



# UNIVERSIDAD VERACRUZANA

FACULTAD DE FÍSICA E INTELIGENCIA ARTIFICIAL

---

---

"UN ESTUDIO EMPÍRICO DE DOS ALGORITMOS  
EVOLUTIVOS PARA CLUSTERING MULTI-OBJETIVO"

T E S I S

Que para obtener el grado de  
Maestro en Inteligencia Artificial

PRESENTA:

**MARÍA GUADALUPE MARTÍNEZ PEÑALOZA**

DIRECTOR:

**DR. EFRÉN MEZURA MONTES**

Xalapa Enríquez, Veracruz

Agosto 2013

# INDICE

<b>CAPÍTULO I: INTRODUCCIÓN.....</b>	<b>1</b>
1.1 ANTECEDENTES.....	1
1.2 DEFINICIÓN DEL PROBLEMA.....	5
1.3 OBJETIVOS.....	6
1.3.1 <i>Objetivo general</i> .....	6
1.3.2 <i>Objetivos específicos</i> .....	6
1.4 HIPÓTESIS.....	6
1.5 JUSTIFICACIÓN.....	6
1.6 ORGANIZACIÓN DEL DOCUMENTO DE TESIS.....	7
<b>CAPÍTULO II: CÓMPUTO EVOLUTIVO .....</b>	<b>9</b>
2.1 ANTECEDENTES HISTÓRICOS.....	9
2.1.1 <i>Teoría de la evolución de Charles Darwin</i> .....	9
2.1.2 <i>Teoría del plasma germinal de August Weismann</i> .....	10
2.1.3 <i>Teoría genética de Gregor Mendel</i> .....	10
2.1.4 <i>Neo-Darwinismo</i> .....	10
2.2 FUNDAMENTOS BIOLÓGICOS.....	11
2.3 CONCEPTOS DEL CÓMPUTO EVOLUTIVO.....	11
2.3.1 <i>Algoritmos Evolutivos (AES)</i> .....	12
2.3.2 <i>Componentes de los algoritmos evolutivos</i> .....	13
2.3.3 <i>Principales paradigmas</i> .....	16
2.3.4 <i>Ventajas de los algoritmos evolutivos</i> .....	19
<b>CAPÍTULO III: OPTIMIZACIÓN .....</b>	<b>20</b>
3.1 CONCEPTOS DE OPTIMIZACIÓN.....	20
3.2 OPTIMIZACIÓN MULTI-OBJETIVO.....	22
3.3 ALGORITMOS PARA OPTIMIZACIÓN MULTI-OBJETIVO.....	26
3.3.1 <i>Técnicas tradicionales</i> .....	26
3.3.2 <i>Algoritmos evolutivos multi-objetivo</i> .....	27
3.4 MÉTRICAS DE DESEMPEÑO PARA AEMO.....	33
3.4.1 <i>Distancia Generacional (GD)</i> .....	34
3.4.2 <i>Cobertura de dos conjuntos (CS)</i> .....	35
3.4.3 <i>Hipervolumen (HV)</i> .....	36
<b>CAPÍTULO IV: CLUSTERING MULTI-OBJETIVO .....</b>	<b>37</b>
4.1 CLUSTERING MULTI-OBJETIVO.....	37
4.2 CLUSTERING MULTI-OBJETIVO CON DETERMINACIÓN AUTOMÁTICA DE K (MOCK).....	38
4.2.1 <i>Inicialización y representación</i> .....	39
4.2.2 <i>Operadores de variación</i> .....	40
4.2.3 <i>Funciones objetivo</i> .....	42
<b>CAPÍTULO V: METODOLOGÍA.....</b>	<b>44</b>
5.1 METODOLOGÍA GENERAL.....	44
5.1.1 <i>Implementación de MOCK usando NSGA-II y SPEA-2</i> .....	44
5.1.2 <i>Calibración de parámetros usando irace</i> .....	45

5.1.3 <i>Diseño experimental</i> .....	48
<b>CAPÍTULO VI: RESULTADOS</b> .....	<b>52</b>
6.1 RESULTADOS DEL PRIMER EXPERIMENTO.....	53
6.1.1 <i>Resultados comparados con la métrica Hipervolumen</i> .....	62
6.1.2 <i>Resultados comparados con la métrica Two Set Coverage</i> .....	66
6.2 RESULTADOS SEGUNDO EXPERIMENTO.....	70
<b>CAPÍTULO VII: CONCLUSIONES</b> .....	<b>74</b>
<b>REFERENCIAS</b> .....	<b>77</b>

## INDICE DE FIGURAS

<b>CAPÍTULO II: CÓMPUTO EVOLUTIVO</b> .....	09
FIGURA 2.1: ALGORITMO EVOLUTIVO GENERAL.....	13
FIGURA 2.2: ESQUEMA GENERAL DE UN ALGORITMO EVOLUTIVO.....	13
FIGURA 2.3: CROMOSOMA CON REPRESENTACIÓN BINARIA.....	14
FIGURA 2.4: REPRESENTACIÓN DEL GENOTIPO Y FENOTIPO.....	14
FIGURA 2.5: REPRESENTACIÓN BINARIA (A), REAL (B) Y ENTERA (C).....	14
FIGURA 2.6: EJEMPLO DE CRUZA DE DOS PUNTOS.....	16
FIGURA 2.7: EJEMPLO DE MUTACIÓN.....	16
FIGURA 2.8: ALGORITMO GENÉTICO SIMPLE.....	17
FIGURA 2.9: ALGORITMO DE ESTRATEGIAS EVOLUTIVAS.....	18
FIGURA 2.10: ALGORITMO BÁSICO DE PROGRAMACIÓN EVOLUTIVA.....	19
<b>CAPÍTULO III: OPTIMIZACIÓN</b> .....	20
FIGURA 3.1: ESPACIOS DE BÚSQUEDA EN UN PROBLEMA DE OPTIMIZACIÓN.....	20
FIGURA 3.2: DOMINANCIA DE PARETO.....	24
FIGURA 3.3: FRENTE DE PARETO PARA UN PROBLEMA DE DOS OBJETIVOS.....	25
FIGURA 3.4: CLASIFICACIÓN DE TÉCNICAS MOAE.....	27
FIGURA 3.5: JERARQUIZACIÓN DE PARETO EN UN PROBLEMA CON DOS OBJETIVOS.....	28
FIGURA 3.6: FUNCIONAMIENTO DEL MOGA.....	29
FIGURA 3.7: FUNCIONAMIENTO DEL NSGA-II.....	31
FIGURA 3.8: PROCEDIMIENTO DEL NSGA-II.....	31
FIGURA 3.9: MALLA ADAPTATIVA DE PAES CON 6 DIVISIONES EN UN PROBLEMA CON DOS OBJETIVOS.....	32
FIGURA 3.10: FUNCIONAMIENTO DE SPEA-2.....	33
FIGURA 3.11: CERCANÍA Y DIVERSIDAD AL FRENTE ÓPTIMO DE PARETO.....	34
FIGURA 3.12: COBERTURA DE DOS CONJUNTOS.....	35
FIGURA 3.13: (A) HIPERVOLUMEN QUE SE CENTRA EN UN PUNTO DE REFERENCIA DADO. (B) HIPERVOLUMEN QUE SE CENTRA EN EL ORIGEN.....	36
<b>CAPÍTULO IV: CLUSTERING MULTI-OBJETIVO</b> .....	37
FIGURA 4.1: (A) ELEMENTOS A AGRUPAR. (B) CONJUNTO DE SOLUCIONES DE PARETO CLUSTERING MULTI-OBJETIVO.....	38
FIGURA 4.2: ESTRUCTURA GENERAL DEL MOCK.....	39

FIGURA 4.3: (A) INICIALIZACIÓN DE INDIVIDUOS CON MST, PRIMER INDIVIDUO. (B) SUB-GRAFOS OBTENIDOS POR MOCK, TERCER INDIVIDUO. ....	40
FIGURA 4.4: EJEMPLO CRUZA UNIFORME EN MOCK. ....	41
FIGURA 4.5: EJEMPLO MUTACIÓN DE VECINOS MÁS CERCANO EN MOCK. ....	42
<b>CAPÍTULO V: METODOLOGÍA</b> .....	44
FIGURA 5.1: ESQUEMA DE FLUJO DE INFORMACIÓN DE IRACE.....	47
<b>CAPÍTULO VI: RESULTADOS</b> .....	52
FIGURA 6.1: FRENTE DE PARETO DE CADA ALGORITMO PARA DERMATOLOGY, CON PARÁMETROS ORIGINALES.....	54
FIGURA 6.2: FRENTE DE PARETO DE CADA ALGORITMO PARA IRIS, CON PARÁMETROS ORIGINALES. ....	54
FIGURA 6.3: FRENTE DE PARETO DE CADA ALGORITMO PARA WINE, CON PARÁMETROS ORIGINALES. ....	55
FIGURA 6.4: FRENTE DE PARETO DE CADA ALGORITMO PARA WISCONSIN, CON PARÁMETROS ORIGINALES. ....	55
FIGURA 6.5: FRENTE DE PARETO DE CADA ALGORITMO PARA YEAST, CON PARÁMETROS ORIGINALES. ....	56
FIGURA 6.6: FRENTE DE PARETO DE CADA ALGORITMO PARA DERMATOLOGY, CON PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	56
FIGURA 6.7: FRENTE DE PARETO DE CADA ALGORITMO PARA IRIS, CON PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE. ....	57
FIGURA 6.8: FRENTE DE PARETO DE CADA ALGORITMO PARA WINE, CON PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE. ....	57
FIGURA 6.9: FRENTE DE PARETO DE CADA ALGORITMO PARA WISCONSIN, CON PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	58
FIGURA 6.10: FRENTE DE PARETO DE CADA ALGORITMO PARA YEAST, CON PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE. ....	58
FIGURA 6.11: FRENTE DE PARETO DE CADA ALGORITMO PARA DERMATOLOGY, CON SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	59
FIGURA 6.12: FRENTE DE PARETO DE CADA ALGORITMO PARA IRIS, CON SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE. ....	60
FIGURA 6.13: FRENTE DE PARETO DE CADA ALGORITMO PARA WINE, CON SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE. ....	60
FIGURA 6.14: FRENTE DE PARETO DE CADA ALGORITMO PARA WISCONSIN, CON SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	61
FIGURA 6.15: FRENTE DE PARETO DE CADA ALGORITMO PARA YEAST, CON SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	61

## INDICE DE TABLAS

<b>CAPÍTULO V: METODOLOGÍA</b> .....	44
TABLA 5.1: ESPACIO DE LOS PARÁMETROS PARA LA CALIBRACIÓN CON IRACE.....	47
TABLA 5.2: RESUMEN DE LAS BASES DE DATOS USADAS PARA PRUEBAS. ....	48
<b>CAPÍTULO VI: RESULTADOS</b> .....	52
TABLA 6.1: PARÁMETROS DE INICIALIZACIÓN DE CADA ALGORITMO, EL SÍMBOLO - SIGNIFICA QUE PARA ESE ALGORITMO NO APLICA .....	52
TABLA 6.2: CONFIGURACIÓN DE PARÁMETROS OBTENIDOS POR IRACE . ....	52
TABLA 6.3: ESTADÍSTICAS DE LA MÉTRICA HIPERVOLUMEN PARA CADA BASE DE DATOS, USANDO CONFIGURACIÓN DE PARÁMETROS ORIGINAL.. ....	62
TABLA 6.4: ESTADÍSTICAS DE LA MÉTRICA HIPERVOLUMEN PARA CADA BASE DE DATOS, USANDO PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	63
TABLA 6.5: ESTADÍSTICAS DE LA MÉTRICA HIPERVOLUMEN PARA CADA BASE DE DATOS, USANDO SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	63
TABLA 6.6: U MANN DE WHITNEY CON MUESTRA DE LA MÉTRICA HIPERVOLUMEN PARA CADA UNA DE LAS BASES DE DATOS, USANDO CONFIGURACIÓN DE PARÁMETROS ORIGINAL.....	64
TABLA 6.7: U MANN DE WHITNEY CON MUESTRA DE LA MÉTRICA HIPERVOLUMEN PARA CADA UNA DE LAS BASES DE DATOS, USANDO PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	65
TABLA 6.8: U MANN DE WHITNEY CON MUESTRA DE LA MÉTRICA HIPERVOLUMEN PARA CADA UNA DE LAS BASES DE DATOS, USANDO SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	65
TABLA 6.9: U MANN DE WHITNEY CON MUESTRA DE LA MÉTRICA HIPERVOLUMEN COMPARANDO LOS RESULTADOS DE CADA CONFIGURACIÓN DE PARÁMETROS. ....	66
TABLA 6.10: ESTADÍSTICAS DE LA MÉTRICA TWO SET COVERAGE PARA CADA BASE DE DATOS, USANDO CONFIGURACIÓN DE PARÁMETROS ORIGINAL. ....	67
TABLA 6.11: ESTADÍSTICAS DE LA MÉTRICA TWO SET COVERAGE PARA CADA BASE DE DATOS, USANDO PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	68
TABLA 6.12: ESTADÍSTICAS DE LA MÉTRICA TWO SET COVERAGE PARA CADA BASE DE DATOS, USANDO SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	69
TABLA 6.13: PROMEDIO DEL VALOR DE F-MEASURE PARA DIEZ EJECUCIONES DE CADA ALGORITMO, USANDO CONFIGURACIÓN DE PARÁMETROS ORIGINAL. ....	71
TABLA 6.14: PROMEDIO DEL VALOR DE F-MEASURE PARA DIEZ EJECUCIONES DE CADA ALGORITMO, USANDO PRIMERA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	71
TABLA 6.15: PROMEDIO DEL VALOR DE F-MEASURE PARA DIEZ EJECUCIONES DE CADA ALGORITMO, USANDO SEGUNDA CONFIGURACIÓN DE PARÁMETROS GENERADA POR IRACE.....	71
TABLA 6.16: COMPARACIÓN DE F-MEASURE PARA LAS TRES CONFIGURACIONES DE PARÁMETROS.....	72

# Capítulo I: Introducción

## 1.1 Antecedentes.

El agrupamiento de datos, también conocido como clustering, es una técnica de aprendizaje automático cuyo objetivo consiste en determinar un conjunto finito de grupos (clusters) para describir un conjunto de datos de acuerdo a las similitudes entre ellos [1] [2].

Existen distintas técnicas de agrupamiento, en general estas pueden ser divididas en tres tipos principales [3]: los que se superponen (llamado clustering no exclusivo), particional y jerárquico. Independientemente del tipo de algoritmo, el objetivo primordial del agrupamiento es maximizar la homogeneidad de cada grupo y la heterogeneidad entre diferentes grupos [3] [4]. En otras palabras, esto quiere decir que los objetos que pertenecen a un mismo grupo deben ser más similares entre sí que los que pertenecen a otros grupos. Ahora bien, el clustering puede ser considerado como un problema desafiante en el área de aprendizaje automático, debido a que es un método de aprendizaje no supervisado, lo cual implica que sus características estructurales no son conocidas, la distribución espacial de los datos, densidad, forma, orientación de grupos, son desconocidos [5], es decir, no hay conocimiento a priori.

Desde una perspectiva de optimización, la agrupación puede considerarse formalmente como un tipo particular de problema NP-hard [6]. Esto ha estimulado la búsqueda de algoritmos de aproximación eficientes, incluyendo no sólo el uso de heurísticas ad hoc para problemas con clases o instancias determinadas, sino también el uso de meta-heurísticas de propósito general (por ejemplo, véase [7]). En particular, los algoritmos evolutivos son meta-heurísticas que se cree que son ampliamente eficaces cuando se abordan problemas NP-hard, pudiendo ofrecer soluciones casi óptimas para este tipo de problemas en un tiempo razonable. Bajo este supuesto, se han propuesto distintos tipos de algoritmos evolutivos para la solución de problemas de agrupamiento. Gran parte de estos algoritmos se basan en la optimización de una función objetivo (es decir, la llamada función de aptitud) la cual guía la búsqueda evolutiva.

Como ya fue mencionado anteriormente, una partición de un conjunto de datos es una colección de  $k$  grupos. El número de grupos,  $k$ , en la mayoría de los algoritmos clásicos de agrupamiento, por lo general debe ser proporcionado de antemano por el usuario. Sin embargo, en algunos casos también se puede estimar de forma automática por el algoritmo de agrupamiento. Dentro de los algoritmos en los que el número de grupos se conoce o se ha establecido con anterioridad, se pueden mencionar como ejemplo Bandyopadhyay y Maulik [8]; Estivill-Castro y Murray [9]; Fränti et al [10]; Kivijärvi et al [11];. Krishna y

Murty [12]; Krovi [13]; Bezdek et al [14]; Kuncheva y Bezdek [15];. Lu et al [16] [17]; Lucasius et al [18]; Maulik y Bandyopadhyay [19]; Merz y Zell [20]; Murthy y Chowdhury [21]. Resulta intuitivo pensar que los algoritmos que asumen un número fijo de grupos ( $k$ ) pueden ser adecuados para aplicaciones en las que existe información sobre el número de grupos. Por ejemplo, en el caso de aplicaciones que sugieren un valor de  $K$  razonable o bien un intervalo posible de los valores que debe tomar  $K$ . Tener esta información conlleva a que algoritmos como los antes referenciados puedan ser aplicados potencialmente para hacer frente al problema de agrupamiento correspondiente. Alternativamente también se pueden considerar algunos algoritmos de agrupamiento convencionales con un  $K$  fijo, por ejemplo el algoritmo  $K$ -means [22] [3]. Sin embargo estos algoritmos son muy sensibles a la inicialización y pueden dar soluciones sub-óptimas. Este problema puede darse con mayor frecuencia en bases de datos complejas. Un enfoque que se ha utilizado para atacar dicho problema consiste en ejecutar el algoritmo varias veces inicializando los centroides en distintas posiciones del espacio de búsqueda, sin embargo hay que tomar en cuenta que la única forma de garantizar la mejor solución de agrupamiento para un valor fijo de  $k$  consistiría en evaluar todas las posibles configuraciones iniciales. Por supuesto, este enfoque no es computacionalmente factible en la práctica, especialmente cuando se trabaja con grandes conjuntos de datos y grandes valores de  $k$ . Por tal razón, se han propuesto otros enfoques, entre ellos, los algoritmos evolutivos que han demostrado ser alternativas prometedoras. Los algoritmos evolutivos evolucionan las soluciones de agrupamiento a través de operadores de variación propios, que utilizan reglas probabilísticas para procesar particiones de datos, en éstos, la búsqueda evolutiva está sesgada hacia soluciones de clustering más prometedoras y tienden a realizar una exploración más eficiente del espacio de búsqueda que los enfoques tradicionales aleatorios (por ejemplo, el de múltiples carreras de  $K$ -means). Además, los enfoques tradicionales aleatorios no hacen uso de información sobre la calidad de las particiones asignadas previamente para generar potencialmente mejores particiones. Por este motivo, estos algoritmos tienden a ser menos eficientes (en un sentido probabilístico) que una búsqueda evolutiva. Además de las ventajas antes mencionadas, se ha mostrado también que los algoritmos evolutivos pueden proporcionar particiones de mejor calidad que las que se encuentran con los algoritmos tradicionales. Esto es posible debido a que los algoritmos evolutivos pueden manejar múltiples soluciones, posiblemente guiadas por diferentes medidas de distancia y diferentes funciones de evaluación.

Para atacar el problema de agrupamiento usando algoritmos evolutivos se han propuesto diversos esquemas de representación: binaria, entera y real. Estos esquemas han sido utilizados por diferentes algoritmos, por ejemplo, Kuncheva y Bezdek [15] hacen uso de representación binaria, este esquema permite que la búsqueda se pueda realizar por medio de operadores clásicos de algoritmos genéticos desarrollados originalmente para manipular genotipos binarios. En el caso de representación entera, se puede citar los algoritmos de Lucasius et al. [18] y Estivill-Castro y Murray [9]. Finalmente si hablamos

de representación real, Fránti et al. [10] y Kivijärvi et al. [11], adoptan este último enfoque y adicionalmente codifican en el genotipo el índice de la agrupación a la que pertenece el objeto.

Hasta el momento, los algoritmos citados consideran que se conoce el número de grupos, o bien, éste se especifica con anterioridad. Los algoritmos que se mencionan a continuación van más allá debido a que buscan el mejor número de conglomerados, es decir, estos algoritmos han sido diseñados con el supuesto de que el número de grupos es desconocido. Algunos algoritmos evolutivos usados para optimizar el número de grupos ( $k$ ) y las particiones correspondientes son descritos por Cole [23], Cowgill et al. [24], Hruschka y Ebecken [25], Casillas et al. [26], Ma et al. [27], Alves et al. [28], Naldi y de Carvalho [29], Handl y Knowles [30], y Pan y Cheng [31].

Cole [23] adopta un esquema de representación entera basada en etiquetas, en donde el genotipo es un vector de números enteros de  $N$  posiciones, cada una de las cuales está asociada con un dato y toma un valor (etiqueta de clúster) sobre el alfabeto  $\{1,2,3,\dots, k\}$  en este caso,  $k$  puede ser interpretado como el número máximo de grupos representados en cada individuo. Este esquema de codificación también ha sido utilizado por Cowgill et al. [24], Hruschka y Ebecken [25], Naldi y de Carvalho [29], y Alves et al. [28], pero algunos de estos autores, además sugieren almacenar el número de conglomerados ( $k$ ) en el genotipo. Ma et al. [27] propuso un algoritmo evolutivo para la agrupación, llamado EvoCluster, que codifica una partición de tal manera que cada gen representa un grupo y contiene las etiquetas de los objetos agrupados en ella. Por lo tanto, una codificación de genotipo de  $k$  grupos ( $C_1, C_2, \dots, C_k$ ) para un conjunto de datos con  $N$  objetos se forma por  $k$  genes, en donde cada uno de los cuales almacena  $n_i$  etiquetas ( $n_1 + n_2 + \dots + n_k = N$ ).

Pan y Cheng [31] adoptan un esquema de codificación binaria sobre la base de un número máximo de grupos que se determina a priori. Cada posición de la cadena corresponde a un centroide inicial. Por lo tanto, el número total de 1s en la cadena corresponde al número de clusters codificados dentro genotipo. Casillas et al. [26] también adoptan el esquema de codificación binaria, pero en este caso se ocupa un vector de  $N-1$  elementos. Estos elementos representan los  $N-1$  aristas de un árbol de expansión mínima (MST) [32], en el que los nodos representan los objetos del conjunto de datos  $N$  y las aristas corresponden a los índices de proximidad entre los objetos.

Handl y Knowles [30], emplean una representación gráfica, la cual se basa en que el genotipo es un vector de números enteros de  $N$  posiciones, es decir, un esquema de codificación entera basada en grafos. Cada posición del genotipo corresponde a un objeto, es decir, la posición  $i$ -ésima del genotipo representa el  $i$ -ésimo objeto del conjunto de datos. Los genes pueden tomar valores del conjunto  $\{1, 2, \dots, N\}$ . Un valor  $j$  asignado a un gen  $i$  significa que existe un vínculo entre los objetos  $i$  y  $j$ , por lo tanto éstos se colocan en el

mismo grupo. La partición codificada en el genotipo se recupera mediante la identificación de todos los componentes conectados en el grafo. Este esquema de codificación es particularmente adecuado en el contexto de un algoritmo evolutivo de clustering que considere múltiples objetivos y es el algoritmo primordial en el que se enfoca la presente investigación, por lo cual se dedica un capítulo específico para la explicación del mismo.

A diferencia de las tareas de aprendizaje supervisado, como la clasificación, el clustering es una tarea de aprendizaje no supervisado, por lo que resulta difícil saber con certeza cuál es la solución correcta u óptima. Esto sugiere que la calidad de una solución de clustering sea evaluada por un conjunto de criterios de validez, en lugar de un único criterio, con el fin de disminuir el sesgo impuesto por un criterio de validez en particular. Normalmente, tomar en consideración diferentes aspectos de la calidad de una solución de clustering puede ser traducido a la incorporación de diversos objetivos, que pueden estar en conflicto, en un método de optimización. Para estos casos, los algoritmos de clustering convencionales no son útiles al optimizar más de un objetivo, por lo que se hace uso de algoritmos evolutivos para atacar dicho problema, el algoritmo evolutivo debe hacer frente a objetivos que pueden estar en conflicto, esto asumiendo que hemos decidido que el algoritmo utilizará dos o más criterios de validez, que a menudo es deseable en la práctica. La metodología que se sigue en este caso, es que el algoritmo evolutivo utilice una función multi-objetivo siguiendo el principio de dominancia de Pareto. De acuerdo a este principio, el algoritmo encontrará el frente óptimo de Pareto, es decir, el conjunto de todas las soluciones no dominadas, por lo tanto se obtienen un conjunto de soluciones de agrupamiento. Cabe destacar que los conceptos de optimización multi-objetivo se describen más adelante.

Un enfoque evolutivo para agrupamiento multi-objetivo es el propuesto por Handl y Knowles [30], ellos utilizan una función de aptitud basada tanto en compactación y conectividad de las agrupaciones. La interacción de estos dos objetivos permite mantener el número de grupos estable, evitando así la convergencia a soluciones triviales, mientras que permite la exploración de las regiones interesantes del espacio de búsqueda. Los autores examinaron dos pasos en el proceso de clustering que podrían beneficiarse de la utilización de múltiples objetivos. En primer lugar, se considera la generación de soluciones de clustering a través de una variante del algoritmo evolutivo PESA-2 (Pareto envelope-based selection algorithm-2) [33], el cual sigue un enfoque llamado selección basada en regiones. En segundo lugar, consideran el problema de la selección de modelos para elegir la mejor solución de clustering en un conjunto de soluciones.

Otro algoritmo evolutivo para la agrupación multi-objetivo se describe en [34]. Los dos objetivos utilizados en este caso son la varianza intra-cluster total y el número de grupos. Ambos objetivos deben reducirse al mínimo, pero ya que están en conflicto entre sí, es necesario utilizar el concepto de dominancia de Pareto, el algoritmo logra encontrar un

conjunto diverso de soluciones de clustering no dominadas, en los que para cada número diferente de clusters, el algoritmo encuentra la varianza intra-cluster más pequeña posible. Por otro lado, Mukhopadhyay et al. [35] usan dos objetivos, en donde el primero también consiste en la variación intra-cluster total, el segundo consiste en una medida de separación entre los grupos, es decir, la distancia inter-cluster. Ripon et al. [36] toman en consideración los mismos objetivos, a diferencia de que en lugar de ser la variación total calculan un promedio de la variación obtenida por todos los clusters, esto es con el fin de obtener un valor normalizado de la medida teniendo en cuenta el número de grupos, que varía para los diferentes individuos de la población del algoritmo evolutivo.

Como se puede observar en los antecedentes anteriormente descritos, los algoritmos evolutivos han sido utilizados con éxito para la realización de agrupamiento, por lo que en la presente tesis se toma en consideración esta idea para llevar a cabo la optimización de dos objetivos en el proceso de clustering, realizando un estudio del comportamiento del agrupamiento usando distintos algoritmos evolutivos para la optimización multi-objetivo.

## **1.2 Definición del problema.**

Cuando se abordan problemas de optimización, la solución deseada puede ser descrita de mejor manera usando una serie de objetivos a optimizar, normalmente al momento de abordar más de un objetivo, éstos pueden entrar en conflicto entre sí lo que se traduce a una mayor complejidad. La optimización multi-objetivo engloba un conjunto de técnicas que tratan de dar solución a éste problema, obteniendo soluciones óptimas de un problema dado en donde se involucra la optimización de más de un objetivo.

El agrupamiento de datos es un ejemplo de un problema en el que buenas soluciones se describen mejor utilizando diferentes criterios, como puede ser la separación entre grupos o la conectividad de los mismos. Debido a que los algoritmos evolutivos (AE) han demostrado ser efectivos para abordar problemas multi-objetivo, en la presente tesis se intenta resolver dicho problema mediante la optimización de dos objetivos (conectividad y desviación global), con lo cual se quiere lograr encontrar mejores soluciones de agrupamiento.

En este trabajo se toma como base al algoritmo propuesto por, Julia Handl and Joshua Knowles llamado MOCK (Multiobjective Clustering with automatic determination of the number of clusters) el cual optimiza dos objetivos y encuentra una aproximación para el frente de Pareto (el conjunto óptimo), en lugar de una única solución. El algoritmo hace uso del algoritmo evolutivo PESA-2 para llegar al conjunto de soluciones óptimas, sin embargo, dicho algoritmo sólo ha sido probado con PESA-2, por lo que los esfuerzos de esta investigación se han centrado en probar otros AE con el fin de realizar un estudio comparativo del funcionamiento de MOCK con AE que han resultado ser más eficientes

para problemas de optimización numérica, tal es el caso de NSGA-II y SPEA-2 y de esta manera poder seleccionar el algoritmo más conveniente para tal problema.

### **1.3 Objetivos**

#### **1.3.1 Objetivo general**

Determinar si el algoritmo MOCK presenta un mejor desempeño al resolver el problema de agrupamiento de datos, mediante la modificación del algoritmo evolutivo multi-objetivo con el que trabaja, comparando y analizando los resultados obtenidos con la versión original.

#### **1.3.2 Objetivos específicos**

- Analizar los conceptos de optimización multi-objetivo.
- Proponer modificaciones al MOCK incorporando los algoritmos NSGA-II y SPEA-2.
- Implementar los algoritmos propuestos para resolver el problema de clustering.
- Probar los algoritmos con distintos conjuntos de datos de la literatura.
- Calibrar los parámetros de los algoritmos implementados usando la herramienta irace.
- Comparar los resultados con otros algoritmos del estado del arte.
- Seleccionar e implementar métricas de desempeño propias de optimización multi-objetivo.
- Realizar pruebas estadísticas para validar los resultados obtenidos.

### **1.4 Hipótesis**

"El uso de los algoritmos evolutivos para optimización multi-objetivo: NSGA-II y SPEA-II, como alternativas para realizar optimización en MOCK, mejora el desempeño de dicho algoritmo para encontrar mejores aproximaciones al frente de Pareto óptimo, lo que implica la obtención de mejores soluciones de agrupamiento".

### **1.5 Justificación.**

El agrupamiento de datos es una técnica ampliamente utilizada en el área de aprendizaje automático con múltiples aplicaciones. Existen varios algoritmos que son utilizados para realizar agrupamiento, el uso de alguno en específico puede variar

dependiendo de la aplicación, complejidad de la base de datos, y otros factores como distribución de los grupos. Sin embargo, para que una solución de agrupamiento pueda ser considerada como buena se deben tomar en cuenta diversos factores, lo cual involucra que al momento de formar los grupos se consideren distintas medidas de calidad del clustering, como ya se mencionó, hablar de la incorporación de más de una medida de calidad puede agregar complejidad al problema de agrupamiento. Tomando en consideración que el agrupamiento puede ser visto como un problema de optimización, las medidas de calidad de clustering deben ser optimizadas, dicho de otro modo, éstas pueden ser vistas como objetivos a optimizar. Ahora bien, las técnicas de clustering tradicionales no toman en cuenta la creación de soluciones de agrupamiento basadas en varias medidas de calidad, por lo tanto el uso de otros métodos como los algoritmos evolutivos, pueden resolver problemas que involucren más de un objetivo obteniendo soluciones prometedoras. Otra manera de justificar el uso de algoritmos evolutivos en el problema del clustering, es que algoritmos como K-means, Fuzzy K-means, etc., tienen la desventaja de que el cluster resultante es sensible a las posiciones iniciales de los centroides por lo que pueden converger en un mínimo local, además de que el número de grupos ( $k$ ) debe ser especificado con anterioridad. Los algoritmos propuestos, serán capaces de determinar el número de grupos de manera automática así como también optimizar más de un objetivo.

Las primeras propuestas de algoritmos evolutivos que resuelven el problema de agrupamiento, tomaban en consideración un sólo objetivo, más adelante surgieron las de optimización multi-objetivo en donde el algoritmo de clustering de optimización multi-objetivo mayormente aplicado es el descrito por Handl y Knowles, y como ya se mencionó es la base de la presente investigación, sin embargo debido a que solo hace uso del algoritmo PESA-2, es importante explorar y explotar nuevas metaheurísticas que resuelvan este problema y que ofrezcan al área de aprendizaje automático mayor diversidad de estrategias para la solución de problemáticas de diversa índole al momento de trabajar con grandes cantidades de datos.

En este trabajo se presentan tres algoritmos para la resolución del problema de agrupamiento de datos usando optimización multi-objetivo, variando el algoritmo evolutivo, promoviendo con ello el uso de otros algoritmos evolutivos multi-objetivo (AEMO), para obtener un algoritmo robusto y útil que obtenga soluciones de agrupamiento comparables o bien mejores que otros algoritmos del estado del arte.

## **1.6 Organización del documento de tesis.**

La tesis está organizada de la siguiente manera:

- **Capítulo 2. Cómputo Evolutivo.** En este capítulo se presentan los antecedentes históricos que dieron origen al cómputo evolutivo, se describen los conceptos principales de ésta área y se mencionan algunos de los paradigmas principales.
- **Capítulo 3. Optimización.** En este capítulo se describen los conceptos fundamentales sobre optimización. Se exponen de forma detallada conceptos sobre optimización multi-objetivo, así como algunos algoritmos clásicos para la resolución de este tipo de problemas. También se describen algunas medidas de desempeño propias de la optimización multi-objetivo.
- **Capítulo 4. Clustering Multi-objetivo.** En este capítulo se presenta el concepto de clustering multi-objetivo, así como también se hace una descripción detallada del algoritmo MOCK, el cual es la base de la presente investigación, se muestran los esquemas de representación, operadores de variación y función de calidad utilizados.
- **Capítulo 5. Metodología.** En este capítulo se describen las modificaciones realizadas al algoritmo MOCK, se presentan los conjuntos de datos con los que se realizan las pruebas, se da una explicación detallada de los experimentos realizados. También se presenta una descripción de irace, paquete usado para la calibración de parámetros.
- **Capítulo 6. Resultados.** En este capítulo se muestran y discuten los resultados obtenidos por las dos versiones de MOCK implementadas (MOCK<sub>NSGA-II</sub> y MOCK<sub>SPEA-2</sub>), comparando los mismos con la versión original (MOCK<sub>PESA-2</sub>) y con otros algoritmos de agrupamiento clásicos tomando en consideración distintas medidas de desempeño y pruebas estadísticas para validar los resultados.
- **Capítulo 7. Conclusiones y trabajos futuros.** En este capítulo se presentan las conclusiones a las que se llegaron basándose en los resultados obtenidos y los futuros trabajos a realizar.

# Capítulo II: Cómputo Evolutivo

La Computación evolutiva, tal como su nombre lo indica, engloba técnicas que simulan los procesos evolutivos fundamentados en la selección natural, es decir, en la supervivencia de los individuos más aptos de una población. El cómputo evolutivo ha sido utilizado como un método alternativo para abordar problemas de búsqueda y optimización con complejidad y dimensionalidad elevada.

En este capítulo se presentan brevemente los antecedentes históricos de la teoría Neo-Darwiniana, los cuales dieron origen al cómputo evolutivo, así como también la forma en que esta área ha adoptado ideas de dicha teoría.

## 2.1 Antecedentes históricos

El paradigma conocido como Neo-Darwinismo es la fusión de la teoría evolutiva propuesta por Charles Darwin, el seleccionismo de August Weismann y de la teoría genética descrita por Gregor Mendel [37]. Para el Neo-Darwinismo, la vida puede ser explicada como un proceso que actúa sobre poblaciones y especies y consta de los siguientes pasos: reproducción, mutación, competencia y selección [37]. A continuación, se presenta más a detalle las teorías que sustentan dicho proceso.

### 2.1.1 Teoría de la evolución de Charles Darwin

En 1859, Charles Darwin en su obra titulada como *"El origen de las especies"*, propuso que la evolución es generada con base en cambios aleatorios de caracteres hereditarios y a un proceso de selección natural [38], con esto se sentaron las bases de lo que en la actualidad conocemos como evolución. Según la teoría de la evolución, las especies existentes en la actualidad son descendientes de otras especies que existieron en el pasado, dichas especies fueron producidas mediante un conjunto de mecanismos los cuales permitieron generar nuevos individuos similares pero con algunas modificaciones, por lo que el proceso más relevante es la selección y mediante la herencia se transmiten características de los padres a los descendientes. Para Darwin, los nuevos individuos poseen mejores características que sus antecesores, cuando éstas características ayuden a los individuos a adaptarse de mejor manera a su ambiente tendrán una mayor probabilidad de supervivencia. A dicho principio se le conoce como proceso de selección natural, el cual se basa en la supervivencia de los individuos mejor adaptados.

### 2.1.2 Teoría del plasma germinal de August Weismann

El biólogo alemán August Weismann [39, 40], a mediados del siglo XIX publicó una obra titulada "*Plasma Germinal: Una Teoría de la Herencia*" [41], en dicha obra se establece la teoría conocida como *Germoplasma*, en donde Weismann indica que los individuos están formados por células germinales (germoplasma) y somáticas, las primeras son las que contienen la información a heredar, y las segundas forman el cuerpo y funciones del individuo. El germoplasma no puede ser alterado o modificado durante la existencia de un individuo; es decir, si el individuo aprende nuevas habilidades estos conocimientos no serán heredados, ya que el germoplasma no los contempla.

### 2.1.3 Teoría genética de Gregor Mendel

En 1855, Johann Gregor Mendel [40] en su obra titulada "*Experimentos de Hibridación en Plantas*" [41], enuncia las leyes de la herencia como resultado de los experimentos que realizó con chícharos, las cuales hoy en día son fundamentales en el campo de la genética, con dichas leyes se establecieron los conceptos de genes dominantes y recesivos y sus relaciones al momento de su transmisión de padres a hijos, las leyes se muestran a continuación:

- **Ley de la Independencia:** los pares de alelos se separan durante la formación de gametos.
- **Ley de la Segregación:** los genes recibidos de los padres se separan durante la producción de los gametos.
- **Ley de la Uniformidad:** las características heredadas son determinadas mediante dos factores provenientes de los padres, lo cual decide si un gene es dominante o recesivo.

### 2.1.4 Neo-Darwinismo

Tal como fue mencionado anteriormente, se conoce como Neo-Darwinismo al conjunto de teorías presentadas por Darwin, Mendel y Weismann en torno al origen de las especies. Este paradigma puede ser explicado como un proceso que involucra:

- **Reproducción:** Es el proceso por el cual se asegura que la información genética de un individuo sea parte de la siguiente generación. Existe la reproducción sexual, en la que intervienen dos individuos que generarán individuos con información genética diferente entre ellos. Y la reproducción asexual, donde un solo individuo es capaz de generar nuevos individuos con réplicas de su información genética, resultando en individuos iguales entre ellos.

- **Mutación:** En este proceso se realiza una modificación de la información genética de un individuo. En algunos casos, las mutaciones pueden resultar benéficas, esto implica que el cambio le otorgará al individuo una mejor adaptación al ambiente.

- **Competencia:** Tal como su nombre lo indica, es el proceso en el cual los individuos tienen una lucha constante por subsistir y reproducirse, heredando su código genético. Este proceso se da por el exceso poblacional de una especie que afecta a los individuos menos aptos al realizarse una selección estocástica.

- **Selección:** Es el proceso en el cual los individuos que son más aptos que otros tendrán una mayor oportunidad de supervivencia y reproducción.

## 2.2 Fundamentos biológicos

Debido a que el cómputo evolutivo, toma aspectos de la teoría Neo-Darwiniana, y ésta se encuentra basada en aspectos biológicos, se mencionan algunos conceptos que puede resultar relevantes para una mejor comprensión.

Como ya se sabe, todos los organismos tienen un material genético asociado el cuál es conocido como **ADN** (ácido Desoxirribonucleico). Un **gene** es una sección de ácido desoxirribonucleico esencial para llevar a cabo cierta función bioquímica definida y es la unidad fundamental de la herencia. Un **cromosoma** es una cadena de ADN formada por genes, la cual es responsable de transmitir la información genética del organismo. Cada gene puede presentar distintas formas alternativas dentro de una posición específica del cromosoma, a esto se le denomina **Alelo**. El **Genoma** es el conjunto total de cromosomas, es decir, de genes de un organismo. Un individuo, puede tener rasgos observables, llamados **fenotipo** y composición genética, la cual no es observable y es llamada **genotipo**. La relación entre el fenotipo y el genotipo es compleja y no lineal. Existen dos fenómenos que determinan esta relación: la pleitropía y la poligenia [37]. La **pleitropía** es el efecto donde un segmento de la información genética afecta a varias rasgos a nivel fenotipo y la **poligenia** es el efecto donde una característica fenotípica es determinada por la interacción de varios genes. Cuando se está hablando de un conjunto de individuos los cuales pueden interactuar de manera conjunta con fines de reproducción nos referimos a una **población**. La **aptitud** de un individuo se usa para determinar la probabilidad de un individuo para sobrevivir y reproducirse en un ambiente específico.

## 2.3 Conceptos del Cómputo Evolutivo

La computación evolutiva, combina los conceptos asociados a la genética con el proceso de selección natural, por lo que se realiza una simulación del proceso de la evolución tomando un conjunto de individuos los cuales serán conocidos como soluciones

potenciales al problema, dichos individuos que conforman la población estarán sujetos a procesos de selección, reproducción y mutación.

Para modelar un proceso evolutivo en una computadora son necesarios los siguientes elementos [42]:

- Tener una representación adecuada para las soluciones potenciales al problema, es decir, para los individuos.
- Una manera de inicializar la población de individuos.
- Un ambiente, modelado por una función de aptitud, con la cual se permita la comparación de unos individuos con otros.
- Operadores de reproducción que actúen sobre los individuos existentes y permitan la generación de nuevos individuos.
- Valores para los parámetros propios de la técnica (probabilidades de cruce y mutación, número de individuos).

### 2.3.1 Algoritmos Evolutivos (AES)

Un Algoritmo Evolutivo (AE) comienza con una etapa de inicialización de los individuos, la cual puede ser completamente aleatoria, o guiada de acuerdo a características del problema a resolver. El proceso evolutivo se lleva a cabo en el ciclo que genera nuevos individuos a partir de la población actual mediante un procedimiento de aplicación de operadores estocásticos, en este ciclo se pueden identificar claramente cuatro etapas importantes, la *evaluación*: en la que se calcula el valor de aptitud (fitness) de cada individuo de la población actual, la *selección*: mediante la cual se eligen a los candidatos adecuados de acuerdo al valor del fitness para la aplicación de los operadores evolutivos, *aplicación de operadores*: que permite generar un conjunto de descendientes a partir de los individuos anteriormente seleccionados y el *reemplazo*: que realiza el intercambio generacional, sustituyendo a los individuos de la generación anterior por los descendientes creados al momento de aplicar los operadores. La condición de paro de la fase iterativa del algoritmo evolutivo puede tomar en cuenta un número de generaciones máximo, o cuando los valores de aptitud permanecen sin cambio, es decir, el proceso se estanca y no obtiene mejoras considerables en los valores de aptitud respecto al valor óptimo del problema, lo cual significa que se ha caído en un óptimo local.

El funcionamiento básico de un Algoritmo Evolutivo (AE) para llevar a cabo la resolución de un problema, aunque los diferentes paradigmas realizan modificaciones o adecuaciones específicas, se muestra en la Figura 2.1 y Figura 2.2.

---

Algoritmo 1: Esquema general de un AE.

---

- 1: Generar la población aleatoria inicial con  $n > 0$  individuos donde cada individuo representa una solución potencial del problema.
  - 2: Se evalúa a cada individuo con la función de aptitud.
  - 3: **repetir:**
  - 4: Seleccionar a los padres.
  - 5: Cruzar padres.
  - 6: Mutar individuos nuevos (hijos).
  - 7: Evaluar a los hijos.
  - 8: Seleccionar individuos que pasan a la siguiente generación.
  - 9: **hasta:** Que se satisfaga una condición de paro.
- 

Figura 2.1: Algoritmo evolutivo general.

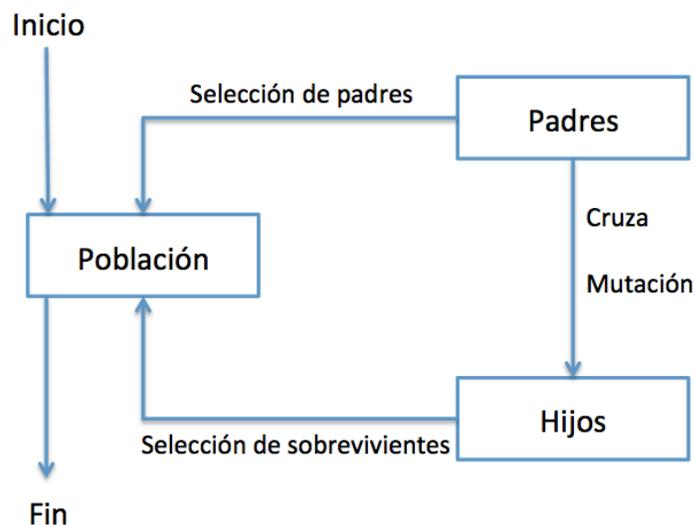


Figura 2.2: Esquema general de un algoritmo evolutivo.

## 2.3.2 Componentes de los algoritmos evolutivos

Los elementos principales de un algoritmo evolutivo se describen a continuación:

### 2.3.2.1 Representación de individuos

Para poder abordar un problema determinado, primero se debe encontrar una manera adecuada de representar las variables de tal forma que se tengan  **cromosomas**  que representen a cada individuo de la población. Los cromosomas, normalmente son arreglos de enteros o reales, en donde cada posición es conocida como  **gen**  y corresponde a una de las variables de decisión (variables del problema planteado). El valor que puede tomar cada gen se conoce como  **alelo**  (ver figura 2.3). La codificación que representan los cromosomas es el  **genotipo** , mientras que la decodificación de los valores del cromosoma de manera que puedan sustituirse en la función de aptitud es el  **fenotipo** . Por lo tanto, se

tienen los dos niveles de representación a nivel fenotípico y genotípico (ver figura 2.4). En caso de usar representación binaria, el cromosoma tendrá en cada alelo los valores de 0 ó 1.

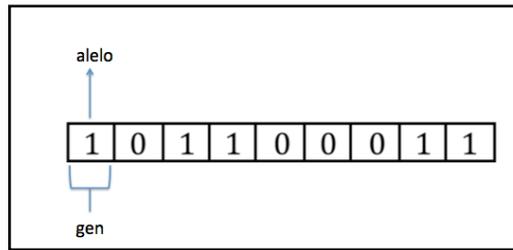


Figura 2.3: Cromosoma con representación binaria

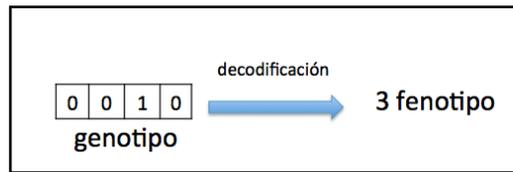


Figura 2.4: Representación del genotipo y fenotipo.

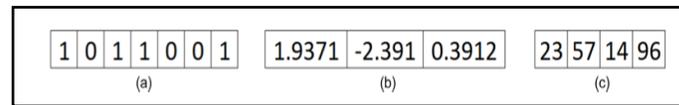


Figura 2.5: Representación binaria (a), real (b) y entera (c).

### 2.3.2.2 Población

Un individuo se define como una solución potencial al problema. La población será un conjunto de individuos, donde la aptitud de cada uno corresponderá a la calidad de la solución, es decir, al ser evaluado el individuo en la función objetivo y comparado con los demás individuos se sabrá qué tanto es mejor o peor que los demás individuos de la población.

### 2.3.2.3 Función de calidad

Dependiendo del problema, la función de calidad es la que queremos minimizar o maximizar, es decir, la función a optimizar. Nos determina qué individuos son más aptos, y éstos tendrán mayor probabilidad de sobrevivir y tener descendencia. El valor que es arrojado por la función objetivo define la **aptitud** de un individuo.

### 2.3.2.4 Mecanismo de selección

La selección de los individuos es de vital importancia en un algoritmo evolutivo debido a que es la que guía la búsqueda hacia una solución buena. El objetivo de la

selección es la obtención del conjunto de padres que participarán en el proceso de reproducción para generar descendencia. Normalmente, los individuos con una calidad mayor (la cual es definida por la función de calidad) tienen mayor probabilidad de ser seleccionados (tal como lo indica el proceso de selección natural). Con esto, se trata de lograr que las generaciones de hijos, superen a las de los padres a lo largo del proceso evolutivo. Existen diferentes técnicas de selección, se pueden distinguir las siguientes:

- Selección proporcional [43]: se eligen a los individuos de acuerdo a la contribución de aptitud que tengan con respecto al total de la población. Algunas propuestas de este tipo de selección son:

1. Ruleta.
2. Muestreo Determinístico.
3. Sobrante Estocástico: con reemplazo y sin reemplazo.
4. Universal Estocástica.

- Selección mediante torneo [44]: se basa en la comparación directa entre individuos, puede ser de dos o más participantes en cada torneo. Se realiza tomando una muestra aleatoria de individuos, la cual representa a los participantes del torneo, donde el ganador (el individuo seleccionado) será el de mejor aptitud.

1. Determinística: siempre se elige al más apto
2. Probabilística: se elige con una cierta probabilidad al individuo más apto, y en caso contrario, se elige al menos apto.

- Selección de estado uniforme [45]: en este caso solamente algunos individuos serán reemplazados por los nuevos (hijos) en cada generación. Esta técnica es utilizada en algoritmos genéticos no generacionales y aprendizaje incremental.

### **2.3.2.5 Operadores de variación**

Los operadores de variación modifican la manera en que se transmite la información genética de padres a hijos. Su objetivo es crear nueva descendencia a partir de una anterior. Podemos distinguir los siguientes operadores.

- Operador de cruce: Utiliza partes de dos o más padres para obtener uno o más hijos tomando las mejores características de los padres para crear mejores hijos. Este operador se aplica con cierta probabilidad, es decir, no siempre se va a realizar el intercambio de genes. Entre las formas más usadas de realizar cruce se encuentra la cruce de  $n$  puntos [46] que fue propuesta para representación binaria (ver figura 2.6). Para representaciones reales y enteras, se puede usar: Cruce de un punto [47], Cruce de dos puntos [48], Cruce uniforme[49,50].

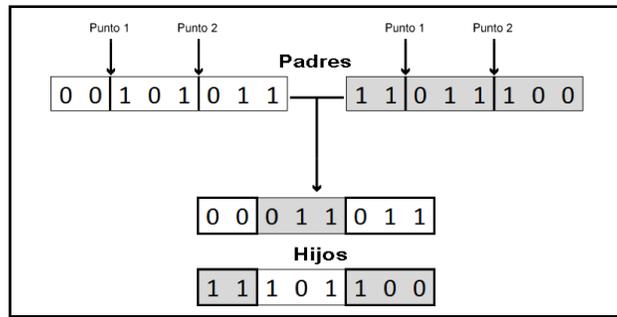


Figura 2.6: Ejemplo de cruce de dos puntos.

- Operador de mutación: Forma un nuevo individuo a partir de modificaciones pequeñas al contenido genético de un solo padre. De la misma manera que la cruce, la mutación se realiza con cierta probabilidad, pero en este caso, la probabilidad indica si el alelo o el gen cambiarán su valor (ver figura 2.7).

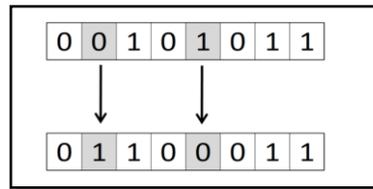


Figura 2.7: Ejemplo de mutación

### 2.3.2.6 Mecanismo de reemplazo

Es similar al mecanismo de selección de padres sólo que difiere en que es utilizado después de la aplicación de los operadores de variación. Su rol es el de distinguir entre individuos con base en su calidad para mantener el tamaño de población fijo. Usualmente es determinista.

### 2.3.3 Principales paradigmas

Existen diferentes paradigmas dentro del cómputo evolutivo, aunque todos los paradigmas se basan en la misma idea de emular la evolución usando una población de soluciones, difieren en el nivel en el cual simulan dicho proceso, en sus operadores primarios y secundarios, además del nivel de representación en el cual trabajan. Los principales paradigmas se describen a continuación:

### 2.3.3.1 Algoritmos Genéticos

Fueron concebidos por John H. Holland [47] a inicios de la década de los 60's. El Algoritmo Genético (AG) tradicionalmente posee una representación binaria, es decir, trabajan a nivel genotipo, su operador primario es la cruce sexual al cual se le da mayor importancia que al operador de mutación, esto es debido a que modela la evolución a nivel de los individuos. Normalmente no cuenta con mecanismos de auto-adaptación. Los individuos son seleccionados de manera probabilística basada en aptitudes. El pseudocódigo de un algoritmo genético simple se muestra en la figura 2.8.

---

Algoritmo 2: Esquema de un AG simple.
1: Generar aleatoriamente una población inicial.
2: Se evalúa a cada individuo con la función de aptitud.
3: <b>repetir:</b>
4:    Seleccionar a los padres.
5:    Cruzar padres.
6:    Mutar individuos nuevos (hijos).
7:    Evaluar a los hijos.
8:    Seleccionar individuos que pasan a la siguiente generación.
9: <b>hasta:</b> Que se satisfaga una condición de paro.

---

Figura 2.8: Algoritmo genético simple.

### 2.3.3.2 Estrategias Evolutivas

Las Estrategias Evolutivas (EE) fueron desarrolladas en Alemania con el objetivo de resolver problemas de Aerodinámica, propuestas por Peter Bienert, Ingo Rechenberg y Hans-Paul Schwefel [51] y consisten en un método de ajustes discretos aleatorios inspirado en el proceso de mutación biológico [52].

En la propuesta inicial, el esquema sólo usaba un individuo que era mutado para producir un descendiente. Más adelante Schwefel introdujo el uso de poblaciones [53]. El nivel de operación de las EE es fenotípico, donde la mutación es el operador principal y se realiza de forma Gaussiana [54], lo que permite que las EE sean auto-adaptativas, debido a que el valor de mutación varía en el tiempo. El operador de cruce es un operador secundario, la cruce puede ser sexual (dos padres) o panmítica (un solo padre). Se utiliza un esquema de selección determinística y extintiva (los peores individuos tienen probabilidad cero de sobrevivir).

La primera versión, sin población, es conocida como  $(1 + 1) - EE$ , donde con un solo padre genera un solo hijo y el mejor sobrevive en la siguiente generación. La forma de generar al hijo es mediante la siguiente expresión:  $x^{t+1} = x^t + N(0, \sigma)$  donde  $t$  se refiere a la generación y  $N(0, \sigma)$  es un vector de números Gaussianos independientes con media cero y

desviación estándar  $\sigma$ . Si el hijo es mejor en aptitud, se mantiene; de lo contrario se elimina.

Rechenberg [55] propone la  $(\mu+1)-EE$ , la cual cuenta con  $\mu$  padres que generan un solo hijo y se seleccionan  $\mu$  individuos para la siguiente generación, es decir, el hijo puede o no reemplazar a algún padre. Más adelante aparecen dos propuestas de Schwefel [56] que son  $(\mu + \lambda) - EE$  y  $(\mu, \lambda) - EE$ , donde se tienen  $\mu$  padres que generan  $\lambda$  hijos. La diferencia entre ambas radica en la estrategia de selección. La primera selecciona de ambas poblaciones a los  $\mu$  mejores individuos y la segunda selecciona  $\mu$  individuos de los  $\lambda$  hijos generados (se requiere que  $\mu \leq \lambda$ ).

La desviación estándar que se utiliza para la generación de los nuevos individuos es un parámetro sumamente importante en la mutación Gaussiana, y el valor de ésta será un factor para el éxito del paradigma. Sin embargo, el valor óptimo de este parámetro depende de la dimensionalidad y naturaleza del problema.

---

Algoritmo 3: Esquema de las EE.
1: Generar aleatoriamente una población inicial.
2: Calcular la aptitud de individuos de la población inicial.
3: <b>repetir:</b>
4:    Seleccionar el conjunto de padres de manera aleatoria
5:    Aplicar operador de cruza.
6:    Aplicar operador de mutación a los descendientes.
7:    Calcular aptitud de todos los descendientes.
8:    Realizar la selección del mejor individuo (“+” o “;”) para la siguiente generación basado en la aptitud.
9: <b>hasta:</b> Que se satisfaga una condición de paro.

---

Figura 2.9: Algoritmo de Estrategias Evolutivas

### 2.3.3.3 Programación Evolutiva

Este paradigma fue propuesto por Lawrence J. Fogel [57] en 1960. En este esquema la inteligencia se ve como un comportamiento adaptativo y le da más importancia al vínculo entre padres e hijos que a los operadores genéticos. En la Programación Evolutiva (PE) no existe un operador de cruza, ya que ésta opera a nivel de las especies y especies distintas no pueden cruzarse [58, 59]., por lo tanto. El operador genético principal es la mutación La selección de individuos es de manera probabilística y opera a nivel fenotipo. En la propuesta original de la PE, no se considera la auto-adaptación de parámetros. El algoritmo básico de PE se muestra en la figura 2.10.

---

Algoritmo 4: Esquema de un algoritmo de PE.

---

- 1: Generar aleatoriamente una población inicial.
  - 2: **repetir:**
  - 3:   Aplicar operador de mutación.
  - 4:   Calcular aptitud de todos los individuos de la población.
  - 5:   Seleccionar mediante torneo estocástico los individuos que sobrevivirán.
  - 6: **hasta:** Que se satisfaga una condición de paro.
- 

Figura 2.10: Algoritmo básico de programación evolutiva.

#### 2.3.3.4 Otras alternativas

Existen otras propuestas más recientes y que comparten características con los paradigmas principales, se pueden citar las siguientes:

- Programación Genética [60].
- Evolución Diferencial [61]
- Optimización mediante cúmulos de partículas [62]
- Optimización por colonia de hormigas [63]
- Sistema inmune artificial [64]
- Algoritmos meméticos [65].

#### 2.3.4 Ventajas de los algoritmos evolutivos

Las principales ventajas que presenta el uso de los algoritmos evolutivos en la resolución de problemas de optimización [66] son entre otras, las siguientes:

- Operan sobre una población (o conjunto de soluciones) lo que evita que la búsqueda se quede atascada en óptimos locales.
- No requieren conocimiento previo sobre el problema a resolverse.
- Pueden combinarse con otras técnicas de búsqueda para mejorar su desempeño.
- Permiten su paralelización de forma sencilla. Son conceptualmente fáciles de implementar y usarse.
- Utilizan operadores probabilísticos, en comparación con las técnicas tradicionales que utilizan operadores determinísticos.
- Generalmente pueden auto-adaptar sus parámetros.

# Capítulo III: Optimización

## 3.1 Conceptos de optimización

El proceso de optimización consiste en encontrar la mejor solución candidata de entre una colección de alternativas. Éste proceso se plantea como un problema estructurado con funciones de variables de decisión, que deben o no satisfacer un conjunto de restricciones impuestas por un problema (ver figura 3.1). Sin embargo no existe ningún método de optimización capaz de solucionar cualquier tipo de problema. Tomando en cuenta número de funciones objetivo, la optimización se puede dividir en dos tipos: mono-objetivo y multi-objetivo. La primera, abarca problemas con una sola función objetivo y la segunda se refiere a problemas donde existen dos o más funciones objetivo.

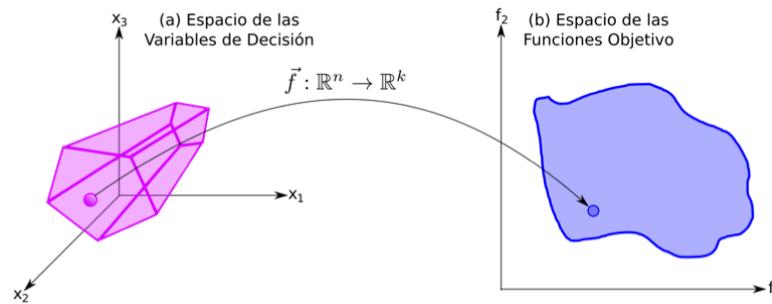


Figura 3.1: Espacios de búsqueda en un problema de optimización.

En problemas mono-objetivo, la noción de óptimo (la mejor solución posible, dadas las restricciones del problema) es un tanto intuitiva. Sin embargo, esto no ocurre en problemas multi-objetivo, donde normalmente sus funciones objetivo están en conflicto, es decir, optimizar una función implica empeorar otra. En consecuencia, en los problemas multi-objetivo se genera un conjunto de soluciones que representan los mejores compromisos posibles entre los objetivos del problema. Matemáticamente, el problema de optimización se describe mediante los siguientes elementos:

- Variables de decisión: Son el conjunto de  $n$  parámetros (de entrada para la función objetivo) cuyos valores arrojan una posible solución al problema. Estas variables pueden ser enteras, reales, o cualquier combinación de ellas. Cada uno de estos parámetros es denotado como  $x^i$ ,  $i = 1, 2, \dots, n$ . El vector  $\vec{x}$  de  $n$  variables se representa como sigue:

$$\vec{x} = [x_1, x_2, \dots, x_n]$$

- **Función objetivo:** pueden ser una(mono-objetivo) o varias (multi-objetivo). Cuando se consideran varias funciones, el problema se denomina optimización multi-objetivo. Estas funciones se expresan en términos de las variables de decisión, y el resultado de su evaluación es el que se desea optimizar (maximizar o minimizar). Para fines de esta tesis, abordaremos problemas multi-objetivo.

- **Restricciones:** Son limitaciones arrojadas por las condiciones del ambiente y los recursos con los que se cuenta, las cuales se deben cumplir o satisfacer para que la solución sea considerada factible, es decir, válida. Si el problema no presenta restricciones, todas las soluciones son válidas. Las restricciones pueden ser expresadas en forma de ecuaciones de igualdad:

$$h(\vec{x}) = 0$$

o desigualdad

$$g_i(\vec{x}) \leq 0$$

El problema de optimización puede ser expresado como uno de programación no lineal cuyo objetivo es:

encontrar  $\vec{x} = [x_1, x_2, \dots, x_n]$  que optimice la función  $f(\vec{x}) \in \mathbb{R}$   
donde  $\vec{x}$  está sujeto a:

$$g_i(\vec{x}) \leq 0, i = 1, 2, \dots, m.$$

$$h(\vec{x}) = 0, i = 1, 2, \dots, p$$

Donde  $\vec{x}$  es el vector de variables de decisión,  $f(\vec{x})$  es la función objetivo,  $g(\vec{x})$  y  $h(\vec{x})$  corresponden a las restricciones de desigualdad e igualdad del problema. Cualquier vector  $\vec{x} \in \mathbb{R}^n$  que satisface las restricciones se denomina *solución factible*, y el conjunto de todas las soluciones factibles se denomina *región factible*. Se dice que  $g(\vec{x})$  es una restricción activa en el óptimo cuando  $g(\vec{x}) = 0$  en ese punto. En caso contrario, se denomina restricción inactiva. Todas las restricciones de igualdad se consideran activas para cualquier punto de la región factible.

Se denomina *óptimo* al valor más pequeño (asumiendo minimización) de  $f(\vec{x})$ . El óptimo global se denota con el par  $(\vec{x}^*, f(\vec{x}^*))$  que constituye una solución óptima. Un óptimo local es el menor valor de  $f(\vec{x})$ , en una vecindad de algún vector  $\vec{x}$ .

### 3.2 Optimización multi-objetivo

Por lo general, en los problemas del mundo real nos enfrentamos a un alto nivel de complejidad, en ciertas ocasiones dicha complejidad puede consistir en que se deben de satisfacer varios objetivos y restricciones.

Los problemas de optimización multi-objetivo están formados por diferentes funciones objetivo que están expresadas en unidades diferentes y pueden encontrarse en conflicto entre sí (al menos parcialmente). Éste tipo de problemas, tienen la particularidad de no tener una solución única, sino un conjunto de soluciones compromiso; a este conjunto se le conoce como conjunto óptimo de Pareto (más adelante se explicará a detalle dicho concepto).

A continuación se define el problema de optimización multi-objetivo, así como también los conceptos más relevantes dentro de la optimización multi-objetivo y se describen las principales técnicas evolutivas existentes para este tipo de optimización.

El problema de optimización multi-objetivo es definido como [67]:

*El problema de encontrar un vector de variables de decisión que satisfaga las restricciones y que optimice una función vectorial cuyos elementos representen las funciones objetivo. Estas funciones forman una descripción matemática de criterios de desempeño que están usualmente en conflicto entre sí. Por lo tanto, el término optimizar significa encontrar aquella solución que daría un valor aceptable al diseñador en todas las funciones objetivo.*

De manera formal esto se puede expresar como sigue:

Encontrar un vector  $\vec{x}^* = [x_1^*, x_2^*, \dots, x_n^*]^T$  el cual debe satisfacer las  $m$  restricciones de desigualdad:

$$g_i(\vec{x}) \leq 0, i = 1, 2, \dots, m.$$

y las  $p$  restricciones de igualdad:

$$h_i(\vec{x}) = 0, i = 1, 2, \dots, p$$

y optimiza la función vector:

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

Donde  $\vec{x} = [x_1, x_2, \dots, x_n]^T$  es un vector de variables de decisión.

Lo cual quiere decir que se requieren determinar los números que pertenecen a la zona factible  $F$  que satisfacen las restricciones de igualdad y desigualdad y que obtienen el valor óptimo para todas las funciones objetivo de la función vector.

Existen 3 tipos de problemas multi-objetivo:

- Minimizar todas las funciones objetivo.
- Maximizar todas las funciones objetivo.
- Minimizar algunas y maximizar las funciones objetivo restantes.

Por simplicidad, se suelen considerar sólo problemas del primer tipo (minimizar todas las funciones objetivo), transformando las funciones objetivo, en caso de ser necesario.

Para los problemas multi-objetivo, el concepto de óptimo tiene otro enfoque que en el caso de la optimización global. La primera noción de optimalidad en un contexto multi-objetivo fue propuesta por Francis Ysidro Edgeworth [68] en 1881, sin embargo, fue Vilfredo Pareto en 1896 quien la generalizó [69].

Entre los conceptos más relevantes y que serán de utilidad para la presente tesis, encontramos los siguientes:

• **Dominancia de Pareto:** Suponiendo minimización, decimos que con dos vectores  $x \vec{1}, x \vec{2} \in F$ , donde  $F$  es la zona factible (o sea, la región del espacio de búsqueda donde se satisfacen todas las restricciones del problema), tenemos que  $x \vec{1}$  domina a  $x \vec{2}$  (denotado mediante  $x \vec{1} \prec x \vec{2}$ ) si y sólo si  $x \vec{1}$  es parcialmente menor que  $x \vec{2}$ . Es decir, en un problema con  $k$  funciones objetivo,

$$x \vec{1} \prec x \vec{2}$$

si y sólo si:

$$\vec{f}_i(\vec{x}_1) < \vec{f}_i(\vec{x}_2), \forall i \in [1, 2, \dots, k] \text{ y} \\ \exists j \in [1, 2, \dots, k], \text{ tal que } \vec{f}_j(\vec{x}_1) < \vec{f}_j(\vec{x}_2),$$

Es decir que para que una solución domine a otra, ésta necesita ser estrictamente mejor en al menos un objetivo, y no peor en ninguno de ellos. Esto es, al comparar dos soluciones A y B, sólo pueden existir tres posibles soluciones:

- A domina a B.
- A es dominada por B.
- A y B no se dominan (son no dominadas entre sí).

La dominancia de Pareto no impone un orden total en el espacio de las funciones objetivo ya que algunas soluciones pueden resultar incomparables. Por lo tanto, la mayoría de los problemas multi-objetivo no tienen una solución única, sino un conjunto de ellas. La figura 3.2 muestra gráficamente la dominancia de Pareto para un problema de optimización con dos objetivos en el cual  $A \preceq C$  tal que A es mejor en ambas funciones,  $A \preceq E$  tal que A es igual con respecto a  $f_2$  pero es mejor con respecto a  $f_1$ ,  $B \preceq E$  tal que B es igual con respecto a  $f_1$  pero mejor con respecto a  $f_2$ ,  $B \preceq D$  tal que B es mejor en ambas funciones, y A y B son incomparables (A y B son óptimos de Pareto).

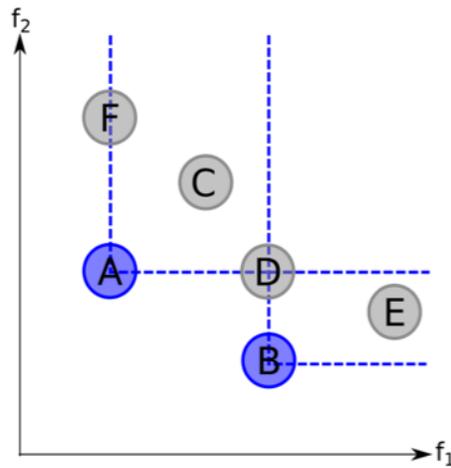


Figura 3.2: Dominancia de Pareto.

- **El conjunto de óptimo de Pareto ( $P^*$ ):** se define como:

$$P^* = \{ \vec{x}_* \in F \mid \neg \exists \vec{x} \in F \mid \vec{f}(\vec{x}) \preceq \vec{f}(\vec{x}_*) \}$$

Es decir, es el conjunto de vectores solución tal que no existe ningún otro vector dentro que domine a cualquiera de éstos. Los elementos del conjunto de óptimos de Pareto tienen vectores objetivo que no pueden ser mejorados sin

empeorar al menos otro objetivo [70]. Los vectores objetivo de las soluciones óptimas de Pareto se denominan vectores o soluciones *no dominadas*. Los vectores de las funciones objetivo correspondientes al conjunto de óptimos de Pareto conforman el denominado frente de Pareto ( $F^*$ ).

- **Frente de Pareto ( $F^*$ ):** Para un problema multi-objetivo determinado  $F(\vec{x})$ , siendo su conjunto de óptimos de Pareto ( $P^*$ ), el Frente de Pareto  $F^*$  o  $F_{real}$ , está definido como (ver figura 3.3):

$$F^* = \{ \vec{u} = \vec{f} = (f_1(\vec{x}_*), \dots, f_k(\vec{x}_*)) \mid \vec{x}_* \in P^* \}$$

Los frentes de Pareto pueden tener diferentes características geométricas tales como concavidad, convexidad, discontinuidades, etc.



Figura 3.3: Frente de Pareto para un problema de dos objetivos.

De esta manera, cuando un algoritmo utiliza la dominancia de Pareto para comparar soluciones, tiene como objetivo, al resolver un problema multi-objetivo, encontrar  $P^*$  y su correspondiente  $F^*$ . Ya que las soluciones a reportar por un algoritmo son finitas, aumenta la importancia de mantener una buena distribución de dichas soluciones. Dicha distribución, más la convergencia de los resultados, se vuelven los indicadores más importantes sobre la calidad de las soluciones reportadas por un algoritmo de optimización multi-objetivo.

Como se puede notar, el objetivo de la optimización multi-objetivo se vuelve complejo ya que no se puede dar mayor importancia a alguna solución no dominada en particular si

no se cuenta con información adicional del problema, ya que todas las soluciones que pertenezcan al conjunto de óptimos de Pareto tienen la misma importancia entre sí. Así, el objetivo principal se concentra en dos aspectos: primero en encontrar un conjunto de soluciones que sean parte o se encuentran muy cercanas al verdadero frente de Pareto, y el segundo aspecto es tener un conjunto de soluciones lo más diversas entre sí, es decir, que no se concentren en una sola parte del frente, sino que estén uniformemente distribuidas a lo largo de éste.

### 3.3 Algoritmos para optimización multi-objetivo

Hoy en día existen muchas técnicas de programación matemática para optimización multi-objetivo [67], sin embargo, la complejidad de muchos problemas de optimización multi-objetivo del mundo real vuelven a estas técnicas inadecuadas o incluso inaplicables para resolverlos. Uno de los problemas a los que se enfrentan éstas técnicas es que normalmente operan sobre un solo individuo a la vez, por lo que se necesitan ejecutar en varias ocasiones para poder encontrar el conjunto de óptimos de Pareto. En contraste, los algoritmos evolutivos tienen la ventaja de trabajar con una población (o conjunto de soluciones), esta característica les permite encontrar varias soluciones del conjunto de óptimos de Pareto en una sola ejecución. También, son menos sensibles a la forma o continuidad del frente de Pareto, además de que son fáciles de usar y no requieren información específica del problema a resolver.

#### 3.3.1 Técnicas tradicionales

Existen diversas técnicas evolutivas que se han propuesto para optimización multi-objetivo según la clasificación de técnicas multi-objetivo con algoritmos evolutivos(MOEA), propuesta por Cohon y Marks [71] se dividen en:

- **Técnicas *a priori*:** Las preferencias del usuario tienen que ser conocidas antes de comenzar la búsqueda. Se especifican las características de la solución deseada, pero, el inconveniente que presentan es que no siempre se pueden saber de antemano dichas características
- **Técnicas *progresivas*:** Las preferencias se van dando conforme la búsqueda avanza y el tomador de decisiones indica si una solución le parece adecuada o no y el proceso actualiza las preferencias conforme el tomador de decisiones lo va indicando, guiando así el proceso de búsqueda.
- **Técnicas *a posteriori*:** En estas técnicas, las preferencias se expresan al final y el tomador de decisiones recibe una información completa de los resultados para así entonces tomar la decisión que mejor le convenga. Es decir, los resultados intentan mostrar todos los compromisos posibles entre las funciones objetivo tratando de generar el verdadero frente de Pareto o al menos una aproximación razonablemente buena.

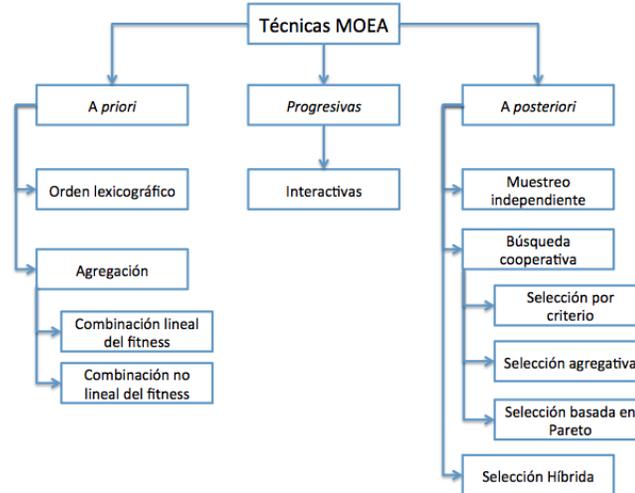


Figura 3.4: Clasificación de técnicas MOAE [87].

### 3.3.2 Algoritmos evolutivos multi-objetivo

La primera implementación formal que sentó las bases para los algoritmos evolutivos multi-objetivo (AEMO) fue propuesta por David Schaffer con su algoritmo genético de evaluación vectorial (VEGA) [72,73]. Posteriormente, surgieron otras propuestas que en general, han mostrado una clara superioridad en su desempeño con respecto a las técnicas clásicas.

Las características fundamentales que contiene un AEMO son [74]:

- Operan sobre un conjunto de soluciones potenciales al problema, lo cual les permite generar varios elementos del conjunto de óptimos de Pareto en una sola ejecución.
- Adoptan un mecanismo de selección cuyo objetivo es preservar las soluciones no dominadas de cada generación.
- Cuentan con un mecanismo que busca preservar la diversidad en la población (el denominado estimador de densidad).
- Cuenta con un mecanismo elitista que preserve las soluciones no dominadas globales.

Las tres últimas características hacen que el AEMO difiera al algoritmo evolutivo mono-objetivo. El mecanismo de selección de un AEMO debe incorporar información sobre las funciones objetivo del problema, lo cual dificulta el uso de un esquema tradicional, basado en aptitud. Para resolver dicho problema se tienen tres criterios principales [75]:

- **Ordenamiento lexicográfico:** Se toma en cuenta una sola función objetivo, y se optimiza en la mayor medida de lo posible. Posteriormente, se toma otra función objetivo y, a partir del resultado de la primera, se le optimiza manteniendo el valor obtenido previamente para los objetivos anteriores. Este procedimiento continúa hasta haber procesado todos los objetivos del problema.
- **Basados en agregación:** El conjunto de funciones objetivo es combinado para obtener una sola función escalar a optimizarse.
- **Basados en jerarquización de Pareto:** propuesta por Goldberg [66], la idea básica es encontrar las soluciones no dominadas y asignar una aptitud con base en la dominancia de Pareto. Estas técnicas son las más populares en la actualidad, por la importancia que tienen, se discuten en mayor detalle en la sección siguiente (ver figura 3.5).

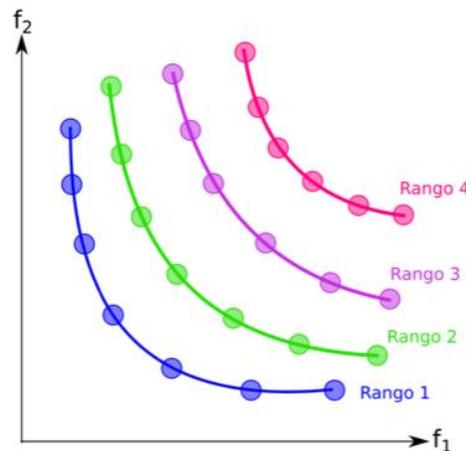


Figura 3.5: Jerarquización de Pareto en un problema con dos objetivos.

### 3.3.2.1 Algoritmos basados en la jerarquización de Pareto.

Goldberg [66] propuso utilizar la dominancia de Pareto como un criterio de selección para diseñar algoritmos evolutivos multi-objetivo. La idea principal es marcar a las soluciones que son no dominadas, asignarles una jerarquía más alta y retirarlos de la población. Este proceso continúa con las siguientes soluciones que son no dominadas y se les asigna otra jerarquía menos alta que la primera. Así, este proceso se realiza hasta que se ha asignado una jerarquía a cada individuo de la población [76]. También se sugirió el uso de nichos para evitar la convergencia a un solo punto.

A continuación se muestran los algoritmos más representativos que utilizan selección basada en una jerarquización de Pareto.

## Multiple Objective Genetic Algorithm (MOGA)

Fue propuesto por Fonseca y Fleming [77]. En este enfoque la jerarquía de un individuo depende del número de individuos en la población actual que lo dominan, es decir:

$$jerarquía(x_i, t) = 1 + p_i^{(t)}$$

donde  $p_i^{(t)}$  es el número de individuos que dominan a  $x_i$  en la generación  $t$ . Así entonces los individuos no dominados tendrán una jerarquía de 1. MOGA hace uso de nichos para distribuir uniformemente las soluciones, además usa restricciones a la cruce y otros mecanismos cuya motivación es evitar una convergencia prematura.

Las principales ventajas de MOGA son su eficiencia y que es fácil de implementar. Entre sus desventajas se encuentra su dependencia del parámetro que determina la distribución de aptitud entre individuos similares  $\sigma_{share}$ . Además de que no pueden existir al mismo tiempo dos vectores diferentes con los mismos valores para las funciones objetivo debido a que la distribución de aptitud se hace en el espacio de los valores de las funciones objetivo y no en el de las variables de decisión [78].

El pseudocódigo del funcionamiento de MOGA se puede ver en la figura 3.6.

---

Algoritmo 5: MOGA	
1:	Inicializar la población.
2:	Evaluar la aptitud de la población.
3:	Asignar una jerarquía basada en la dominancia de Pareto.
4:	Realizar el cómputo de los nichos.
5:	Asignar una aptitud lineal escalable.
6:	Asignar una aptitud de compartición.
7:	<b>Para i=1 hasta total generaciones</b>
8:	Seleccionar con muestreo estocástico
9:	Aplicar operador cruce de un punto.
10:	Aplicar operador de mutación.
11:	Calcular aptitud.
12:	Asignar una jerarquía basada en la Dominancia de Pareto.
13:	Realizar el cómputo de los nichos.
14:	Asignar una aptitud lineal escalable.
15:	Asignar una aptitud de compartición
16:	<b>Fin ciclo</b>

---

Figura 3.6: Funcionamiento del MOGA.

## Non-dominated Sorting Genetic Algorithm (NSGA)

Fue propuesto por Srinivas y Deb [79] y utiliza la idea original de Goldberg al jerarquizar la población por capas. Así entonces, los individuos que están en la primera capa tienen la máxima aptitud (por ser no dominados con respecto a toda la población) y se obtienen muchas copias de estos individuos con respecto al resto de la población. El NSGA

usa también compartición de aptitud para distribuir los individuos a lo largo del frente de Pareto.

Entre las principales ventajas de NSGA se encuentra que es muy eficiente debido a que objetivos múltiples se reducen a un valor temporal de aptitud el cual jerarquiza a la población en subconjuntos de individuos.

Cabe mencionar que a diferencia de MOGA, NSGA divide la aptitud de los individuos (nichos) en el dominio de las variables del problema (nivel fenotípico) con el objeto de asegurar una mejor distribución de los individuos y permitir la coexistencia de soluciones equivalentes.

### **Non-dominated Sorting Genetic Algorithm II (NSGA-II)**

Fue propuesto por Deb et al. [80]. Esta es una nueva versión del NSGA más eficiente (computacionalmente hablando) además de que se agrega elitismo y un operador que ayuda a mantener la diversidad sin requerir de parámetros adicionales. Emplea una jerarquización de Pareto y un mecanismo de preservación de la diversidad basada en el agrupamiento.

El NSGA-II estima la densidad de las soluciones alrededor de una solución particular en la población, calculando la distancia promedio hacia los dos puntos a los lados de cada objetivo del problema. Este valor es llamado *distancia de agrupamiento*.

Durante la selección, el NSGA-II utiliza un comparador que toma en consideración tanto la jerarquía de Pareto de un individuo, como su distancia de agrupamiento. De esta manera, se prefieren las soluciones no dominadas, y como criterio de desempate se toma la que se encuentre en la región menos densa del espacio de las funciones objetivo.

El NSGA-II no utiliza un archivo externo como otros algoritmos. Sin embargo, cuenta con un mecanismo elitista que consiste en combinar los mejores padres con los mejores hijos, tal como lo hace la selección ( $\mu + \lambda$ ).

Este algoritmo se ha vuelto tan popular que es altamente citado y la gran mayoría de los nuevos algoritmos de optimización se comparan contra él para validar su desempeño. Sin embargo, el desempeño propio del NSGA-II se degrada rápidamente al aumentar el número de objetivos del problema [81].

El pseudocódigo y el funcionamiento de NSGA-II se puede ver en la figura 3.7 y 3.8.

---

**Algoritmo 6: NSGA-II**

---

- 1: Generar aleatoriamente una población inicial.
  - 2: Evaluar la aptitud de la población.
  - 3: Asignar un nivel basado en la dominancia de Pareto -"ordenar".
  - 4: Generar la población siguiente:
  - 5:     Seleccionar mediante torneo binario
  - 6:     Aplicar cruce y mutación.
  - 7:     **Para  $i=1$  hasta total generaciones**
  - 8:     Para la población padre e hijo
  - 9:     Asignar un nivel basado en dominancia de Pareto y ordenar.
  - 10:     Generar el conjunto de frentes no dominados.
  - 11:     Sumar soluciones a la siguiente generación, empezando por la primera jerarquía y utilizar el factor de agrupamiento (crowding) en cada frente.
  - 12:     Seleccionar los puntos en el frente más bajo y que estén fuera de la distancia del factor de agrupamiento
  - 13:     Crear la siguiente generación
  - 14:     Seleccionar mediante torneo binario.
  - 15:     Aplicar cruce y mutación.
  - 16:     **Fin ciclo**
- 

Figura 3.7: Funcionamiento del NSGA-II

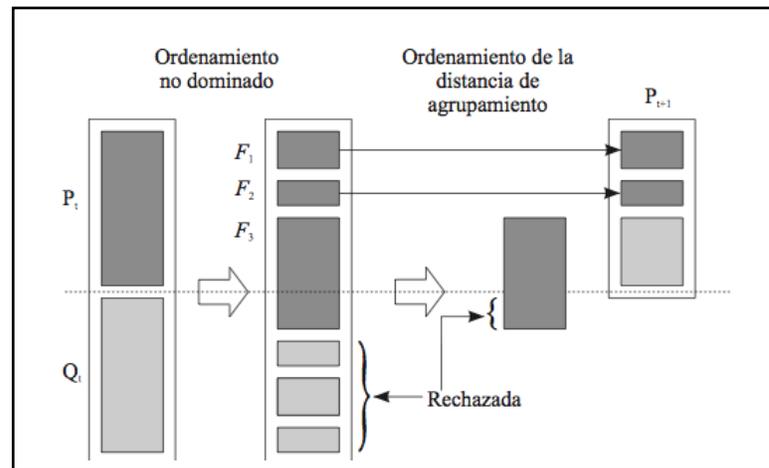


Figura 3.8: Procedimiento del NSGA-II.

### Pareto Achieved Evolution Strategy (PAES)

Fue propuesta por Knowles y Corne [82]. PAES consiste en una estrategia evolutiva de dos miembros (en la que un padre genera un solo hijo) en combinación con un archivo externo el cual almacena los individuos no dominados generados a lo largo del proceso evolutivo. Así, cuando se encuentra un individuo no dominado, se compara con los individuos del archivo externo, y en caso de que sea no dominado con respecto a éstos, una

La malla adaptativa se encarga de elegir a los individuos que salen del archivo externo. La malla ayuda a mantener diversidad y a distribuir uniformemente las soluciones no dominadas producidas (ver figura 3.9). Además no requiere parámetros y su complejidad computacional es menor que la de los nichos.

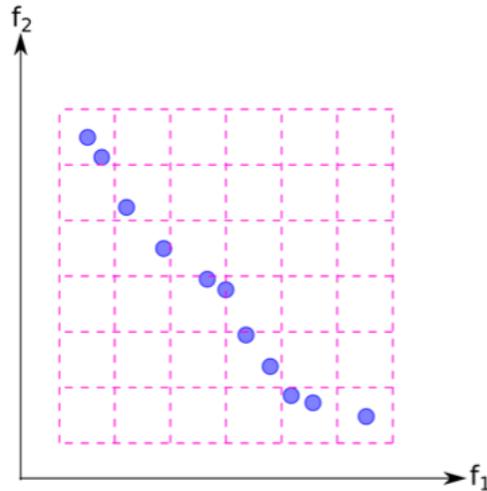


Figura 3.9: Malla adaptativa de PAES con 6 divisiones en un problema con dos objetivos.

### **Strength Pareto Evolutionary Algorithm (SPEA)**

Fue propuesto por Zitzler y Thiele [83]. Este algoritmo hace uso de un archivo externo que contiene las soluciones no dominadas del problema. A cada generación, los individuos no dominados se copian al archivo externo y se les asigna un valor extra el cual es proporcional al número de individuos que lo dominan. De tal forma, la aptitud de cada individuo se calcula de acuerdo con este valor extra de todos los individuos del archivo externo que lo dominen. Además, se adopta una técnica de cúmulos para mantener diversidad.

### **Strength Pareto Evolutionary Algorithm II (SPEA-2)**

Fue propuesto por Zitzler et al. [84]. Tiene tres diferencias principales con respecto al anterior:

- Para calcular la aptitud de los individuos se toman en cuenta la cantidad de individuos que lo dominan, y la cantidad de individuos que son dominados por él.
- Utiliza una densidad vecinal que guía la búsqueda más eficientemente.
- Usa un esquema del truncamiento del archivo externo para garantizar la preservación de las soluciones en los extremos.

El pseudocódigo del funcionamiento de SPEA-2 se puede ver en la figura 3.10.

---

**Algoritmo 7: SPEA-2**

---

- 1: Generar aleatoriamente una población inicial P.
  - 2: Crear un archivo externo vacío E.
  - 3: **Para i=1 hasta total generaciones**
  - 4:   Evaluar la aptitud de cada individuo en P y E.
  - 5:   Copiar todos los individuos no dominados de P y E en E.
  - 6:   Usar operador de truncamiento para quitar elementos de E si el tamaño de E es muy grande.
  - 7:   Si el tamaño del archivo no se ha excedido entonces usar individuos dominados en P para llenar E.
  - 8:   Seleccionar con torneo binario con reemplazo.
  - 9:   Aplicar cruce y mutación a los seleccionados
  - 10: **Fin ciclo**
- 

Figura 3.10: Funcionamiento de SPEA-2.

### **Pareto Envelope-based Selection Algorithm (PESA, PESA-2)**

Fue propuesto por Corne et al. [85], este algoritmo combina los aspectos positivos de SPEA y PAES. Al igual que SPEA, PESA considera dos poblaciones (Usa una población interna pequeña y una gran población externa.). Emplea una división del hyper-grid del fenotipo (o sea la función objetivo) para mantener la diversidad. La selección se basa en una medida de agrupamiento de los grids, con la que se decide cuál solución se colocará en la población externa que almacenará las soluciones no dominadas encontradas a lo largo del proceso. La versión extendida de PESA [86] es llamada PESA-2, ésta usa selección basada en hiperboxes en vez de individuos. Los hiperboxes en el espacio objetivo son seleccionados basándose en el número de soluciones que residen en los hiperboxes. Después de que se seleccionan, una solución azar de los hiperboxes elegidos se mantiene. Este procedimiento de selección basado en regiones ha mostrado un mejor desempeño que el procedimiento de selección basado en individuos de PESA.

### **3.4 Métricas de desempeño para AEMO.**

Para poder realizar una comparación entre los resultados obtenidos por distintos algoritmos multi-objetivo, se han ideado métricas las cuales ayudan a determinar la eficiencia con la que éstos resuelven los problemas multi-objetivo.

Como se había mencionado anteriormente, la optimización multi-objetivo debe de alcanzar dos metas particulares:

- Encontrar las soluciones que se encuentren en el frente óptimo de Pareto.
- Y que éstas soluciones se encuentren bien distribuidas en dicho frente.

Suponiendo que se tienen dos conjuntos de soluciones no dominadas obtenidas por diferentes AEMO para un mismo problema (ver figura 3.11). El algoritmo 1 obtiene soluciones que convergen perfectamente al frente óptimo de Pareto, pero carecen totalmente de diversidad, en caso contrario, en el algoritmo 2 se obtienen soluciones con una buena diversidad, pero sin converger sobre el frente óptimo.

Se puede decir entonces, que un AEMO es bueno si las metas mencionadas anteriormente se satisfacen adecuadamente. Ahora bien, debido a que se requiere evaluar el desempeño de los algoritmos tomando en cuenta éstas dos metas, una sola métrica es incapaz de medir tal desempeño, por lo tanto, existen diferentes métricas que de manera separada, puede medir tanto la cercanía al frente como la diversidad [80].

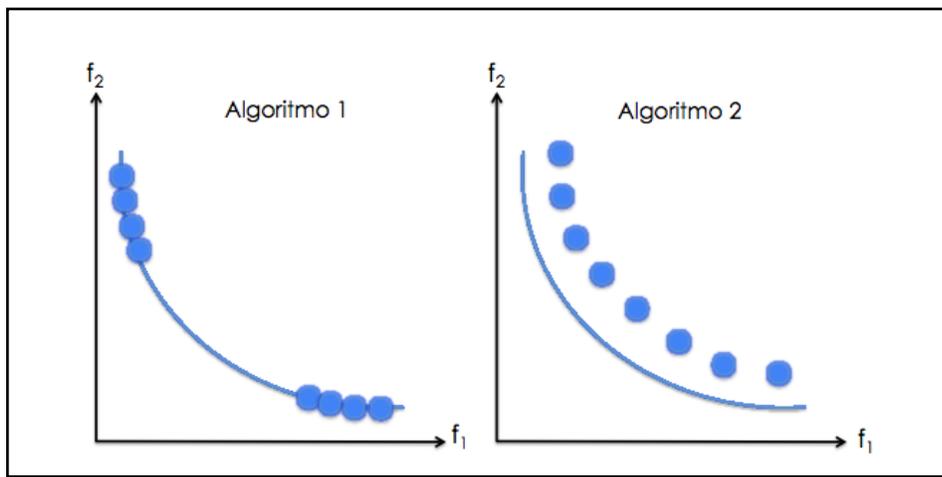


Figura 3.11: Cercanía y diversidad al frente óptimo de Pareto.

Para poder realizar una comparación de los dos algoritmos estudiados en esta tesis, se utilizan tres métricas: dos unarias y una binaria. Las métricas unarias, requieren solamente un conjunto de vectores no dominados, a diferencia de las métricas binarias, las cuales requieren dos conjuntos de vectores no dominados, pertenecientes a cada uno de los algoritmos a comparar. A continuación se describen cada una de ellas:

### 3.4.1 Distancia Generacional (GD)

La distancia generacional (GD) propuesta por Van Veldhuizen y Lamont [88, 89] es una métrica unaria que indica qué tan lejos o cerca en promedio están los puntos obtenidos por un algoritmo multi-objetivo ( $PF_{know}$ ) del frente de Pareto verdadero ( $PF_{true}$ ). Matemáticamente se expresa de la siguiente forma:

$$GD = \frac{(\sum_{i=1}^n d_i^p)^{\frac{1}{p}}}{n}$$

donde  $n$  es el número de soluciones generadas por el algoritmo ( $PF_{know}$ ),  $p = 2$ , y  $d_i$  es la distancia euclidiana entre cada vector y el individuo más cercano al verdadero frente de Pareto ( $PF_{true}$ ). Un resultado de 0 indica que se alcanzó el verdadero frente de Pareto, es decir,  $PF_{true} = PF_{know}$  y cualquier otro valor nos indica qué tan lejos se encuentra del mismo.

### 3.4.2 Cobertura de dos conjuntos (CS)

Fue propuesta en [90], ésta métrica compara la cobertura entre dos conjuntos de soluciones utilizando la dominancia de Pareto. Al igual que GD es una métrica para evaluar la cercanía al frente óptimo de Pareto, con la diferencia de es de tipo binaria. Se considera  $X'$  y  $X''$  como dos conjuntos de vectores. CS es definido como un mapeo del par ordenado  $(X', X'')$  en el intervalo de  $[0,1]$ . Matemáticamente se define como:

$$CS(X', X'') = \frac{|\{a'' \in X''; \exists a' \in X': a \succ a''\}|}{|X''|}$$

Si todos los puntos en  $X'$  dominan o son iguales a todos los puntos en  $X''$ , entonces por definición  $CS=1$ .  $CS=0$  en caso contrario. Ésta no es una métrica de distancia para saber qué tan cerca están los conjuntos unos del otro. Cuando  $CS(X', X'')=1$  y  $CS(X'', X')=0$ , se dice que  $X'$  es mejor que  $X''$ . En la figura 3.12, se observa que el conjunto  $X'$  domina a todos los puntos de  $X''$  por lo tanto, el conjunto  $X'$  es mejor que el conjunto  $X''$  y el valor de  $CS(X', X'')$  es igual a 1.

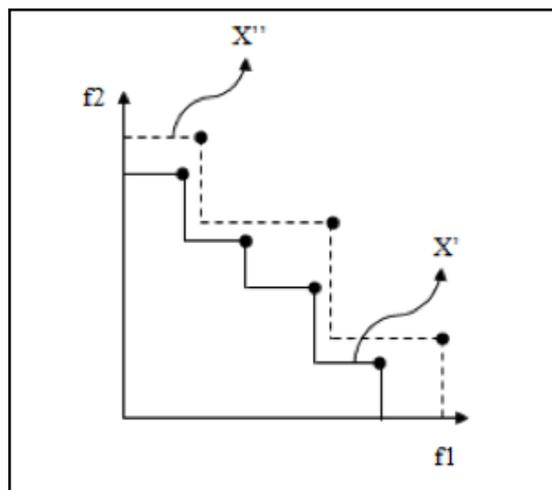


Figura 3.12: Cobertura de dos conjuntos.

### 3.4.3 Hipervolumen (HV)

Es una métrica unaria, propuesta originalmente por Zitzler and Thiele en 1998 [91], quienes lo definen como “el tamaño del espacio cubierto o del espacio dominado”. Esta métrica mide el volumen (más de 3 objetivos) o área (2 objetivos) del espacio objetivo que es cubierto por el conjunto de soluciones y que está limitado por un punto de referencia  $y_{ref}$ . Por ejemplo, para hacer el cálculo del hipervolumen para dos funciones objetivo, se define como el área de cobertura del  $PF_{know}$  en relación con el espacio objetivo de dichas funciones. Esto equivale a la suma de todas las áreas rectangulares, delimitadas por un punto de referencia  $(f_1(x), f_2(x))$  (ver figura 3.13). Según Coello et al. , matemáticamente, se describe como [92]:

$$HV = \bigcup_I \{vol_i | vec_i \in PF_{know}\}$$

Este indicador tiene dos ventajas importantes, las cuales son [93]:

- 1: Es sensible a cualquier tipo de mejora, es decir, cuando se calcula el hipervolumen para un conjunto  $X'$  de soluciones, que domina a otro conjunto  $X''$  de soluciones, entonces el aporte del hipervolumen será de mayor calidad para el primer conjunto que para el segundo.
- 2: Como resultado de la primera propiedad, el hipervolumen garantiza que, cualquier aproximación al conjunto  $X'$  de soluciones que alcanza el valor máximo de calidad posible para un problema particular contiene todo el conjunto de óptimos de Pareto.

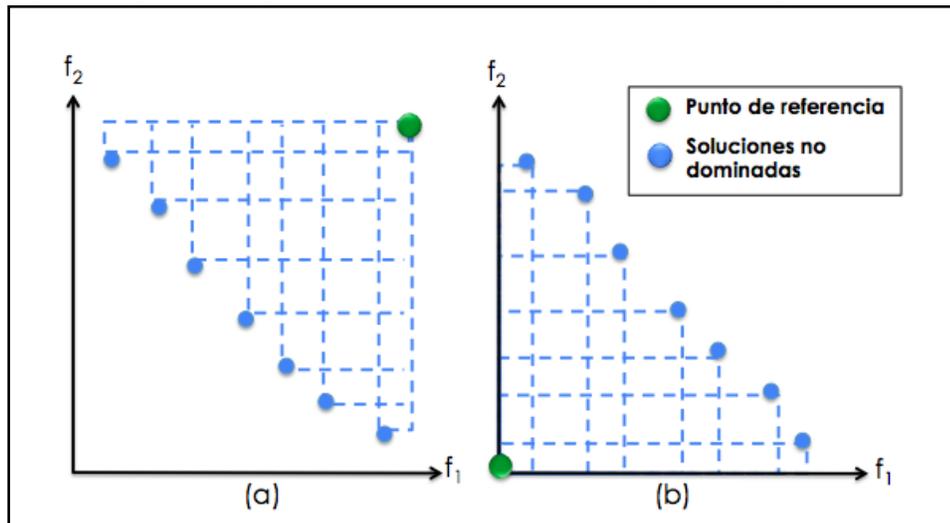


Figura 3.13: (a) Hipervolumen que se centra en un punto de referencia dado. (b) Hipervolumen que se centra en el origen.

# Capítulo IV: Clustering Multi-objetivo

## 4.1 Clustering Multi-objetivo

Para poder comprender, como se aplica la optimización multi-objetivo al proceso de agrupamiento, lo primero es entender cuál es el objetivo del mismo. El agrupamiento de datos (clustering) consiste en la clasificación de objetos de acuerdo a similitudes entre ellos, para que después se organicen los datos en grupos. Las técnicas de clustering están incluidas entre los métodos de aprendizaje no supervisado, debido a que no utilizan conocimiento de identificadores de clases. Por lo anterior, se puede decir, que un cluster es un grupo de objetos, los cuales son más similares entre sí que los miembros de otros clusters. Estos grupos ofrecen una posible clasificación o categorización de los elementos.

Dentro del área de aprendizaje automático, existen diferentes algoritmos de clustering, los cuales realizan la tarea de agrupamiento de datos, sin embargo, los algoritmos de clustering tradicionales no son útiles cuando se ha de optimizar más de un objetivo y, en estos casos, es necesario aplicar otro tipo de métodos.

En el Clustering multi-objetivo se evalúa cada objetivo de forma simultánea para cada solución de agrupamiento [30, 94]. La solución final es un conjunto de soluciones de clustering representadas en un frente de Pareto, donde cada solución tiene un equilibrio diferente entre los objetivos que han sido optimizados, por lo tanto, se obtienen varias soluciones de agrupamiento con ajustes diferentes de los objetivos. Por ejemplo, en la figura 4.1 (b) se muestra el conjunto de Pareto obtenido aplicando clustering multi-objetivo para el ejemplo de la figura 4.1 (a) en donde se optimizan de manera simultánea las distancias intra-cluster e inter-cluster. Como se puede observar, cada uno de los ejes representa un objetivo a optimizar. En el conjunto de Pareto, todas las soluciones son no dominadas, esto significa que no existe una solución que sea peor que las otras para ambos objetivos.

El concepto de clustering multi-objetivo se introdujo por primera vez en el año 1992 [94]. En 2004, Handl y Knowles proponen un algoritmo de clustering multi-objetivo denominada VI-ENNA[95], el cual más adelante es mejorado, obteniéndose el algoritmo MOCK [96,30], algoritmo tomado como base para el presente trabajo. MOCK es uno de los algoritmos más conocidos de clustering multi-objetivo que realiza agrupación particional. La optimización multi-objetivo se basa en el algoritmo evolutivo PESA-2.

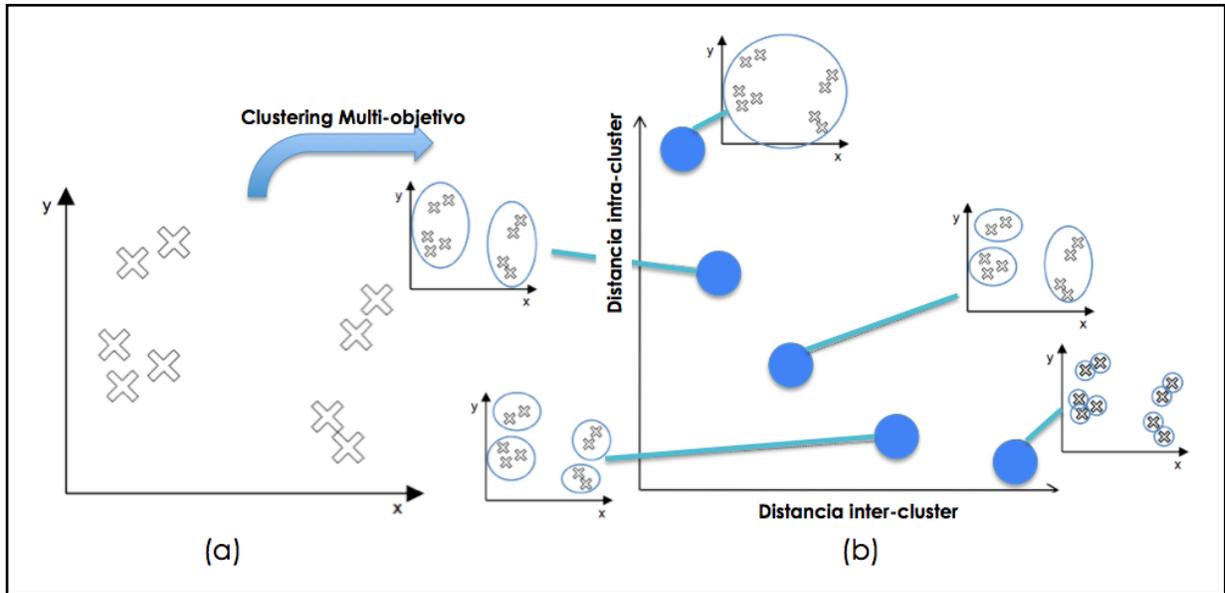


Figura 4.1: (a) Elementos a agrupar. (b) Conjunto de soluciones de Pareto Clustering Multi-objetivo.

## 4.2 Clustering Multi-objetivo con determinación automática de K (MOCK)

Tal como fue mencionado anteriormente, MOCK (Multiobjective clustering with automatic determination of de numer of clusters) es considerado el primer algoritmo para la realización de agrupamiento multi-objetivo, dicho algoritmo propuesto por Julia Handl y Joshua Knowles en el 2004, realiza el agrupamiento tomando en cuenta dos objetivos complementarios basados en la conectividad y la compactación de los grupos formados, la optimización multi-objetivo empleada por MOCK permite obtener un conjunto de soluciones óptimas, en donde cada una de ellas representa un posible agrupamiento de los datos, cada solución encontrada puede presentar un valor de  $k$  distinto, es decir con diferente número de clusters. La determinación automática del valor de  $k$  es una gran ventaja de MOCK, debido a que en la mayoría de los algoritmos de clustering tradicionales se necesita especificar éste valor, por lo que la solución obtenida depende en gran medida del mismo. La optimización en MOCK se realiza con el AEMO PESA-2. Para poder aplicar el PESA-2, es necesario definir una codificación genética adecuada que represente un posible agrupamiento, un o más operadores de variación (cruza y mutación) y más de dos objetivos a optimizar (ver figura 4.2). A continuación se describe más a detalle el funcionamiento de MOCK, así como su representación y los operadores que usa.

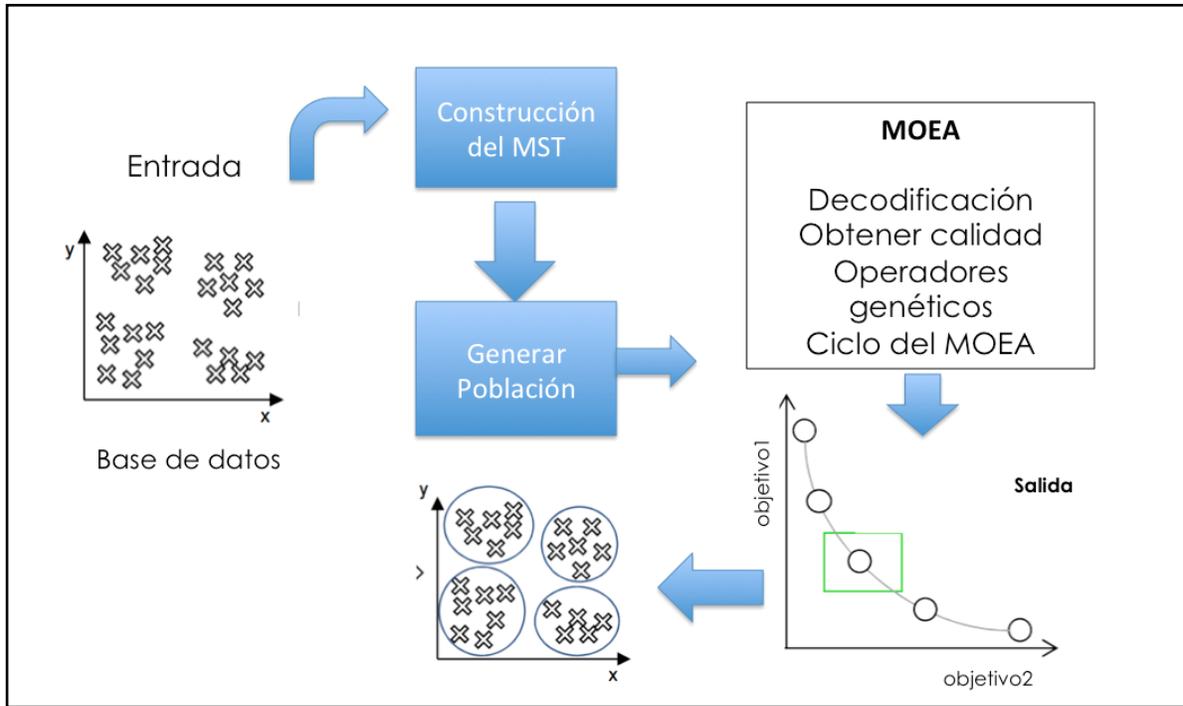


Figura 4.2: Estructura general del MOCK.

#### 4.2.1 Inicialización y representación

Para la codificación de los individuos se emplea una representación que hace uso de grafos, propuesta en [97]. En dicha representación, cada individuo  $g$  consiste en  $N$  genes  $g_1, g_2, \dots, g_N$ , en donde  $N$  es la cantidad de datos que se van a agrupar, y cada gen  $g_i$  puede tomar valores en su alelo  $j$  en un rango de  $\{1, \dots, N\}$ . Por lo tanto un valor  $j$  asignado al  $i$ -ésimo gen, es interpretado como un vínculo entre los datos  $i$  y  $j$ : hablando en términos de clustering, esto significa que los datos  $i$  y  $j$  se encontrarán en el mismo grupo (ver figura 4.3 (a) y (b)). La decodificación de esta representación requiere la identificación de todos los sub-grafos, lo cual se traduce a cada uno de los grupos de la solución final de agrupamiento. Todos los datos que pertenezcan al mismo sub-grafo son asignados a un cluster. La representación utilizada, tiene la ventaja de que no se necesita determinar el número de clusters que se quieren formar, éste proceso se lleva a cabo automáticamente durante el proceso de decodificación, por lo tanto se pueden encontrar soluciones con diferentes números de clusters en una sola ejecución del algoritmo. Además, la representación hace posible la aplicación de los operadores de variación de manera sencilla. Para la inicialización, primeramente se calcula el árbol de expansión mínima (MST) que se puede formar con todo el conjunto de datos, éste árbol se forma usando el algoritmo de Prim [98]. El primer individuo de la población será aquel que se obtenga al aplicar el algoritmo para encontrar el árbol de expansión mínima. El  $i$ -ésimo individuo de la población se forma quitando los  $(i-1)$  links más largos existentes entre los datos, cuando se elimina un link, en el genotipo la correspondencia entre el gen y el valor del alelo es la

misma en la figura 4.3 (b) se muestra el tercer individuo de la población inicial. Este esquema de inicialización permite que se obtenga una población inicial con soluciones que presenten diferente valor de  $k$ .

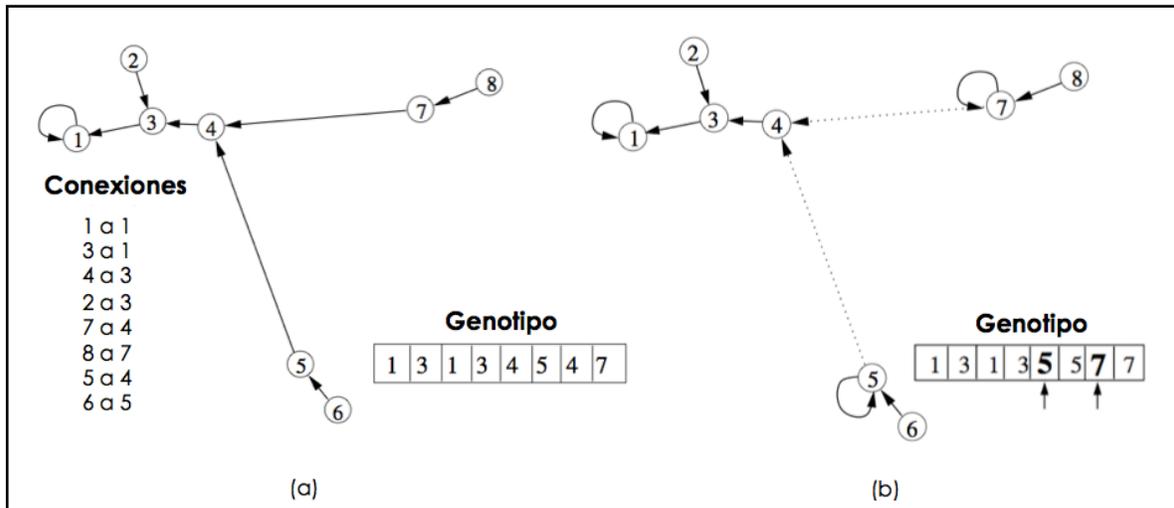


Figura 4.3: (a) Inicialización de individuos con MST, primer individuo. (b) Sub-grafos obtenidos por MOCK, tercer individuo.

## 4.2.2 Operadores de variación.

### 4.2.2.1 Operador de cruza.

Para la recombinación en MOCK se utiliza la cruza uniforme, en lugar de la cruza de un punto o dos puntos, esto es debido a que es imparcial con respecto a la ordenación de genes y puede generar cualquier combinación de alelos a partir de los dos padres (en un solo suceso de cruce). Un ejemplo de cruza uniforme aplicado al problema en cuestión se puede observar en la figura 4.4 en donde se tienen dos padre A y B con su correspondiente genotipo y generan al hijo C.

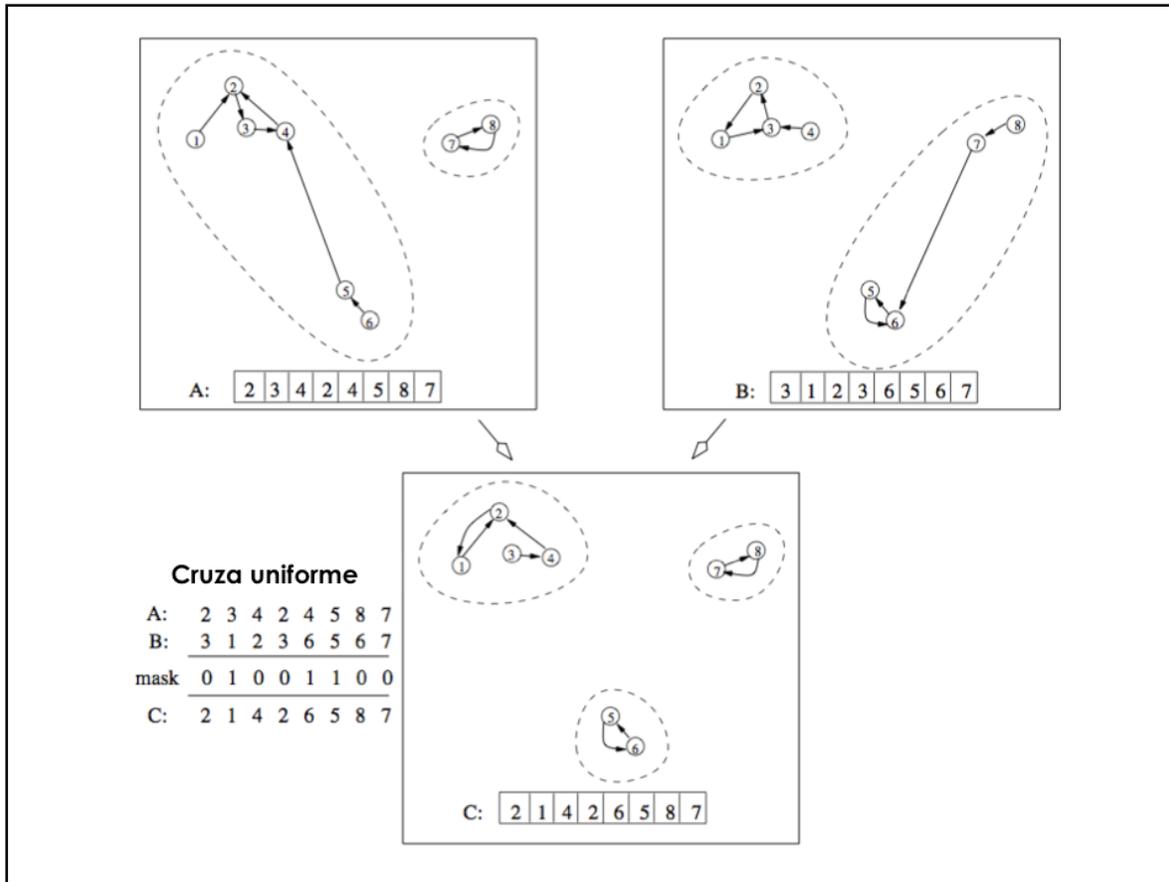


Figura 4.4: Ejemplo cruce uniforme en MOCK.

#### 4.2.2.2 Operador de mutación

Debido a que el espacio de búsqueda es muy grande con  $N^N$  posibles combinaciones, MOCK utiliza un operador de mutación especial donde cada dato sólo puede ser ligado a uno de sus  $L$  vecinos más cercanos. Por lo tanto,  $g_i \in \{nn_{i1}, \dots, nn_{iL}\}$ , donde  $nn_{iL}$  denota al  $l$ -ésimo vecino más cercano del dato  $i$ . Esto reduce el espacio de búsqueda a  $L^N$ . Para poder aplicar el operador antes descrito, es necesario hacer un cálculo previo de la lista de vecinos más cercanos de cada dato, el número  $L$  de vecinos, será un parámetro que el usuario debe ingresar a MOCK. La mutación entonces es un evento simple en donde se cambia el valor del alelo de una posición en un individuo (ver figura 4.5).

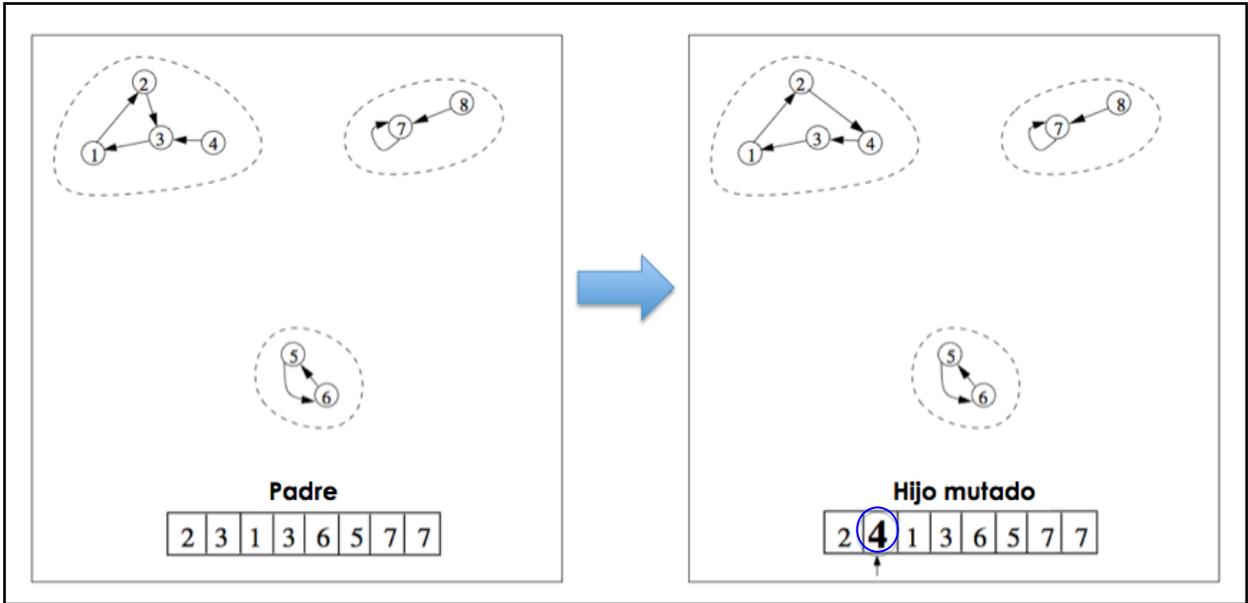


Figura 4.5: Ejemplo mutación de vecinos más cercano en MOCK.

### 4.2.3 Funciones objetivo

Para los objetivos a considerar para la optimización, en MOCK se selecciona un criterio que refleje una buena solución de clustering. Se seleccionaron dos objetivos complementarios: uno basado en la compactación y el otro en la conectividad de los clusters. Para expresar la compactación de los clusters, se calcula la desviación global del agrupamiento. Esta se obtiene sumando las distancias totales entre cada dato y su correspondiente centroide en un grupo determinado. Matemáticamente se puede expresar de la siguiente forma:

$$Dev(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k)$$

Donde  $C$  es el conjunto de todos los clusters,  $\mu_k$  es el centroide del cluster  $C_k$  y  $\delta(.,.)$  es una medida de distancia (distancia euclidiana). Como un objetivo, la desviación global puede ser minimizada. Este criterio es similar al criterio de varianza intra-cluster.

Para expresar la conectividad entre grupos, se usa una medida que evalúa el grado en que un vecino de un dato es colocado en el mismo cluster del dato actual. Matemáticamente se expresa de la siguiente manera:

$$Conn(C) = \sum_{i=1}^n \left( \sum_{j=1}^L x_{i,nn_i(j)} \right), \text{ donde } x_{r,s} = \begin{cases} \frac{1}{j} & \text{si } \exists C_k: r, s \in C_k \\ 0 & \text{otherwise} \end{cases}$$

$nn_i(j)$  es el  $j$ -ésimo vecino más cercano del dato  $i$ , y  $L$  es un parámetro que determina el número de vecinos que contribuyen a la medida de la conectividad. Este objetivo también debe de ser minimizado. Esta medida es similar al criterio de consistencia de vecino más cercano introducido por Ding et al. [99]. La diferencia radica en el uso de penaltis  $\frac{1}{j}$  donde Ding et al. usa una constante, esto permite dar más énfasis a los vecinos más cercanos, lo que garantiza una distinción más sutil entre la calidad de las soluciones de agrupamiento además de la obtención de grupos de tamaños significativamente más pequeños que  $L$ .

Como se puede observar, tanto para el operador de mutación como para calcular el objetivo de conectividad es necesario que previamente se implemente un método para encontrar los vecinos más cercanos.

Un aspecto importante en la elección de estas funciones objetivo es su potencial para balancear el número de clusters, es decir, aumentar o disminuir el número de clusters. Ya que mientras el valor objetivo asociado con la desviación global mejora con un aumento del número de grupos, lo contrario sucede para la conectividad. La interacción de los dos objetivos es importante con el fin de explorar las partes sensibles del espacio de la solución, y no converger a soluciones triviales.

# Capítulo V: Metodología

## 5.1 Metodología general

El algoritmo de clustering multi-objetivo (MOCK) descrito en el capítulo anterior, realiza la optimización usando el AEMO PESA-2, el objetivo de la presente investigación es incorporar a MOCK los AEMO NSGA-II y SPEA-2 para observar si mejora su desempeño obteniendo frentes de Pareto más prometedores. La metodología que se seguirá para realizar dicha tarea consta de 5 etapas principales:

- Se implementó la versión de MOCK con los algoritmos NSGA-II y SPEA-2.
- Implementación de métricas de desempeño de AEMO.
- Calibración de parámetros usando irace.
- Ejecución de pruebas con bases de datos reales de distinto tamaño.
- Elaboración de pruebas estadísticas.

### 5.1.1 Implementación de MOCK usando NSGA-II y SPEA-2.

Para la implementación de MOCK, se tomó como base el reporte técnico de Handl y Knowles [96]. La implementación fue realizada en Matlab, al igual que el algoritmo de MOCK original, se usó una representación entera para la codificación de los individuos, la inicialización fue realizada usando el algoritmo de expansión mínima, generando el  $i$ -ésimo individuo de la población inicial removiendo el link más largo encontrado en el grafo que representa al primer individuo de la población. Debido a que los algoritmos evolutivos involucran aleatoriedad, una vez que se genera una población de tamaño  $N$ , en donde  $N$  es el total del conjunto de datos a ser agrupados, se eligen al azar cierta cantidad de individuos de la población generada disminuyendo de esta manera el tamaño de la población.

Una vez que se generó la población inicial y se implementaron los operadores de mutación y cruce, así como también la función a optimizar tal como se realiza en MOCK, se hizo una adaptación de los algoritmos NSGA-II y SPEA-2, para ello se consideró dejar los mismos operadores que utiliza MOCK con el fin de poder realizar una comparación más directa entre los tres algoritmos. Cabe destacar que se consideraron dos restricciones, la primera se refiere a que el número de clusters en una solución se encuentre en un rango de 1 a 25, la segunda a que un grupo debe de estar formado por más de un elemento. El manejo de restricciones en problemas de optimización multi-objetivo ha sido menos abordado que en problemas mono-objetivo [102,103]. Sin embargo, existen propuestas competitivas, una de estas es la combinación de las reglas de factibilidad propuestas por Deb [104] con dominancia de Pareto para el ordenamiento no dominado [105]. En la

presente investigación se adoptó este criterio. Por lo tanto, para tomar en cuenta las restricciones, se hace una modificación a la selección por torneo que ocupan los algoritmos dándole prioridad a la elección de soluciones que no violen las restricciones, para después comparar soluciones no dominadas, esto se realiza tomando en consideración lo siguiente:

- a) Entre dos soluciones factibles, se prefiere a la que domina a la otra. Si ambas soluciones son factibles y no dominadas entre sí, entonces se elige una al azar.
- b) Entre una solución factible y una no factible, se prefiere a la factible.
- c) Entre dos soluciones no factibles, se elige a la que tenga un menor valor del total de restricciones violadas.

### **5.1.2 Calibración de parámetros usando irace.**

Un aspecto fundamental en el área de cómputo evolutivo consiste en encontrar una buena configuración de los valores de los parámetros para los algoritmos, esto con el fin de que se obtengan mejores resultados de la función objetivo que se trata de minimizar o maximizar con el algoritmo evolutivo. Debido a la importancia de los parámetros, es preferible probar los algoritmos con distintos valores de los mismos, en ocasiones estos valores son seleccionados de forma manual, sin embargo para la presente investigación se utiliza una herramienta que realiza la determinación automática de los parámetros, dicha herramienta consiste en un paquete del entorno R llamado irace (Iterated Racing for Automatic Algorithm Configuration). A continuación se describe brevemente aspectos relacionados con R y el funcionamiento de irace.

R es un entorno de software libre en el cual se puede programar, realizar cómputo estadístico (modelos lineales y no lineales, pruebas estadísticas clásicas, análisis de series de tiempo, clasificación, etc.) y representación gráfica. Se puede compilar y ejecutar en una amplia variedad de plataformas. R puede ser extendido fácilmente a través de paquetes. Al momento de instalar la distribución de R se tienen ciertos paquetes incluidos, sin embargo muchos más están disponibles a través de la familia CRAN del sitio de Internet del proyecto R [106].

Irace, es uno de los paquetes adicionales que pueden ser incorporados a R, implementa un procedimiento llamado iteración de carreras, el cual es una extensión de F-race (I/F-race) propuesto por Balaprakash, Birattari y Stützle [107] y más desarrollado por Birattari, Yuan, Balaprakash y Stützle [108]. El principal uso de irace es la configuración automática de los algoritmos de optimización, es decir, encuentra la configuración más apropiada de un algoritmo dado un conjunto de instancias de un problema de optimización. No obstante, los métodos de configuración automáticos son también aplicables a cualquier

sistema que tiene una serie de parámetros configurables, y cuyo rendimiento en un determinado problema a menudo depende de los ajustes particulares de estos parámetros. Un ejemplo de ello son los algoritmos evolutivos, ya que trabajan con distintos parámetros.

El paquete irace que se utiliza para la calibración de parámetros en la presente investigación ya ha sido ampliamente probado en varios proyectos. López Ibáñez y Stützle [109] configuran automáticamente los parámetros del algoritmo de colonia de hormigas. Montes de Oca, Aydin y Stützle [110] diseñaron un algoritmo de optimización de enjambre de partículas incremental para problemas de optimización continua a gran escala por medio de una configuración automática.

Tal como se mencionó anteriormente, irace utiliza un procedimiento llamado iteración de carreras, para este trabajo se usa la implementación de I/F-race la cual hace uso de un método conocido como análisis no paramétrico de dos vías de Friedman el cuál tendrá por objetivo encontrar los parámetros que minimicen o maximicen una determinada función (costo), usualmente para algoritmos evolutivos mono-objetivo esta corresponde al fitness (función de aptitud) del problema que se quiere resolver. El método de iteración de carreras para la configuración automática consiste en tres pasos básicos: (1) toma muestras de nuevas configuraciones de acuerdo con una distribución particular, (2) selección de las mejores configuraciones de los recién incluidos en la muestra por medio de las carreras, y (3) la actualización de las muestras con el fin de desviar el muestreo hacia las mejores configuraciones. Estos tres pasos se repiten hasta que se cumpla un criterio de terminación. En las carreras, cada parámetro configurable tiene una distribución de muestreo independiente, que es o bien una distribución normal para los parámetros numéricos, o una distribución discreta para parámetros categóricos. La actualización de las distribuciones consiste en modificar la distribución del muestreo, la media y la desviación estándar en el caso de la distribución normal, o los valores de probabilidad discretos en las distribuciones discretas.

Las carreras comienzan con un conjunto finito de candidatos, en cada uno de los pasos los candidatos son evaluados y se descartan a los peores, la carrera continúa con las configuraciones de supervivientes restantes. Este procedimiento continúa hasta alcanzar un número mínimo de configuraciones supervivientes o un criterio de paro predefinido que puede ser una serie de experimentos totales.

Ahora bien, el programa irace requiere tres entradas principales (Figura 5.1):

- Una descripción del espacio de los parámetros  $X$ , es decir, los parámetros que se deben configurar, sus tipos, rangos y restricciones.
- El conjunto de instancias  $\{I_1, I_2, \dots\}$ , el cual es finito y representativo de la muestra.
- La configuración propia de irace.

Además, irace requiere una función (o un programa auxiliar) llamado hookRun, que es responsable de la aplicación de una configuración particular de los parámetros y de devolver el correspondiente valor de costo.

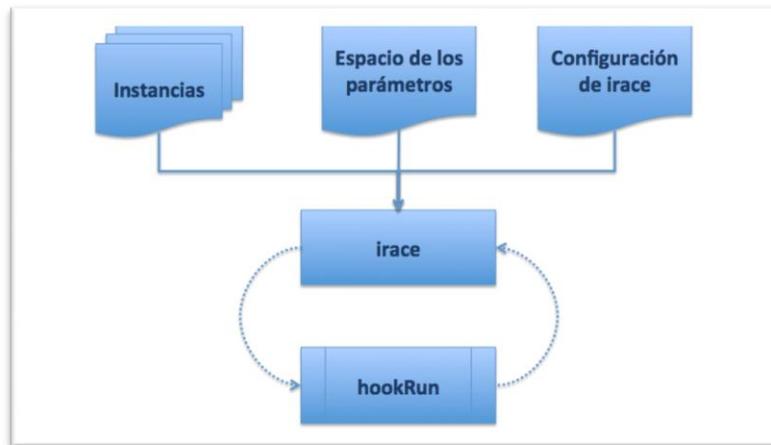


Figura 5.1: Esquema de flujo de información de irace.

### 5.1.2.1 Calibración de parámetros de MOCK.

Como ya fue mencionado, un aspecto fundamental en el área de cómputo evolutivo es encontrar una configuración adecuada de los parámetros de los algoritmos. Con el fin de llevar a cabo la calibración con irace, para el archivo del conjunto de instancias y la configuración propia de irace se tomaron como entradas los valores por default. En lo que respecta al espacio de los parámetros, se realiza la calibración de tres parámetros de MOCK: número de generaciones, L (cantidad de vecinos más cercanos) y probabilidad de cruza. El resto de los parámetros del algoritmo no fue considerado, ya que se tomaron los valores propuestos en el reporte técnico del MOCK original, esto debido a que son parámetros que dependen de la cantidad de instancias de cada base de datos a probar. El conjunto del espacio de los parámetros se muestra en la Tabla 5.1.

Tabla 5.1: Espacio de los parámetros para la calibración con irace.

<b>Parámetro</b>	<b>Tipo</b>	<b>Rango</b>
Generaciones	entero	(100 , 300)
Lvecinos	entero	(10, 30)
Probabilidad de cruza	real	(0,1)

El hookRun para este caso, es la versión del algoritmo de MOCK usando NSGA-II, se eligió esta versión, debido a que con ciertas pruebas preliminares realizadas, se observó que en la mayoría de los conjuntos de datos  $MOCK_{NSGA-II}$  se comportaba mejor que las otras

dos versiones. Cabe destacar que irace toma en cuenta un sólo valor (costo) para realizar la calibración de los parámetros, es decir, lleva a cabo dicha calibración tratando de minimizar el costo (función de aptitud) obtenido del hookRun. Sin embargo, en este caso tanto  $MOCK_{NSGA-II}$ , como las otras dos versiones son algoritmos Multi-objetivo, por lo tanto el resultado final de la ejecución de los algoritmos no es una única solución sino un frente de Pareto con un conjunto de soluciones lo cual implica que el algoritmo no regresa un valor correspondiente a la función de aptitud, por tal motivo se necesitó establecer un método para obtener un valor que pudiera ser utilizado por irace, para la realización de esto se utiliza la métrica de hipervolumen, el valor obtenido de dicha métrica representa el área de espacio cubierto por el frente obtenido con el algoritmo multi-objetivo, un valor más pequeño de hipervolumen implicaría que se tuvo un mejor acercamiento al frente óptimo de Pareto y por lo tanto se obtienen mejores soluciones. Entonces, el algoritmo ejecutable con el cual trabajará irace regresa el valor de hipervolumen y dicho valor debe ser minimizado tomando en consideración distintas configuraciones de parámetros.

### 5.1.3 Diseño experimental

Para probar el rendimiento de MOCK, se hacen pruebas en cinco bases de datos reales obtenidas del repositorio de Machine Learning [100]. Las características de cada una de ellas se detallan en la Tabla 5.2 en donde  $N$  es el número total de instancias en la base de datos,  $N_i$  es el número de items para el cluster  $i$ , y  $D$  y  $k$  se refieren al total de atributos y número de clusters, respectivamente. Como se puede observar, las bases datos son de diferente tamaño, número de atributos, tipos de datos, etc.

Para las bases de datos, se hizo un preprocesamiento que consistió en dos pasos sencillos: los valores faltantes fueron reemplazados por cero y los vectores de datos fueron normalizados, para tener media cero y desviación estándar de uno en cada dimensión (atributos).

Tabla 5.2: Resumen de las bases de datos usadas para pruebas.

Nombre	$N$	$N_i$	$D$	$k$	Tipo
<b>Dermatology</b>	366	112, 61, 72, 49, 52, 20	34	6	Entero
<b>Iris</b>	150	3 x 50	4	3	Continuo
<b>Wine</b>	178	59, 71, 48	13	3	Continuo
<b>Wisconsin</b>	699	485, 241	9	2	Entero
<b>Yeast</b>	1484	463, 429, 244, 163, 51, 44, 37, 30, 20, 5	8	10	Continuo

El rendimiento de los algoritmos implementados se evalúa de dos formas diferentes:

- a) Una comparación entre los algoritmos implementados y MOCK original, desde el punto de vista de la computación evolutiva, tomando en cuenta métricas para determinar la eficiencia con la que los algoritmos resuelven los problemas multi-objetivo.
- b) Una comparación contra algoritmos clásicos de agrupamiento desde el punto de vista de aprendizaje automático.

Los experimentos que se describen a continuación, se ejecutan con los valores de los parámetros descritos en el reporte técnico del MOCK original y con los valores que se obtuvieron al calibrar usando la herramienta irace.

#### **5.1.3.1 Primer experimento: Comparación entre algoritmos evolutivos.**

El algoritmo MOCK en su versión original ocupa a PESA-2 para la obtención de soluciones, la implementación de las otras versiones de MOCK cambiando los algoritmos evolutivos con los cuales trabaja es la parte fundamental de este trabajo, debido a ello se requiere hacer un comparativo del comportamiento de las tres versiones de MOCK ( $MOCK_{PESA-2}$ ,  $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$ ) para observar con cual AEMO el algoritmo MOCK presenta un mejor rendimiento, es decir, se obtienen soluciones más cercanas al frente de Pareto y mejor distribuidas. Para la realización de este comparativo el primer experimento consiste en la ejecución de cada uno de los algoritmos a comparar para las bases de datos de la Tabla 5.2, al terminar la ejecución se obtiene como resultado los frentes de Pareto generados por cada algoritmo, una vez obtenidos dichos frentes, se aplican las métricas de desempeño Hipervolumen y Two Set Coverage las cuales fueron descritas en el capítulo 3. Se reportaran resultados de estadísticas de los valores obtenidos para cada métrica llevando a cabo 10 ejecuciones independientes por algoritmo y base de datos.

#### **5.1.3.2 Segundo experimento: Comparación con otros algoritmos de clustering.**

Debido a que existen otros algoritmos de agrupamiento que han resultado ser eficientes, y con el fin de justificar que la solución de agrupamiento final al realizar un algoritmo de clustering multi-objetivo es mejor o bien comparable con algunos algoritmos tradicionales. El experimento dos consiste en la comparación de las tres versiones de MOCK ( $MOCK_{PESA-2}$ ,  $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$ ), un algoritmo de clustering jerárquico aglomerativo, el algoritmo k-means y un algoritmo meta-clustering. Estos tres últimos son descritos brevemente a continuación.

- Clustering jerárquico aglomerativo: También conocido como ascendente, es decir, comienza el análisis con tantos grupos como individuos haya. A partir de estas unidades iniciales se van formando grupos, de forma ascendente, hasta que al final

de las iteraciones todos los casos tratados se encuentran en un mismo cluster. En el experimento realizado con clustering jerárquico, se toma en cuenta la métrica average-link para ir realizando la fusión de instancias, la cual en cada paso permite que se unan los dos grupos tal que tienen la mínima distancia promedio entre sus puntos.

- K-means: La idea principal es definir k centroides (cantidad de grupos a formar) y luego tomar cada punto de la base de datos y situarlo en la clase de su centroide más cercano. El siguiente paso es recalcularse el centroide de cada grupo y volver a distribuir todos los objetos según el centroide más cercano. Para el experimento realizado, se le especifica a k-means que realice 100 iteraciones, y los centroides iniciales son seleccionados aleatoriamente.
- Meta-clustering: Este enfoque hace uso de multi clasificadores, es decir, generar las etiquetas finales combinando más de un algoritmo de clustering. Para el experimento que se realiza, se emplea el algoritmo K-means y clustering jerárquico usando la métrica average-link. Se ejecuta el algoritmo con valores de  $K=[2...20]$ .

Ahora bien, para realizar la comparación entre cada uno de los algoritmos antes mencionados, es necesario tomar en consideración alguna medida de evaluación de la calidad del clustering. La función de evaluación elegida es F-measure, esta es una función de evaluación externa la cual requiere que se tenga un conocimiento de las etiquetas de clase correctas para cada instancia de las bases de datos a analizar, esto es comparado con el modelo que es generado por un algoritmo de clustering determinado. Esencialmente la medida se basa en la idea de precisión y recuperación. Para el cálculo, se considera que  $n_{i,j}$  da el número de elementos de la clase  $i$  pertenecientes al cluster  $j$ . Para cada clase  $i$  y cluster  $j$  la precisión y recuperación son definidas como  $p(i,j) = \frac{n_{i,j}}{n_j}$  y  $r(i,j) = \frac{n_{i,j}}{n_i}$  y el valor correspondiente de F-measure se expresa como:

$$F(i,j) = \frac{(b^2 + 1) * p(i,j) * r(i,j)}{b^2 * p(i,j) + r(i,j)}$$

Donde  $b=1$ , para obtener una ponderación igual de  $p(i,j)$  y  $r(i,j)$ . El valor total de F-measure para una partición se calcula de la siguiente forma:

$$F = \sum_i \frac{n_i}{n} \max_j \{F(i,j)\}$$

Donde  $n$  es el tamaño total de la base de datos.  $F$  es un valor que se encuentra en el intervalo  $[0,1]$  y debe maximizarse. Por lo que un valor más cercano a 1 indica que los datos fueron clasificados de mejor manera.

Para obtener los resultados de F-measure de este experimento también se ejecutaron los algoritmos 10 veces. En este caso hay que considerar que los algoritmos de clustering multi-objetivo ( $MOCK_{PESA-2}$ ,  $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$ ) regresan muchas soluciones de agrupamiento, por lo que para cada ejecución independiente, se debe elegir una solución del frente de Pareto, al igual que en el MOCK original, la solución elegida para este experimento es aquella con el mayor valor de F-measure, lo cual implica la elección de una solución con la cual se realizará un mejor agrupamiento. Una vez que es seleccionada la solución, la medida F-measure, como ya se mencionó, considerará el modelo obtenido por dicha solución, es decir, el resultado de las etiquetas de clase para cada instancia de la base datos analizada. Este experimento tiene como fin reportar estadísticas de los resultados de F-measure para cada algoritmo.

# Capítulo VI: Resultados

En esta sección se muestran y discuten los resultados obtenidos por las dos versiones de MOCK implementadas con los algoritmos NSGA-II y SPEA-2, los cuales toman como base la versión original del MOCK que usa el AEMO PESA-2. Los resultados del rendimiento de las versiones implementadas son comparadas conforme a los experimentos descritos en el capítulo anterior, cabe destacar que se presentan resultados de 10 ejecuciones independientes.

Ya que el objetivo principal de esta tesis es hacer una comparación directa con MOCK en su versión original, primero se tomaron los mismos parámetros que usaron los autores al reportar sus resultados, lo mismo sucede con los operadores de variación. El valor de los parámetros de inicialización para los algoritmos en la primera prueba se muestran en la Tabla 6.1.

Tabla 6.1: Parámetros de inicialización de cada algoritmo, el símbolo - significa que para ese algoritmo no aplica.

<b>Parámetro</b>	<b>Generaciones</b>	<b>Población interna</b>	<b>Población externa</b>	<b>Prob. mutación</b>	<b>Prob. cruza</b>	<b>L</b>
<b>Algoritmo</b>						
MOCK <sub>PESA2</sub>	200	Max(50,N/20)	1000	1/N	0.7	20
MOCK <sub>NSGA-II</sub>	200	Max(50,N/20)	-	1/N	0.7	20
MOCK <sub>SPEA-2</sub>	200	Max(50,N/20)	Max(50,N/20)	1/N	0.7	20

Como fue detallado en el capítulo anterior, también se hizo la calibración de parámetros usando irace, los resultados obtenidos por irace dieron la configuración de parámetros que se presenta en la Tabla 6.2. Una vez que se obtuvieron estos resultados, se procedió a ejecutar los algoritmos con las configuraciones obtenidas. Cabe destacar que los parámetros que se calibraron son: Generaciones, L, Probabilidad de cruza, los demás parámetros fueron tomados de acuerdo al reporte técnico de MOCK, es decir, tanto la población interna, población externa y la probabilidad de mutación toman los mismos valores que en la Tabla 6.1 .

Tabla 6.2: Configuración de parámetros obtenidos por irace.

	<b>Parámetro</b>	<b>Valor</b>
<b>Primera configuración</b>	Generaciones	292
	L	22
	Probabilidad de Cruza	0.6

<b>Segunda configuración</b>	Generaciones	246
	L	20
	Probabilidad de Cruza	0.7

Los resultados que se muestran a continuación, se distribuyen de la siguiente manera:

1. *Resultados del primer experimento (Comparación entre algoritmos evolutivos):* Se presentan resultados obtenidos con las tres configuraciones de parámetros (valores originales de parámetros según reporte técnico, primera configuración obtenida por irace, segunda configuración obtenida por irace), para cada una de las configuraciones se muestran gráficas comparativas de los frentes de Pareto obtenidos, resultados de la métrica hipervolumen y resultados de la métrica Two set Coverage. Se discuten los resultados.
2. *Resultados del segundo experimento (Comparación con otros algoritmos de clustering):* Al igual que con el primer experimento, se presentan resultados con las tres configuraciones de parámetros para las versiones de MOCK. Para cada configuración se muestran tablas comparativas entre algoritmos.

## 6.1 Resultados del primer experimento

Para la comparación directa de los resultados obtenidos por los algoritmos implementados y el algoritmo original, de acuerdo al área de cómputo evolutivo, como ya fue mencionado anteriormente se lleva a cabo a través de las métricas de Hipervolumen y Two Set Coverage (Cobertura). A continuación se muestran los resultados obtenidos por dichas métricas, pruebas estadísticas para cada una y los frentes de Pareto que se obtienen después de la ejecución de los algoritmos.

En las Figuras 6.1- 6.15 se puede observar el comportamiento de los algoritmos ya que se muestran los frentes de Pareto obtenidos de los algoritmos  $MOCK_{PESA-2}$ ,  $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$  para cada de las bases de datos con las que se realizaron los experimentos. Debido a que se realizaron 10 corridas independientes, se graficó el Frente de la ejecución que se encuentra en la mediana. Se identifica como  $MOCK_{PESA-2}$  al algoritmo original.

Los frentes que se obtienen al ejecutar los algoritmos con los parámetros originales se muestran en las Figuras 6.1 a 6.5.

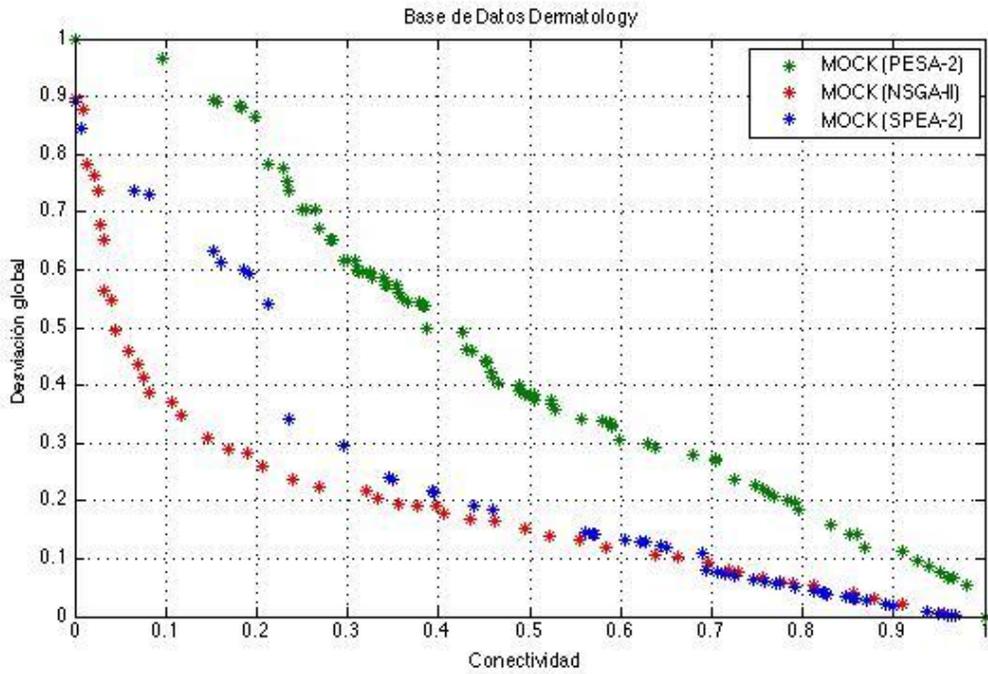


Figura 6.1: Frente de Pareto de cada algoritmo para Dermatology, con parámetros originales.

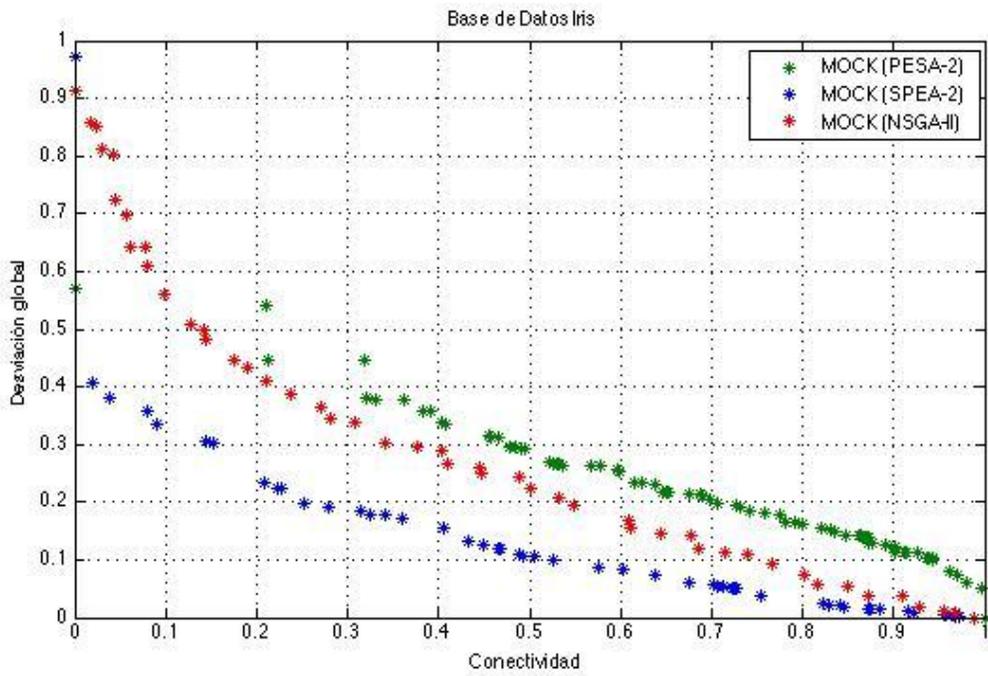


Figura 6.2: Frente de Pareto de cada algoritmo para Iris, con parámetros originales.

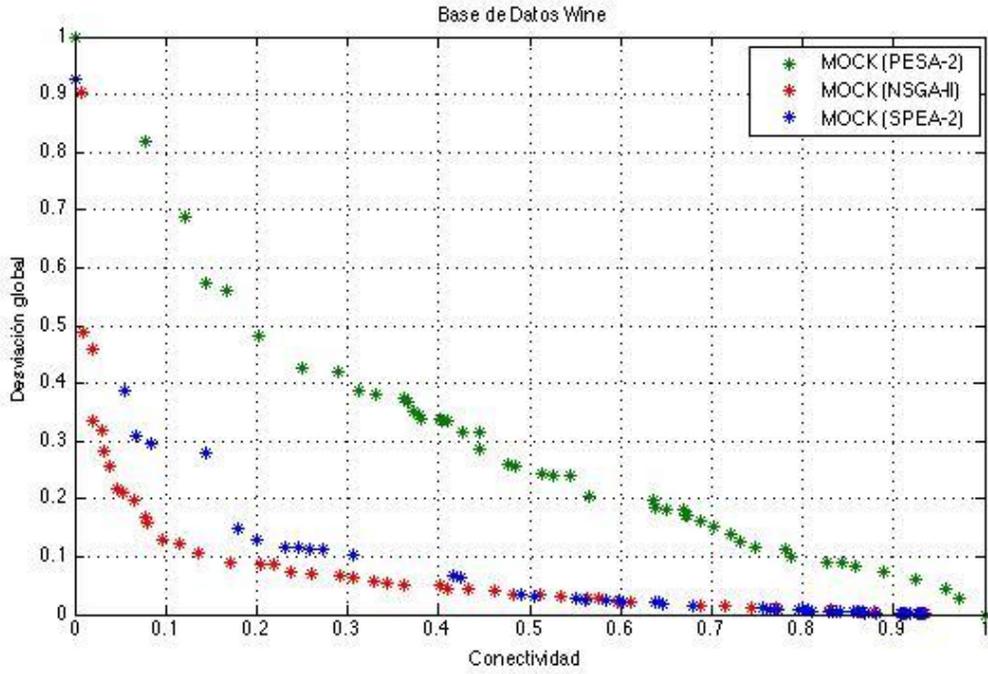


Figura 6.3: Frente de Pareto de cada algoritmo para Wine, con parámetros originales.

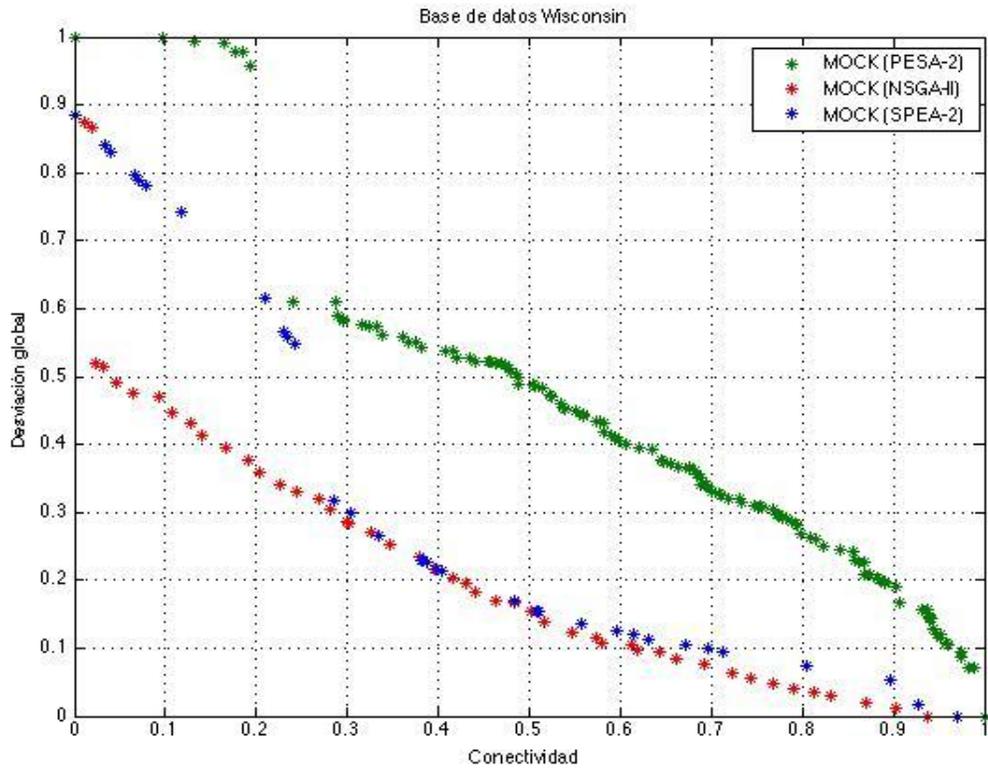


Figura 6.4: Frente de Pareto de cada algoritmo para Wisconsin, con parámetros originales.

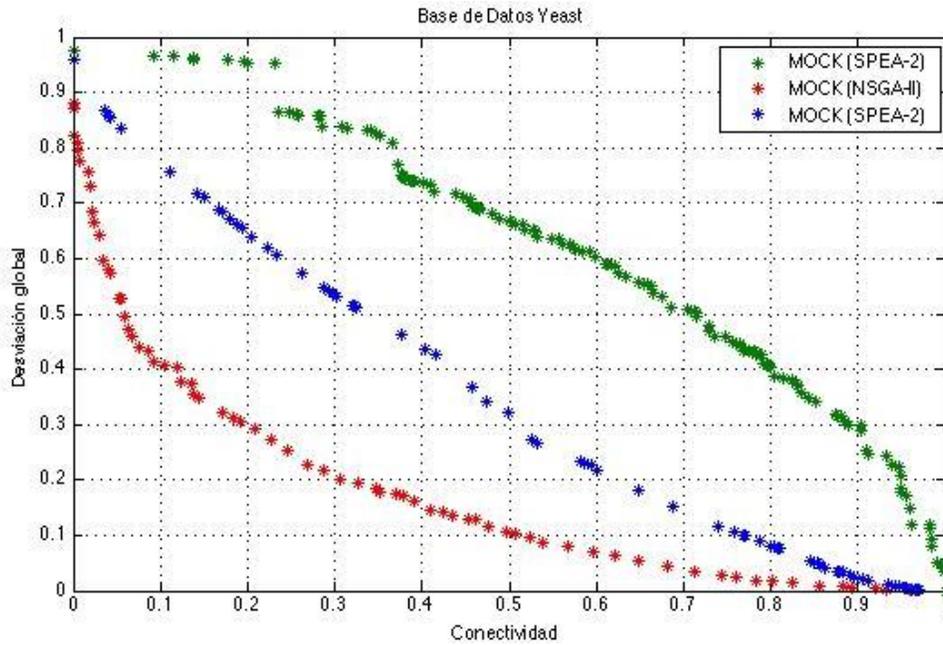


Figura 6.5: Frente de Pareto de cada algoritmo para Yeast, con parámetros originales.

En las Figuras 6.6 -6.10, se muestran los frentes obtenidos de la ejecución de los algoritmos según la primera configuración de parámetros que generó irace, es decir, cambiando el número de generaciones a 292, L igual a 22 y probabilidad de cruce de 0.6. Los tres algoritmos se ejecutan tomando en cuenta dichos valores.

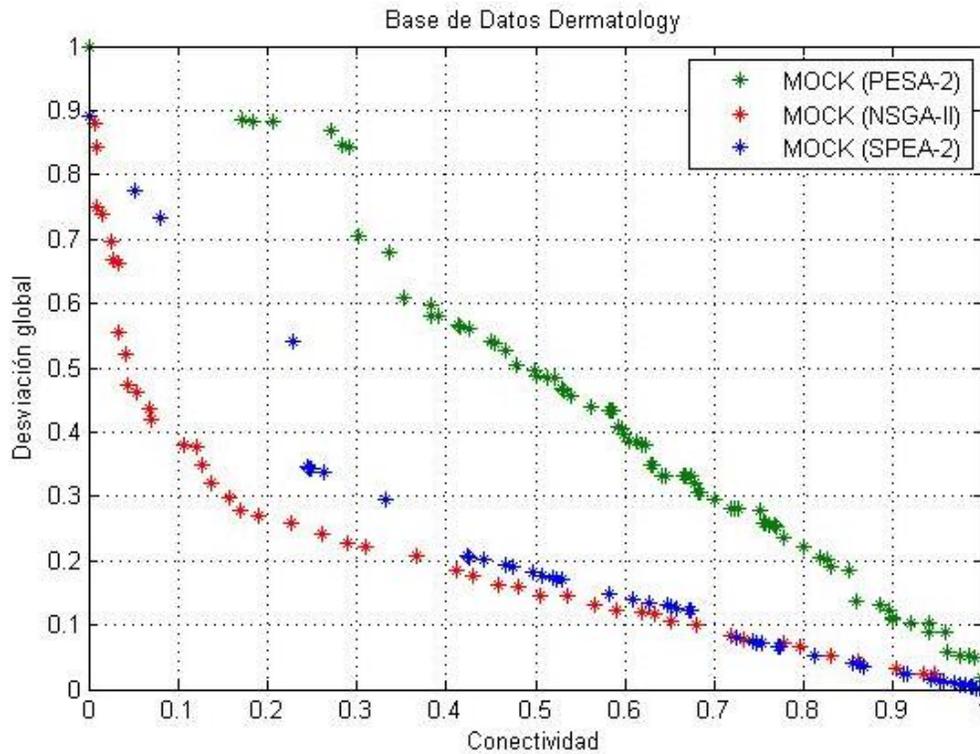


Figura 6.6: Frente de Pareto de cada algoritmo para Dermatology, con primera configuración de parámetros generada por irace.

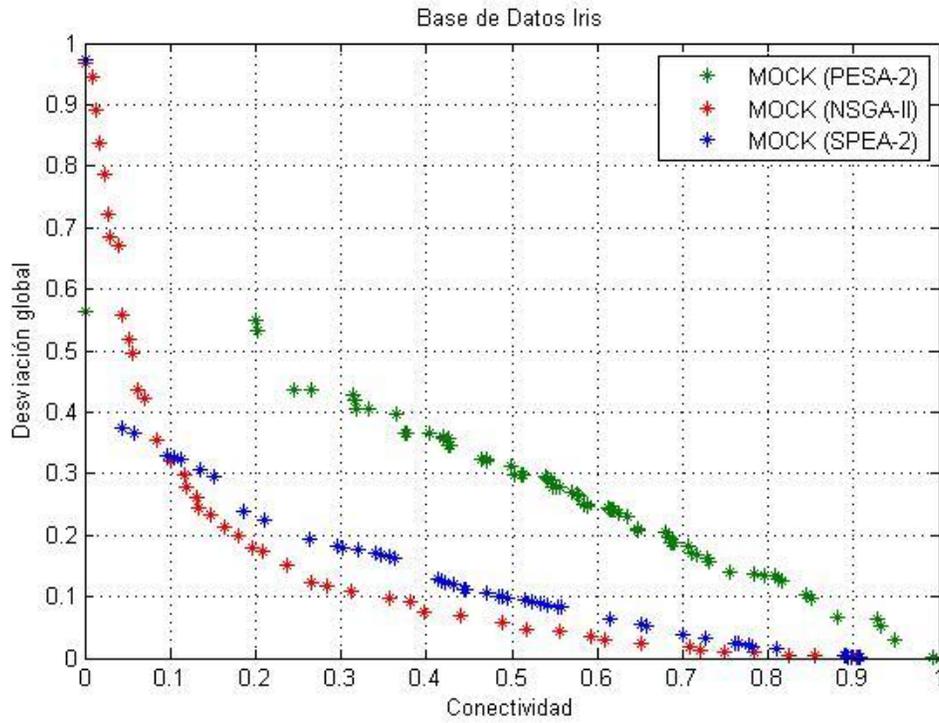


Figura 6.7: Frente de Pareto de cada algoritmo para Iris, con primera configuración de parámetros generada por irace.

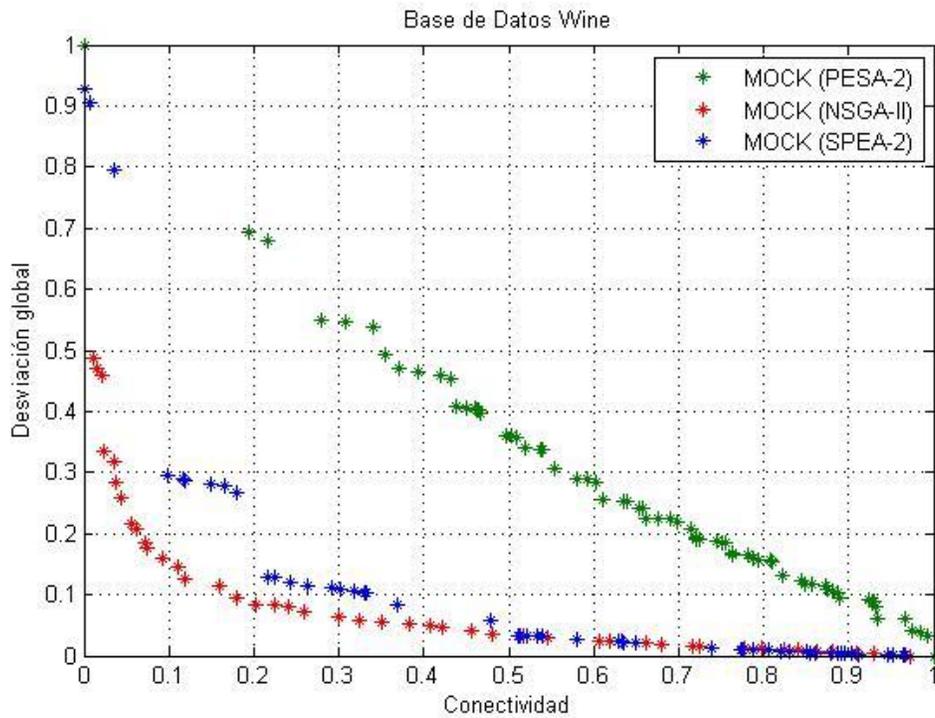


Figura 6.8: Frente de Pareto de cada algoritmo para Wine, con primera configuración de parámetros generada por irace.

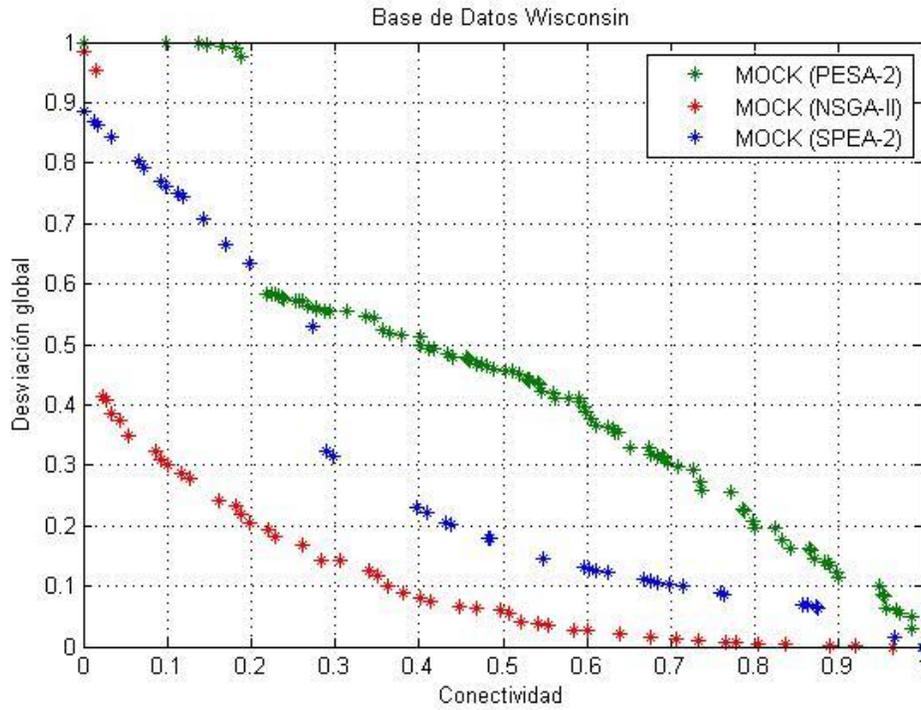


Figura 6.9: Frente de Pareto de cada algoritmo para Wisconsin, con primera configuración de parámetros generada por irace.

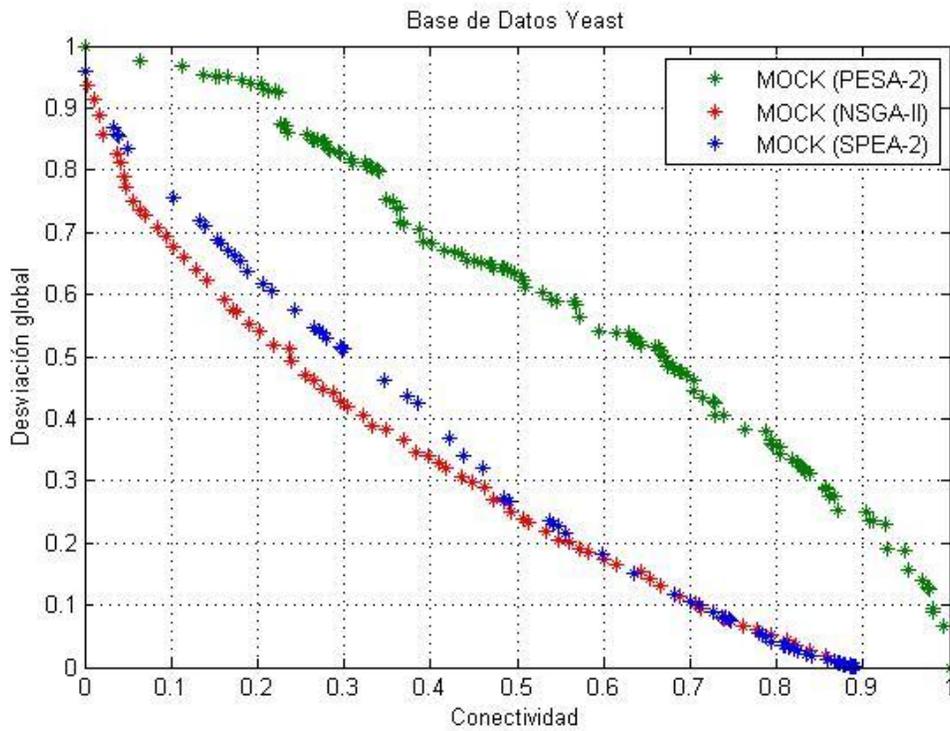


Figura 6.10: Frente de Pareto de cada algoritmo para Yeast, con primera configuración de parámetros generada por irace.

Por último, en las Figuras 6.11-6.15, se muestran los frentes obtenidos al ejecutar los tres algoritmos con la segunda configuración de parámetros obtenida por irace, en donde el número de generaciones es igual a 246, L igual a 20 y probabilidad de cruce 0.7.

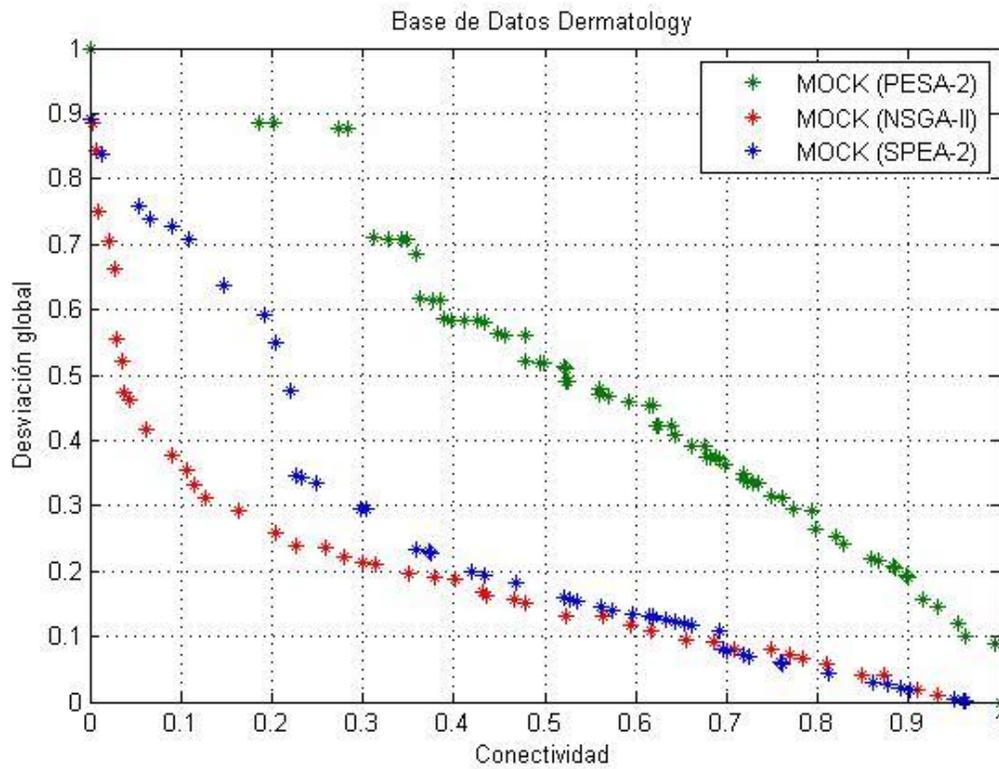


Figura 6.11: Frente de Pareto de cada algoritmo para Dermatology, con segunda configuración de parámetros generada por irace.

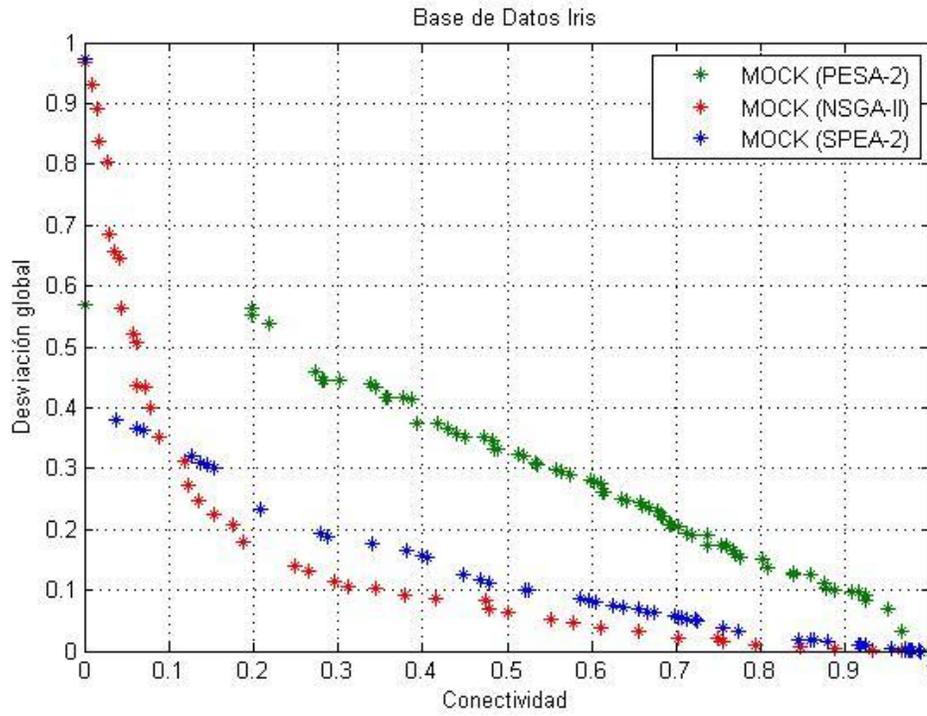


Figura 6.12: Frente de Pareto de cada algoritmo para Iris, con segunda configuración de parámetros generada por irace.

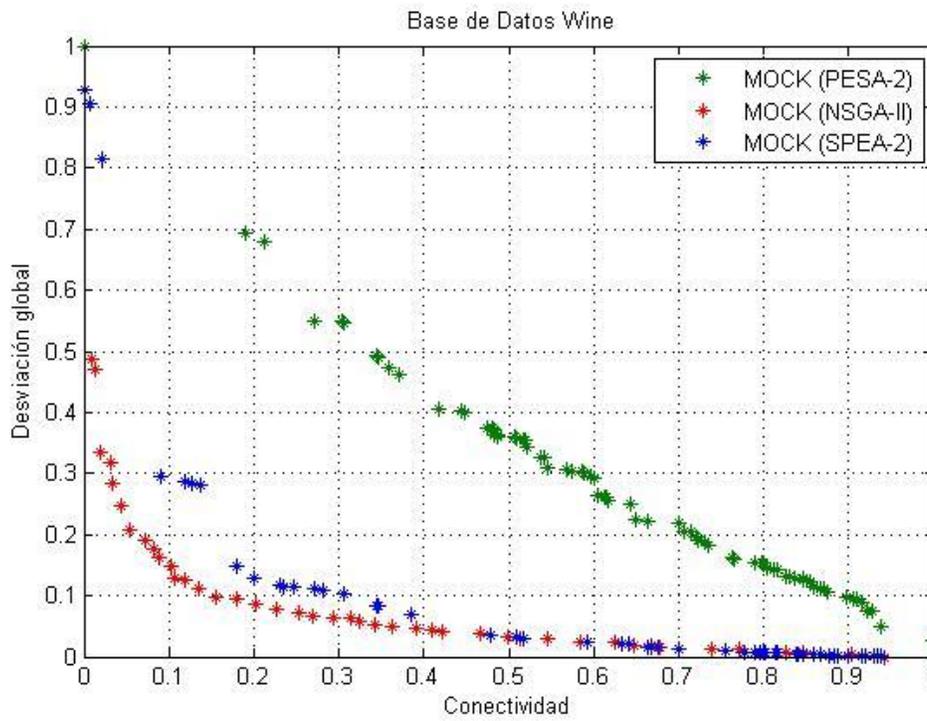


Figura 6.13: Frente de Pareto de cada algoritmo para Wine, con segunda configuración de parámetros generada por irace.

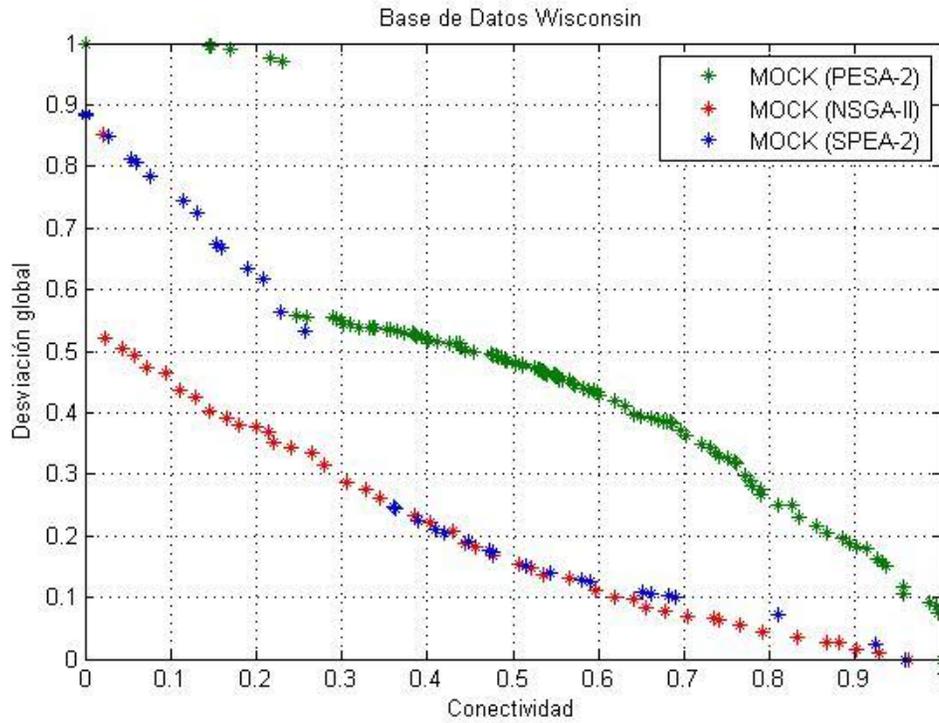


Figura 6.14: Frente de Pareto de cada algoritmo para Wisconsin, con segunda configuración de parámetros generada por irace.

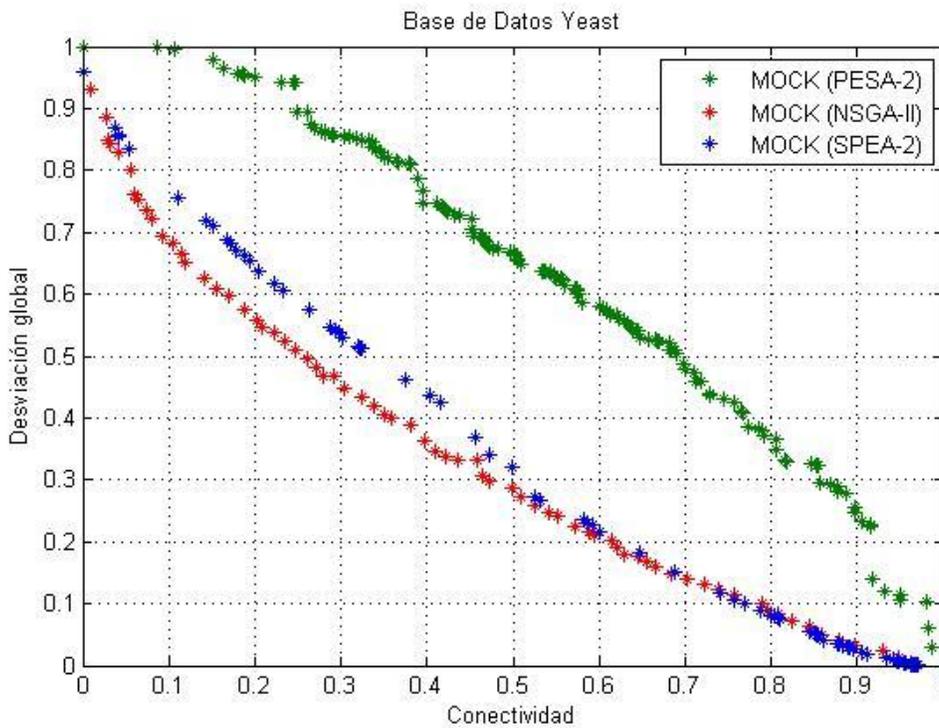


Figura 6.15: Frente de Pareto de cada algoritmo para Yeast, con segunda configuración de parámetros generada por irace.

En las gráficas anteriores, se puede observar a simple vista que las dos nuevas versiones que se implementaron obtienen mejores resultados en comparación con el algoritmo original. Sin embargo, se observa que usando los parámetros propuestos por los autores de MOCK, el algoritmo  $MOCK_{NSGA-II}$  supera a los otros algoritmos en todas las bases de datos a excepción de Iris. Los tres algoritmos, encuentran soluciones cubriendo gran parte del área de soluciones. Para las otras configuraciones de parámetros, se logra que  $MOCK_{NSGA-II}$  sea el mejor algoritmo para todas las bases de datos incluyendo iris.

Para las bases de datos Dermatology y Wine,  $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$  presentan soluciones similares, lo cual significa que aunque aparentemente  $MOCK_{NSGA-II}$  parece comportarse de mejor manera, las soluciones que se obtienen con dicho algoritmo no dominan por completo a las obtenidas por  $MOCK_{SPEA-2}$ . Ahora bien, tomando como referencia los frentes mostrados anteriormente, a continuación se muestran los resultados de métricas que ayudan a determinar con mayor precisión cual algoritmo presenta un mejor comportamiento.

### 6.1.1 Resultados comparados con la métrica Hipervolumen

En las Tablas 6.3- 6.5 se presentan las estadísticas obtenidas al aplicar la métrica Hipervolumen a los resultados obtenidos para cada uno de los algoritmos, se coloca en negritas el mejor resultado. El valor de la media representa el área que puede un algoritmo cubrir del espacio objetivo. Cabe destacar que el punto de referencia para realizar el cálculo fue el origen (0,0).

Tabla 6.3: Estadísticas de la métrica Hipervolumen para cada base de datos, usando configuración de parámetros original. Desv. Est.:Desviación estándar.

	Algoritmo	Mejor	Media	Mediana	Desv. Est.	Peor
Dermatology	$MOCK_{PESA-2}$	0.4598	0.46038	0.4602	0.000434102	0.4613
	$MOCK_{NSGA-II}$	<b>0.1766</b>	0.1767	0.1767	<b>5.67646E-05</b>	0.1768
	$MOCK_{SPEA-2}$	0.2421	0.2420	0.2421	0.000352767	0.2422
Iris	$MOCK_{PESA-2}$	0.3108	0.3115	0.3115	0.000286744	0.3119
	$MOCK_{NSGA-II}$	0.2575	0.25756	0.2576	6.99206E-05	0.2575
	$MOCK_{SPEA-2}$	<b>0.1340</b>	0.13403	0.1340	<b>4.83046E-05</b>	0.1341
Wine	$MOCK_{PESA-2}$	0.3048	0.30548	0.3054	0.000480278	0.3063
	$MOCK_{NSGA-II}$	<b>0.0602</b>	0.06024	0.0602	<b>8.43274E-05</b>	0.0604
	$MOCK_{SPEA-2}$	0.0821	0.08233	0.0824	0.000125167	0.0824
Wisconsin	$MOCK_{PESA-2}$	0.5106	0.51102	0.5110	0.000402216	0.5117
	$MOCK_{NSGA-II}$	<b>0.1973</b>	0.19743	0.1975	9.48683E-05	0.1975
	$MOCK_{SPEA-2}$	0.2640	0.26401	0.2640	<b>3.16228E-05</b>	0.2641
Yeast	$MOCK_{PESA-2}$	0.6459	0.64655	0.6462	0.000535931	0.6473
	$MOCK_{NSGA-II}$	<b>0.1606</b>	0.16077	0.1607	0.000188856	0.1612
	$MOCK_{SPEA-2}$	0.3480	0.34813	0.3481	<b>8.23273E-05</b>	0.3483

Tabla 6.4: Estadísticas de la métrica Hipervolumen para cada base de datos, usando primera configuración de parámetros generada por irace. Desv. Est.:Desviación estándar.

	Algoritmo	Mejor	Media	Mediana	Desv. Est.	Peor
Dermatology	MOCK <sub>PESA-2</sub>	0.5058	0.50606	0.5060	<b>0.000250333</b>	0.5064
	MOCK <sub>NSGA-II</sub>	<b>0.1796</b>	0.18047	0.1809	0.000561842	0.1809
	MOCK <sub>SPEA-2</sub>	0.2431	0.24517	0.2454	0.000763108	0.2456
Iris	MOCK <sub>PESA-2</sub>	0.2997	0.30001	0.3001	<b>0.000341402</b>	0.3008
	MOCK <sub>NSGA-II</sub>	<b>0.1148</b>	0.11527	0.1154	0.000359166	0.1159
	MOCK <sub>SPEA-2</sub>	0.1228	0.12358	0.1239	0.000458984	0.1239
Wine	MOCK <sub>PESA-2</sub>	0.3775	0.37771	0.3777	<b>0.000196921</b>	0.3781
	MOCK <sub>NSGA-II</sub>	<b>0.0609</b>	0.06291	0.06405	0.001604473	0.0642
	MOCK <sub>SPEA-2</sub>	0.1009	0.10386	0.1050	0.002046786	0.1053
Wisconsin	MOCK <sub>PESA-2</sub>	0.4774	0.47769	0.4777	0.000314289	0.4781
	MOCK <sub>NSGA-II</sub>	<b>0.1085</b>	0.10857	0.1086	<b>4.21637E-05</b>	0.1086
	MOCK <sub>SPEA-2</sub>	0.2597	0.26176	0.2608	0.003225317	0.2678
Yeast	MOCK <sub>PESA-2</sub>	0.6130	0.61321	0.6133	<b>0.000119722</b>	0.6133
	MOCK <sub>NSGA-II</sub>	<b>0.2939</b>	0.29420	0.2943	0.000210819	0.2944
	MOCK <sub>SPEA-2</sub>	0.3194	0.31991	0.3201	0.000357305	0.3203

Tabla 6.5: Estadísticas de la métrica Hipervolumen para cada base de datos, usando segunda configuración de parámetros generada por irace. Desv. Est.:Desviación estándar.

	Algoritmo	Mejor	Media	Mediana	Desv. Est.	Peor
Dermatology	MOCK <sub>PESA-2</sub>	0.5378	0.53841	0.5379	0.001015929	0.5403
	MOCK <sub>NSGA-II</sub>	<b>0.1698</b>	0.17021	0.1703	<b>0.000191195</b>	0.1703
	MOCK <sub>SPEA-2</sub>	0.2443	0.24568	0.2461	0.000737564	0.2461
Iris	MOCK <sub>PESA-2</sub>	0.3247	0.32486	0.3248	0.000183787	0.3251
	MOCK <sub>NSGA-II</sub>	<b>0.1199</b>	0.12069	0.1206	0.000433205	0.1214
	MOCK <sub>SPEA-2</sub>	0.1327	0.13284	0.1328	<b>0.000142980</b>	0.1331
Wine	MOCK <sub>PESA-2</sub>	0.3689	0.36922	0.3693	0.000282056	0.3697
	MOCK <sub>NSGA-II</sub>	<b>0.0599</b>	0.06008	0.0602	<b>0.000154919</b>	0.0602
	MOCK <sub>SPEA-2</sub>	0.0826	0.08460	0.0843	0.002180214	0.0877
Wisconsin	MOCK <sub>PESA-2</sub>	0.5198	0.52043	0.5205	0.000275076	0.5207
	MOCK <sub>NSGA-II</sub>	<b>0.2018</b>	0.20203	0.2021	<b>0.000125167</b>	0.2021
	MOCK <sub>SPEA-2</sub>	0.2613	0.26444	0.2659	0.001988411	0.2659
Yeast	MOCK <sub>PESA-2</sub>	0.6388	0.63894	0.6389	<b>0.000126491</b>	0.6391
	MOCK <sub>NSGA-II</sub>	<b>0.3193</b>	0.31958	0.3197	0.000244040	0.3198
	MOCK <sub>SPEA-2</sub>	0.3473	0.34778	0.3479	0.000193218	0.3479

Debido a que se tomó como referencia el origen, un menor valor de hipervolumen identificará al mejor algoritmo, lo cual indica que el área que el mismo cubre es menor y por lo tanto se aproxima más al frente óptimo de Pareto. De acuerdo con los resultados obtenidos por esta métrica, usando la configuración de parámetros original, se observa que para las bases de datos Dermatology, Wine, Wisconsin y Yeast, MOCK<sub>NSGA-II</sub> obtiene mejores resultados que los otros dos algoritmos. Comparando MOCK<sub>NSGA-II</sub> y MOCK<sub>SPEA-2</sub>

para Dermatology y Wine, se observa también que  $MOCK_{NSGA-II}$  tiene un mejor comportamiento, a pesar de que existan soluciones de  $MOCK_{SPEA-2}$  que dominan al primero.

En cuanto a las otras configuraciones de parámetros se observa que  $MOCK_{NSGA-II}$  es el mejor algoritmo, obteniendo valores menores de hipervolumen incluso en la base de datos de iris, se puede notar también que el cambio de parámetros varía el comportamiento de los algoritmos y no es preciso definir cual configuración funciona de mejor manera, ya que depende de cada base de datos. Tomando en cuenta el  $MOCK_{NSGA-II}$ , se puede notar que la configuración de parámetros original funciona bien para la base de datos Yeast obteniendo un valor de hipervolumen mucho mejor que el que se obtiene con las otras dos configuraciones. La segunda configuración obtiene mejores valores en las bases de datos Iris y Wisconsin, mientras que la tercera configuración obtiene mejores resultados en las bases de datos Dermatology y Wine.

Como se puede observar anteriormente, con las tres configuraciones de parámetros las nuevas versiones ( $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$ ) mejoran los resultados en comparación con la versión original ( $MOCK_{PESA-2}$ ), sin embargo no se sabe con precisión si hay una diferencia significativa entre los resultados de los algoritmos  $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$  por lo cual, se aplican pruebas estadísticas para poder dar una conclusión más precisa sobre los resultados obtenidos. Se aplicó la prueba U Mann de Whitney o Wilcoxon [101] comparando los resultados de un algoritmo con cada uno los resultados de los otros dos algoritmos, se muestra lo obtenido por la prueba en las Tablas 6.6 - 6.8.

Tabla 6.6: U Mann de Whitney con muestra de la métrica Hipervolumen para cada una de las bases de datos, usando configuración de parámetros original.

Base datos	$MOCK_{PESA-2}$ vs $MOCK_{NSGA-II}$		$MOCK_{PESA-2}$ vs $MOCK_{SPEA-2}$		$MOCK_{NSGA-II}$ vs $MOCK_{SPEA-2}$	
	p-value	H	p-value	H	p-value	H
<b>Dermatology</b>	1.0515e-04	1	8.8040e-05	1	7.5738e-05	1
<b>Iris</b>	1.0927e-04	1	8.9829e-05	1	1.0651e-04	1
<b>Wine</b>	8.6863e-05	1	1.0447e-04	1	7.5212e-05	1
<b>Wisconsin</b>	1.2543e-04	1	7.3650e-05	1	6.6688e-05	1
<b>Yeast</b>	1.4332e-04	1	1.3093e-04	1	1.3093e-04	1

Tabla 6.7: U Mann de Whitney con muestra de la métrica Hipervolumen para cada una de las bases de datos, usando primera configuración de parámetros generada por irace.

Base datos	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>		MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>		MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	
	p-value	H	p-value	H	p-value	H
<b>Dermatology</b>	1.3093e-04	1	1.4332e-04	1	1.2620e-04	1
<b>Iris</b>	1.2161e-04	1	1.2313e-04	1	1.1641e-04	1
<b>Wine</b>	1.5028e-04	1	1.5566e-04	1	1.4332e-04	1
<b>Wisconsin</b>	9.3489e-05	1	1.3993e-04	1	9.5991e-05	1
<b>Yeast</b>	1.2543e-04	1	1.1495e-04	1	1.2543e-04	1

Tabla 6.8: U Mann de Whitney con muestra de la métrica Hipervolumen para cada una de las bases de datos, usando segunda configuración de parámetros generada por irace.

Base datos	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>		MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>		MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	
	p-value	H	p-value	H	p-value	H
<b>Dermatology</b>	9.5991e-05	1	1.2934e-04	1	8.6863e-05	1
<b>Iris</b>	1.1208e-04	1	1.2698e-04	1	1.0380e-04	1
<b>Wine</b>	1.2934e-04	1	1.5475e-04	1	1.2855e-04	1
<b>Wisconsin</b>	1.2161e-04	1	1.3743e-04	1	1.0380e-04	1
<b>Yeast</b>	1.5116e-04	1	1.3579e-04	1	1.2855e-04	1

De acuerdo al resultado de la prueba para la métrica hipervolumen, se observa que en todas las comparaciones realizadas y para las tres configuraciones de parámetros se obtuvo un p-value menor a 0.05 por lo que se puede asumir un rechazo de la hipótesis nula ( $h=1$ ), la cual supone la igualdad de las medianas, esto significa que los algoritmos presentan un comportamiento diferente, por lo cual si existe diferencia significativa entre cada uno de ellos. Con esto se puede decir que usando los parámetros originales, MOCK<sub>NSGA-II</sub> es mejor con respecto a MOCK<sub>SPEA-2</sub> y a MOCK<sub>PESA-2</sub>, sin embargo para Iris MOCK<sub>SPEA-2</sub> es mejor que MOCK<sub>NSGA-II</sub>. Finalmente para las otras dos configuraciones, con los resultados de las pruebas estadísticas, se puede corroborar que MOCK<sub>NSGA-II</sub> es el mejor algoritmo en todas las bases de datos y presenta un comportamiento diferente a los demás algoritmos.

Ya que los algoritmos se ejecutaron con tres configuraciones de parámetros diferentes, anteriormente se mostraron resultados de pruebas por cada configuración, sin embargo es necesario también realizar pruebas comparando las tres configuraciones entre sí, esto es para ver si existen una diferencia significativa en el comportamiento de los algoritmos al cambiar los parámetros (Tabla 6.9).

Tabla 6.9: U Mann de Whitney con muestra de la métrica Hipervolumen comparando los resultados de cada configuración de parámetros.

Base datos	Algoritmo	U Mann de Whitney					
		Parámetros Original		Parámetros Original		Parámetros PrimerConf.	
		vs		vs		vs	
		Parámetros PrimerConf.	Parámetros SegConf.	Parámetros SegConf.	Parámetros SegConf.	Parámetros SegConf.	Parámetros SegConf.
p-value	H	p-value	H	p-value	H		
Dermatology	MOCK <sub>PESA-2</sub>	1.3415e-04	1	1.3093e-04	1	1.4504e-04	1
	MOCK <sub>NSGA-II</sub>	1.0247e-04	1	7.5738e-05	1	8.5697e-05	1
	MOCK <sub>SPEA-2</sub>	9.4733e-05	1	8.6863e-05	1	0.0148	1
Iris	MOCK <sub>PESA-2</sub>	1.0927e-04	1	1.1280e-04	1	1.3253e-04	1
	MOCK <sub>NSGA-II</sub>	1.2161e-04	1	1.0858e-04	1	1.0247e-04	1
	MOCK <sub>SPEA-2</sub>	1.0181e-04	1	1.0181e-04	1	1.1788e-04	1
Wine	MOCK <sub>PESA-2</sub>	1.3993e-04	1	1.3661e-04	1	1.5932e-04	1
	MOCK <sub>NSGA-II</sub>	9.4109e-05	1	0.0168	1	1.2161e-04	1
	MOCK <sub>SPEA-2</sub>	1.1714e-04	1	1.1936e-04	1	1.5116e-04	1
Wisconsin	MOCK <sub>PESA-2</sub>	1.3661e-04	1	1.4764e-04	1	1.4764e-04	1
	MOCK <sub>NSGA-II</sub>	8.5119e-05	1	1.0247e-04	1	7.5212e-05	1
	MOCK <sub>SPEA-2</sub>	0.0183	1	0.4445	0	0.0226	1
Yeast	MOCK <sub>PESA-2</sub>	1.2855e-04	1	1.5116e-04	1	1.3579e-04	1
	MOCK <sub>NSGA-II</sub>	1.3993e-04	1	1.4332e-04	1	1.3993e-04	1
	MOCK <sub>SPEA-2</sub>	1.1714e-04	1	1.1714e-04	1	1.1495e-04	1

Se puede notar que en todas las pruebas realizadas a excepción de una, se presenta un p-value menor a 0.05 con un valor de H=1, por lo que se confirma que con el cambio de los parámetros los algoritmos presentan comportamientos diferentes. En el caso de la base de datos de Wisconsin con el algoritmo MOCK<sub>SPEA-2</sub>, se puede notar que el p-value fue mayor tomando las muestras de las ejecuciones usando los parámetros originales y la segunda configuración, por lo tanto para este caso no existe diferencia significativa en el comportamiento del algoritmo al utilizar la configuración original o la segunda configuración.

### 6.1.2 Resultados comparados con la métrica Two Set Coverage.

Para complementar los resultados obtenidos con hipervolumen, en las Tablas 6.10 - 6.12 se presentan las estadísticas obtenidas al aplicar la métrica Two Set Coverage a los resultados de cada uno de los algoritmos, se coloca en negritas el mejor resultado. La comparación se realizó con cada uno de los frentes obtenidos en las 10 ejecuciones de un algoritmo contra cada uno de los frentes obtenidos del algoritmo con el que se compara, esto con el fin de conocer que algoritmo es mejor con respecto al número de soluciones que

domina y el área que ocupa del espacio objetivo. De la misma manera que con la métrica de hipervolumen, se muestran las tablas para cada configuración de parámetros.

Tabla 6.10: Estadísticas de la métrica Two Set Coverage para cada base de datos, usando configuración de parámetros original. Desv. Est.:Desviación estándar. El signo ✓ Si existe diferencia significativa. El signo =: No existe diferencia significativa.

	Algoritmo	Mejor	Media	Mediana	Desv. Est.	Peor	U Mann de Whitney		
							p-value	h	
Dermatology	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
Iris	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.7600</b>	0.6920	0.6800	0.05266245	0.6200	1.0997	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.4000	0.3880	0.4000	<b>0.01932184</b>	0.3600	e-04		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.0411	0.03150	0.0274	<b>0.00661773</b>	0.0274	9.6624	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>0.9863</b>	0.97808	0.9726	0.00707465	0.9726	e-05		
Wine	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0.0274	0.02192	0.0274	0.00707465	0.0137	6.3864	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>0.9800</b>	0.97800	0.9800	<b>0.00632456</b>	0.9600	e-05		
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	0.0400	0.0380	0.0400	<b>0.00632456</b>	0.0200	4.7314	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	<b>0.9600</b>	0.9440	0.9400	0.00843274	0.9400	e-05		
Wisconsin	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	e-05		
Yeast	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.6400</b>	0.5940	0.5800	0.02319004	0.5800	1.0181	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.4200	0.4000	0.4000	<b>0.02108185</b>	0.3800	e-04		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
Yeast	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.7800</b>	0.7580	0.7500	<b>0.019888579</b>	0.7400	1.3253	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.3400	0.2960	0.3000	0.046951510	0.2400	e-04		
Yeast	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
Yeast	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>1</b>	0.9933	0.9933	0.00711513	0.9865	8.4544	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.0135	0.0027	0	<b>0.00569210</b>	0	e-05		

Tabla 6.11: Estadísticas de la métrica Two Set Coverage para cada base de datos, usando primera configuración de parámetros generada por irace. Dev. Est.:Desviación estándar. El signo ✓ Si existe diferencia significativa. El signo =: No existe diferencia significativa.

	Algoritmo	Mejor	Media	Mediana	Dev. Est.	Peor	U Mann de Whitney		
							p-value	h	
Dermatology	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
Iris	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.7800</b>	0.75600	0.7400	0.020655911	0.7400	9.8543	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.2000	0.19000	0.2000	<b>0.016996732</b>	0.1600	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.0263	0.01975	0.0198	0.006904306	0.0132	9.9839	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>0.9868</b>	0.98287	0.9868	<b>0.006327901</b>	0.9737	e-05		
Wine	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0.0260	0.0156	0.0131	0.005481281	0.0130	4.7314	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>0.9870</b>	0.9857	0.9870	<b>0.004110961</b>	0.9740	e-05		
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.6600</b>	0.64600	0.6400	0.009660918	0.6400	7.3650	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.0600	0.05600	0.0600	<b>0.008432740</b>	0.0400	e-05		
Wisconsin	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	e-05		
Yeast	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.7000</b>	0.67400	0.6800	0.018973666	0.6400	5.8493	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.3400	0.33800	0.3400	<b>0.006324555</b>	0.3200	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
Wisconsin	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-II</sub>	0.0594	0.05049	0.0495	<b>0.005619698</b>	0.0396	1.0858	1	✓
	MOCK <sub>SPEA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>0.9505</b>	0.94357	0.9406	0.006681991	0.9307	e-04		
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.9400</b>	0.91800	0.9200	<b>0.011352924</b>	0.9000	1.0858	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.2400	0.22200	0.2300	0.019888579	0.2000	e-04		
Yeast	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>1</b>	1	1	<b>0</b>	1	e-05		
Yeast	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.7027</b>	0.68380	0.6892	0.015846135	0.6622	1.3253	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.1892	0.17164	0.1622	<b>0.015669588</b>	0.1486	e-04		

Para esta métrica Two set coverage el valor óptimo es 1, este valor indica que tanto porcentaje de las soluciones encontradas por un algoritmo dominan a las del algoritmo contra el que es comparado. De tal manera que, si todas las soluciones de un algoritmo dominan o son iguales a todas las del otro algoritmo, se obtendrá un valor para Two set coverage igual a 1, o bien 0 en caso contrario. Para poder decir que un algoritmo es mejor que el otro, es preferible tener valores cercanos a 1.

Tabla 6.12: Estadísticas de la métrica Two Set Coverage para cada base de datos, usando segunda configuración de parámetros generada por irace. Desv. Est.:Desviación estándar. El signo ✓ Si existe diferencia significativa. El signo =: No existe diferencia significativa.

	Algoritmo	Mejor	Media	Mediana	Desv. Est.	Peor	U Mann de Whitney		
							p-value	h	
Dermatology	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
Iris	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.7800</b>	0.74800	0.7400	0.013984118	0.7400	7.5212	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.1800	0.17600	0.1800	<b>0.008432740</b>	0.1600	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.0411	0.02603	0.0274	<b>0.011995652</b>	0.0137	1.3093	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	<b>0.9863</b>	0.97671	0.9863	0.012996961	0.9589	e-04		
Wine	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0.0541	0.03646	0.0405	<b>0.009133236</b>	0.0270	1.3497	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	<b>0.9865</b>	0.97029	0.9730	0.012427429	0.9459	e-04		
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.7600</b>	0.71400	0.7000	0.023190036	0.7000	5.8493	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.0800	0.07800	0.0800	<b>0.006324555</b>	0.0600	e-05		
Wisconsin	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
Yeast	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.6600</b>	0.62400	0.6400	0.030983867	0.5800	1.0049	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.4400	0.39800	0.3800	<b>0.028982753</b>	0.3800	e-04		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
Wisconsin	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-II</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.5800</b>	0.56400	0.5600	<b>0.008432740</b>	0.5600	7.3650	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.3800	0.37400	0.3800	0.009660918	0.3600	e-05		
Yeast	MOCK <sub>PESA-2</sub> vs MOCK <sub>NSGA-II</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>NSGA-II</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
	MOCK <sub>PESA-2</sub> vs MOCK <sub>SPEA-2</sub>	0	0	0	0	0	1.5938	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>PESA-2</sub>	1	1	1	0	1	e-05		
Yeast	MOCK <sub>NSGA-II</sub> vs MOCK <sub>SPEA-2</sub>	<b>0.6892</b>	0.65405	0.64865	0.022275609	0.6351	1.3661	1	✓
	MOCK <sub>SPEA-2</sub> vs MOCK <sub>NSGA-II</sub>	0.3514	0.32702	0.32430	<b>0.017803982</b>	0.3108	e-04		

En las tablas anteriores observamos que en todos los casos los algoritmos implementados dominan a todas las soluciones del algoritmo original ya que se obtiene un valor igual a 1 desde MOCK<sub>NSGA-II</sub> a MOCK<sub>PESA-2</sub> y desde MOCK<sub>SPEA-2</sub> a MOCK<sub>PESA-2</sub>, así como se obtiene un valor de 0 en el caso inverso, por lo tanto podemos afirmar que las nuevas versiones son mejores que MOCK<sub>PESA-2</sub>.

De igual manera que en la métrica de hipervolumen, se aplicó la prueba estadística U Mann de Whitney , se observa en todas las comparaciones que el p-value es pequeño y  $h=1$ , esto quiere decir, que existe una diferencia significativa entre todos los algoritmos, lo cual coincide con los resultados obtenidos con hipervolumen. Debido a que los algoritmos tienen comportamientos diferentes de acuerdo a las pruebas realizadas, los valores marcados en negritas pueden tomarse para identificar al mejor algoritmo. De aquí que se afirma que a excepción de Iris, **MOCK<sub>NSGA-II</sub>** es mejor para todas las bases de datos usando cualquiera de las tres configuraciones de parámetros propuestas.

Cabe destacar que la calibración de parámetros fue realizada tomando en consideración el hipervolumen, por lo tanto, para esta métrica no se presentan pruebas estadísticas de comparación de los algoritmos con las diferentes configuraciones de parámetros.

## **6.2 Resultados segundo experimento**

Como ya fue mencionado en el diseño experimental del capítulo anterior, con el objetivo de evaluar el rendimiento de los algoritmos de acuerdo a la calidad del clustering, se hace un comparativo con otros algoritmos de agrupamiento clásicos usando la medida de evaluación F-measure una vez que es seleccionada una solución del frente de Pareto. Debido a que un valor más alto obtenido por dicha medida de evaluación significa que se encuentra una solución de agrupamiento ideal, para todos los algoritmos que se comparan se elige la solución del frente que maximice mejor el F-measure. Los resultados que se obtienen tomando en cuenta este criterio son interesantes porque permiten obtener información sobre el rendimiento de los algoritmos al encontrar soluciones de alta calidad, dando la ventaja de que a diferencia de los otros algoritmos de agrupamiento clásicos, se obtiene un escenario en el que el experto tiene la oportunidad de probar una serie de soluciones alternativas.

En las Tablas 6.13 -6.15 se muestra el valor de la media de los resultados obtenidos al realizar 10 ejecuciones independientes tomando en consideración la métrica F-measure , haciendo un comparativo del algoritmo MOCK en su versión original, algunos algoritmos de agrupamiento clásicos y las nuevas implementaciones de MOCK. El mejor resultado se muestra en negritas. Las tablas muestran resultados de acuerdo a cada configuración de parámetros con las que se hicieron los experimentos.

Tabla 6.13: Promedio del valor de F-measure para diez ejecuciones de cada algoritmo, usando configuración de parámetros original.

Algoritmo Base de datos	Clustering jerárquico	k-means	Meta-clustering	MOCK <sub>PESA-2</sub>	MOCK <sub>NSGA-II</sub>	MOCK <sub>SPEA-2</sub>
Dermatology	0.898637	0.953200	0.782163	0.958893	<b>0.960392</b>	0.958471
Iris	0.809857	0.817795	0.774389	0.834559	0.983729	<b>0.985699</b>
Wine	0.925527	0.919785	0.925524	0.945101	<b>0.950094</b>	0.945283
Wisconsin	0.965966	0.965825	0.849309	0.971545	<b>0.974915</b>	0.970961
Yeast	0.448316	0.422493	0.309288	0.493356	<b>0.803160</b>	0.785482

Tabla 6.14: Promedio del valor de F-measure para diez ejecuciones de cada algoritmo, usando primera configuración de parámetros generada por irace.

Algoritmo Base de datos	Clustering jerárquico	k-means	Meta-clustering	MOCK <sub>PESA-2</sub>	MOCK <sub>NSGA-II</sub>	MOCK <sub>SPEA-2</sub>
Dermatology	0.898637	0.953200	0.782163	0.939852	<b>0.958497</b>	0.953284
Iris	0.809857	0.817795	0.774389	0.896429	<b>0.992410</b>	0.988101
Wine	0.925527	0.919785	0.925524	0.944720	<b>0.949748</b>	0.949028
Wisconsin	0.965966	0.965825	0.849309	0.975968	<b>0.983718</b>	0.971382
Yeast	0.448316	0.422493	0.309288	0.513901	<b>0.800158</b>	0.804610

Tabla 6.15: Promedio del valor de F-measure para diez ejecuciones de cada algoritmo, usando segunda configuración de parámetros generada por irace.

Algoritmo Base de datos	Clustering jerárquico	k-means	Meta-clustering	MOCK <sub>PESA-2</sub>	MOCK <sub>NSGA-II</sub>	MOCK <sub>SPEA-2</sub>
Dermatology	0.898637	0.953200	0.782163	0.937496	<b>0.967906</b>	0.952018
Iris	0.809857	0.817795	0.774389	0.837957	<b>0.987496</b>	0.986174
Wine	0.925527	0.919785	0.925524	0.944958	<b>0.951094</b>	0.947928
Wisconsin	0.965966	0.965825	0.849309	0.970968	<b>0.973972</b>	0.970988
Yeast	0.448316	0.422493	0.309288	0.509658	<b>0.800106</b>	0.801957

Con el fin de obtener un comparativo más claro tomando en consideración las tres configuraciones de parámetros en la tabla 6.16 se muestra un resumen del mejor valor obtenido de F-measure para cada uno de los algoritmos en cada base de datos, el resultado en negritas representa el mejor valor.

Tabla 6.16: Comparación de F-measure para las tres configuraciones de parámetros.

Base de datos	Algoritmo	Parámetros originales	Parámetros Primera configuración	Parámetros Segunda configuración
Dermatology	MOCK <sub>PESA-2</sub>	<b>0.958893</b>	0.939852	0.937496
	MOCK <sub>NSGA-II</sub>	0.960392	0.958497	<b>0.967906</b>
	MOCK <sub>SPEA-2</sub>	<b>0.958471</b>	0.953284	0.952018
Iris	MOCK <sub>PESA-2</sub>	0.834559	<b>0.896429</b>	0.837957
	MOCK <sub>NSGA-II</sub>	0.983729	<b>0.992410</b>	0.987496
	MOCK <sub>SPEA-2</sub>	0.985699	<b>0.988101</b>	0.986174
Wine	MOCK <sub>PESA-2</sub>	<b>0.945101</b>	0.944720	0.944958
	MOCK <sub>NSGA-II</sub>	0.950094	0.949748	<b>0.951094</b>
	MOCK <sub>SPEA-2</sub>	0.945283	<b>0.949028</b>	0.947928
Wisconsin	MOCK <sub>PESA-2</sub>	0.971545	<b>0.975968</b>	0.970968
	MOCK <sub>NSGA-II</sub>	0.974915	<b>0.983718</b>	0.973972
	MOCK <sub>SPEA-2</sub>	0.970961	<b>0.971382</b>	0.970988
Yeast	MOCK <sub>PESA-2</sub>	0.493356	<b>0.513901</b>	0.509658
	MOCK <sub>NSGA-II</sub>	<b>0.803160</b>	0.800158	0.800106
	MOCK <sub>SPEA-2</sub>	0.785482	<b>0.804610</b>	0.801957

Como se puede observar en las tablas anteriores, el algoritmo de MOCK en sus tres versiones demuestra ser más robusto que los algoritmos de agrupamiento contra los que se compara. Las soluciones de agrupamiento finales generadas tanto por MOCK en su versión original como las obtenidas con los algoritmos implementados, resultan ser comparables o bien mejores que la mejor solución obtenida por los otros métodos, lo cual indica que tanto como MOCK como las nuevas versiones, exploran soluciones de alta calidad. Los resultados bajos de F-measure para los métodos tradicionales se ven afectados debido a que la gran parte de estos algoritmos pueden llegar a fallar al tratar con bases de datos de diversas formas y dimensiones, por ejemplo, k-means y el clustering jerárquico aglomerativo fallan para bases de datos con formas de clusters muy alargados.

Ahora bien, debido a que lo que más nos interesa en la presente investigación es la comparación de las nuevas versiones de MOCK con el algoritmo original, en la tabla también se muestran los resultados de estos tres algoritmos, se observa claramente que los algoritmos implementados obtienen valores mayores de F-measure, lo cual implica que las soluciones de agrupamiento obtenidas por dichos algoritmos reflejan una mejor calidad de clustering. Según este criterio de comparación de los algoritmos, MOCK<sub>NSGA-II</sub> presenta un

mejor desempeño obteniendo modelos de agrupamiento que maximizan la medida de evaluación. Con la configuración de parámetros original, para la base de datos Iris se observa que  $MOCK_{SPEA-2}$  obtuvo un mejor resultado, sin embargo al variar los parámetros  $MOCK_{NSGA-II}$  supera a  $MOCK_{SPEA-2}$  en todas las bases de datos incluso con iris, esto puede deberse a que la calibración se realizó tomando en cuenta precisamente esta base de datos y por consiguiente irace logró encontrar valores mejores para los parámetros usando el algoritmo  $MOCK_{NSGA-II}$ .

Estos resultados, pueden relacionarse con los obtenidos en el primer experimento, ya que tanto en las gráficas de los frentes de Pareto como en los resultados que se obtienen con cada una de las métricas se observa claramente que los nuevos algoritmos, se aproximan más al frente de Pareto óptimo, y obtienen mejores valores en cada métrica, por lo que se puede decir que cuando un algoritmo obtiene soluciones más cercanas al frente óptimo de Pareto mayor será el valor de F-measure que se pueda encontrar en cada una de las soluciones que conforman a dicho frente y por lo tanto la calidad de agrupamiento mejora significativamente.

## Capítulo VII: Conclusiones

En este trabajo se hizo uso del cómputo evolutivo para resolver el problema de agrupamiento de datos, para atacar dicho problema se tomó como referencia el algoritmo MOCK (Multiobjective clustering with automatic determination of the number of clusters), el cual ya ha sido probado con diferentes conjuntos de datos dando resultados favorables y comparativos con otros algoritmos clásicos de agrupamiento. El agrupamiento en este caso es visto como un problema de optimización multi-objetivo, por lo tanto MOCK ocupa el algoritmo PESA-2 para llevar a cabo el proceso evolutivo, con el cual se encuentran un conjunto de soluciones donde cada una de ellas representa un posible agrupamiento. Debido a que se resolvió un problema multi-objetivo, se sabe que existen otros algoritmos para optimización multi-objetivo que han demostrado dar mejores resultados que el que utiliza MOCK, por tal motivo, en este trabajo se implementaron otras dos versiones de MOCK, incorporando los algoritmos NSGA-II y SPEA-2.

Con las versiones implementadas  $MOCK_{NSGA-II}$  y  $MOCK_{SPEA-2}$ , se realizó un estudio comparativo entre ambos algoritmos y el MOCK original, los algoritmos se probaron en cinco bases de datos reales con diferentes características.

Retomando la hipótesis realizada al principio de este trabajo, "El uso de los algoritmos evolutivos para optimización multi-objetivo: NSGA-II y SPEA-II, como alternativas para realizar optimización en MOCK, mejora el desempeño de dicho algoritmo para encontrar mejores aproximaciones al frente de Pareto óptimo, lo que implica la obtención de mejores soluciones de agrupamiento", dados los experimentos realizados con los conjunto de datos de prueba, se puede afirmar que tanto el algoritmo NSGA-II como SPEA-II mejoran el comportamiento de MOCK y encuentran soluciones más cercanas al frente óptimo de Pareto. Siendo el NSGA-II con el que se encuentran resultados más favorables para la mayor parte de los conjuntos de datos. Los resultados obtenidos fueron evaluados y analizados con las métricas Two set coverage e Hipervolumen, dichas métricas son propias en el área del cómputo evolutivo y fueron usadas con el fin de presentar información más confiable sobre el comparativo realizado, cabe destacar que para corroborar los resultados también se hizo uso de las correspondientes pruebas estadísticas. Sin embargo, otro punto importante que trata la hipótesis es el hecho de que se encuentren buenas soluciones de agrupamiento, para evaluar este aspecto también se analizaron los resultados desde el punto de vista de aprendizaje automático llevando a cabo una comparación con algoritmos convencionales para la realización de clustering, se hizo uso de una medida de calidad de

clustering llamada F-measure, con tal medida se determinó qué tan buena es una solución elegida del frente de Pareto, los resultados que se presentan corresponden a la solución en donde se minimiza más el F-measure, estos resultados complementan los experimentos realizados tomando en consideración el área evolutiva demostrando que en efecto, con las nuevas versiones se encuentran soluciones de agrupamiento más prometedoras que las encontradas con la versión original de MOCK e incluso con otros algoritmos del estado del arte. Este hecho también corrobora que los algoritmos evolutivos pueden ser usados en el agrupamiento dando resultados favorables y con mayores ventajas que otros algoritmos tradicionales, como por ejemplo el hecho de que el valor de  $k$  (número de grupos) no necesite ser especificado con anterioridad, el algoritmo evolutivo realiza una mayor exploración y explotación del espacio de búsqueda evitando caer en óptimos globales por lo se garantiza que se encuentren las mejores soluciones y se provee al experto de un conjunto de soluciones en lugar de una sola solución, lo cual le permite poder elegir de acuerdo al problema la que considere más conveniente.

Por otro lado, ya que un aspecto importante al tratar con algoritmos evolutivos es la calibración de los parámetros, se hizo uso del paquete irace para la realización de dicha tarea, esto con el fin de ajustar de manera más adecuada los algoritmos y por ende que éstos logren optimizar de mejor manera el problema que atacan. También se hicieron experimentos con los resultados obtenidos por irace, los tres algoritmos fueron probados con tres combinaciones de parámetros diferentes, una de ellas es la especificada por los autores de MOCK y las otras dos fueron las obtenidas por dicho paquete. Los resultados que se obtuvieron al variar la configuración de parámetros son similares, esto probablemente se deba a que los autores reportaron resultados con parámetros que ya habían sido sujetos a prueba con anterioridad. Tales resultados, por lo tanto de igual manera demostraron que NSGA-II y SPEA-II presentan un mejor rendimiento. Cabe destacar que a pesar de la calibración, no se puede determinar cual es la configuración más apropiada para un determinado algoritmo, ya que esta depende en gran medida del conjunto de datos con el que se trabaje, por lo que parámetros que pueden estar bien calibrados para una base de datos y un determinado algoritmo no son los más apropiados si se considera otra base de datos aún con el mismo algoritmo.

Como trabajos futuros, se pueden probar los algoritmos con otros conjuntos de datos para observar si se comportan de manera similar, igualmente sería interesante modificar los operadores de variación utilizados ya que se tomaron los mismos que proponen los autores, cambiar dichos operadores podría modificar en gran medida el comportamiento de los algoritmos, por lo que probablemente encontrar operadores más adecuados para el tipo de representación entera mejoraría aún más el desempeño de los algoritmos, lo cual podría traducirse a que se encuentren soluciones de agrupamiento más prometedoras. Ahora bien, tal como ya se mencionó los algoritmos implementados también fueron comparados con otros algoritmos de clustering tradicionales, sin embargo los experimentos que se realizaron

en este caso solo consideraban una medida de evaluación, como trabajo a futuro también se propone llevar a cabo un estudio más completo tomando en consideración el uso de diversas medidas de evaluación que ayuden a complementar los análisis obtenidos.

Para este trabajo, la calibración de parámetros realizada solo se llevó a cabo tomando como referencia el conjunto de datos de iris y el algoritmo NSGA-II, el hecho de haber realizado la calibración de esta manera posiblemente puede sesgar la búsqueda de la mejor combinación de parámetros para trabajar con dicha base de datos y con el algoritmo antes mencionado, por lo tanto esto no garantiza que los parámetros encontrados sean los mejores para todos los conjuntos de datos, este hecho sugiere que como trabajo a futuro se pueda realizar un estudio en donde el paquete irace pueda ir probando el algoritmo tomando como entrada los distintos conjuntos de datos con los que se requiera trabajar, de tal manera se garantiza que mientras irace se encuentre en ejecución eventualmente irá encontrando parámetros que se adapten a más de un conjunto de datos.

Por último, ya que se sabe que la optimización multi-objetivo como su nombre lo indica puede atacar problemas con dos o más objetivos, un trabajo más a futuro puede consistir en tratar de incorporar algunos otros objetivos para la resolución del problema de agrupamiento, esto implicaría el estudio de algoritmos evolutivos multi-objetivo que sean eficientes cuando se trabaja con más de dos objetivos.

# REFERENCIAS

- [1] L. Kaufman, P. J. Rousseeuw, *Finding Groups in Data – An Introduction to Cluster Analysis*, Wiley Series in Probability and Mathematical Statistics, 1990.
- [2] B. S. Everitt, S. Landau, M. Leese, *Cluster Analysis*, Arnold Publishers, 2001.
- [3] A. K. Jain, R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.
- [4] L. J. Arabie, G. Hubert, P. DeSoete, *Clustering and Classification*, World Scientific, 1999.
- [5] C. Fralley, A. E. Raftery, “How Many Clusters? Which Clustering Method? Answer via Model-Based Cluster Analysis”, *The Computer Journal*, Vol. 41, pp. 578-588, 1998.
- [6] E. Falkenauer, *Genetic Algorithms and Grouping Problems*, John Wiley & Sons, 1998.
- [7] V. J. Rayward-Smith, “Metaheuristics for Clustering in KDD”, *In Proc. IEEE Congress on Evolutionary Computation*, pp. 2380-2387, 2005.
- [8] S. Bandyopadhyay, U. Maulik, “An Evolutionary Technique based on *k*-Means Algorithm for Optimal Clustering in RN”, *Information Sciences*, Vol. 146, pp. 221-237, 2002.
- [9] V. Estivill-Castro, A. T. Murray, “Spatial Clustering for Data Mining with Genetic Algorithms”, *In Proc. Int. ICSC Symposium on Engineering of Intelligent Systems*, pp. 317-323, 1997.
- [10] P. Fränti, J. Kivijärvi, T. Kaukoranta, O. Nevalainen, “Genetic Algorithms for Large-Scale Clustering Problems”, *The Computer Journal*, Vol. 40, pp. 547-554, 1997.
- [11] J. Kivijärvi, P. Fränti, O. Nevalainen, “Self-Adaptive Genetic Algorithm for Clustering”, *Journal of Heuristics*, Vol. 9, pp. 113-129, 2003.
- [12] K. Krishna, N. Murty, “Genetic K-means Algorithm”, *IEEE Trans. on Systems, Man and Cybernetics – Pt. B*, Vol. 29, pp. 433-439, 1999.
- [13] R. Krovi, “Genetic Algorithms for Clustering: A Preliminary Investigation”, *In Proc. of the 25th Hawaii Int. Conference on System Sciences*, Vol. 4, pp. 540-544, 1992.
- [14] J. C. Bezdek, S. Boggavaparu, L. O. Hall, A. Bensaid, “Genetic Algorithm Guided Clustering”, *In Proc. IEEE Congress on Evolutionary Computation*, pp. 34-40, 1994.
- [15] L. I. Kuncheva, J. C. Bezdek, “Selection of Cluster Prototypes from Data by a Genetic Algorithm”, *In Proc. 5th European Congress on Intelligent Techniques and Soft Computing*, pp. 1683-1688, 1997.
- [16] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, S. J. Brown, “Incremental Genetic k-Means Algorithm and its Application in Gene Expression Data Analysis”, *BMC Bioinformatics*, Vol. 28, 172, 2004.
- [17] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, S. J. Brown, “FGKA: A Fast Genetic K-means Clustering Algorithm”, *In Proc. ACM Symposium on Applied Computing*, pp. 622-623, 2004.

- [18] C. B. Lucasius, A. D. Dane, G. Kateman, "On *k-Medoid* Clustering of Large Data Sets with the Aid of a Genetic Algorithm: Background, Feasibility and Comparison", *Analytica Chimica Acta*, Vol. 282, pp. 647-669, 1993.
- [19] U. Maulik, S. Bandyopadhyay, "Genetic Algorithm- based Clustering Technique", *Pattern Recognition*, Vol. 33, pp. 1455-1465, 2000.
- [20] P. Merz, A. Zell, "Clustering Gene Expression Profiles with Memetic Algorithms", *In Proc. Parallel Problem Solving from Nature*, LNCS 2439, pp. 811-820, 2002.
- [21] C. A. Murthy, N. Chowdhury, "In Search of Optimal Clusters using Genetic Algorithms", *Pattern Recognition Letters*, Vol. 17, pp. 825-832, 1996.
- [22] J. B. McQueen, "Some Methods of Classification and Analysis of Multivariate Observations", In: *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-297, 1967.
- [23] R. M. Cole, *Clustering with Genetic Algorithms*, MSc Thesis, University of Western Australia, Australia, 1998.
- [24] M. C. Cowgill, R. J. Harvey, L. T. Watson, A Genetic Algorithm Approach to Cluster Analysis, *Computational Mathematics and its Applications*, Vol. 37, pp. 99-108, 1999.
- [25] E. R. Hruschka, N. F. F. Ebecken, "A Genetic Algorithm for Cluster Analysis", *Intelligent Data Analysis*, Vol. 7, pp. 15-25, 2003.
- [26] A. Casillas, M. Y. González de Lena, R. Martínez, "Document Clustering into an Unknown Number of Clusters Using a Genetic Algorithm", *In Proc. Int. Conference on Text Speech and Dialogue*, LNCS 2807, pp. 43-49, 2003.
- [27] P.C.H.Ma, K.C.C.Chan, X.Yao, D.K.Y.Chiu, "An Evolutionary Clustering Algorithm for Gene Expression Microarray Data Analysis", *IEEE Transactions on Evolutionary Computation*, Vol. 10, pp. 296-314, 2006.
- [28] V. S. Alves, R. J. G. B. Campello, E. R. Hruschka, "Towards a Fast Evolutionary Algorithm for Clustering", *In Proc. IEEE Congress on Evolutionary Computation*, pp. 6240-6247, 2006.
- [29] M. C. Naldi, A. C. P. L. F. de Carvalho, "Clustering Using Genetic Algorithm Combining Validation Criteria", *In Proc. 15th European Symposium on Artificial Neural Networks*, pp. 139-147, Bruges, Belgium, 2007.
- [30] J. Handl, J. Knowles, "An Evolutionary Approach to Multiobjective Clustering", *IEEE Transactions on Evolutionary Computation*, Vol. 11, pp. 56-76, 2007.
- [31] S. Pan, K. Cheng, Evolution-Based Tabu Search Approach to Automatic Clustering, *IEEE Transactions on Systems, Man, and Cybernetics, Part C – Applications and Reviews*, v. 37, n. 5, pp. 827-838, 2007.
- [32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, 2nd Ed., 2001.

- [33] Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: 'PESA-II: region-based selection in evolutionary multiobjective optimization'. Proc. Genetic Evolutionary Computation Conf., 2001.
- [34] E. E. Korkmaz, J. Du, R. Alhadj, K. Barker, Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. *Intelligent Data Analysis, Vol. 10, No. 2*, pp, 163-182, 2006.
- [35] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, Multiobjective genetic fuzzy clustering of categorical attributes. *Proc. 10th Int. Conf. on Information Technology*, pp. 74-79. IEEE Computer Society, 2007.
- [36] K.S.N. Ripon, C.-H. Tsang, S. Kwong, M.-K. Ip, Multi-objective evolutionary clustering using variable- length real jumping genes genetic algorithm. *Proc. of the 18th Int. Conf. on Pattern Recognition (ICPR'06)*. IEEE Computer Society, 2006.
- [37] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on neural networks*, 5(1):3-14, January 1994.
- [38] Darwin, C. R. The Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life. Penguin Books, Nueva York, EE. UU, 1959.
- [39] John Jenkins, editor. *Genetics*. Houghton Mifflin Company, Boston, Massachusetts, 1984.
- [40] E. Baldwin. *Genetica elemental*. Limusa, 1983.
- [41] Mendel, G. J. Experiments in plant hybridisation. *Journal of Royal Horticultural Society* 26, 1-32 (1901).
- [42] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.
- [43] David E. Goldberg and Kalyanmoy Deb. A comparison of selection schemes used in genetic algorithms. In G.J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69-93. Morgan Kaufmann, San Mateo, California, 1991.
- [44] A Wetzel. Evaluation of Efectiveness of Genetic Algorithms in Combinatorial Optimization. No Publicado, 1983.
- [45] D. Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. David Schaffer, editor, *Proceedings of the Third Conference on Genetic Algorithms*, pages 116-121, San Mateo, California, jun 1989. George Mason University, Morgan Kaufmann Publishers.
- [46] DeJong, K. A. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, EE. UU., 1975.
- [47] John H. Holland. *Adaptation in Natural an Artificial Systems*. MIT Press, Cambridge, Massachusetts, second edition, 1992.
- [48] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In David Fogel, editor, *Proceedings of the first IEEE Conference on Evolutionary Computation*, pages 579-584, Orlando, Florida, 1994. IEEE Press.

- [49] David H. Ackley, editor. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Boston, Massachusetts, 1995.
- [50] Gilbert Syswerda. Uniform Crossover in Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, California, jun 1989. George Mason University, Morgan Kaufmann Publishers.
- [51] Hans-Paul Schwefel, editor. *Evolution and Optimization Seeking*. John Wiley & Sons, New York, 1995.
- [52] Toscano Pulido, G. Optimización multiobjetivo usando un micro algoritmo genético. Master's thesis, Universidad Veracruzana - LANIA, 2001. Director de tesis: Dr. Carlos A. Coello Coello.
- [53] Schwefel, H.-P. *Numerical Optimization of Computer Models*. John Wiley, Chichester, U.K., 1981.
- [54] Bäck, T., and David B. Fogel, y. Z. M., Eds. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, Nueva York, EE. UU., 1997.
- [55] Rechenberg, I. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, Alemania, 1973.
- [56] Schwefel, H.-P. ((adaptive mechanisms in der biologischen evolution und ihr einfluß auf die evolutionsgeschwindigkeit.)) inf. tec. del working group of bionics and evolution techniques at the institute for measurement and control technology. Tech. Rep. 215/3, Technical University of Berlin, 1974.
- [57] K. Deb and S. Agrawal. A Niche-Penalty Approach for Constraint Handling in Genetic Algorithms. In *Proceedings of the ICANNGA*, Portoroz, Slovenia, 1999.
- [58] Fogel, D. B., Ed. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. The Institute of Electrical and Electronic Engineers, New York, EE. UU., 1995.
- [59] Fogel, D. B., Ed. *Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Algorithms*. The Institute of Electrical and Electronic Engineers, New York, EE. UU., 1998.
- [60] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, EE. UU., 1992.
- [61] Storn, R., and Price, K. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-12, International Computer Science, Berkeley, California, March 1995.
- [62] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.
- [63] Dorigo, M., Maniezzo, V., and Colnari, A. The ant system: Optimization by a colony of cooperating agents. In *IEEE Transactions on Systems, Man, and Cybernetics-Part B* 26 (1996), pp. 29–41.

- [64] de Castro, L. N., and Timmis, J. *Artificial Immune Systems: A new Computational Intelligence Approach*. Springer Verlag, 2002.
- [65] Pablo Moscato. *Memetic Algorithms: a Short Introduction*. Eds. McGraw - Hill, 1999.
- [66] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
- [67] Andrzej Osyczka. Multicriteria optimization for engineering design. In John S. Gero, editor, *Design Optimization*, pages 193–227. Academic Press, 1985.
- [68] Edgeworth, F. Y. *Mathematical Physics*. P. Keagan, 1881.
- [69] Vilfredo, P. *Cours D'Economie Politique*. F. Rouge, 1896.
- [70] Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–836, Piscataway, New Jersey, May 2002. IEEE Service Center.
- [71] Cohon, J., and Marks, D. A review and evaluation of multiobjective programming techniques. *Water Resources Research* 11, 2 (1975), 208–220.
- [72] Schaffer, J. D. Multiple objective optimization with vector evaluated genetic algorithms. PhD thesis, Vanderbilt University, Nashville TN, EE. UU., 1984.
- [73] Schaffer, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms* (1985), L. Erlbaum, Ed., pp. 93–100.
- [74] Zitzler, E., Teich, J., and Bhattacharyya, S. S. Evolutionary Algorithm Based Exploration of Software Schedules for Digital Signal Processors. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)* (San Francisco, California, July 1999), W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 2, Morgan Kaufmann, pp. 1762–1769.
- [75] Zitzler, E., and Thiele, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* 3, 4 (November 1999), 257–271.
- [76] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In Schaffer J.D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, Washington, D.C., 1989. Morgan Kauffman Publishers.
- [77] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.
- [78] Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3), 1997.

- [79] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.
- [80] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II., *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.
- [81] K. Ikeda, H. Kita, and S. Kobayashi. Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal? In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pages 957–962 vol. 2, 2001.
- [82] David W. Corne, Joshua D. Knowles, and Martin J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.
- [83] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.
- [84] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.
- [85] Corne, D. W., J. D. Knowles, M. J. Oates. The Pareto Envelope-Based Selection Algorithm for Multi-Objective Optimization. – In: *PPSN 2000. LNCS, Vol. 1917*, Springer, Heidelberg (K. Deb, et al., Eds.), 2000, 839-848.
- [86] Corne, D. W., N. R. Jerram, J. D. Knowles, M. J. Oates. PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization. – In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, San Francisco, California, Morgan Kaufmann Publishers (Lee Spector et al., Eds.), 2001, 283-290.
- [87] Carlos A. Coello Coello, David A. Van Veldhuizen, Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.
- [88] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.
- [89] David A. Van Veldhuizen and Gary B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, pages 204–211, Piscataway, New Jersey, July 2000. IEEE Service Center.
- [90] Zitzler, E., Deb, K., & Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, Vol. 8(2), pages. 173-195, 2000.

- [91] Zitzler, E. and Thiele, L. (1998). Multiobjective Optimization Using Evolutionary Algorithms: A Comparative Case Study. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, PPSN, volume 1498 of Lecture Notes in Computer Science, pages 292–304. Springer.
- [92] Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- [93] Zitzler, E., Brockhoff, D., and Thiele, L. (2006). The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, EMO, volume 4403 of Lecture Notes in Computer Science, pages 862–876. Springer.
- [94] Ferligoj, A. and Batagelj, V. (1992). Direct multicriterion clustering. *Journal of Classification*, 9:43–61.
- [95] Handl, J. and Knowles, J. (2004). Evolutionary multiobjective clustering. *Lecture notes in computer science*, pages 1081–1091.
- [96] Handl, J. and Knowles, J. (2004). Multiobjective clustering with automatic determination of the number of clusters. In *Technical Report TR-COMPSYSBIO-2004-02*. UMIST.
- [97] Y. J. Park and M. S. Song. A genetic algorithm for clustering problems. In *Proceedings of the Third Annual Conference on Genetic Programming*, pages 568-575, Madison, WI, 1998. Morgan Kaufmann.
- [98] Robin J. Wilson and John J. Watkins. *Graphs: an introductory approach: a first course in discrete mathematics*. John Wiley and Sons, New York, NY, 1990.
- [99] Ch. Ding and X. He. K-nearest-neighbour consistency in data clustering: incorporating local information into global optimization. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 584-589, New York, NY, 2004. ACM Press.
- [100] C. Blake and C. Merz. UCI repository of machine learning databases. Department of Information and Computer Sciences, University of California, Irvine. <http://www.ics.uci.edu/mllearn/MLRepository.html>, 1998.
- [101] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* doi:10.1016/j.swevo.2011.02.002, pages 1-16, 2011.
- [102] Yen, G.G. (2009), “An adaptive penalty function for handling constraint in multi-objective evolutionary optimization,” (Vol. 198, ed. E. Mezura-Montes, Springer-Verlag, Studies in Computational Intelligence Series, ISBN:978-3-642-00618-0, pp. 121–143.
- [103] Ray, T., Singh, H.K., Isaacs, A., and Smith, W. (2009), “Infeasibility Driven Evolutionary Algorithm for Constrained Optimization,” (Vol. 198, ed. E. Mezura-Montes, Springer-Verlag, Studies in Computational Intelligence Series, ISBN:978-3-642-00618-0, pp. 145–165.

- [104] Deb, K. (2000), “An Efficient Constraint Handling Method for Genetic Algorithms,” *Computer Methods in Applied Mechanics and Engineering*, 186(2/4), 311–338.
- [105] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002), “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197.
- [106] R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.
- [107] Balaprakash P, Birattari M, Stützle T (2007). “Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement.” In T Bartz-Beielstein, MJ Blesa, C Blum, B Naujoks, A Roli, G Rudolph, M Sampels (eds.), *Hybrid Metaheuristics*, volume 4771 of *Lecture Notes in Computer Science*, pp. 108–122. Springer-Verlag, Heidelberg, Germany.
- [108] Birattari M, Yuan Z, Balaprakash P, Stützle T (2010). “F-Race and Iterated F-Race: An Overview.” In T Bartz-Beielstein, M Chiarandini, L Paquete, M Preuss (eds.), *Experimental Methods for the Analysis of Optimization Algorithms*, pp. 311–336. Springer-Verlag, Berlin, Germany.
- [109] López-Ibáñez M, Stützle T (2010). “Automatic Configuration of Multi-Objective ACO Algorithms.” In M Dorigo, et al. (eds.), *Swarm Intelligence*, 7th International Conference, ANTS 2010, volume 6234 of *Lecture Notes in Computer Science*, pp. 95–106. Springer- Verlag, Heidelberg, Germany.
- [110] Montes de Oca MA, Aydin D, Stützle T (2011). “An Incremental Particle Swarm for Large-Scale Continuous Optimization Problems: An Example of Tuning-in-the-loop (Re)Design of Optimization Algorithms.” *Soft Computing*, 15(11), 2233–2255.