misma especie. Aquellas variaciones se las conoce como mutaciones.

Una mutación es el resultado de alteraciones en el genotipo de las especies y en ocasiones también se ve afectado el fenotipo de la misma.

Las mutaciones se pueden originar al azar y por diferentes causas. Pueden ocurrir de manera espontánea, debido a errores al replicar el ADN o a lesiones ocasionadas al mismo, o ser provocadas de manera artificial con substancias químicas y radiación. (Universidad del País Vasco, 2004)

1.3. Evolución

La evolución es el proceso en el cual los individuos de una especie van cambiando poco a poco a lo largo de generaciones.

Es una de los conceptos más controversiales dentro de la biología y que ha generado varias teorías en su afán de dar explicación a las razones de los cambios que se dan dentro de individuos de una misma especie.

1.3.1. El creacionismo

> *“Y pasó Dios a decir: "Haga brotar la tierra hierba, vegetación que dé semilla, árboles frutales que lleven fruto según sus géneros, cuya semilla esté en él, sobre la tierra". Y llegó a ser así. Y la tierra empezó a producir hierba, vegetación que da semilla según su género y árboles que llevan fruto, cuya semilla está en él según su género. Entonces Dios vio que era bueno. Y llegó a haber tarde y llegó a haber mañana, un día tercero”.*
>
> *Génesis 1, 11-13.* (Genésis)

Según la teoría del creacionismo, todo origen de la vida tiene un origen divino dado por un ser superior. Ésta teoría fue aceptada durante muchos siglos hasta el aparecimiento de teorías científicas; incluso hoy en día es la teoría aceptada dentro de círculos religiosos más cerrados.

1.3.2. Ànaximandro (610 a 546 A.C.)

La teoría propuesta por el filósofo griego Anaximandro postula que la vida se originó gracias al calentamiento de la tierra y del agua.

Los primeros seres en surgir se asemejaban a peces, los cuales contenían en su interior unos seres con forma de embriones los cuales a llegar a su etapa adulta salían de su contenedor como hombres y mujeres adultos.

1.3.3. Aristóteles (384 a 322 A.C.)

La teoría propuesta por Aristóteles indica que dentro de la naturaleza existe una escala que va desde los objetos inanimados hasta ir evolucionando a plantas, luego animales hasta finalmente llegar al ser humano. A esta teoría la llamo Scala Natura o Escala de la Naturaleza.

1.3.4. Lamarck (1744-1829)

La teoría propuesta por Jean Batiste Lamack en 1809 postula que las especies van evolucionando de organismos simples a otros más complejos, lo cual les permite adaptarse a las adversidades del medio que las rodea, siendo el medio el que obliga a las especies a cambiar.

Tomó el ejemplo de las jirafas en las cuales su órgano más desarrollado llegó a ser su cuello, el cual les permite alimentarse de las hojas más altas de los árboles, algo que las otras especies no pueden hacer. Con esto quiso demostrar que al evolucionar una especie ésta va heredando las características que se fueron potenciando para la supervivencia, mientras que aquellas que no se las usaron se van atrofiando hasta desaparecer.

1.3.5. Darwin (1809-1882)

El famoso naturista británico Charles Darwin propuso en 1859 la famosa teoría de la evolución de las especies dentro de su trabajo *"El origen de las especies la segunda gran unificación del siglo XIX: la teoría de la evolución biológica"*.

En este trabajo Darwin postuló que las características que tienen todas las especies en la actualidad se fueron derivando de otras mediante un proceso llamado descendencia con modificación. En este proceso las pequeñas modificaciones que aparecen en cada generación les permiten irse adaptando de mejor o peor manera al medio que las rodea. Aquellas especies que están mejor adaptadas son las que irán teniendo descendencia gracias al proceso de selección natural. Por ello son las características con mayores posibilidades de adaptación las que se irán heredando y reforzando en cada generación donde como resultado las especies con las características que se conocen en la actualidad.

1.4. Mecanismos de Evolución

La evolución biológica es el resultado de los cambios que se dan a nivel genético en las especies a lo largo del tiempo.

1.4.1. Población

La población es el conjunto de individuos a los cuales se les estudiará sus características tanto fenotípicas como genotípicas.

Al momento de estudiar una población es importante conocer las diferencias entre fenotipo y genotipo, puesto que dentro de una población se tiene gran variedad de fenotipos pero pueden tener el mismo genotipo. Esto se puede dar gracias a la influencia del medio de las especies pertenecientes a la población.

1.4.2. Herencia

Aquellas características que se pasan de una generación a otra se las conoce como características hereditarias.

En las especies en donde existe reproducción sexual, la mitad de las características genéticas (cromosomas) son aportadas por la célula sexual masculina y la otra por la femenina.

1.4.3. Mutación

La mutagénesis es la rama de la genética que estudia los efectos generados por factores externos como la radiación, en los genes de los individuos expuestos.

Estos factores externos generan alteraciones en los cromosomas, los cuales pueden afectar en menos o mayor proporción a un individuo. A estas alteraciones se las conoce como mutaciones.

2. COMPUTACIÓN EVOLUTIVA

El concepto de la teoría de la evolución siempre ha sido tema de discusión, tanto en la comunidad científica como en la sociedad en general.

Desde los antiguos pensadores griegos, quienes ya comenzaron a plantear sus teorías sobre la evolución hasta llegar a Darwin con su famosa publicación sobre “El Origen de las Especies”, la teoría de la evolución ha generado gran curiosidad en el ser humano y ha permitido también el tratar de reproducir esas teorías con modelos computacionales.

Conceptos tales como mutación, reproducción, selección natural, han sido usados para desarrollar algoritmos, los cuales al ser aplicados como modelos computacionales, permiten resolver de manera más eficiente problemas cotidianos, cosa que simples modelos matemáticos no pueden resolver de manera óptima o en muchos casos les es imposible resolverlos.

A ésta rama de la computación que se fusiona con los conceptos ya antes mencionados se la conoce como Computación Evolutiva.

Para tener una mejor comprensión sobre ésta rama es necesario conocer inicialmente sobre la Inteligencia Artificial y su historia, puesto que de ésta se desprende la Computación Evolutiva.

2.1. La Inteligencia Artificial y su relación con la Computación Evolutiva

> *"Primera Ley: un robot no puede dañar a un ser humano o, con su inactividad, permitir que un ser humano sufra daño.*
>
> *Segunda Ley: un robot tiene que obedecer las órdenes dadas por los seres humanos, salvo cuando tales órdenes vulneren la Primera Ley.*
>
> *Tercera Ley: un robot debe proteger su propia existencia, siempre que esta protección no vulnere la Primera o la Segunda Ley.*

Ley Cero: Un robot no puede perjudicar a la Humanidad ni, por omisión, permitir que la Humanidad sufra daño."

Isaac Asimov, "Las tres leyes de la robótica" (Asimov, 2004, pág. vii)

Desde el momento en que el ser humano comenzó a tener un pensamiento racional y comenzó a desarrollar herramientas para facilitar su vida, ha deseado crear máquinas que pudiesen pensar y actuar por sí mismas de manera similar a los mismos humanos.

Actualmente existen diferentes aproximaciones a la Inteligencia Artificial. El tema del presente estudio lo podemos ubicar de la siguiente manera (de la Herrán Gascón, 1998):

- Inteligencia artificial.
    - Enfoque simbólico o `top-down'.
    - Enfoque subsimbólico o `bottom-up'.
        - Redes neuronales artificiales.
        - **Computación evolutiva o algoritmos evolutivos**.
            - Solidificación o recocido simulado (simulated annealing).
            - Algoritmos genéticos.
            - Estrategias evolutivas.
            - Clasificadores genéticos.
            - Programación genética.

La inteligencia artificial nos da una aproximación de lo que es la inteligencia humana y así también de cómo la naturaleza actúa. La Computación Evolutiva toma como modelo a la naturaleza para poder generar algoritmos que permitan resolver problemas.

2.2. Historia de la Computación Evolutiva

"En la naturaleza todos los seres vivos se enfrentan a problemas que deben resolver con éxito." (Baños Navarro, 2008)

A partir de la década de 1930, los científicos poco a poco fueron tratando de integrar los comportamientos de la evolución natural para el desarrollo de algoritmos de aprendizaje.

En los años 50's, Alan Turing propuso que debía existir una conexión entre aquellos comportamientos y el aprendizaje de máquina, con lo cual pretendía desarrollar un programa para jugar ajedrez basándose en los mismos.

Formalmente aquellas técnicas fueron poco a poco aplicadas en los procesos de controles estadísticos, aprendizaje de máquina y en la optimización de funciones.

En 1954, Barricelli, desarrolló uno de los primeros sistemas, el cual usaba los mismos principios básicos que se usan hoy en día en la disciplina conocida como Vida Artificial.

En 1958, Friedberg propuso un sistema el cual podría mejorarse a sí mismo basándose en proposiciones iniciales y escogiendo siempre la mejor solución.

Laurence J. Fogel, conocido como el padre de la programación evolutiva, introdujo una de las primeras técnicas evolutivas, la cual consistía en la evolución de autómatas, los cuales tenían estados finitos por medio de mutaciones. Esta técnica usada es muy similar a las usadas en la actualidad dentro de la Computación Evolutiva.

2.2.1. Conceptos Básicos

El principal propósito de la mayoría de los sistemas desarrollados que utilizan inteligencia artificial, es la solución de problemas complejos de optimización y con múltiples objetivos. La

computación Evolutiva utiliza técnicas basadas en los conceptos de la evolución de las especies para el diseño, implementación y resolución de aquellos problemas en el computador.

Aquellos modelos desarrollados se los conoce como algoritmos evolutivos. Los algoritmos evolutivos se basan en los conceptos de los procesos de selección, reproducción y mutación de las especies, tomando cada variable como un individuo.

Cada individuo de la población es evaluado y se mide su aptitud de supervivencia en un ambiente dado, la selección se enfoca en aquellos individuos con la más alta aptitud, con la recombinación y mutación se modifican a los individuos de la población generando nuevas medidas de aptitud (poblaciones).

Dentro de la Computación Evolutiva se ha desarrollado algoritmos robustos de búsqueda para poder simular aquel comportamiento usando técnicas heurísticas. Las técnicas heurísticas permiten encontrar la solución de un problema mediante métodos como el análisis modal de fallo y efecto y los árboles de fallo.

En el análisis modal de fallo y efecto (AMFE) se emplea métodos de clasificación para identificar las fallas dentro de un proceso y ejecutar posibles soluciones. (James, 1996).

Mientras que en los árboles de fallos se busca descomponer sistemáticamente un suceso complejo denominado suceso TOP en sucesos intermedios hasta llegar a sucesos básicos. (Civil, 2008)

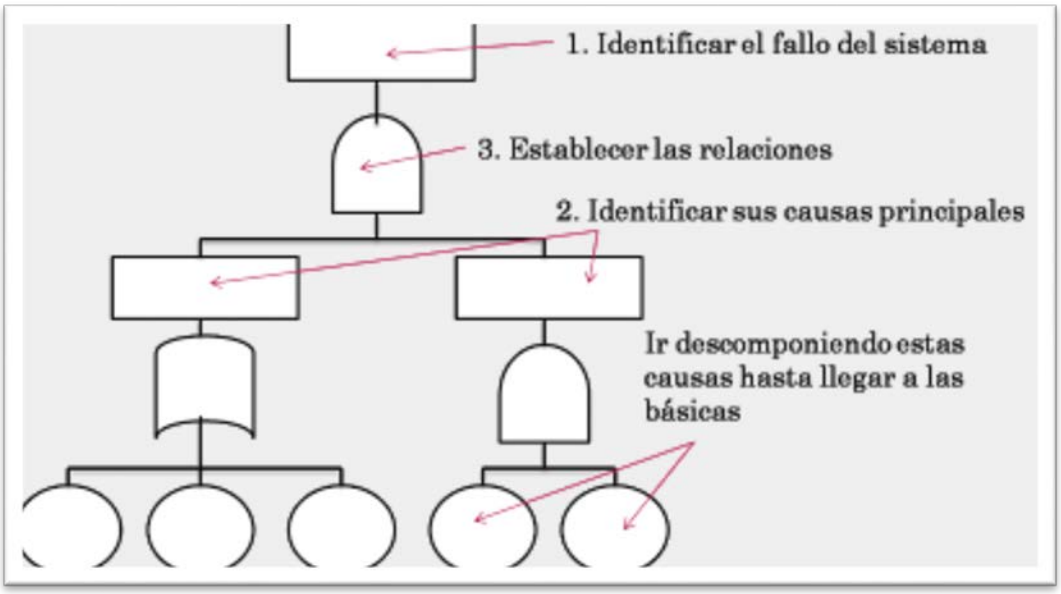


Fig. 2.1 Ejemplo de un árbol de fallo (Beltrán, 2010)

2.3. Algoritmos Genéticos

Existe gran variedad de algoritmos de búsqueda. Aquellos que se basan en los procesos de selección natural y que también utilizan conceptos utilizados en el campo de la genética se los conoce como “Algoritmos genéticos”.

Con los Algoritmos Genéticos (AG), no es necesario conocer el valor de la gradiente de la función lo cual resulta muy útil al momento de analizar problemas donde la información suele ser escasa o incluso inaccesible. También se los suele utilizar en problemas donde la función objetivo no está definida o no tiene una respuesta exacta.

Al momento de analizar un problema dado con un algoritmo genético, las entradas son el conjunto de soluciones posibles al problema. La llamada Función de Aptitud, es la métrica que permite evaluar cuantitativamente cada una de las entradas.

Los AG son una herramienta robusta para la solución de problemas complejos. Se los ha utilizado en una variedad de problemas dando como resultado, en la mayoría de casos, soluciones más eficientes a las obtenidas por otros métodos.

2.3.1. Clases de Algoritmos Genéticos

2.3.1.1. Algoritmos Genéticos Generacionales

Modelo también conocido como "Algoritmo Genético Canónico". Se basa en el comportamiento de los insectos, los cuales cuando producen una nueva generación, los padres se alejan o mueren. Dentro del algoritmo, en cada iteración se crea una población completamente nueva.

2.3.1.2. Algoritmos Genéticos de Estado Fijo o Modelo Estacionario

En un modelo cuya convergencia es rápida. Se basa en el comportamiento de los mamíferos, donde tanto padres como hijos conviven en la misma población; generando competencia entre cada individuo. Dentro del algoritmo en cada iteración se escogen parejas de individuos (aleatoriamente) a los cuales se les aplica los operadores genéticos de reproducción, cruce o mutación, dando como resultado la nueva generación de individuos. Pero antes también se evalúa la aptitud de cada individuo, ya sea padre o hijo, para continuar con una nueva iteración. (Universidad del País Vasco, 2004)

2.3.1.3. Algoritmos Genéticos Paralelos

Es un modelo que permite evaluar de manera simultánea varias soluciones, las cuales están representadas por un individuo dentro de la población.

Modelo de Islas (Algoritmo de Grano Grueso)

La población dada es dividida en varias sub-poblaciones a las cuales se les examina, de manera simultánea, con el Algoritmo Genético escogido. Luego de cierto número de iteraciones, se genera un intercambio de individuos entre las sub-poblaciones (emigración), lo cual permite tener una mayor diversidad entre las sub-poblaciones.

La definición de la tasa de migración entre las sub-poblaciones es importante ya que permitirá la rápida convergencia o no del problema. (Universidad del País Vasco, 2004)

Modelo Celular (Algoritmo de Grano Fino)

Se coloca a los individuos de la población dentro de una matriz, cada individuo se reproducirá con aquellos que sean sus vecinos; siendo seleccionados al azar o al más apto. El individuo hijo ocupará un lugar cercano a los padres.

Luego de varias iteraciones, se puede observar grupos grande de individuos con características genéticas semejantes, por lo cual se podría decir que se tiene tendencias evolutivas similares al modelo de las islas. (Universidad del País Vasco, 2004)

2.3.2. Algoritmo Genético Simple

Los Algoritmos Genéticos Simples trabajan con cadenas binarias (0,1) con una distribución uniforme en la población inicial. La población inicial se genera de manera aleatoria y cada individuo puede ser 0 o 1 con igual probabilidad (0.5).

En cada iteración evalúa la cadena resultante con el valor real. El algoritmo se detiene cuando se ha alcanzado el número de iteraciones (generaciones) predefinido para el problema.

2.3.2.1. Elementos de un Algoritmo Genético

Varios autores indican que un Algoritmo Genético Simple contiene los siguientes elementos: (Zitzler, 1999)

1. **Selección**: copia cadenas individuales de acuerdo al criterio dado por la función de aptitud o función objetivo.

   La función permite "premiar" a las mejores soluciones, de tal forma que éstas se propaguen con mayor rapidez, mientras que a las malas soluciones se las penaliza de tal forma que tienen menor probabilidad de reproducción.

2. **Cruce**: se selecciona aleatoriamente dos individuos (padres), cada uno determina la probabilidad de cruce o no.

   Si existe cruce:

   - De un punto: el punto de corte (k) se selecciona de manera aleatoria entre 1 y Lc-1 (Lc: longitud del cromosoma o cadena), desde aquel punto se procede a intercambiar cada gen. Como se puede observar en la Fig. 2.2 El Hijo1 en la primera parte de la cadena tiene los elementos heredados del Padre y la segunda parte los de la Madre. Mientras que el Hijo2 la primera parte de la cadena pertenece a la Madre y la segunda al Padre.

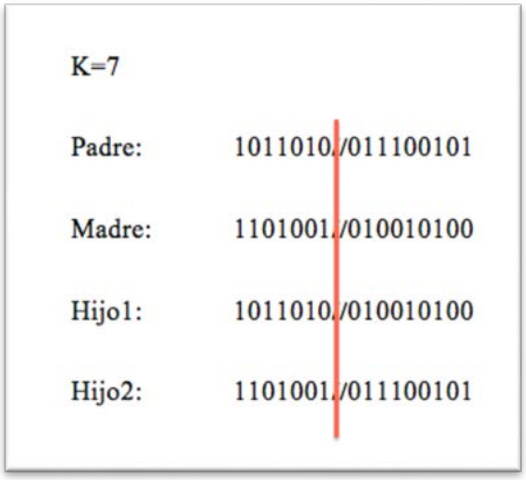


Fig. 2.2 Ejemplo cruce en un solo punto[3]

- De dos puntos: se escoge dos puntos de corte K0 y K1, de manera aleatoria y se procede de igual manera que en la de un punto.

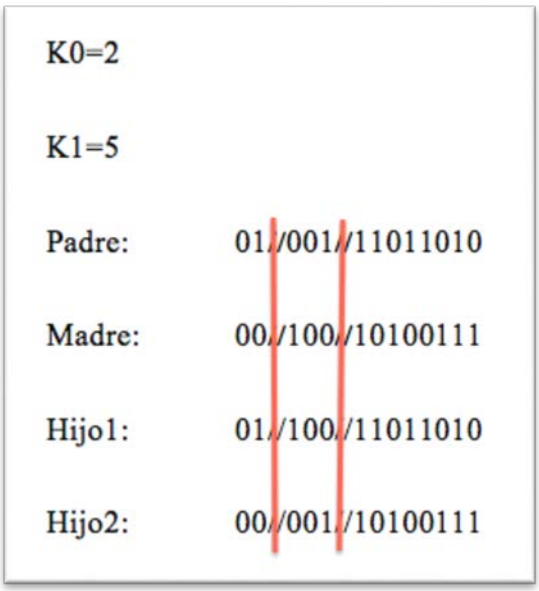


Fig. 2.3 Ejemplo cruce en dos puntos

[3] La línea roja indica el punto de corte de la cadena (cromosoma).

- Uniforme: cada gen se hereda de un padre elegido aleatoriamente

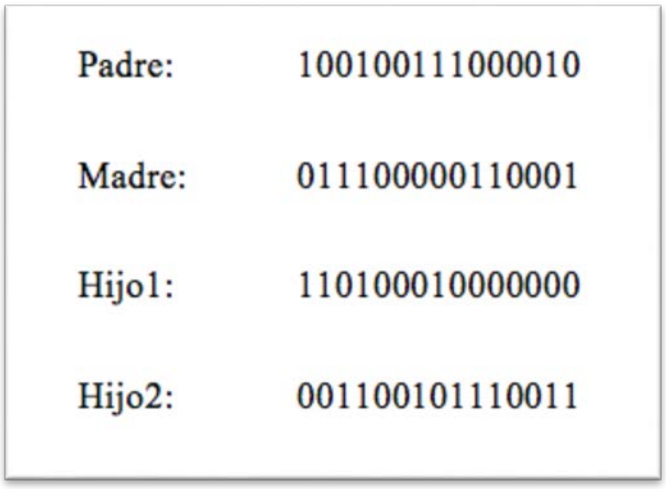


Fig. 2.4 Ejemplo cruce uniforme

Dentro de los algoritmos genéticos la operación de cruce se da por un porcentaje dado (probabilidad), lo que hace que no todas las parejas de individuos se crucen, y éstos pasarán a la siguiente generación sin cambios.

Dentro de la técnica conocida como “Elitismo” el individuo con mejor aptitud no se cruza con ningún otro hasta que dentro de alguna de las nuevas generaciones aparezca otro mejor y lo reemplace.

3. **Mutación**: este proceso se aplica después de cada cruce. Se elige un individuo al azar, cada uno tiene una probabilidad muy pequeña de mutación, para evitar que todos los individuos de una población pasen por ese proceso. Se toma la cadena o cromosoma y se la altera. (Jacob, 2001)

2.3.2.2. Esquema Básico

1. Los algoritmos genéticos parten de una población inicial (posibles soluciones)
2. Se selecciona los individuos (mejor calidad)
3. Se realiza el cruce
4. Se generan las mutaciones de manera aleatoria
5. Genera la nueva generación
6. Se continúa de igual manera en las siguientes iteraciones hasta que la función converja.

En pseudocódigo:

```
BEGIN /* Algoritmo Genético Simple */

    Generar una población inicial.

    Computar la función de evaluación de cada individuo.

    WHILE NOT Terminado DO

    BEGIN /* Producir nueva generación */

        FOR Tamaño  población/2 DO

        BEGIN /*Ciclo Reproductivo */

            Seleccionar dos individuos de la anterior generación, para el cruce (probabilidad
            de selección proporcional a la función de evaluación del individuo).

            Cruzar con cierta probabilidad los dos individuos obteniendo dos descendientes.

            Mutar los dos descendientes con cierta probabilidad.

            Computar la función de evaluación de los dos descendientes mutados.

            Insertar los dos descendientes mutados en la nueva generación.

        END

        IF la población ha convergido THEN

        Terminado:= TRUE

    END

END
```

Fig. 2.5 Pseudocódigo de un Algoritmo Genético Simple

3. OPTIMIZACIÓN EVOLUTIVA MULTIOBJETIVO

Los algoritmos de optimización usan conceptos de genética para codificar un problema como una generación y luego simular mediante la aplicación de operadores matemáticos genéticos (selección, cruce y mutación) para determinar el mejor individuo o solución en las múltiples iteraciones de un número finito de generaciones. La *función de aptitud* es la que permite definir si la solución dada es mejor o peor que otra de acuerdo al objetivo definido en el problema.

Al momento de querer resolver cualquier problema en el cual se tienen varias soluciones posibles, generalmente se sigue el sistema de la ruta más corta, en la cual se analiza la ruta más eficiente para lograr conseguir un solo objetivo. Sin embargo en problemas de optimización reales no siempre se tiene un solo objetivo al cual satisfacer. Si bien se podría usar el método de la ruta más corta para obtener la solución, un objetivo por cada vez que se lo corra; no es la manera más efectiva de satisfacer todos a la vez. Para ello se utiliza la Optimización Evolutiva Multiobjetivo, la cual permite evaluar todos los objetivos a la vez.

3.1. Conceptos Básicos

El procesos de optimizar sistemática y simultáneamente una colección de funciones objetivo es llamado Optimización Multiobjetivo (OMO)[4].

La optimización multiobjetivo implica minimizar o maximizar funciones con múltiples objetivos que se encuentran sujetas a un grupo de restricciones. Como ejemplos se puede usar dentro de problemas de análisis de diseños equilibrados, para seleccionar productos óptimos o diseños de procesos, o cualquier otra aplicación en la cual se necesite una solución óptima y que necesite tener un equilibrio entre dos o más objetivos en conflicto.

Enfoques comunes para la optimización multiobjetivo incluyen:

[4] En sus siglas en inglés MOO (multi-objetcive optimization)

Obtener una meta: reduce los valores de una función vectorial lineal o no lineal para obtener los valores dados en el vector objetivo. La importancia relativa de los objetivos se indica usando un vector con los pesos. Los problemas de obtención de metas pueden estar sujetos a restricciones lineales y no lineales.

Minimización: minimiza los valores del peor caso de un grupo de funciones multivariadas, posiblemente sujetas a restricciones lineales y no lineales.

Algoritmo genético multiobjetivo: resuelve problemas de optimización multiobjetivo encontrando un grupo de puntos equitativamente distribuidos en el Frente de Pareto. Este enfoque es usado para optimizar problemas uniformes con o sin limitaciones y restricciones lineales.

Ambos, la obtención de una meta y problemas de minimización pueden ser resueltos transformando el problema en un problema de optimización con restricciones estándar y luego usando un enfoque de un conjunto activo para encontrar la solución.

La optimización multiobjetivo se originó principalmente para poder resolver problemas de economía (teoría de equilibrio y bienestar), teorías de juego y matemática; aunque en la actualidad se la aplica en una gran variedad de ramas. Por sus orígenes muchos de los términos usados dentro de la optimización multiobjetivos se derivan de esos tres primeros campos.

3.1.1. Términos Básicos

Preferencias: las preferencias se refieren a los puntos dentro del conjunto de criterios que se basan para la toma de decisiones. Estos puntos de toma de decisiones influyen directamente en el conjunto de puntos de soluciones potenciales.

Función de preferencia: una función de preferencia es una función abstracta, la cual contiene el conjunto de preferencias del tomador de decisiones.

Función de utilidad: en economía, la utilidad es modelada con la función de utilidad, la cual está dada por el grado de satisfacción de un individuo o grupo, en ese caso se enfoca en el nivel de satisfacción de la toma de decisiones. En términos de la optimización multiobjetivo, una función de utilidad individual está definida por cada objetivo y representa la importancia de cada uno. La función de utilidad **U** es la combinación de las funciones de utilidad individuales, es una representación matemática de las preferencias de la toma de decisiones. Se la usa para aproximar la función de preferencia que en muchos casos no puede ser expresada de una forma matemática.

Criterio Global: el Criterio Global es una función escalar que combina matemáticamente funciones con múltiples objetivos y que no involucra necesariamente utilidad ni preferencia.

Teoría de Juego: Stadler (1988). "la aproximación de la matemática y la economía (a problemas con múltiples objetivos), fueron eventualmente unificadas con la creación de la teoría de Juego por Borel en 1921"

Según la Teoría del Juego, un juego es cualquier situación de conflicto o cooperación de dos o más jugadores, en la que existen múltiples posibilidades de estrategias o movimientos. En la Teoría del Juego se representa la optimización multiobjetivo con múltiples tomadores de decisión, en la cual cada uno de ellos controla ciertas variables. En caso de que todos los jugadores cooperen, los resultados son similares a solo tener un tomador de decisión para el problema de optimización.

3.1.1.1. Dominancia y Eficiencia

Eficiencia e ineficiencia: dado un punto $x^* \in X$ es eficiente si no existe otro $x \in X$ tal que $F(x) \leq F(x^*)$, caso contrario x^* es ineficiente.

Frontera eficiente: es el conjunto de todos los puntos x^* eficientes.

Puntos dominados y no dominados: El vector de funciones objetivos $F(x^*) \in Y$ (espacio de criterios factibles) no es dominado si no existe otro vector $F(x) \in Y$ tal que

$F(x) \leq F(x^*)$ con al menos un $f_i(x) < f_i(x^*)$; caso contrario es dominado.

3.1.2. Definición de un problema de optimización multiobjetivo

Un problema de optimización multiobjetivo siempre se puede representar de la siguiente manera:

$$(min_o \max)\, f(x) \qquad x \in X$$

Donde ***f*** es la función escalar y ***X*** es el conjunto de restricciones que se definen por:

$$X = \{x \in R^m : h(x) = 0, g(x) \geq 0\}$$

En un problema de optimización multiobjetivo se representa de la siguiente manera:

$$(min_o \max)\, F(x) = [f_1(x), f_2(x), \ldots . f_k(x)]^T$$

Sujeto a $g_j(x) \geq 0$ j=1,2….,m

$$h_l(x) = 0 \qquad l = 1,2,3 \ldots, e$$

Donde

k: número de funciones objetivo

m: número de restricciones de desigualdad

$\vec{x} \in E^n$: es el vector de las variables de decisión donde la solución del problema de describe en términos de $x_i \rightarrow (x_1, x_2, \ldots . . x_n)$

n: número de variables independientes en el espacio de decisión ***X***

$F(y) \in E^k$: es el vector de las funciones objetivo en el espacio ***Y***

e: número de restricciones de igualdad

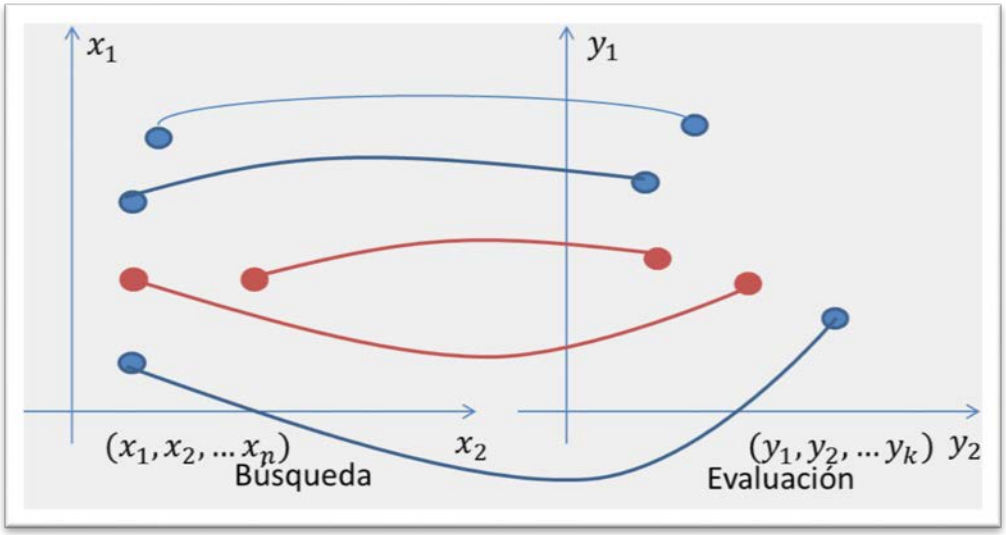


Fig. 3.1 Ilustración de un problema de optimización multiobjetivo[5], (Ziztler, Laumans, & Bleuler, 2002)

3.2. Óptimo de Edgeworth-Pareto

Al tener más de una función objetivo, es raro que un solo valor de ***x*** satisfaga a la vez a todos los objetivos ***F(x,)*** para ello es necesario encontrar un conjunto de soluciones que satisfagan los diversos objetivos más que tratar de encontrar un solución óptima única.

En 1881 Ysidro Edgeworth propuso tal problema y en 1906 Pareto lo generalizó y planteó el concepto predominante conocido como *Óptimo de Pareto.*

[5] En azul se ilustra el conjunto de Pareto en X, el Frente de Pareto en Y y en rojo se ilustra la aproximación al conjunto de Pareto en X y la aproximación al Frente de Pareto en Y.

3.2.1. Optimalidad de Pareto

El conjunto de soluciones óptimas encontradas en el espacio de decisión ***X*** es conocido como el *conjunto de Pareto* $x^* \subseteq x$y su imagen en el espacio objetivo se lo conoce como *frente de Pareto.*

El espacio al cual pertenece el vector objetivo se lo conoce como *espacio objetivo* y la imagen del grupo factible se conoce como *conjunto obtenido.*

3.2.1.1. Óptimo de Pareto:

Un punto $x^* \in X$ es el *óptimo de Pareto* si no existe otro punto $x \in X$ para los cuales

$F(x) \leq F(x^*)$ y $f_i(x) < f_i(x^*)$ para al menos una función

Todos los puntos del *óptimo de Pareto* se encuentran en el espacio de soluciones posibles ***Z*** (Athan and Papalambros, 1996; Chen et al., 2000)

3.2.1.2. Óptimo de Pareto débil

En un problema optimización multiobjetivo, se dice que un punto x* es un *óptimo de Pareto débil* cuando no existe otro punto $x \subseteq X$(conjunto de restricciones planteadas) tal que *F(x)<F(x*)* ($f_i(x) < f_i(x^*)$) para todo $i \in \{1,2, \ldots . . k\}$

Un punto se dice que es un *óptimo de Pareto débil* si no existe otro que mejore todas las funciones objetivo de manera simultánea.

3.2.1.3. Óptimo Estricto de Pareto

En un problema de optimización multiobjetivo se dice que un punto *x** es un *óptimo estricto de Pareto* o una solución eficiente estricta si y sólo si no existe ninguna $x \in X$ tal que *F(x)<F(x*)* con al menos una desigualdad estricta.

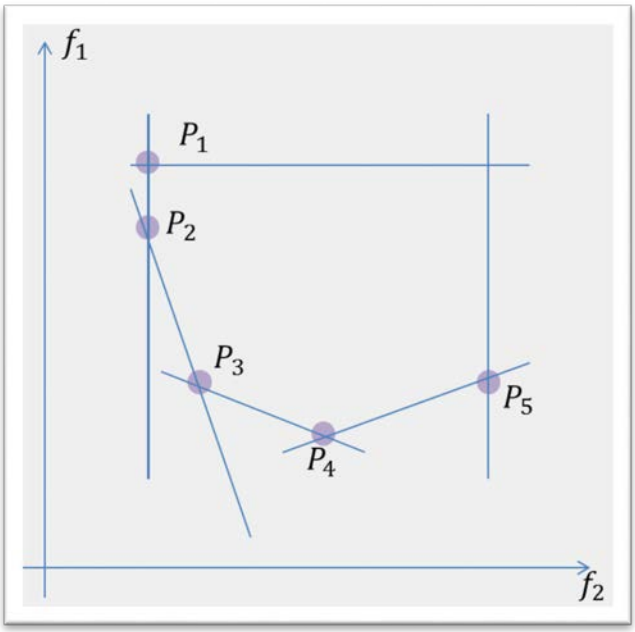


Fig.3.2 En la figura se muestra los puntos P_1 y P_5 son óptimos de Pareto débiles, los puntos P_2, P_3, P_4 son óptimos estrictos. (Coello Coello & Sarker)

3.2.1.4. Puntos del óptimo de Pareto Local

El concepto de los puntos del *óptimo de Pareto locales* es similar al anterior sólo que se enfoca en el vecindario de soluciones factibles de *x*.

Tomado de otra manera,

si *B(x*,Ɛ)* es una esfera de radio Ɛ>0 cercana al punto *x**,

es necesario que para algún Ɛ>0 no exista ningún $x \in X \cap B(x^*, \varepsilon)$

tal que $F(x) \leq F(x^*)$

($f_i(x) \leq f_i(x^*)$ para todo $i \in \{1, \ldots . k\}$) con al menos una desigualdad estricta

3.2.2. Dominancia de Pareto

Se dice que el vector *F(x)* domina a *F(x*) si y solo si* $F(x) \leq F(x^*)$

$$f_i(x) \leq f_i(x^*) \text{ para todo } i \in \{1, \ldots . k\}$$

Dados dos vectores $x\ y\ x^* \in X$ existen tres posibilidades para sus vectores objetivo correspondientes:

- $f_i(x) \leq f_i(x^*): \forall i$: se dice que el vector *F(x)* domina a *F(x*)* y *x* domina a *x**
- $f_i(x^*) \leq f_i(x): \forall i$ se dice que el vector objetivos *F(x*)* es dominado por F*(x)* y que *x** es dominado por *x*.
- $f_i(x^*) \nleq f_i(x)$ y $f_i(x) \nleq f_i(x^*)$ se dice que el vector objetivo *F(x)* es diferente de *F(x*)* y vice-versa y que el *x* es indiferente de *x** y vice-versa.

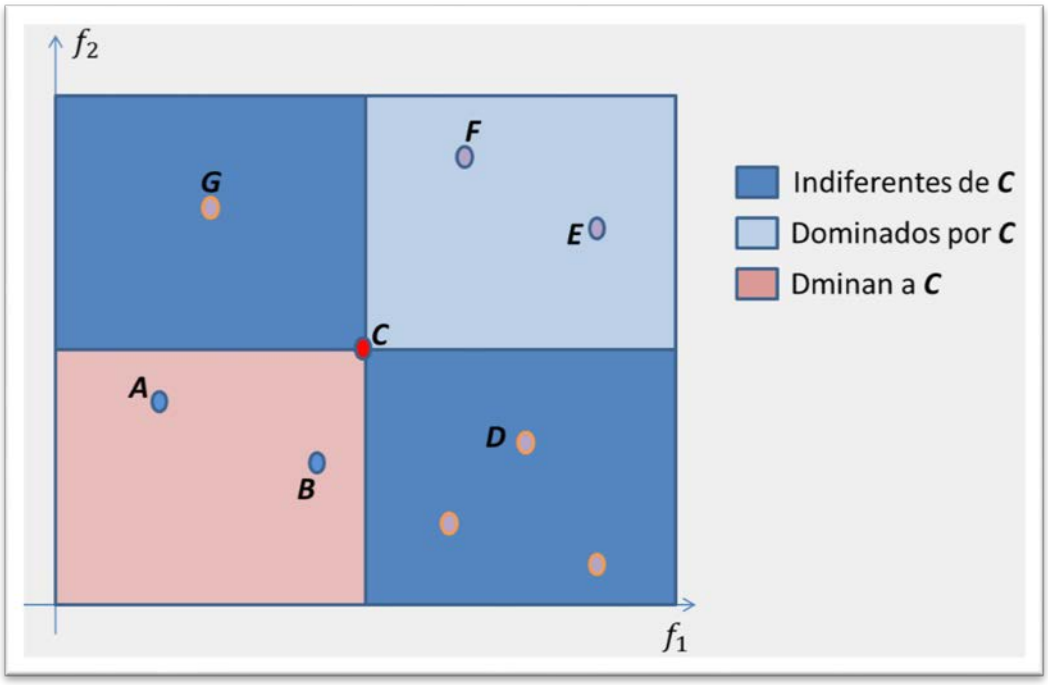


Fig.3.3 Dominancia de Pareto en un problema de minimización con dos objetivos (Coello Coello & Sarker)

3.2.3. Conjunto Soluciones Óptimas de Pareto

$x^* \in Z$ es un óptimo de Pareto si no existe otra $x \in X$ y que x domine a x^* a lo cual se le denomina a *F(x*)* como *Punto Eficiente* o *Punto Óptimo de Pareto.* Al conjunto de todas esas soluciones se lo conoce como *conjunto Eficiente* o *conjunto óptimo de Pareto.*

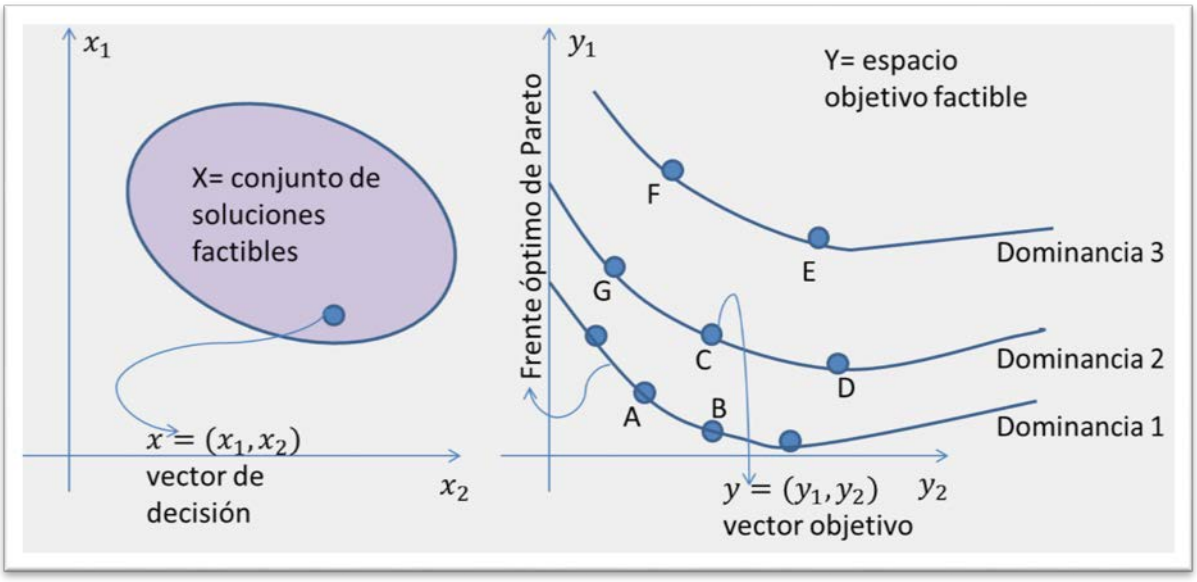


Fig.3.4 Conjunto de soluciones factibles, espacio objetivo factible y grado de dominancia en un problema de minimización con dos objetivos. (Coello Coello & Sarker)

3.2.4. Frente de Pareto

A la imagen en *y* del conjunto óptimo de Pareto se lo conoce como *Frente óptimo de Pareto* o como *Curva o Superficie de Pareto.* La forma del frente de Pareto indica el tipo de compensación (trade-off) entre las diferentes funciones objetivo.

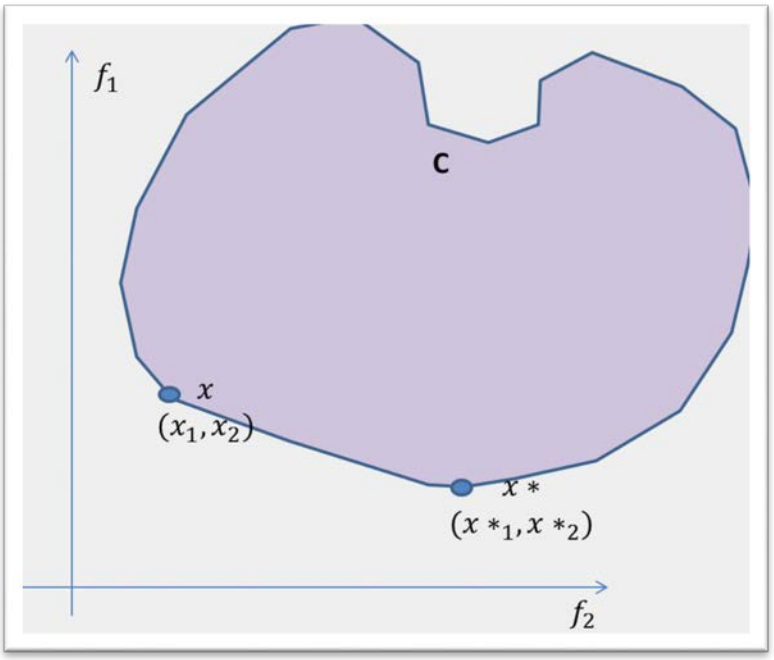


Fig.3.5 Ejemplo de un frente de Pareto[6] (Coello Coello & Sarker)

3.3. Algoritmos representativos

Para poder resolver un problema de optimización multiobjetivo se utilizan los llamados algoritmos evolutivos, los cuales permiten evaluar de manera simultánea el conjunto de posibles soluciones del problema, evaluando en cada iteración las soluciones aproximadas al frente de Pareto sin ser afectado por la forma o continuidad del frente de Pareto.

Las ventajas de los algoritmos evolutivos son:

- Permiten encontrar soluciones aproximadas al frente de Pareto.
- Permiten tener una muestra más equitativa entre las funciones a optimizar.

[6] En la figura se muestra los puntos *x* y *x** que definen el frente de Pareto, a estos puntos se los conoce como puntos no dominados

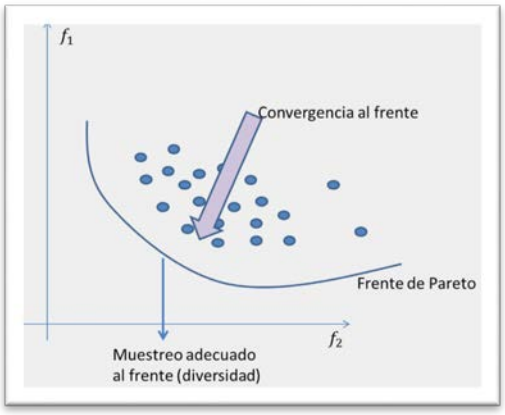


Fig.3.6 Propósitos en la resolución de un problema multiobjetivo (Coello Coello & Sarker)

3.3.1. Esquema de un Algoritmo Evolutivo de Optimización Multiobjetivo

```
Inicializar(P(0))
generacion = 0
mientras (no CriterioParada)
{
        Evaluar(P(generacion))
        Operador de diversidad(P(generacion))
        Asignar fitness(P(generacion))
        Padres = Seleccion(P(generacion))
        Hijos = Operadores de Reproduccion(Padres)
        NuevaPop = Reemplazar(Hijos,P(generacion))
        generacion ++
        P(generacion) = NuevaPop
} retornar frente de Pareto
```

Fig. 3.7 Pseudocódigo Algoritmo Evolutivo Multiobjetivo (Coello Coello & Sarker)

Los operadores de diversidad y de asignación de fitness no son usados dentro de los algoritmos evolutivos genéticos pero en el caso de los algoritmos evolutivos de optimización multiobjetivo, son conceptos muy importantes.

Operador de diversidad: permite evitar la convergencia a un sector en el *frente de Pareto*.

Operador de asignación: permite a los individuos más aptos la posibilidad de perpetuar su descendencia considerando los valores de las funciones objetivo y las medidas para evaluar la diversidad.

3.3.2. Tipos de algoritmos evolutivos multiobjetivo

En los problemas de optimización multiobjetivo se pueden clasificar a los algoritmos en dos principales tipos basados en el mecanismo usada para la asignación de fitness:

Algoritmos no basados en Pareto	Algoritmos basados en Pareto
Estos algoritmos utilizan mecanismos sencillos para la asignación del fitness pero no siempre llevan a la solución del problema	Estos algoritmos se basan en la dominancia o no de un individuo respecto a otro para su jerarquización. La dominancia se basa en el concepto del óptimo de Pareto.

Cuadro 3.1 Comparación entre algoritmos basados en Pareto y no basados.

3.3.2.1. Algoritmos no basados en Pareto

No siempre garantizan la convergencia en la solución más óptima y no son adecuados en problemas con más de tres objetivos.

Este tipo de algoritmos pueden usar diferentes mecanismos para la asignación del fitness de los cuales se hablarán brevemente a continuación. (Universidad de la República, 2007)[7]

- Agregación lineal de objetivos:

  Se asigna el fitness como una suma ponderada

$$F(x) = \sum_{i=0}^{n} w_i f_i(x)$$

- Ordenamiento lexicográfico:

  1) Se inicia dando una jerarquía a las funciones objetivo según su importancia.
  2) Se resuelve en el orden de la jerarquía.
  3) Se agregan restricciones a las funciones resueltas previamente.

- Ɛ-constraint (Restricción Ɛ):

  Se escoge una de las funciones objetivo a la cual se va a optimizar. Los demás son tomadas como restricciones las cuales tienen ciertos niveles de holgura denominados Ɛ; a los cuales se les va a modificando sus valores hasta llegar a generar el conjunto de Pareto.

[7] Clasificación sacada de la página 16-20

- Optimización del vector objetivo:

  Se busca encontrar el *sector ideal* por medio de diferentes métodos de optimización. Este método es usado en el algoritmo VEGA.

- Satisfacción de metas:

  Inicialmente se fijan *metas* o valores a cumplir por cada una de las funciones objetivo.

  El objetivo de este mecanismo es llegar a satisfacer cada una de las *metas* definidas.

3.3.2.2. Algoritmos basados en Pareto

Se basan principalmente en la dominancia de Pareto para asignar el fitness.

Se trata de mantener la diversidad de la población mediante distintos mecanismos tales como: (Universidad de la República, 2007)

- Fitness sharing
- Restricción en el cruce
- Crowding
- Isolation by distance

También se puede clasificar a los algoritmos basados en Pareto por su momento de aparición en dos generaciones:

- **Primera Generación:** son algoritmos más simples que los de la *segunda generación*. Utiliza los conceptos de jerarquía de Pareto y nichos.

Este tipo de algoritmos ya no son muy usados. Ej.: NSGA[8], NPGA[9], MOGA[10] y VEGA[11].

- **Segunda Generación:** Se utiliza el concepto de *elitismo* usando la selección y usando una población alterna a partir de 1990 su uso comenzó a ser mucho más común. Ej.: SPEA[12], SPEA II[13], NSGA II[14], MOMGA[15], MOMGA II, PAES[16], PESA[17], PESA II[18], entre otros.

3.4. Medidas de Rendimiento de los Algoritmos

Para poder resolver un problema de optimización multiobjetivo que tenga que ver con uno de la vida real, es necesario usar algoritmos de optimización multiobjetivo. Estos algoritmos generan grupos de soluciones más homogéneas (balanceadas) entre sí dentro de los criterios del problema. Se han propuesto una variedad de métricas para evaluar los resultados de la optimización en problemas multiobjetivo. Existen varios indicadores que permiten evaluar cada uno de esos valores y comparar sus resultados.

A continuación se revisará cada uno de éstas métricas y cuál es su funcionalidad dentro de los problemas de optimización multiobjetivo.

[8] Non Dominated Sorting Algorithm
[9] Niched Pareto Genetic Algorithm
[10] Multo-Objetctive Genetic Algorithm
[11] Vector Evaluated Genetic Algorithm
[12] Strenght Pareto Evolutionary Algorithm
[13] Strength Pareto Evolutionary Algorithm 2
[14] Elitist Non Dominated Sorting Algorithm
[15] Multi- Objective messy Genetic Algorithm
[16] Pareto Archived Evolution Strategy
[17] Pareto Envelope-based Selection Algorithm
[18] Improved Pareto Envelope-based Selection Algorithm

3.4.1. Hipervolumen

El hipervolumen es el espacio de *n* dimensiones contenido dentro de un grupo de puntos que se encuentran en la región del espacio objetivo (*y*). Estas dimensiones son dominadas por *P* y limitado por un punto de referencia *R.*

Se representa de tal forma:

$$X \leq Y$$

$$\text{Si } x_i \leq y_i \text{ para cada } i \in \{i, \dots k\}$$

El hipervolumen de una población *P* se define como:

$$HYP(P) \coloneqq \int_{\mathbb{R}^k} A_p\,(x)dx$$

donde

$$A_p(x)\colon \mathbb{R}^k \rightarrow \{0,1\}$$

$$A_p(x) = 1 \text{ si R es dominado por P caso contrario } A_p = 0$$

El objetivo de un Algoritmo Evolutivo multiobjetivo es encontrar el conjunto P^* de tamaño μ que maximice el hipervolumen tal que

$$HYP(P^*) = HYP_\mu(Y)$$

Donde se define para toda $Y \subset \mathbb{R}^k$

$$maxHYP_\mu(Y) = supHYP(P)$$

Si el conjunto en el espacio objetivo *Y* es finito es supremo de la definición $maxHYP_\mu(Y)$ se convierte en un máximo, pero para conjuntos infinitos es necesario el supremo.

La contribución de un punto *p* a la población *P* es:

$$CON_p(p) \coloneqq HYP(P) - HYP(P - p)$$

La contribución del hipervolumen de un individuo dominado es cero. Representa el valor de la dispersión de las soluciones a través del *frente de Pareto* y la cercanía al *frente óptimo de Pareto*.

3.4.2. Cobertura

Otra medida que permite evaluar los trade-off producidos por diferentes algoritmos evolutivos multiobjetivo es la cobertura.

(S) es una medida de cuan débilmente dominado está el espacio objetivo por un conjunto no dominado *(A)* dado.

3.4.2.1. Tamaño del espacio dominado

Siendo $A = (x_1, x_2, \dots \dots x_k) \subseteq X$

La función *S(A)* da el volumen dado por la unión de los politopos $p_1 \dots . . p_k$, donde cada p_i está formado por la intersección de los hiperplanos originados en x_i junto con los ejes. Para cada eje en el espacio objetivo existe un hiperplano perpendicular al eje y que pasa a través del punto $(f_1(x_i), f_2(x_i) \dots f_k(x_i))$

3.4.2.2. Cobertura de dos conjuntos

Teniendo $A, B \subseteq X$ son dos conjuntos de vectores de decisión.

La función $\mathcal{C}$(cobertura) presenta el par ordenado (A,B) en el intervalo [0,1]

$$\mathcal{C}(A,B) \coloneqq \frac{|\{b \in B | \exists a \in A : a \geq b\}|}{|B|}$$

Si $\mathcal{C}(A,B) = 1$ significa que todos los vetores de decisión en B están débilmente dominados por A. Caso contrario ninguno de los puntos en B están débilmente dominados por A.

3.4.2.2.1. <u>Diferencia de cobertura de dos conjuntos</u>

Hay casos en los cuales la medida de $\mathcal{C}$ no permite decidir si un frente obtenido es mejor que otro para ello se desarrollo otra medida que permite resolver ese problema.

Teniendo $A, B \subseteq X$ son dos conjuntos de vectores de decisión.

La función $\mathcal{D}$ (diferencia de cobertura de dos conjuntos) está definida por:

$$\mathcal{D}(A,B) \coloneqq \mathcal{S}(A+B) - \mathcal{S}(B)$$

Lo que como resultado da el tamaño de espacio débilmente dominado por A pero débilmente domina por B.

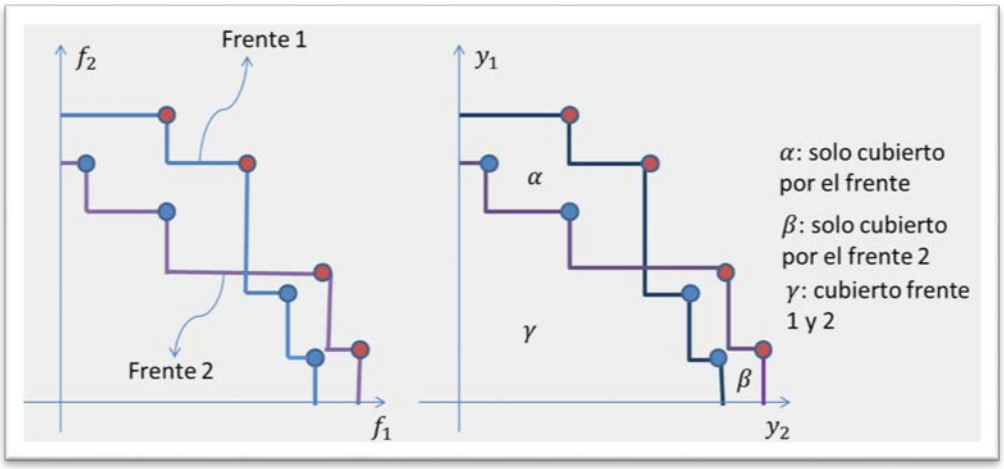


Fig. 3.8 Problema potencial con la medida $\mathcal{C}$(izq) y la la ilistración de la alternativa de la medida $\mathcal{D}$(der). (Coello Coello & Sarker)

3.4.3. Distancia Generacional (GD)

La distancia generacional es la medida que permite estimar cuán lejos están los elementos del $PF_{obtenido}$(*Frente de Pareto*) del PF_{global} y se define

$$GD = \sqrt{\sum_{i=1}^{n} d_i^2}$$

Donde

n= número de vectores en $PF_{obtenido}$

d_i= distancia Euclidiana (medida en el espacio objetivo) entre cada uno de los vectores y el vector más cercano de PF_{global}

Si *GD*=0 todos los elementos de $PF_{obtenido}$ también se encuentran en PF_{global} por lo tanto cualquier otro valor indicará cuán lejos se está del *Frente de Pareto* original del problema analizado.

3.4.4. Distancia Generacional Invertida (IGD[19])

Este indicador permite saber la distancia entre los elementos en el óptimo de Pareto y el conjunto de vectores no dominados.

Si GD=0 todos los elementos generados se encuentran en el *Frente de Pareto* y cubren toda su extensión.

[19] Inverted Generational Distance

3.4.5. Medidas de diversidad

En esta sección se revisarán algunas medidas de diversidad.

3.4.5.1. Medida de Espaciado (1)

La medida de espaciado (Δ) mide cuan equitativamente están distribuidos los puntos del conjunto aproximado dentro del espacio objetivo.

$$\Delta = \sum_{i=1}^{|PF|} \frac{d_i - \bar{d}}{|PF|}$$

Donde

PF: Frente de Pareto

d_i: Distancia Euclidiana entre dos vectores consecutivos dentro del frente no dominado del conjunto aproximado.

$\bar{d}$; promedio de las distancias

Nota: la métrica de Δ es apta solo para problemas con dos objetivos.

3.4.5.2. Medida de Espaciado (2)

La medida Δ' permite evaluar cuan equitativamente se encuentran los puntos del conjunto aproximado, dentro del espacio objetivo.

$$\Delta' = \sqrt{\frac{1}{n-1}\sum_{i+1} n(\bar{d} - d_i)^2}$$

Donde

$$d_i = min_j(\left|f_1^i(x) - f_1^j(x)\right| + \left|f_2^i(x) - f_2^j(x)\right|)$$

$$i,j=1,2,\ldots..n$$

$\bar{d}$: promedio de todas las d_i

n: tamaño del *Frente de Pareto* conocido

4. OPTIMIZACIÓN DE PROBLEMAS MULTIOBJETIVOS CON ALGORITMOS EVOLUTIVOS

En este capítulo se revisará la aplicación de Algoritmos Evolutivos para la optimización de diferentes tipos problemas con múltiples objetivos.

Los algoritmos evolutivos generan grupos de soluciones óptimas a problemas de la vida real. En este caso se estudiará el problema de optimización de problemas con múltiples objetivos y su resolución con el algoritmo evolutivo NSGA II[20].

4.1. Problemas de optimización con múltiples objetivos y algoritmos evolutivos

Para poder resolver problemas de optimización con múltiples objetivos es necesario encontrar los diseños tales que puedan satisfacer de forma satisfactoria todos los objetivos demandados.

Por ejemplo dentro de un problema de distribución de agua en necesario que el costo permita satisfacer la demanda de agua hacia los diferentes nodos (consumidores) y que a su vez la calidad no se vea afectada.

Los sistemas de distribución de agua generalmente se diseñan en base al costo que tendrá, por lo que con la optimización del diseño se busca reducir el costo final. Es imposible el enfocase solo en ese objetivo sin tomar en cuenta otros que son tan importantes como el costo dentro de un sistema de distribución de agua.

Para obtener un modelo eficiente se debe minimizar los costos, maximizar la confiabilidad y la presión en todos los nodos (demanda).

[20] Elitist Non Dominated Sorting Algorithm

Algunas de las mejoras sugeridas podrían ser:

Mejoras en los componentes de la red (agregar nuevas tuberías, bombas y tanques).

El diámetro de las tuberías también afecta; si el diámetro es más pequeño, el costo es bajo, pero a menor diámetro la presión del agua es mayor permitiendo tener una mayor cantidad de nodos a los que se puede llegar (mayor área de cobertura).

4.1.1. Métodos de optimización evolutiva multiobjetivo

A los métodos de optimización evolutiva multiobjetivo se los puede clasificar en dos grupos: *elitistas* y *no elitistas* (ver Capítulo 2). A continuación se nombrarán algunos de los más comunes de cada uno

No Elitista	Elitista
• VEGA (Vector Evaluated Genetic Algorithm)(*Schaffer 1985*) • MOGA (Multi-Objective Genetic Algorithm)(*Fonseca & Fleming 1993*) • NSGA (Non dominated Sorting Genetic Algorithm)(*Srinivas & Deb 1994*)	• SPEA (Strength Pareto Evolutionary Algorithm)(*Zitzler & Thiele 1998*) • NSGA II (Elitist Non dominated Sorting Genetic Algorithm)(*Deb et al. 2000*) • PAES (Pareto Archived Evolution Strategy) (*Knowles & Corne 2000*)

Cuadro 4.1 Ejemplos de Algoritmos Evolutivos Multiobjetivo Elitistas y no Elitistas

Dentro de los problemas de optimización de redes de distribución de agua se utilizan varios de los Algoritmos Evolutivos Multiobjetivo, siendo uno de los más comunes el NSGA II.

4.2. Algoritmo Genético de Ordenamiento Elitista No Dominado

Con problemas de múltiples objetivos se pueden encontrar un grupo de varias soluciones óptimas (Soluciones Óptimas de Pareto) y no una sola solución, lo que junto al hecho que no siempre se tiene toda la información completa, es difícil saber si una solución es mejor que otra.

En los métodos clásicos de optimización, para poder resolver esa dificultad se suele transformar el problema de múltiples objetivos a enfocarse en una solución óptima de Pareto a la vez. Esto obliga a realizar varias simulaciones encontrando diferentes soluciones cada vez.

Los Algoritmos Evolutivos son capaces de encontrar varias soluciones óptimas de Pareto en una sola simulación ya que trabajan con una población de soluciones y permiten mantener una variedad de grupos de soluciones.

Uno de los primeros algoritmos no dominados propuestos para solucionar este tipo de problemas fue el NSGA.

El algoritmo NSGA se caracteriza por establecer un valor de aptitud (fitness) ficticio basándose en el orden de dominancia de las soluciones (Universidad de la República, 2007)

1) Se asigna el fitness a los elementos no dominados
2) Se eliminan los elementos no dominados de la población
3) Se repite el proceso hasta no tener más elementos.

Las desventajas encontradas al usar este algoritmo fueron:

- El ordenamiento no dominado era muy complejo al ser implementado en una computadora (consume muchos recursos) $\rightarrow O(MN^3)$ donde:
    - M: número de funciones objetivo
    - N: tamaño de la población

Al tener una población de mayor tamaño es necesaria mayor capacidad de procesamiento.

- Falta de elitismo: el elitismo permite un rendimiento más rápido de los algoritmos genéticos y permite preservar las mejores soluciones a lo largo de las diferentes generaciones.
- Es necesario definir el parámetro de contribución: para poder asegurar la diversidad de la población es necesario establecer un valor de contribución (σ_{share})

Para ello en el 2000 Deb et al. propusieron una versión mejorada elitista del NSGA: el NSGA II.

4.2.1. Características del NSGA II

El NSGA II es un Algoritmo Evolutivo de múltiples objetivos elitista (ver Capítulo 2) cuyo criterio de selección se basa en la dominancia de Pareto.

Sus principales características son:

- Rápida aproximación de ordenamiento no dominado en el cual todos los individuos son ordenados de acuerdo a su nivel de no dominancia (ver Capítulo 3), permitiendo disminuir la complejidad en la búsqueda de dominancia de $O(MN^3)$ a $O(MN^2)$.
- Al implementar el elitismo se almacena todas las soluciones no dominadas y se realza las propiedades de convergencia.
- Permite la preservación de la diversidad de la población basándose en la distancia de hacinamiento (*crowding distance*), gracias a ello no requiere que se especifiquen parámetros adicionales por lo que no se depende del σ_{share} usado en el NSGA original.
- Se implementa restricciones utilizando una definición de dominancia que no usa funciones de penalización.

4.2.2. Algoritmos de ordenamiento del NSGA II

El algoritmo NSGA II tiene dos tipos de algoritmos de ordenamiento:

- Algoritmo de ordenamiento no dominado (*non dominated sorting algorithm*)
  - Busca las soluciones más próximas al *Frente de Pareto.*
  - Para cada solución se calcula cuantas otras soluciones dominan y cuantas son dominadas por esta.
  - En cada iteración se crea un nuevo frente con las soluciones retiradas del conjunto.
- Ordenamiento por la distancia de hacinamiento (*crowding distance sorting*)
  - Busca las soluciones mejor distribuidas en el espacio.
  - Una función calcula la distancia entre las soluciones que se encuentran en el mismo frente para garantizar un mejor ajuste (*fitness*) con las áreas menos pobladas dentro del espacio de búsqueda.
  - Se recalcula la distancia de las otras soluciones con una fórmula normalizada que utiliza la diferencia entre la distancia de cada solución y las dos más próximas a esta.

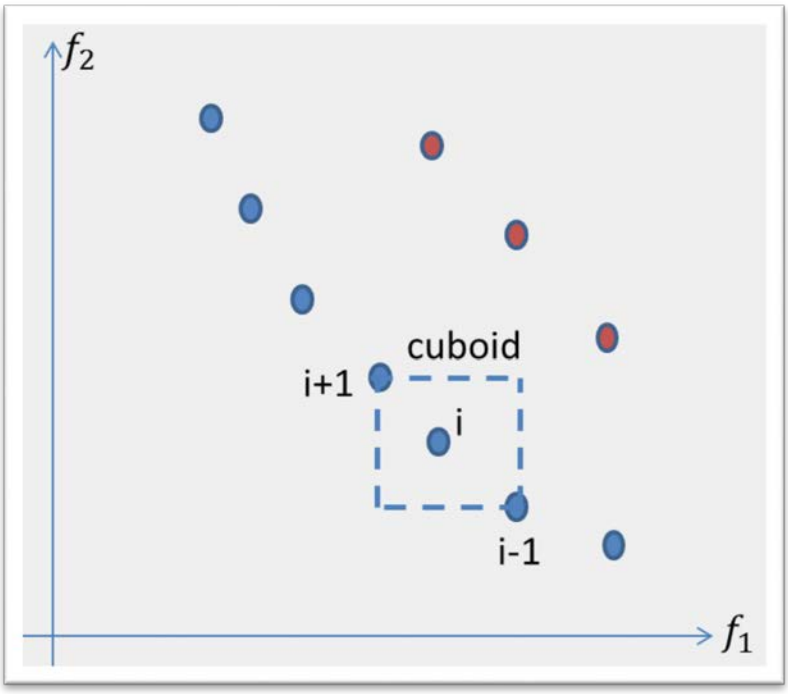


Fig.4.1 Cálculo de *crowding distance* los puntos ● son las soluciones que pertenecen al mismo frente no dominado[21] (Zitzler, 1999)

4.2.3. Procedimiento del algoritmo NSGA II

Un algoritmo genético convencional está formado por cuatro etapas: Generación de la población inicial, cruce, mutación y selección. Con estos operadores, luego de cada iteración es posible ir mejorando los resultados de una generación a otra.

[21] En la figura la distancia de hacinamiento (*crowding distance*) de la **i** a la solución en el frente es la longitud promedio del lado del *cuboid*.

Fig. 4.2 Diagrama de bloques de un algoritmo genético clásico

El algoritmo NSGA II es un algoritmo genético multiobjetivo que se basa en el concepto de dominancia (Pareto) para clasificar a la población en frentes de acuerdo con su grado de dominancia.

Los individuos que se encuentran primeros son considerados como las mejores soluciones de esa generación. Con este concepto es posible encontrar las mejores soluciones que se encuentren más cercanas al óptimo de Pareto y que se adaptan mejor al problema presentado. Algunos aspectos importantes a tomar en cuenta son:

- Dividir la población utilizando el criterio de *Dominancia* y agregar el concepto de *Elitismo* para que el algoritmo de prioridad a las mejores soluciones.
- Los individuos del frente ***n*** son mejores que los que se encuentran en el ***n+1***.

Para la selección de individuos el NSGA II utiliza los mecanismos de Ordenamiento rápido no dominado (*Fast Non dominated Sorting*) y Distancia de Hacinamiento (*Crowding Distance*).

Inicialmente se tiene una población no clasificada a la cual se le aplica el criterio de dominancia y se comparan todos los miembros entre sí para su clasificación.

Se clasifican a los individuos en frentes de acuerdo a los valores del criterio de dominancia. Asignando al primer frente a los mejores y al último los peores. Esta etapa continúa hasta que se hayan clasificado a todos los individuos de la población.

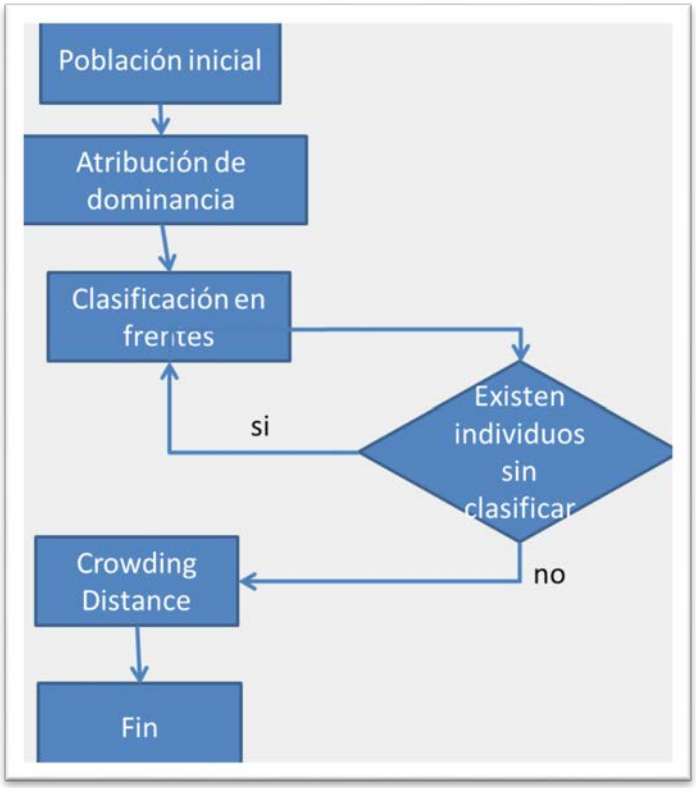


Fig. 4.3 Diagrama de bloque del algoritmo NSGA II

A continuación a los individuos de cada frente se les irá ordenando con el operador de diversidad (crowding distance). El ordenamiento por distancia de hacinamiento (*crowding distance*) irá ordenando a los individuos de tal manera que no existan aglomeraciones en un mismo punto.

4.2.3.1. Ordenamiento Rápido no dominado (Fast Non dominated Sorting)

El algoritmo de ordenamiento rápido no dominado se ejecuta en dos etapas:

1) En la primera se analizan los individuos de la población **P**, comparándolos entre sí para clasificarlos por su grado de dominancia **np** (número de individuos que dominan a **p**, siendo **p** un individuo de la población **P**).

   p es dominada por **x** individuos de la población **P**, por lo tanto **np=x**

   Si al final de la etapa existe un individuo **p** tal que **np=0**, se dice que no es dominado por ningún otro de la población **P**, por lo tanto ese individuo será parte del primer frente (mejores individuos de la población actual).

```
Para cada p (pertenece a) P
        para cada q (pertenece a) P
                si (p domina a q) entonces
                        Almacenar q en S_p
                caso contrario
                        incrementa np
                fin si
        fin para
        si np (es igual a) 0 entonces
                Almacenar p en F_1
        fin si
Fin
S_p: Lista de individuos dominados por p.
```

Fig. 4.4 Algoritmo primera etapa Ordenamiento Rápido No dominado

2) En la segunda etapa se separa a los individuos en las diferentes categorías (frentes) de acuerdo a sus valores de dominancia **np**.

   Al asignar a un individuo a un frente, este es retirado del contexto de la población decreciendo el valor de **np** de los individuos dominados por este. El proceso se repite hasta que no existan más individuos en la población.

```
Inicializar i=0
Mientras F_i ≠ 0
        H=0
        para cada p (pertenece a) F_i
                para cada q (pertenece a) S_p
                        disminuye el valor de nq
                        si nq==0 entonces
                                almacenar el valor de q en H
                                incrementar el valor de i en uno
                        fin si
                        Almacenar los valores de H en F_i (el cual será el frente actual
                                                           en la siguiente iteración)

                        fin para
                fin para
        fin mientras
Fin
```

Fig. 4.5 Algoritmo segunda etapa Ordenamiento Rápido No dominado

4.2.3.2. Distancia de Hacinamiento (Crowding Distance)

La distancia de hacinamiento (*crowding distance*) permite garantizar que haya un mayor espacio entre dos resultados a lo largo del *Frente de Pareto.*

La distancia de hacinamiento (*crowding distance*) mide la distancia entre un punto inicial **i** dentro de la población y dos puntos localizados en los extremos del punto seleccionado (**i+1**) y (**i-1**).

Lo que se desea lograr con este proceso es encontrar esos puntos extremos y priorizar los más distantes durante el proceso de selección y así tener una distribución más uniforme a lo largo del *Frente de Pareto*.

La fórmula usada para calcular la distancia de dos individuos en relación a un punto medio es

$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{I_{j+1}^m} - f_m^{I_{j-1}^m}}{f_m^{max} - f_m^{min}}$$

Donde

$d_{I_j^m}$: es la distancia del j-esimo elemento del conjunto I en relación al objetivo m.

$f_m^{I_{j+1}^m}$ y $f_m^{I_{j-1}^m}$: son los valores del m-ésimo objetivo en las posiciones j+1 y j-1 respectivamente.

$f_m^{max} - f_m^{min}$: es la diferencia entre los objetivos máximo y mínimo del m-esimo objetivo.

Se repite el procedimiento dependiendo la cantidad **m** de objetivos existentes en el problema.

Distancia de cada punto en el conjunto **S**
Asignar *crowding distance* **(S)**
i=|S| (**|S|** representa el número de elementos en el conjunto **S**)
Para cada **I** del conjunto $S[i]_{distancia} = 0$
    para cada objetivo **m**
        ordenar **(S,m)** (ordena cada elemento de **S** respecto a **m**)
    $S[0]_{distancia}$ o $S[l]_{distancia}$ escoger el valor más alto
    para **i=1** en (l-1) para todo los otros puntos
        $S[i]_{distancia}$=$S[i]_{distancia}$+(S[i+1].m-S[i-1].m)

Fig. 4.6 Algoritmo en seudocódigo de ordenamiento por distancia de hacinamiento

4.2.3.3. Seudocódigo NSGA II

```
Algoritmo NSGA-II (G(N,E), s, T)
Inicio
    Para (cada t ∈ T) haga
        Mientras Que (exista un ruta del nodo s a nodo t) haga
            Conjunto_Rutas ← Ruta desde s hasta t;
        Fin Mientras Que
    Fin Para
    Generar aleatoriamente P_0
    Hacer ordenamiento rápido no dominado de P_0
    Aplicar los operadores de selección, cruce y mutación para generar una población hija Q_0
    Hacer t ←1
    Hacer R_t ← Ø
    Mientras Que t < g_max y Cont_Convergencia < g_max
        Hacer R_t ← P_t ∪ Q_t
        Calcular el número de saltos y el retardo de los miembros R_t
        F ß OrdenamientoRápidoNoDominado (R)
        Mientras Que ( | P_t+1 | < N ) hacer
            CalcularDistanciaCrowding (F_i)
            P_t+1 ← P_t+1 ∪ F_i
        Fin Mientras Que
        Ordenar P_t+1 en forma descendente, utilizando el operador ≻_γ
        Escoger los N primeros elementos de P_t+1.
        Generar Q_t+1, aplicando los operadores de selección cruce y mutación sobre P_t+1
        Hacer t ← t + 1
    Fin Mientras Que
Fin
```

Fig. 4.7 Seudocódigo algoritmo NSGA II

4.3. Rendimiento del método NSGA II

Para la comprobación del algoritmo NSGA II se utilizó el **MOEA Framework** A Free and Open Source Java Framework for Multiobjective Optimization. Versión 2.1

http://www.moeaframework.org. A continuación se siguió los pasos de instalación descritos en el Anexo Manual de usuario MOEAFramework-2.1.

4.3.1. Licencia MOEA Framework

Copyright 2011-2014 David Hadka. All Rights Reserved.

The MOEA Framework is free software: you can redistribute it and/ir modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The MOEA Framework is dristributes in the hope that it will be useful, but WHITHOUT ANY WARRANTY; whithout even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesset General Public License for more details.

You sould have received a copy of the GNU Lesser General Public License along with the MOEA Framework. Of not, see http://www.gnu.org/licenses/

4.3.2. Descripción del framework, selección del algoritmo y selección del problema

El framework tiene implementado varios algoritmos dentro de la carpeta src/org/moeframework/algorithm/ tales como:

- ε-MOEA
- NSGA-II
- ε-NSGA-II
- MOEA/D
- GDE3
- RandomSearch

- JMetalAlgorithms
  - AbYSS
  - CellDE
  - DENSEA
  - FastPGA
  - IBEA
  - MOCell
  - MOCHC
  - OMOPSO
  - PAES
  - PESA-II
  - SMPSO
  - SMSEMOA
  - SPEA2

Para comprobar la eficacia del algoritmo NSGA II sobre otros algoritmos de tipo elitista basados en pareto se comparará los resultados del NSGA II contra los resultados del algoritmo SPEA II que también pertenece a la segunda generación de algoritmos basado en pareto de tipo elitista.

El framework también ya tiene implementados varios problemas para poder ser probados con los diferentes algoritmos entre los cuales se hallan:

- CF1
- CF2
- CF3
- CF4
- CF5
- CF6
- CF7
- CF8
- CF9
- CF10
- DTLZ1
- DTLZ2
- DTLZ3
- DTLZ4
- DTLZ7
- LZ1
- LZ2
- LZ3
- LZ4
- LZ5
- LZ6
- LZ7
- LZ8
- LZ9
- UF1
- UF2
- UF3
- UF4
- UF5
- UF6
- UF7
- UF8
- UF9
- UF10
- UF11
- UF12
- UF13
- WFG1
- WFG2
- WFG3
- WFG4
- WFG5
- WFG6
- WFG7
- WFG8
- WFG9
- ZDT1
- ZDT2
- ZDT3
- ZDT4
- ZDT5
- ZDT6
- Belegundu
- Binh
- Binh2
- Binh3

- Binh4
- Fonseca
- Fonseca2
- Jimenez
- Kita
- Kursawe
- Laumanns
- Lis
- Murata
- Obayashi
- OKA1
- OKA2
- Osyczka
- Osyczka2
- Poloni
- Quagliarella
- Rendon
- Rendon2
- Schaffer
- Schaffer2
- Srinivas
- Tamaki
- Tanaka
- Viennet
- Viennet2
- Viennet3
- Viennet4

Para la prueba de los algoritmos se escogió el grupo de problemas de prueba DTLZ(1,2,3,4), para el cual se hizo 5 corridas para 3, 4 y 5 objetivos por cada método.

4.3.2.1. Especificaciones

El framework contiene tres clases principales las cuales permiten comprobar el funcionamiento del algoritmo. Executor, Instrumenter y Analyser.

Clase Executor: permite la construcción y ejecución de los algoritmos.

Clase Instrumenter: permite analizar el rendimiento de los algoritmos. Captura el comportamiento del algoritmo durante la corrida y como van cambiando los elementos en el tiempo.

Clase Analyser: permite comparar los resultados del algoritmo enfocándose en la aproximación al frente de Pareto y lo compara con un conjunto de referencia.

4.3.3. Conjunto de Problemas DTLZ

Los problemas seleccionados se basan en la serie de problemas DTLZ (Deb, Thiele, Laumanns, & Zitzler, 2002)

El problema de prueba DTLZ1 es un problema multiobjetivo con un frente óptimo de Pareto lineal y se lo representa de la siguiente manera:

$$f_1(x) = \frac{1}{2} \cdot x_1 \cdot x_2 \cdot \ldots \cdot x_{M-1} \cdot (1 + g(x))$$
$$f_2(x) = \frac{1}{2} \cdot x_1 \cdot x_2 \cdot \ldots \cdot (1 - x_{M-1}) \cdot (1 + g(x))$$
$$\vdots$$
$$f_M(x) = \frac{1}{2} \cdot (1 - x_1) \cdot (1 + g(x))$$
$$g(x) = 100 \cdot [k + \sum_{i=M}^{n} (x_i - 0.5)^2 - cos(20 \cdot \pi \cdot (x_i - 0.5))]$$

Donde:

$n = M + k - 1$ siendo $k=5$

$x_i \in [0,1]$ siendo $i=1,\ldots,n$

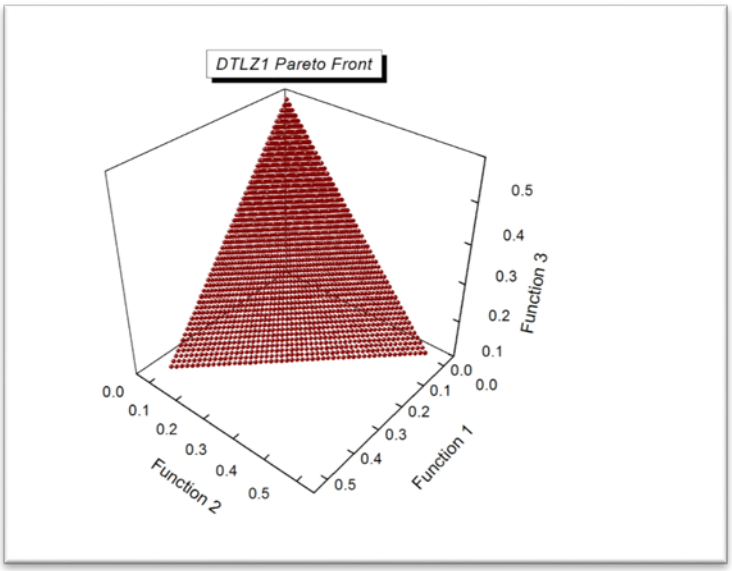


Fig. 4.8 Representación del frente de Pareto para el problema DTLZ1

El problema de prueba DTLZ2 tiene un frente óptimo de Pareto cuya forma es cóncava y se lo representa de la siguiente manera:

$$
\begin{aligned}
f_1(x) &= \cos(x_1\frac{\pi}{2}) \cdot \cos(x_2\frac{\pi}{2}) \cdot \ldots \cdot \cos(x_{M-1}\frac{\pi}{2}) \cdot (1+g(x)) \\
f_2(x) &= \cos(x_1\frac{\pi}{2}) \cdot \cos(x_2\frac{\pi}{2}) \cdot \ldots \cdot \operatorname{sen}(x_{M-1}\frac{\pi}{2}) \cdot (1+g(x)) \\
f_3(x) &= \cos(x_1\frac{\pi}{2}) \cdot \cos(x_2\frac{\pi}{2}) \cdot \ldots \cdot \operatorname{sen}(x_{M-2}\frac{\pi}{2}) \cdot (1+g(x)) \\
&\vdots \\
f_{M-1}(x) &= \cos(x_1\frac{\pi}{2}) \cdot \operatorname{sen}(x_2\frac{\pi}{2})(1+g(x)) \\
f_M(x) &= \operatorname{sen}(x_1\frac{\pi}{2}) \cdot (1+g(x)) \\
g(x) &= \sum_i = (x_i - 0.5)^2
\end{aligned}
$$

Donde:

$n = M + k - 1$ siendo *k=10*

$x_i \in [0,1]$ siendo *i=1,…,n*

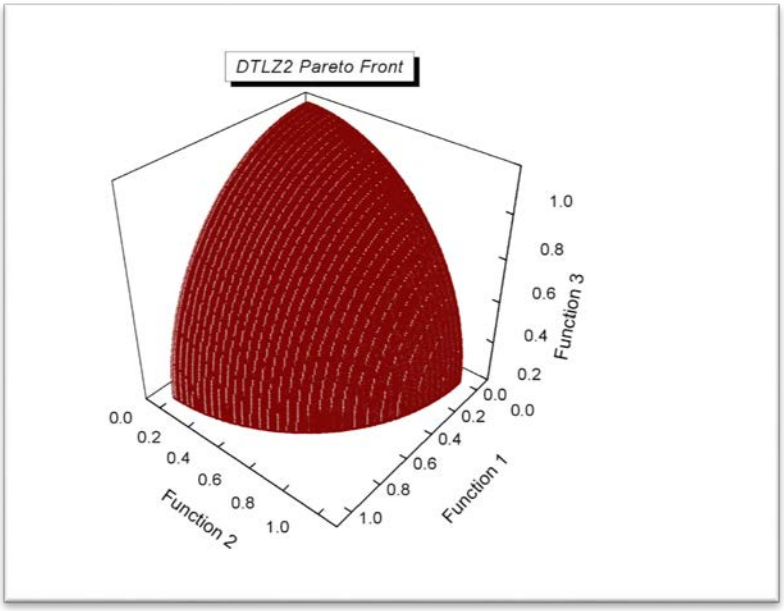


Fig. 4.9 Representación del frente de Pareto para el problema DTLZ2

El problema DTLZ3 tiene un frente óptimo de Pareto cóncavo y multimodal cuya forma es similar al DTLZ2 y se representa de la siguiente manera:

$$
\begin{aligned}
f_1(x) &= \cos(x_1\frac{\pi}{2}) \cdot \cos(x_2\frac{\pi}{2}) \cdot \ldots \cdot \cos(x_{M-1}\frac{\pi}{2}) \cdot (1+g(x)) \\
f_2(x) &= \cos(x_1\frac{\pi}{2}) \cdot \cos(x_2\frac{\pi}{2}) \cdot \ldots \cdot \operatorname{sen}(x_{M-1}\frac{\pi}{2}) \cdot (1+g(x)) \\
f_3(x) &= \cos(x_1\frac{\pi}{2}) \cdot \cos(x_2\frac{\pi}{2}) \cdot \ldots \cdot \operatorname{sen}(x_{M-2}\frac{\pi}{2}) \cdot (1+g(x)) \\
&\vdots \\
f_{M-1}(x) &= \cos(x_1\frac{\pi}{2}) \cdot \operatorname{sen}(x_2\frac{\pi}{2}) \cdot (1+g(x)) \\
f_M(x) &= \operatorname{sen}(x_1\frac{\pi}{2}) \cdot (1+g(x)) \\
g(x) &= 100 \cdot [k + \sum_{i=M}^{n} (x_i - 0.5)^2 - \cos(20 \cdot \pi \cdot (x_i - 0.5))]
\end{aligned}
$$

Donde:

$n = M + k - 1$ siendo *k=10*

$x_i \in [0,1]$ siendo *i=1,…,n*

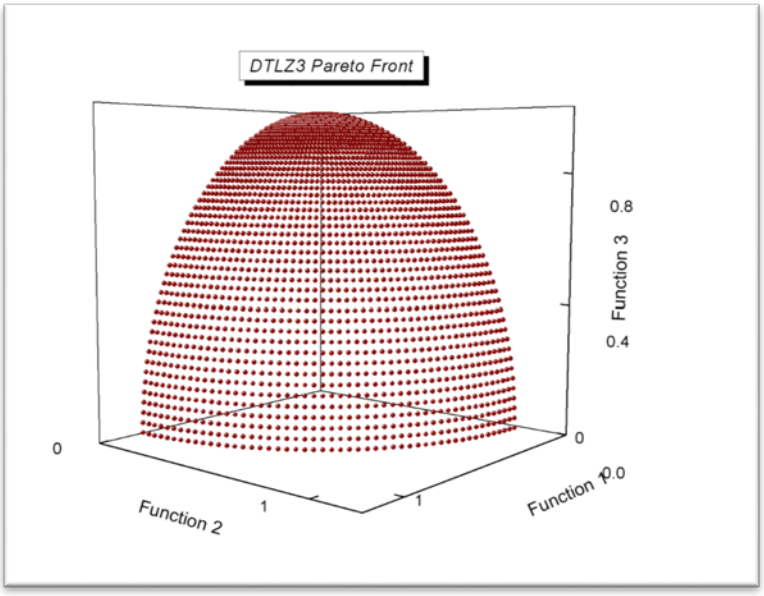


Fig. 4.10 Representación del frente de Pareto para el problema DTLZ3

El problema de prueba DTLZ4 tiene un frente óptimo de Pareto cóncavo, separable y multimodal cuya forma es similar al DTLZ2 y se representa de la siguiente manera:

$$
\begin{aligned}
f_1(x) &= \cos(x_1^{\alpha}\frac{\pi}{2}) \cdot \cos(x_2^{\alpha}\frac{\pi}{2}) \cdot \ldots \cdot \cos(x_{M-1}^{\alpha}\frac{\pi}{2}) \cdot (1+g(x)) \\
f_2(x) &= \cos(x_1^{\alpha}\frac{\pi}{2}) \cdot \cos(x_2^{\alpha}\frac{\pi}{2}) \cdot \ldots \cdot \operatorname{sen}(x_{M-1}^{\alpha}\frac{\pi}{2}) \cdot (1+g(x)) \\
f_3(x) &= \cos(x_1^{\alpha}\frac{\pi}{2}) \cdot \cos(x_2^{\alpha}\frac{\pi}{2}) \cdot \ldots \cdot \operatorname{sen}(x_{M-2}^{\alpha}\frac{\pi}{2}) \cdot (1+g(x)) \\
&\vdots \\
f_{M-1}(x) &= \cos(x_1^{\alpha}\frac{\pi}{2}) \cdot \operatorname{sen}(x_2^{\alpha}\frac{\pi}{2}) \cdot (1+g(x)) \\
f_M(x) &= \operatorname{sen}(x_1^{\alpha}\frac{\pi}{2}) \cdot (1+g(x)) \\
g(x) &= \sum_i (x_i - 0.5)^2
\end{aligned}
$$

Donde:

$n = M + k - 1$ siendo *k=10* y $\alpha = 100$

$x_i \in [0,1]$ siendo *i=1,…,n*

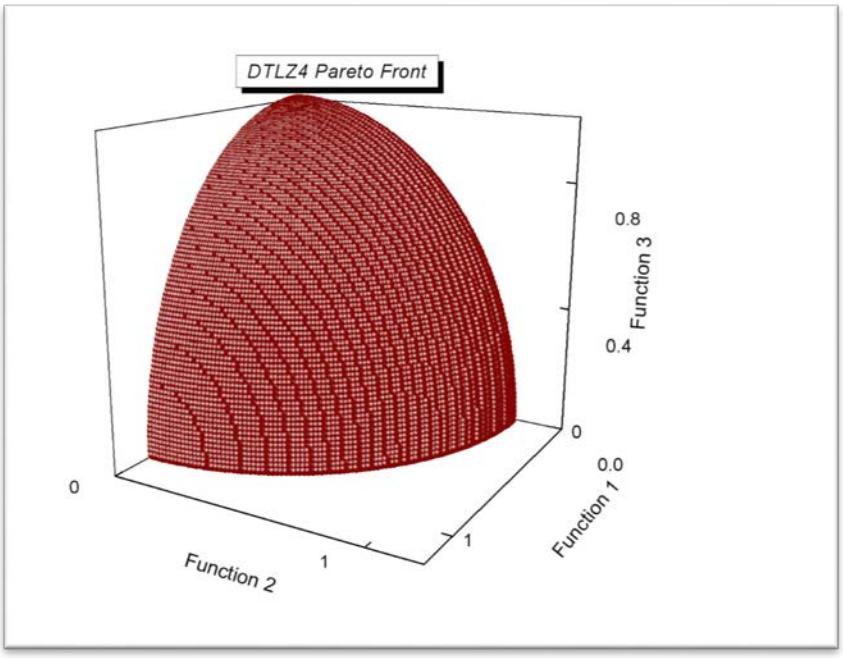


Fig. 4.11 Representación del frente de Pareto para el problema DTLZ4

El problema DTLZ7 tiene un frente óptimo de Pareto discontinuo y se representa de la siguiente manera:

$$
\begin{aligned}
f_1(x) &= x_1 \\
f_2(x) &= x_2 \\
&\vdots \\
f_{M-1}(x) &= x_{M-1} \\
f_M(x) &= (1 + g(x_M)) \cdot h(f_1, f_2, \ldots, f_{M-1} g(x)) \\
g(x) &= 1 + \frac{9}{k} \cdot \sum_{i=2}^{n} x_i \\
h(f_1, f_2, \ldots, f_{M-1} g(x)) &= M - \sum_{i=1}^{M-1} \left(\frac{f_i}{1 + g(x)} (1 + \operatorname{sen}(3 \cdot \pi \cdot f_i)) \right)
\end{aligned}
$$

Donde:

$n = M + k - 1$ siendo $k=20$

$x_i \in [0,1]$ siendo $i=1,\ldots,n$

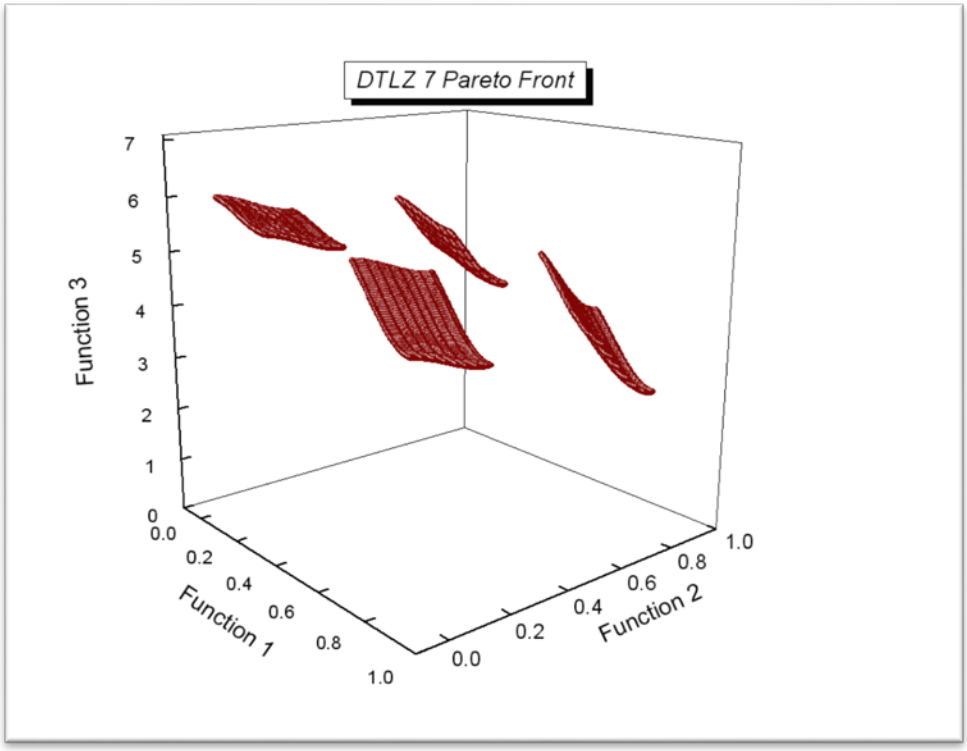


Fig. 4.12 Representación del frente de Pareto para el problema DTLZ7

4.3.4. Ejecución del Algoritmo

Se ejecuta la clase ***Analyser*** para los algoritmos NSGA II y SPEA2 con 50 corridas cada uno, cada corrida con 10000 evaluaciones para cada problema.

Los resultados están organizados de tal forma que en cada nivel se muestran las diferentes métricas estudiadas en el capítulo 3.

Dentro de cada una de las métricas analizadas se tienen los valores mínimo, media y máximo de cada una.

Finalmente se analiza la significancia estadística de los resultados obtenidos presentado en el campo ***Indifferent*** **.** Si este campo presenta algún valor, este indica que los valores de la media de la muestra de esas medidas son indiferentes, lo cual lleva a concluir que el desempeño de los métodos es el mismo.

El MOEA Framework utiliza los métodos de Mann-Whitney y Kruskal-Wallis para probar la significancia estadística.

4.3.4.1. DTLZ1

Resultados:

		DTLZ1					
		3 Objetivos		4 Objetivos		5 Objetivos	
		NSGAII	SPEA2	NSGAII	SPEA2	NSGAII	SPEA2
Hypervolume	Min	0,967768138	0,998816647	0	0	0,99842502	0,9994464
	Media	0,996594871	0,999889079	0	0	0,99951601	0,99994225
	Max	0,999547169	0,999995071	0	0	0,99994885	0,99999903
	Indiferente			SPEA2	NSGAII		
Generational Distance	Min	0,008073606	0,002425644	26,86009214	7,575965824	0,00320647	0,00320802
	Media	0,435792362	0,066814602	38,28923766	24,66845309	0,00562355	0,00834058
	Max	1,367833583	0,465222414	50,5591399	37,31458643	0,00921475	0,01385365
	Indiferente						
Inverted GenerationalDistance	Min	0,002137809	0,002211755	1,196172159	0,159455128	0,0024013	0,0021848
	Media	0,004064844	0,004859349	2,991546109	0,857431737	0,00290647	0,00277095
	Max	0,007889881	0,007545976	6,440154783	1,599090123	0,00348215	0,00368731
	Indiferente					SPEA2	NSGAII
Additive Epsilon	Min	0,019780037	0,004199481	15,84737694	2,495310231	0,02297137	0,01471669
	Media	0,044679215	0,014505953	39,10583221	10,83358918	0,0925687	0,05120584

Indicator	Max	0,103985346	0,037081629	82,36444358	20,65715944	0,15305232	0,08753951
	Indiferente						
Maximum Pareto Front Error	Min	0,357425726	0,104116158	637,4627454	315,3170592	0,09274228	0,11365763
	Media	21,2318801	3,737394102	882,0597057	686,9400508	0,19204463	0,27891021
	Max	56,50870801	28,17077218	1307,840786	1075,920885	0,3452367	0,41910616
	Indiferente						
Spacing	Min	1,833236409	0,319409062	5,578834809	4,577913741	20,2810258	13,5206782
	Media	9,102791246	3,678940569	11,16171234	10,02204722	28,5838977	17,3222114
	Max	22,35567801	10,65777401	25,23505442	22,11793738	36,1570705	28,3065927
	Indiferente						
Contribution	Min	0	0,0	0	0	0	0
	Media	0	0,003690037	0	0	0,00667408	0,0083426
	Max	0,033210332	0,195571956	0	0	0,02669633	0,04449388
	Indiferente	SPEA2	NSGAII	SPEA2	NSGAII	SPEA2	NSGAII

4.3.4.2. DTLZ2

Resultados:

		DTLZ2					
		3 Objetivos		4 Objetivos		5 Objetivos	
		NSGAII	SPEA2	NSGAII	SPEA2	NSGAII	SPEA2
Hypervolume	Min	0,419842433	0,453754041	0,324276909	0,429328174	0,90865057	0,96836117
	Media	0,43205462	0,459029408	0,369160354	0,467431105	0,95462686	0,98391561
	Max	0,444356402	0,464254035	0,418395314	0,491314862	0,9760003	0,98995491
	Indiferente						
Generational Distance	Min	0,000707804	6,33E-04	0,012056147	0,010841048	0,00624671	0,00392964
	Media	0,001092281	0,001077463	0,016150392	0,01344255	0,00948593	0,00792286
	Max	0,001662046	0,001950976	0,025734904	0,017354859	0,01418328	0,01409139
	Indiferente	SPEA2	NSGAII				
Inverted Generational Distance	Min	0,00093413	0,000745086	0,011881352	0,009544223	0,00327187	0,00272298
	Media	0,001038497	0,000767452	0,013483914	0,010413841	0,00382736	0,00293351
	Max	0,001155356	0,000789074	0,015120796	0,011598645	0,00471761	0,0034196
	Indiferente						
Additive	Min	0,098800402	0,066371492	0,186985907	0,148266936	0,17734744	0,13325156

Epsilon Indicator	Media	0,119547012	0,07839528	0,240616683	0,19709357	0,22799778	0,16520561
	Max	0,182227178	0,109943154	0,293259659	0,244111239	0,28841527	0,20466038
	Indiferente						
Maximum Pareto Front Error	Min	0,024005689	0,024338251	0,194052897	0,184368645	0,13624454	0,14489516
	Media	0,035828563	0,039388081	0,392171822	0,333704445	0,22569147	0,26959008
	Max	0,089545025	0,13068376	0,679331253	0,478000223	0,35600211	0,48045347
	Indiferente						
Spacing	Min	0,045613922	0,019168347	0,107574046	0,042870014	0,24913043	0,0974442
	Media	0,055461063	0,023901491	0,1346839	0,053782004	0,31932129	0,12106675
	Max	0,063593885	0,031781586	0,173742621	0,067969906	0,37598084	0,1587288
	Indiferente						
Contribution	Min	0,006544503	0,00710546	0	0	0,00043937	0,00966608
	Media	0,009629768	0,010471204	0	0	0,00395431	0,01493849
	Max	0,013649963	0,015145849	0	0	0,01230228	0,02855888
	Indiferente			SPEA2	NSGAII		

4.3.4.3. DTLZ3

Resultados:

		DTLZ3					
		3 Objetivos		4 Objetivos		5 Objetivos	
		NSGAII	SPEA2	NSGAII	SPEA2	NSGAII	SPEA2
Hypervolume	Min	0,252687411	0	0	0	0,97315875	0,99664779
	Media	0,459438719	0,481707497	0	0	0,99480812	0,99910363
	Max	0,475201932	0,487498558	0	0	0,9995049	0,99996648
	Indiferente			SPEA2	NSGAII		
Generational Distance	Min	0,000152375	0	44,02251475	33,18975651	0,00448949	0,00410028
	Media	0,001379396	0,001103893	55,86237165	42,31297603	0,0082056	0,01422074
	Max	0,001748811	0,002110288	71,77168873	54,48512488	0,01710392	0,02026381
	Indiferente						
Inverted Generational Distance	Min	0,000945099	0,00075037	5,947163843	1,931503127	0,00360947	0,00313091
	Media	0,001049947	0,000775902	9,629075083	4,285526179	0,00404219	0,00363801
	Max	0,008350209	0,013156387	14,46586975	6,694302248	0,00583975	0,00438156
	Indiferente						

Additive Epsilon Indicator	Min	0,086266165	0,06806479	62,93244873	28,78635746	0,08554882	0,04697538
	Media	0,115049604	0,087710595	111,6631039	56,61074995	0,16353835	0,10128076
	Max	0,689923621	1	189,7861111	90,05834748	0,29324558	0,16010594
	Indiferente						
Maximum Pareto Front Error	Min	0,006685843	0	791,3050936	796,1162022	0,13956764	0,20020569
	Media	0,043088943	0,038675276	1136,549473	963,0111025	0,24961613	0,43043773
	Max	0,078172462	0,16976587	1559,065532	1264,975946	0,4724963	0,62105569
	Indiferente	SPEA2	NSGAII				
Spacing	Min	0,008557244	0	29,92483918	18,09218269	119,393327	54,7117178
	Media	0,056462826	0,022848003	60,56884309	39,59048674	147,342953	86,8354386
	Max	0,067660731	0,028417422	81,16434562	84,84554478	189,881596	121,664391
	Indiferente						
Contribution	Min	0,005928854	0,000197628	0	0	0,0009058	0,00271739
	Media	0,009387352	0,010474308	0	0	0,00634058	0,00951087
	Max	0,012648221	0,017391304	0	0	0,03442029	0,05434783
	Indiferente			SPEA2	NSGAII		

4.3.4.4. DTLZ4

Resultados:

		DTLZ4					
		3 Objetivos		4 Objetivos		5 Objetivos	
		NSGAII	SPEA2	NSGAII	SPEA2	NSGAII	SPEA2
Hypervolume	Min	0	0	0,32732321	0,20725887	0,90465054	0,95825077
	Media	0,386097923	0,411277149	0,383947443	0,43427466	0,951006	0,98067476
	Max	0,400354349	0,419954799	0,441255856	0,502170877	0,97373271	0,98875666
	Indiferente						
Generational Distance	Min	0,000480899	0	0,011048968	0,009227415	0,00865616	0
	Media	0,001476589	0,001139662	0,014725849	0,014031744	0,01419422	0,0116892
	Max	0,002902261	0,00415762	0,024335695	0,024660855	0,02071989	0,02086655
	Indiferente			SPEA2	NSGAII		
Inverted Generationa lDistance	Min	0,000977039	0,000795119	0,01143754	0,009575798	0,00339373	0,00271989
	Media	0,00112918	0,00082379	0,012842945	0,011629382	0,00427423	0,00321395
	Max	0,014007575	0,014007561	0,014656417	0,056299154	0,00533265	0,00780203

	Indiferente			SPEA2	NSGAII		
Additive Epsilon Indicator	Min	0,09515259	0,067906527	0,182742646	0,167680514	0,17683296	0,137699
	Media	0,127917181	0,089040813	0,239920496	0,239905911	0,24322594	0,18520763
	Max	1,000003568	1,000001582	0,273503072	0,715151186	0,30068855	0,28440648
	Indiferente			SPEA2	NSGAII		
Maximum Pareto Front Error	Min	1,45926E-06	0	0,171539841	0,170433368	0,18357475	0
	Media	0,04633454	0,044519235	0,362943209	0,324854573	0,36527264	0,32412728
	Max	0,129724629	0,201602563	0,746288695	0,737691683	0,56300747	0,52216818
	Indiferente	SPEA2	NSGAII	SPEA2	NSGAII		
Spacing	Min	0	0	0,097030058	0,0034597	0,24913043	0,03066135
	Media	0,057957522	0,022849669	0,129953063	0,051744753	0,31932129	0,12749177
	Max	0,065716872	0,029296495	0,166315355	0,079805173	0,37598084	0,19852082
	Indiferente			SPEA2	NSGAII		
Contribution	Min	0	0	0	0	0	0,00133869
	Media	0,009500707	0,010208207	0	0	0,00178492	0,00624721
	Max	0,012532848	0,01637356	0	0	0,00580098	0,04462294
	Indiferente			SPEA2	NSGAII		

4.3.4.5. DTLZ7

Resultados:

		DTLZ7					
		3 Objetivos		4 Objetivos		5 Objetivos	
		NSGAII	SPEA2	NSGAII	SPEA2	NSGAII	SPEA2
Hypervolume	Min	0,364204566	0,358471078	0,008726038	0,006917679	0,55460298	0,61684882
	Media	0,395726666	0,399196469	0,034965424	0,040744795	0,65598085	0,67585171
	Max	0,417982122	0,420818056	0,074411649	0,076187557	0,69686596	0,71623671
	Indiferente	SPEA2	NSGAII	SPEA2	NSGAII		
Generational Distance	Min	0,001515604	0,001502303	0,028443899	0,0287476	0,00833201	0,00448844
	Media	0,002433253	0,002482746	0,036682796	0,041061844	0,01289875	0,00789028
	Max	0,003995665	0,004949666	0,049694714	0,053294452	0,01758403	0,01149985
	Indiferente	SPEA2	NSGAII				
Inverted Generationa	Min	0,00111949	0,000972669	0,014445539	0,013377261	0,00449729	0,00404223
	Media	0,00130518	0,001053467	0,018840284	0,017669921	0,00508038	0,00431098

lDistance	Max	0,001561268	0,00995732	0,02302569	0,054571892	0,00561155	0,00469686
	Indiferente			SPEA2	NSGAII		
Additive Epsilon Indicator	Min	0,062125756	0,061578739	0,280476802	0,230831703	0,16019716	0,10501719
	Media	0,093637868	0,076514616	0,376243131	0,302469734	0,19451494	0,13526708
	Max	0,19468586	0,351259501	0,474871144	0,765000902	0,31085785	0,18457437
	Indiferente						
Maximum Pareto Front Error	Min	0,048375576	0,043869501	0,494407284	0,598724683	0,2062578	0,14160209
	Media	0,073375885	0,079179059	0,734565227	0,920342997	0,27354646	0,25259123
	Max	0,203488804	0,278159444	1,311098493	1,351807119	0,43753957	0,4284811
	Indiferente	SPEA2	NSGAII				
Spacing	Min	0,052975188	0,021827932	0,141406799	0,047605549	0,23378901	0,11637161
	Media	0,068425238	0,040924789	0,173302219	0,102416623	0,29952842	0,17716522
	Max	0,10051165	0,119131152	0,225853375	0,161494735	0,38418919	0,31422791
	Indiferente						
Contribution	Min	0,001610306	0	0	0	0,00174902	0,00174902
	Media	0,010198604	0,006977992	0	0	0,0080892	0,01071272
	Max	0,030595813	0,030595813	0	0	0,02842151	0,02404897
	Indiferente	SPEA2	NSGAII	SPEA2	NSGAII		

4.3.5. Diagnóstico de rendimiento del NSGA II

Con MOEA Framework permite analizar el rendimiento del algoritmo luego de diferente número de evaluaciones.

Para este caso se analiza el algoritmo NSGAII con los problemas de prueba DTLZ(1,2,3,4 y 7).

4.3.5.1. Hypervolumen

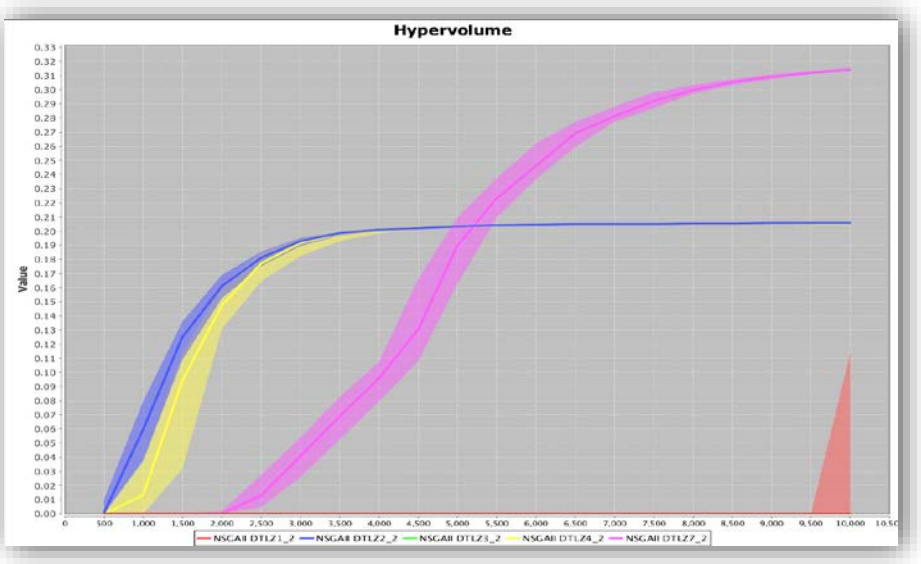


Fig.4.13 Hypervolumen para el algoritmo NSGAII con 2 objetivos para los diferentes problemas.

4.3.5.2. Distancia Generacional

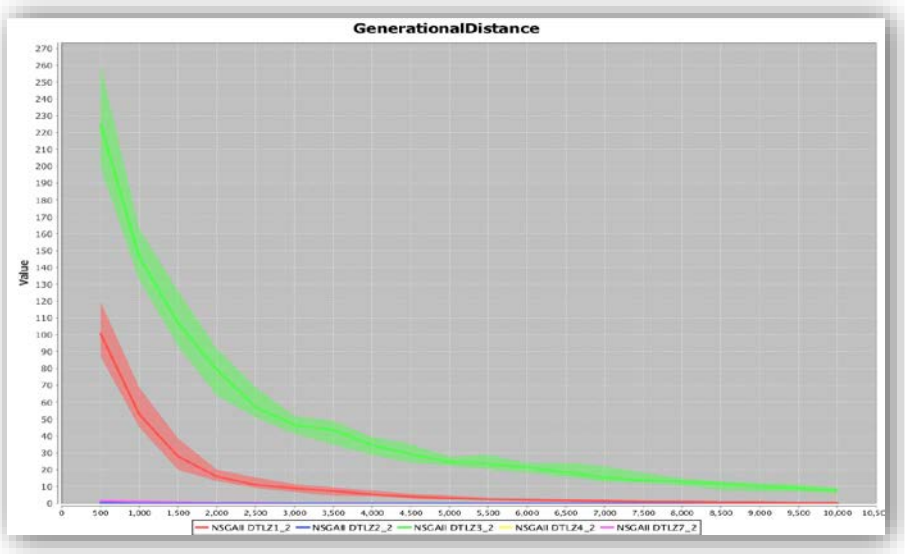


Fig. 4.14 Distancia Generacional para NSGAII con 2 objetivos para los diferentes problemas

4.3.5.3. Distancia Generacional Inversa

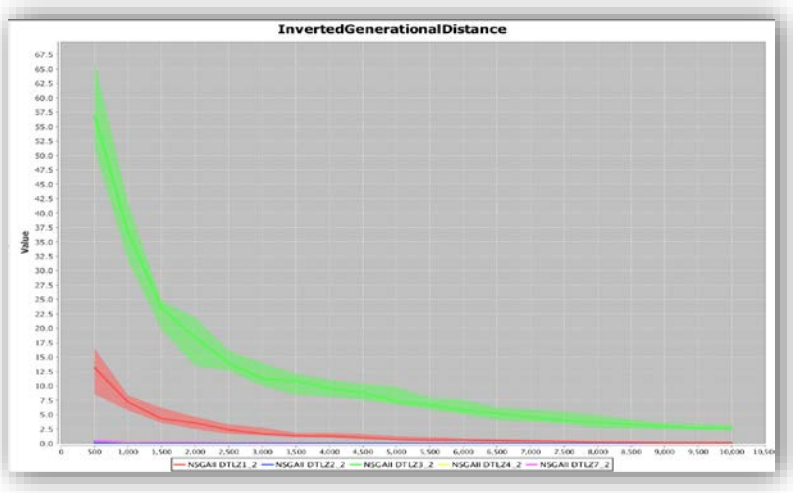


Fig. 4.15 Distancia Generacional Inversa para NSGAII con 2 objetivos para los diferentes problemas

4.3.5.4. Cobertura

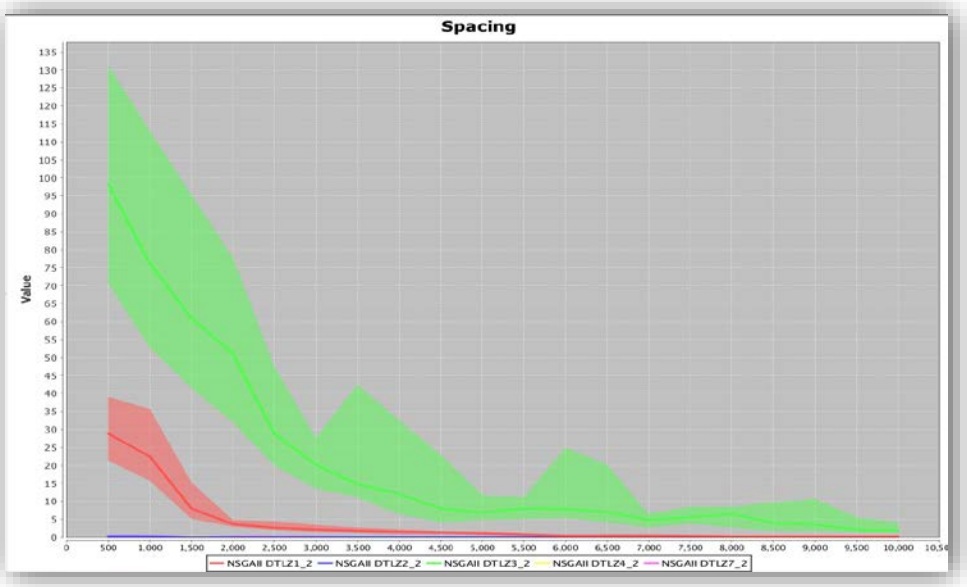


Fig. 4.16 Cobertura para el algoritmo NSGAII con 2 objetivos para los diferentes problemas

Para ver los cambios del conjunto de aproximación para problema ver Anexos V.

Después de analizar los datos que se presentan, se puede concluir que los tres primeros problemas de prueba DTLZ tuvieron una mejor convergencia usando el algoritmo NSGAII, donde a pesar que el algoritmo no pudo encontrar el verdadero frente de Pareto su aproximación fue la que mejor se ajusta.

4.3.5.5. Resumen

Algunos algoritmos no pueden encontrar todas las soluciones óptimas de Pareto en problemas de optimización multiobjetivo que no convergen, por lo que es necesario tener cierto conocimiento específico del problema para así poder agregar a la información dada los pesos y valores deseados para cada objetivo requerido.

Luego de ser probado en diferentes problemas complejos y siendo sus resultados comparados con otros Algoritmos Evolutivos Multiobjetivo de tipo elitista (SPEA), se puede decir que el NSGA II, en la mayoría de los casos es capaz de obtener mejores soluciones de dispersión y una mejor convergencia cerca del *Frente Óptimo de Pareto*, en comparación con el otro algoritmo analizados.

La significancia estadística permite evaluar si, en este caso los valores de las muestras generadas por cada algoritmo evaluado, no tienen ninguna relación entre sí. Esto permite saber cuál es el algoritmo que mejor se ajusta a las necesidades. En el caso que sean estadísticamente significantes, indica que ambas muestras tienen un comportamiento similar y que su rendimiento fue el mismo.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- La noción de tener crear objetos sin vida que puedan ser totalmente autómatas y recrear el comportamiento humano o animal ha estado presente en el pensamiento del ser humano desde épocas remotas. En las cuales a pesar de no tener la tecnología necesaria para poder crearlos, se desarrollaron varias teorías y técnicas que más adelante fueron las que aportaron de gran manera en el desarrollo de tales tecnologías.
- Gracias a las investigaciones realizadas dentro del campo de la Computación Evolutiva para la optimización de problemas, se han desarrollado una variedad de herramientas que han permitido tanto a investigadores como ingenieros en diferentes áreas el poder mejorar procesos sin necesidad de realizar complicadas teorías matemáticas que también pueden resolver este tipo de problemas pero con un mayor gasto de recursos por parte de los investigadores.
- Las técnicas de solución de problemas multiobjetivo permiten a los investigadores el poder encontrar soluciones óptimas dentro de grandes espacios de búsqueda sin la necesidad de tener conocimiento específico del problema.
- La necesidad de resolver problemas con múltiples objetivos ha llevado al desarrollo de una variedad de algoritmos evolutivos capaces de resolverlos. Estos algoritmos se basan principalmente en los conceptos desarrollados por la Teoría de la Evolución de Darwin.

Recomendaciones

- Para comprender bien el área de la Computación Evolutiva se recomienda comenzar su estudio con los conceptos de biología, la genética y principalmente la teoría de la evolución de Darwin; de los cuales la Computación Evolutiva se basó para su desarrollo.
- Las técnicas de la Computación Evolutiva permiten el diseño y optimización de diferentes sistemas. Dichas técnicas utilizadas tienen un rendimiento mucho mejor que los modelos matemáticos tradicionales, en los cuales en caso de necesitar ciertos cambios dentro del ajuste de la función, o de números de parámetros, muchas veces es necesario cambiar el modelo completo. Por ellos se recomienda el uso de algoritmos evolutivos ya que son mucho más sencillos pues se adaptan con mucha mayor facilidad y generan un menor costo.

Bibliografía

Aguirre, H., & Tanaka, K. (2007). "Working Principles, Behavior, and Performance of MOEAs on MNK-Landscapes". *European Journal of Operational Research, Elsevier, vol. 181*, 1670-1690.

Aguirre, H., Oyama, A., & Tanaka, K. (2013). "Adaptive ε-Sampling and ε-Hood for Evolutionary Many-Objective Optimization". *Proc. 7th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2013), Lecture Notes in Computer Science (Springer), vol.7811*, 322-336.

American Association of Artificial Intelligence and Massachusetts of Technology Artificial Intelligence Lab. (29 de 04 de 2008). *Tech.* Recuperado el 18 de 03 de 2014, de History of AI: http://usatoday30.usatoday.com/tech/news/2001-06-20-ai-history.htm

Asimov, I. (2004). *Yo, Robot.* Granada, España: EDHASA.

Aula Virtual de Biología. (2002). Recuperado el 18 de 02 de 2014, de Proteinas: http://www.um.es/molecula/prot.htm

Baños Navarro, R. (2008). *Meta-Heurísticas híbridas para optimización mono-objetivo y multi-objetivo. Paralelización y aplicaciones.* Almería: Editorial Universidad de Almería.

Beltrán, M. (13 de 10 de 2010). *FTA ó Fault Tree Analysis.* Recuperado el 23 de 08 de 2014, de redindustria: http://redindustria.blogspot.com/2010/10/fta-o-fault-tree-analysis-y-ii.html

C.A. (1966). *Biología.* Mexico: VILLE.

Cabrero, C. e. (1995). Allele Frequency Distribution of Four PCR-Amplified Loci in the Spanish population. *Forensic Science Int. 71*(153), 71.

Celaya, I. T. (s.f.). *Instituto Tecnológico de Celaya.* Recuperado el 13 de 08 de 2014, de Introducción a la Optimización Multiobjetivo: http://www.iqcelaya.itc.mx/~vicente/NotasSeminario3.pdf

Chércoles, D. M. (16 de 02 de 2007). *Universidad de Oviedo.* Recuperado el 15 de 07 de 2014, de Historia de la Automática: http://isa.uniovi.es/~gojea/funding/documentos/historia%20automatica.pdf

Civil, P. (2008). *MÉTODOS PARA LA EVALUACIÓN CUALITATIVA DE FRECUENCIAS DE OCURRENCIA.* Recuperado el 23 de 08 de 2014, de GUÍA TÉCNICA: Métodos cualitativos para el análisis de riesgos : http://www.proteccioncivil.es/catalogo/carpeta02/carpeta22/guiatec/Metodos_cualitativos/cuali_33.htm

Coello Coello, C. A. (09 de 2002). *CINVESTAV-IPN.* Recuperado el 02 de 08 de 2014, de Introducción a la Optimización Evolutiva Multiobjetivo: http://neo.lcc.uma.es/pdf-charlas/MOEA.pdf

Coello Coello, C. A., & Sarker, R. (s.f.). Recuperado el 03 de 08 de 2014, de http://seit.unsw.adfa.edu.au/staff/sites/rsarker/ch-7-sarker.pdf

Coello, C. A. (09 de 2002). *Networking and Emeging Optimization.* Recuperado el 19 de 06 de 2013, de http://neo.lcc.uma.es/pdf-charlas/MOEA.pdf

Coello, C., Van Veldhuizen, D., & Lamont, G. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems.* Boston: Kluwer Academic Publishers.

Darwin, C. (1859). *El Origen de las Especies.*

de la Herrán Gascón, M. (1998). Computación Evolutiva. *Computación Evolutiva*. (I. J. Herrera, Ed.) Bilbao, España.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms.* Chichester: John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). "A Fast and Elitis Multiobjective Genetic Algorithm: NSGA-II". *IEEE Transactions on Evolutionary Computation, vol.6,no.2*, 182-197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002). Scalable multi-objective optimization test problems. *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on . I*, págs. 825-830. Honolulu, HI: IEEE.

Dr. Raisman, J., & Dr. Gonzalez, A. M. (2013). *La replicación del ADN.* Obtenido de HIPERTEXTOS DEL ÁREA DE LA BIOLOGÍA: http://www.biologia.edu.ar/adn/adntema1.htm

Dr. Tiessen Favier, A. (2004). *El alfabeto de los genes*. Recuperado el 23 de 08 de 2014, de Fundacion Ciencia Activa: http://www.ciencia-activa.org/DivergenAlfabetoGenes.htm

Dr. Túpac Valdivia, Y. J. (23 de 11 de 2011). *Universidad Católica San Pablo.* Recuperado el 02 de 08 de 2014, de http://www.ucsp.edu.pe/~ytupac/speaks/2011/UNJBG_CIIS_MOGA.pdf

Dr. Vega Rodríguez, M. Á. (s.f.). *Universidad de Extremadura.* Recuperado el 02 de 08 de 2014, de Grupo de Investigación ARCO: http://jornadascdisc.ceta-ciemat.es/programa/Jornadas-CDISC-MiguelAVega.pdf

Farmani, R., Walters, G., & Savic, D. (s.f.). "Evolutionary multi-objective optimization of the design and operation of water distribution network: total cost vs. reliability vs. water quality". *Journal of Hydroinformatics, Vol. 08, No. 3*, 165-179.

Gascón, M. d. (1998). *Biblioteca CF+S.* Recuperado el 19 de 06 de 2013, de http://habitat.aq.upm.es/boletin/n21/amher1.html

Genésis. (s.f.).

Griffiths, A. J. F, Gelbart, W. M., Miller, J. H., . . . R. C. (2000). *Genética Moderna.* McGraw-Hill Interamericana.

Grosan, C. (s.f.). *Performance metrics for multiobjective optimization evolutionary.* Recuperado el 23 de 07 de 2014, de http://www.cs.ubbcluj.ro/~cgrosan/grosanCAIM.pdf

Igl, G. (08 de Septiembre de 2008). *Desde Mendel hata las moléculas.* Recuperado el 19 de Febrero de 2014, de http://genmolecular.wordpress.com/principios-de-biologia-molecular/

Irizar Mesa, M., López Romeo, I., & Solana Hernández, I. (1998). Algoritmos Genéticos: Evolución, Genética y Computación. *La Revista Cubana de Computación. GIGA Número 4*, 26–33.

Jacob, C. (2001). *"Illustrating Evolutionary Computation with Mathematica".* San Francisco: Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann.

James, P. (1996). *Gestión de la Calidad Total.* Prentice Hall. Obtenido de http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/601a700/ntp_679.pdf

Lacadena, J. R. (1999). *Genética General: conceptos fundamentales.* Madrid: Síntesis.

Línea, P. e. (2011). *Profesor en Línea.* Recuperado el 18 de Febrero de 2014, de Profesor en Línea: http://www.profesorenlinea.cl

McCorduck , P. (2004). *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence.* Natick, Massachusetts, Estados Unidos de Américaéri.

McCorduck, P. (2004). *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence.* Natick: A. K. Peters.

Muñoz de la Peña Castrillo, F. (Septiembre de 2002). *Aula Tecnológica Siglo XXI.* Recuperado el 18 de 02 de 2014, de Nutrición Navega y Aprende: http://www.aula21.net/nutricion/proteinas.htm

Peñuela Meneses, C. A., & Granada Echeverria, M. (08 de 2007). Optimización Mutiobjetivo Utilizando un Algoritmo Genético y un Operador Elitista basado en un Ordenamiento no dominado (NSGA II). *Scientia et Technica*. Pereira: Universidad Tecnológica de Pereira.

Radziukynienė, I., & Žilinskas, A. (2008). *Evolutionary Methods for Multi-Objective Portfolio Optimization.* Londres.

RAMIREZ, A. (s.f.). *Pontificia Universidad Javeriana de Bogotá.* Recuperado el 19 de 06 de 2013, de http://javeriana.edu.co/fear/ecologia/documents/ALBERTORAMIREZMETODOLOGIADELAINVESTIGACIONCIENTIFICA.pdf

Reehuis, E., Kruisselbrink, J., Deutz, A., Back, T., & Emmerich, M. (2011). "Multiobjective Optimization of Water Distribution Networks Using SMS-EMOA". *Proc. Intl Conf. Evolutionary and Deterministic Methods for Design, Optimization, and Control with Application to Industrial Problems (EUROGEN 2011)*, 269-279.

SCPM Isaac Newton. (01 de 2006). Recuperado el 15 de 07 de 2014, de Números Revista Didáctica de las Matemáticas: http://www.sinewton.org/numeros/numeros/43-44/Articulo33.pdf

Stansfield, W. D. (1992). *Teoría y problemas de Genética.* Bogotá: McGraw-Hill Latinoamericana.

Tecnología, A. I. (11 de 10 de 2013). *La organización del ADN influye en la evolución de las especies.* Recuperado el 23 de 08 de 2014, de Agencia Iberoamericana para la Difusión de la Ciencia y la Tecnología: http://www.dicyt.com/noticias/la-organizacion-del-adn-influye-en-la-evolucion-de-las-especies

Universidad de la República. (2007). *Facultad de Ingeniería.* Recuperado el 03 de 08 de 2014, de Universidad de la República - Uruguay: http://www.fing.edu.uy/inco/cursos/geneticos/ae/2007/Clases/clase12.pdf

Universidad del País Vasco. (09 de 11 de 2004). *Universidad del País Vasco.* Recuperado el 07 de 07 de 2014, de Campus de Gipuzkoa: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf

Vela, F. J. (s.f.). *Repositorio Institucional de la Universidad de Málaga.* Recuperado el 19 de 06 de 2013, de http://riuma.uma.es/xmlui/bitstream/handle/10630/4066/investigacion6.pdf?sequence=1

Zitzler, E. (11 de 11 de 1999). Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Zurich, Suiza. Recuperado el 23 de 07 de 2014, de http://www.tik.ee.ethz.ch/~sop/publicationListFiles/zitz1999a.pdf

Ziztler, E., Laumans, M., & Bleuler, S. (02 de 2002). A Tutorial onEvolutionary Multiobjective Optimization. *A Tutorial onEvolutionary Multiobjective Optimization*. Zurich, Suiza.

GLOSARIO DE TÉRMINOS

Ácido nucleico: Biomoléculas formadas por macropolímeros de nucleótidos, o polinucleótidos. Está presente en todas las células y constituye la base material de la herencia que se transmite de una a otra generación. Existen dos tipos, el ácido desoxirribonucleico (ADN) y el ácido ribonucleico (ARN).

Adaptación: Proceso por el que un animal o un vegetal se acomoda al medio ambiente y a sus cambios

Algoritmo: Conjunto de Instrucciones que especifican la secuencia de operaciones a realizar, en orden, para resolver un sistema específico o clase de problema.

Algoritmo evolutivo: Son métodos de optimización y búsqueda de soluciones basados en los postulados de la evolución biológica.

ADN: Abreviatura de ácido desoxirribonucleico (en inglés deoxyribonucleic acid o DNA). Es la molécula que contiene y transmite la información genética de los organismos excepto en algunos tipos de virus (retrovirus). Está formada por dos cadenas complementarias de nucleótidos que se enrollan entre sí formando una doble hélice que se mantiene unida por enlaces de hidrógeno entre bases complementarias. Los cuatro nucleótidos que forman el ADN contienen las bases adenina (A), guanina (G), citosina (C) y timina (T). Dado que en el ADN la adenina se empareja sólo con la timina y la citosina sólo con la guanina, cada cadena del ADN puede ser empleada como molde para fabricar su complementaria.

AMFE: **Análisis Modal de Fallo y Efecto.** Consiste en sistematizar el estudio de un proceso/producto, identificar los puntos de fallo potenciales, y elaborar planes de acción para combatir los riesgos.

Árboles de fallo: Técnica que consiste en un proceso deductivo basado en las leyes del Algebra de Boole, que permite determinar la expresión de sucesos

complejos estudiados en función de los fallos básicos de los elementos que intervienen en él.

ARN: Ácido nucleico formado por nucleótidos en los que el azúcar es ribosa, y las bases nitrogenadas son adenina, uracilo, citosina y guanina. Actúa como intermediario y complemento de las instrucciones genéticas codificadas en el ADN. Existen varios tipos diferentes de ARN, relacionados con la síntesis de proteínas. Así, existe ARN mensajero (ARNm), ARN ribosómico (ARNr), ARN de transferencia (ARNt) y un ARN heterogéneo nuclear (ARN Hn). El ARN es normalmente el producto de la transcripción de un molde de ADN, aunque en los retrovirus el ARN actúa de plantilla y el ADN de copia.

Biología: Ciencia que trata del estudio de los seres vivos y de los fenómenos vitales en todos sus aspectos

Biología Molecular: Parte de la biología que trata de los fenómenos biológicos a nivel molecular. En sentido restringido comprende la interpretación de dichos fenómenos sobre la base de la participación de las proteínas y ácidos nucleicos.

Computación evolutiva: Conjunto de técnicas que basándose en la simulación de los procesos naturales y la genética se utilizan para resolver problemas complejos de búsqueda y aprendizaje.

Convergencia: Cuando una sucesión numérica posee un límite finito se denomina convergente, de lo contrario, si es infinito recibe la denominación de divergente

Cromosoma: Componente de las células, de estructura filamentosa, portador de los factores de la herencia o genes. Se hallan en número constante, que en la especie humana, es de 22 pares más dos cromosomas sexuales, en total 46 cromosomas. Los cromosomas son muy visibles en el núcleo celular durante la mitosis.

Distancia Euclidiana: La distancia euclidiana entre los puntos p y q es la longitud del segmento de línea que los conecta.

Enzima: Sustancias que en cantidades mínimas produce cambios químicos, sin intervenir en ella misma en la reacción. Catalítico producido por organismos vivos.

Evolución: Cambios primero molecular, después celular, y por último de organismos, a lo largo de la historia como resultado de mutaciones en el ADN, de su reproducción y de procesos de selección.

Fenotipo: Conjunto de características observables de un organismo o grupo, fruto de la interacción entre su genotipo y el ambiente en que éste se expresa.

Gen: Unidad elemental del cromosoma, constituida por ADN, que transporta el mensaje hereditario.

Genética: Ciencia que trata de la reproducción, herencia, variación y el conjunto de fenómenos y problemas relativos a la descendencia.

Genoma: Complemento cromosómico básico que contiene toda la información genética del individuo.

Genotipo: Conjunto de los alelos de un individuo en uno, varios o todos sus loci**.**

Herencia: Proceso por el cual determinados rasgos o características se transmiten de padres a hijos. Implica la separación y recombinación de genes durante la meiosis y las posibles influencias posteriores sobre el material genético durante la embriogénesis.

Hiperplano: En un espacio vectorial de dimensión n en relación a un origen fijo, el hiperplano es un conjunto de puntos cuyas n coordenadas escalares verifican una relación de primer grado.

Inteligencia artificial: Interesante rama de la informática que se encarga del desarrollo de programas que tienen como base el razonamiento análogo del ser humano y aprovecharlos con fines tecnológicos.

Kruskal-Wallis (prueba): es un método no paramétrico para probar si un grupo de datos proviene de la misma población. Intuitivamente, es idéntico al ANOVA con los datos reemplazados por categorías. Es una extensión de la prueba de la U de Mann-Whitney para 3 o más grupos.

Lógica silogística: lógica basada en los trabajos del filósofo griego Aristóteles. Un silogismo es, según la definición de Aristóteles, "un discurso (logos) en el cual, establecidas ciertas cosas, resulta necesariamente de ellas, por ser lo que son, otra cosa diferente"

Man-Whitney (Prueba): Es una prueba no paramétrica aplicada a dos muestras independientes. Es, de hecho, la versión no paramétrica de la habitual prueba t de Student.

Maximizar/minimizar: (función) Encontrar el valor máximo (o mínimo) de la función objetivo dentro de los bordes de la zona factible delimitada por las restricciones del problema.

Mutación: Alteración en los cromosomas, bien sea en número o bien porque éstos carecen de ciertas partes.

Mutagénesis: Una sustancia o agente físico que causa mutaciones, es decir, que altera de forma permanente el ADN de las células.

Población: Agrupación de ejemplares de una cierta especie que comparten un hábitat

Politopo: (geometría) la generalización a cualquier dimensión de un polígono bidimensional, y un poliedro tridimensional.

Programación lógica: Consiste en la aplicación del corpus de conocimiento sobre lógica para el diseño de lenguajes de programación; no debe confundirse con la

disciplina de la lógica computacional. La programación lógica gira en torno al concepto de predicado, o relación entre elementos.

Proteína: Molécula orgánica rica en nitrógeno, compuesta esencialmente por aminoácidos. Todas las membranas celulares están formadas por proteínas, por lo que constituyen un compuesto fundamental.

Razonamiento lógico-formal: Razonamiento descontextualizado, forma de razonar guiada por reglas donde el contenido no importa a la hora de decidir si una conclusión es correcta.

Redes neuronales: **S**on un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales.

Reproducción sexual: Proceso de crear un nuevo organismo descendiente a partir de la combinación de material genético de dos organismos de una misma especie; el cual se produce en organismos eucariotas.

Selección natural: Proceso a través del cual, los organismos mejor adaptados desplazan a los menos adaptados mediante la acumulación lenta de cambios genéticos favorables en la población a lo largo de las generaciones.

Significación Estadística: En estadística, un resultado se denomina estadísticamente significativo cuando no es probable que haya sido debido al azar. Una "diferencia estadísticamente significativa" solamente significa que hay evidencias estadísticas de que hay una diferencia entre las variables estudiadas. No significa que la diferencia sea grande, importante, o significativa en el sentido estricto de la palabra, sólo indica que hay diferencias.

Sistemas expertos: Sistemas que incorporan en la base de conocimiento del sistema el conocimiento de un experto e intentan simular el razonamiento humano por medio de un conjunto de programas de computación.

Anexos

Anexo I. Manual de Usuario MOEAFramework

Descarga directa: https://sourceforge.net/projects/moeaframework/files/MOEAFramework-2.1/MOEAFramework-2.1-Manual.pdf/download

Anexo II. Código Fuente MOEA Framework

Descarga directa: https://sourceforge.net/projects/moeaframework/files/MOEAFramework-2.1/MOEAFramework-2.1-Source.tar.gz/download

Anexo III. Código NSGA II en java

Anexo IV. Código Análisis NSGA II vs. SPEA2

Anexo V. Conjunto de Aproximación para NSGAII para dos objetivos

Código NSGA II en java

El código del algoritmo NSGA II se encuentra en:

MOEAFramework-2.1/src/org/moeaframework/algorithm/ NSGAII.java

Código:

```
/* Copyright 2009-2014 David Hadka
 *
 * This file is part of the MOEA Framework.
 *
 * The MOEA Framework is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or (at your
 * option) any later version.
 *
 * The MOEA Framework is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
 * or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public
 * License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with the MOEA Framework.  If not, see <http://www.gnu.org/licenses/>.
 */
package org.moeaframework.algorithm;

import org.moeaframework.core.EpsilonBoxDominanceArchive;
import org.moeaframework.core.EpsilonBoxEvolutionaryAlgorithm;
import org.moeaframework.core.Initialization;
import org.moeaframework.core.NondominatedSortingPopulation;
import org.moeaframework.core.Population;
import org.moeaframework.core.Problem;
import org.moeaframework.core.Selection;
```

```
import org.moeaframework.core.Solution;
import org.moeaframework.core.Variation;

/**
 * Implementación del algoritmo NSGA-II, con la posibilidad de adjuntar de manera opcional a
                                    un
 * &epsilon;-archivo de dominancia.
 * <p>
 * Referencias:
 * <ol>
 *   <li>Deb, K. et al.  "A Fast Elitist Multi-Objective Genetic Algorithm:
 *       NSGA-II."  IEEE Transactions on Evolutionary Computation, 6:182-197,
 *       2000.
 *   <li>Kollat, J. B., and Reed, P. M.  "Comparison of Multi-Objective
 *       Evolutionary Algorithms for Long-Term Monitoring Design."  Advances in
 *       Water Resources, 29(6):792-807, 2006.
 * </ol>
 */
public class NSGAII extends AbstractEvolutionaryAlgorithm implements
                                    EpsilonBoxEvolutionaryAlgorithm {

                                    /**
                                     * Operador de selección.
                                     */
                                    private final Selection selection;

                                    /**
                                     * Operador de variación.
                                     */
                                    private final Variation variation;

                                    /**
                                     * Construye el algoritmo NSGA-II con los componentes
                                    especificados.
```

```
 *
 * @param problem el problema a resolver
 * @param population la población que será usada para almacenar las soluciones
 * @param archive el archivo usado para almacenar el resultado; puede ser {@code null}
 * @param selection el operador de selección
 * @param variation el operador de variación
 * @param initialization el que inicializa el método
*/
public NSGAII(Problem problem, NondominatedSortingPopulation population,
        EpsilonBoxDominanceArchive archive, Selection selection,
        Variation variation, Initialization initialization) {
 super(problem, population, archive, initialization);
 this.variation = variation;
 this.selection = selection;
}

@Override
public void iterate() {
 NondominatedSortingPopulation population = getPopulation();
 EpsilonBoxDominanceArchive archive = getArchive();
 Population offspring = new Population();
 int populationSize = population.size();

 while (offspring.size() < populationSize) {
        Solution[] parents = selection.select(variation.getArity(),
                        population);
        Solution[] children = variation.evolve(parents);

        offspring.addAll(children);
 }
```

```
        evaluateAll(offspring);

        if (archive != null) {
                archive.addAll(offspring);
        }

        population.addAll(offspring);
        population.truncate(populationSize);
    }

    @Override
    public EpsilonBoxDominanceArchive getArchive() {
        return (EpsilonBoxDominanceArchive)super.getArchive();
    }

    @Override
    public NondominatedSortingPopulation getPopulation() {
        return (NondominatedSortingPopulation)super.getPopulation();
    }

}
```

Código Análisis NSGA II vs. SPEA2

```
/*this class was developed by Claudia Subia 4th September 2014.

*

*/

import java.io.IOException;

import org.moeaframework.Analyzer; import org.moeaframework.Executor;


public class NSGAIIvsSPEA2 {


    /**

     * @param args the command line arguments

     */

    public static void main(String[] args) throws IOException{

/*---3 objetivos---*/

        System.out.format("/--------------------3 objetivos--------------/%n");

Analyzer analyzer_3ob = new Analyzer()

        .withProblem("DTLZ1 3")

        .includeAllMetrics()

        .showStatisticalSignificance();

        Executor executor_3 = new Executor()

                .withProblem("DTLZ1 3")

                .withMaxEvaluations(10000);

        analyzer_3ob.addAll("NSGAII",

                executor_3.withAlgorithm("NSGAII").runSeeds(50));
```

```
        analyzer_3ob.addAll("SPEA2",
                executor_3.withAlgorithm("SPEA2").runSeeds(50));
        analyzer_3ob.printAnalysis();



/*---4 objetivos---*/
        System.out.format("/--------------------4 objetivos--------------/%n");

Analyzer analyzer_4ob = new Analyzer()
        .withProblem("DTLZ1 4")
        .includeAllMetrics()
        .showStatisticalSignificance();
        Executor executor_4 = new Executor()
                .withProblem("DTLZ1 4")
                .withMaxEvaluations(10000);
        analyzer_4ob.addAll("NSGAII",
                executor_4.withAlgorithm("NSGAII").runSeeds(50));
        analyzer_4ob.addAll("SPEA2",
                executor_4.withAlgorithm("SPEA2").runSeeds(50));
        analyzer_4ob.printAnalysis();



/*---5 objetivos---*/
        System.out.format("/--------------------5 objetivos--------------/%n");
```

```
Analyzer analyzer_5ob = new Analyzer()
        .withProblem("DTLZ1 5")
        .includeAllMetrics()
        .showStatisticalSignificance();
        Executor executor_5 = new Executor()
                .withProblem("DTLZ1 5")
                .withMaxEvaluations(10000);
        analyzer_5ob.addAll("NSGAII",
                executor_5.withAlgorithm("NSGAII").runSeeds(50));
        analyzer_5ob.addAll("SPEA2",
                executor_5.withAlgorithm("SPEA2").runSeeds(50));
        analyzer_5ob.printAnalysis();
}


}
```

Conjunto de Aproximación para NSGAII para dos objetivos

DTLZ1

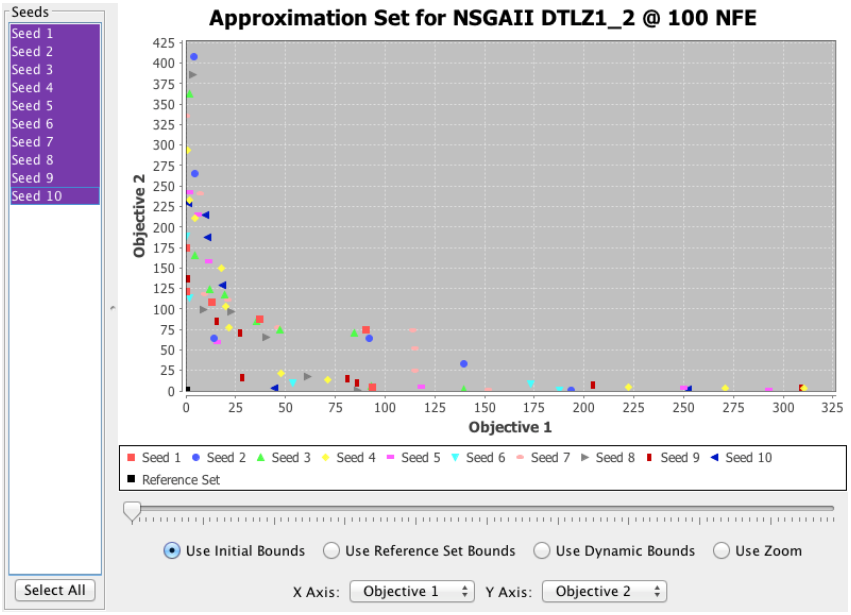


Fig. A 1 Conjunto de aproximación para NSGA II para el problema DTLZ1 con 100 evaluaciones

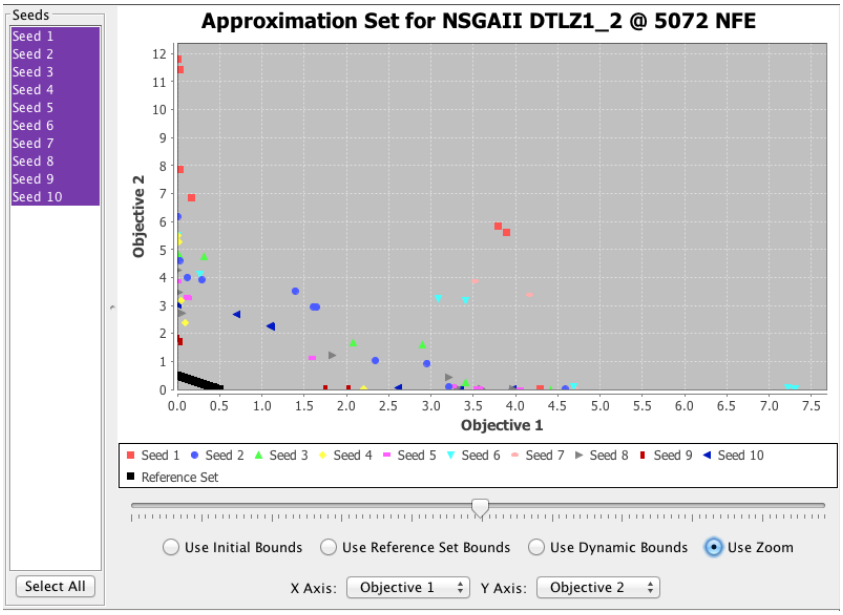


Fig. A,2 Conjunto de aproximación para NSGA II para el problema DTLZ1 con 5072 evaluaciones

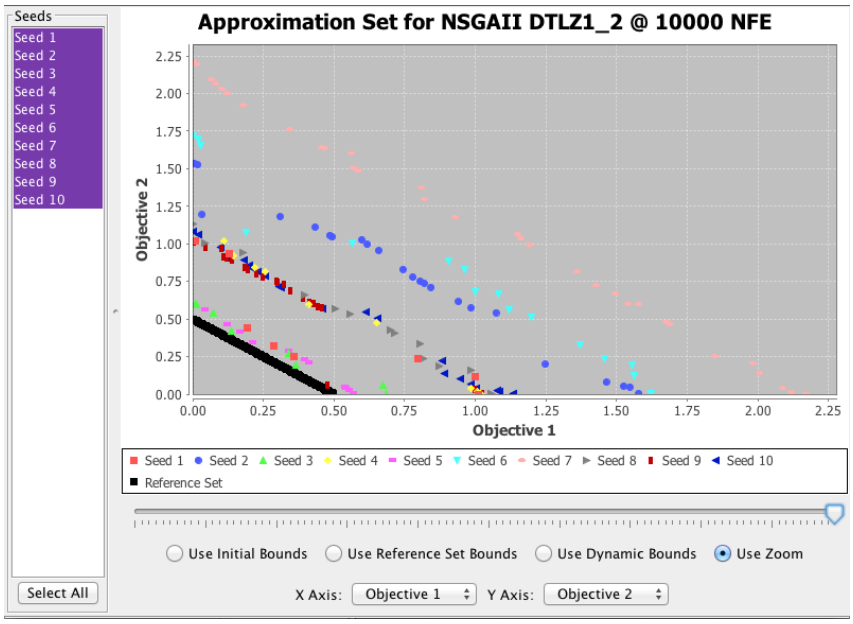


Fig. A.3 Conjunto de aproximación para NSGA II para el problema DTLZ1 con 10000 evaluaciones

DTLZ2

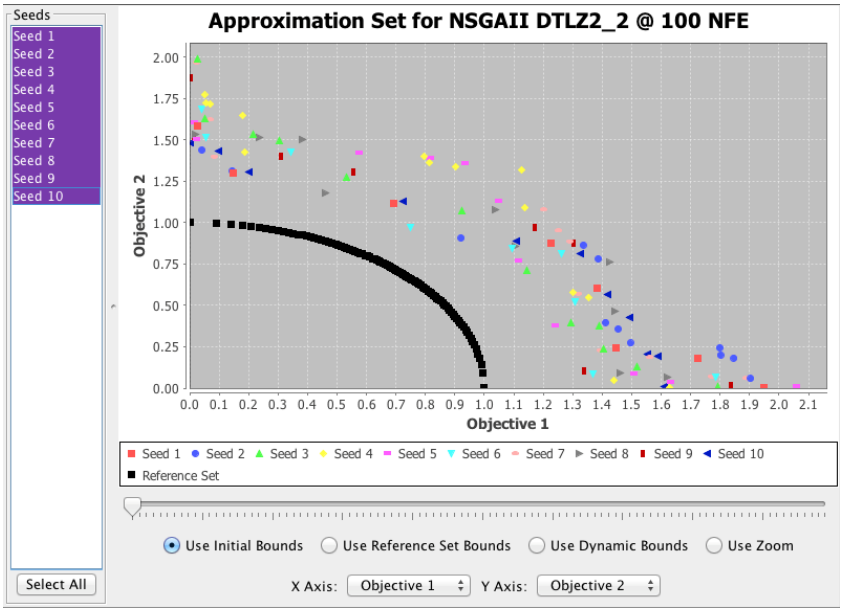


Fig. A.4 Conjunto de aproximación para NSGA II para el problema DTLZ2 con 100 evaluaciones

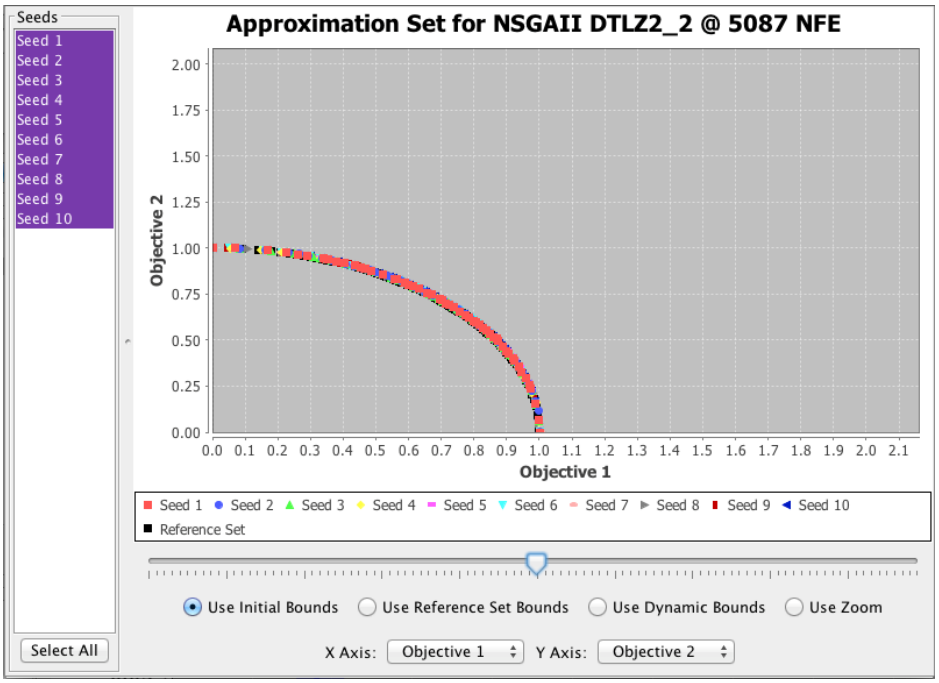


Fig. A.5 Conjunto de aproximación para NSGA II para el problema DTLZ2 con 5087 evaluaciones

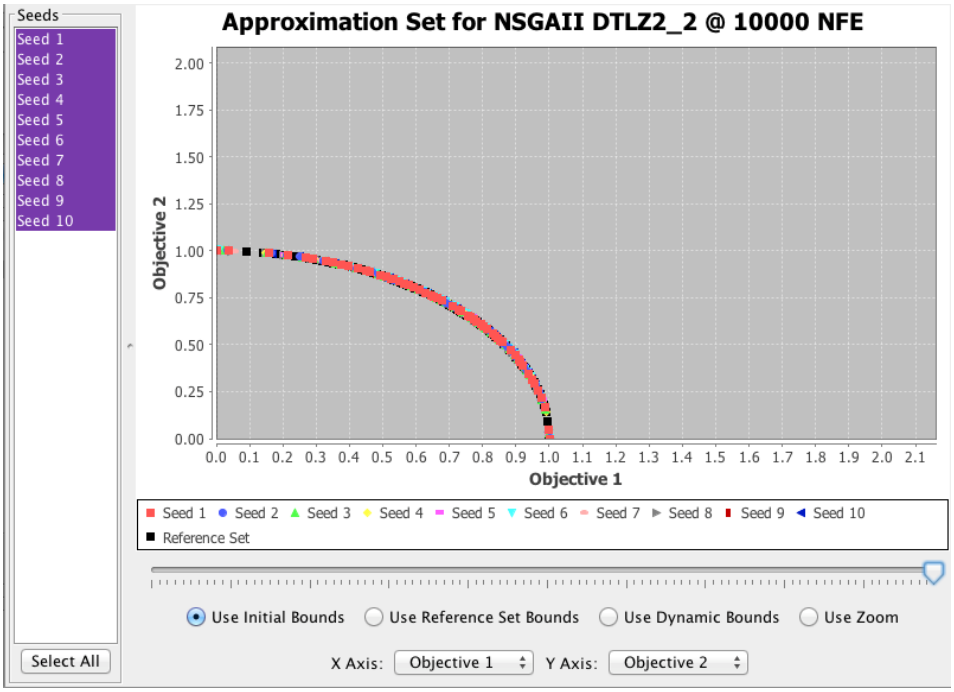


Fig. A.6 Conjunto de aproximación para NSGA II para el problema DTLZ2 con 10000 evaluaciones

DTLZ3

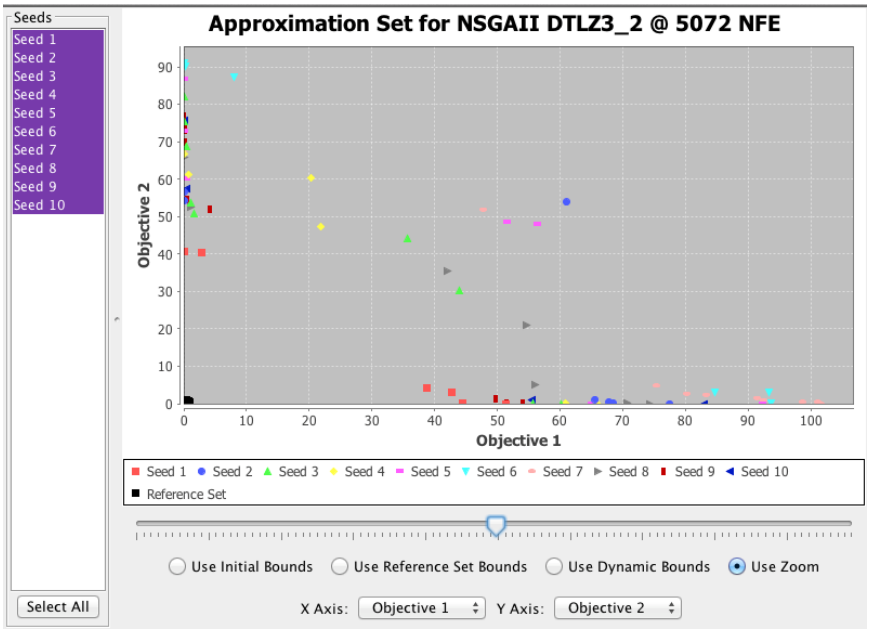


Fig. A.7 Conjunto de aproximación para NSGA II para el problema DTLZ3 con 100 evaluaciones

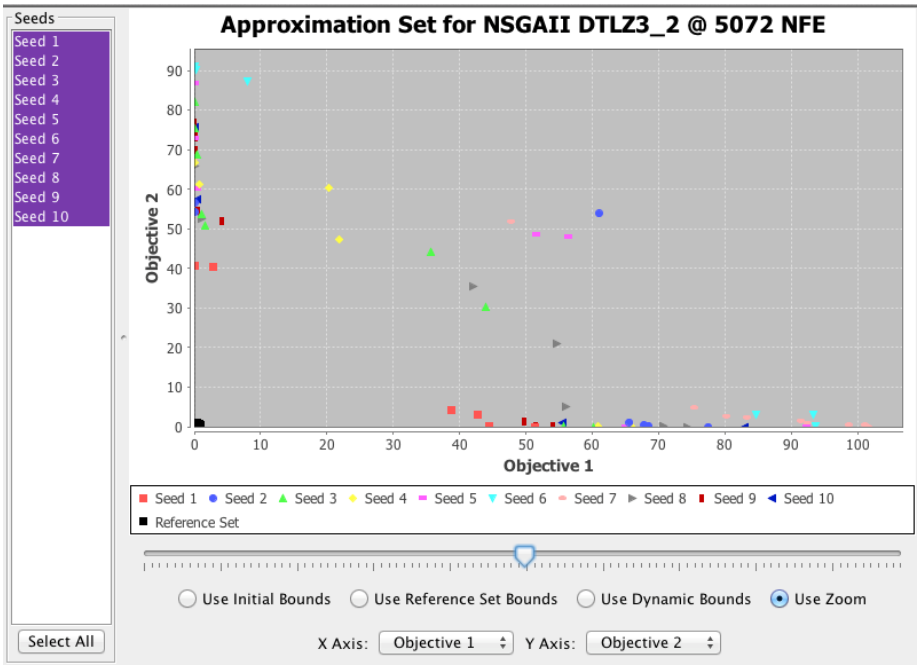


Fig. A.8 Conjunto de aproximación para NSGA II para el problema DTLZ3 con 5072 evaluaciones

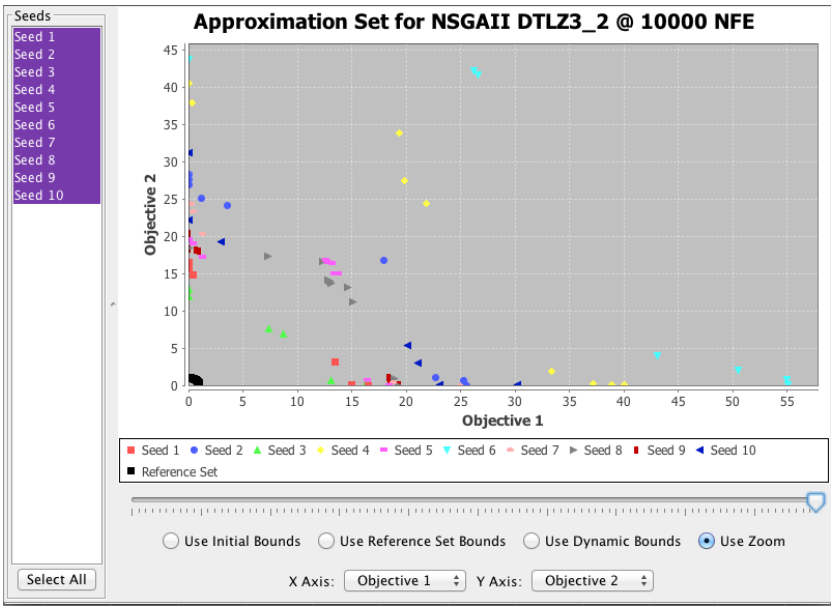


Fig. A.9 Conjunto de aproximación para NSGA II para el problema DTLZ3 con 10000 evaluaciones

DTLZ4

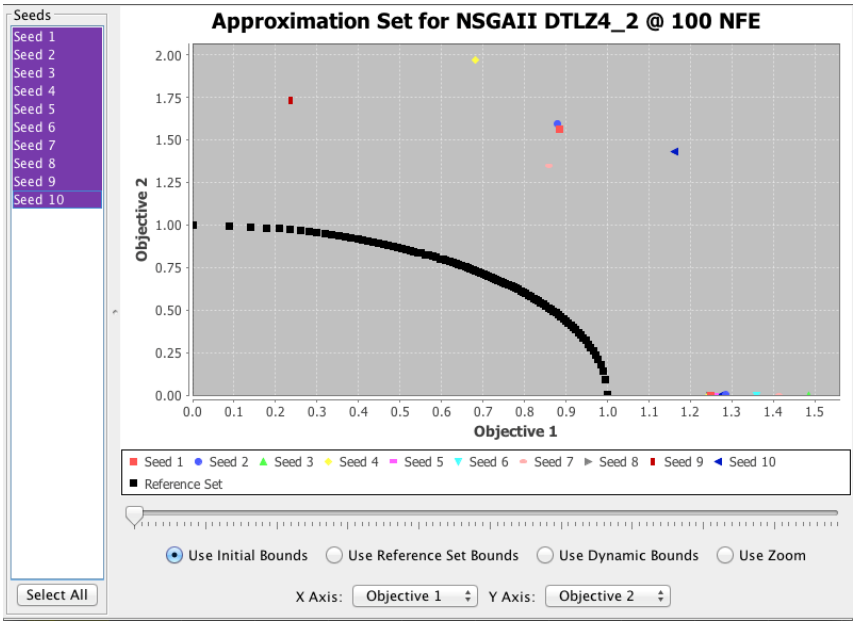


Fig. A.10 Conjunto de aproximación para NSGA II para el problema DTLZ4 con 100 evaluaciones

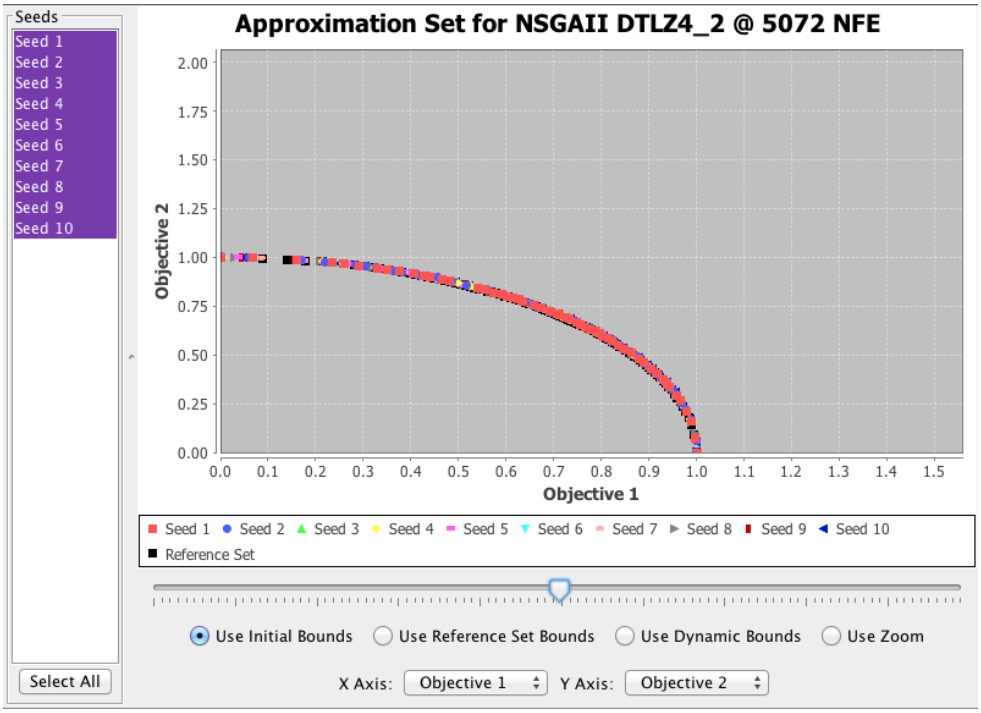


Fig. A.11 Conjunto de aproximación para NSGA II para el problema DTLZ4 con 5072 evaluaciones

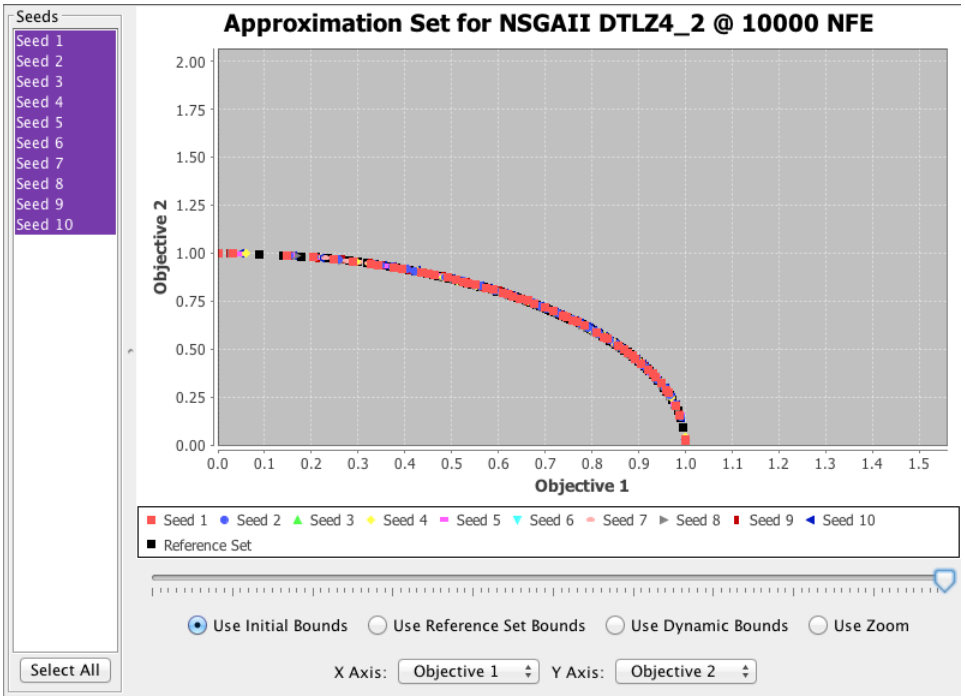


Fig. A.12 Conjunto de aproximación para NSGA II para el problema DTLZ4 con 10000 evaluaciones

DTLZ7

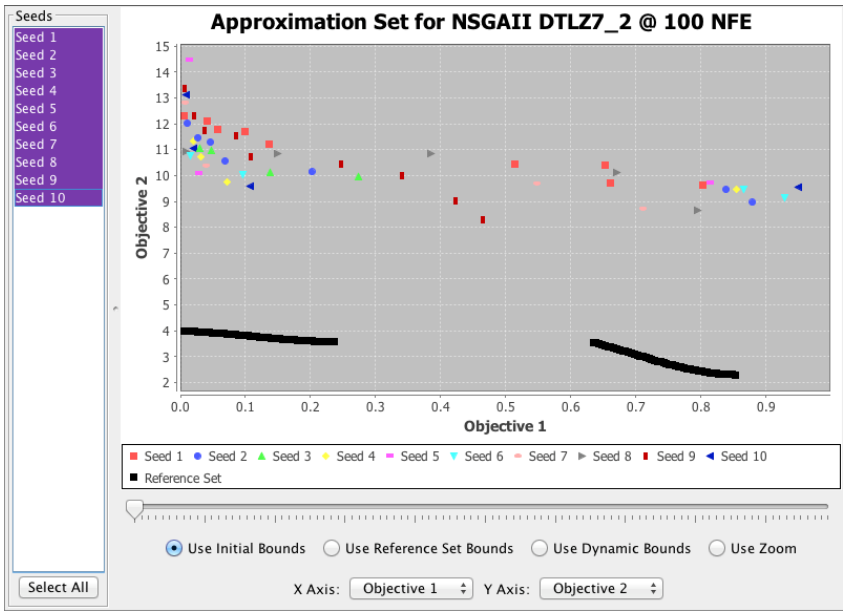


Fig. A.13 Conjunto de aproximación para NSGA II para el problema DTLZ7 con 100 evaluaciones

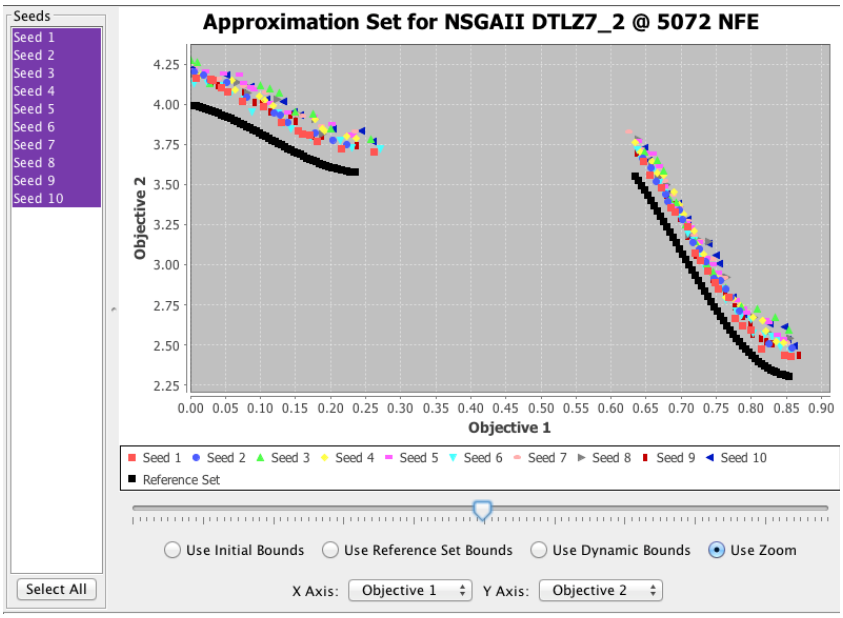


Fig. A.14 Conjunto de aproximación para NSGA II para el problema DTLZ7 con 5072 evaluaciones

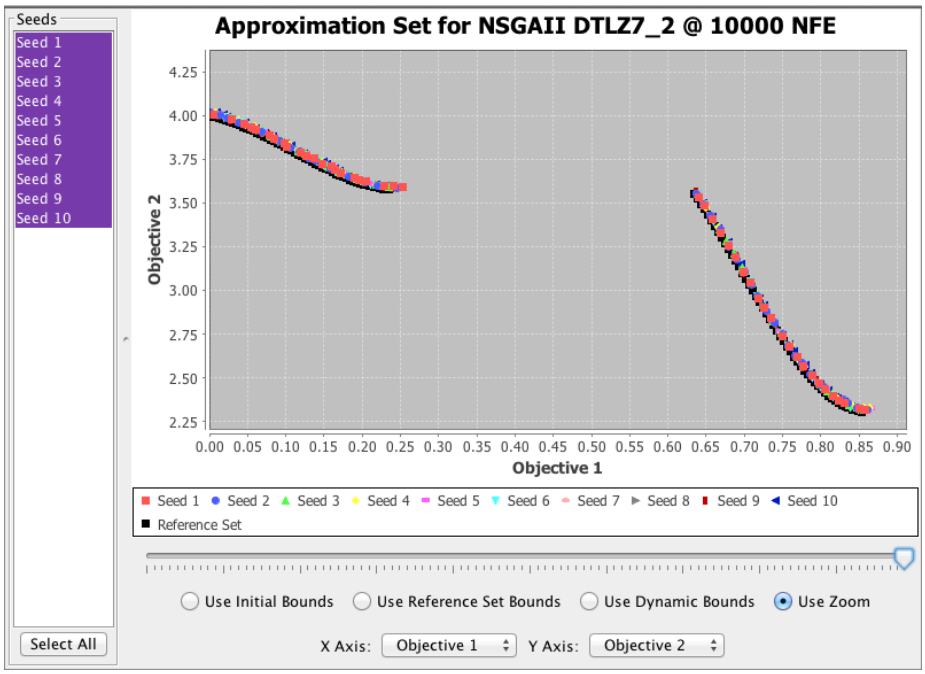


Fig. A.15 Conjunto de aproximación para NSGA II para el problema DTLZ7 con 10000 evaluaciones