Diseño e implementación de un modelo subrogado en Evolución Diferencial para optimización con restricciones

TESIS DOCTORAL

Mariana Edith Miranda Varela

Universidad Veracruzana
Centro de Investigación en Inteligencia Artificial
Doctorado en Inteligencia Artificial

Enero 2018

Diseño e implementación de un modelo subrogado en Evolución Diferencial para optimización con restricciones

Tesis para obtener el grado de
Doctorado en Inteligencia Artificial

Dirigida por el Doctor
Efrén Mezura Montes

Universidad Veracruzana
Centro de Investigación en Inteligencia Artificial
Doctorado en Inteligencia Artificial

Enero 2018

Agradecimientos

Sé el cambio que quieres ver en el mundo
Mahatma Gandhi

A mi familia por su apoyo y comprensión a lo largo de esta etapa de mi vida.

Al Dr. Efrén Mezura por brindarme la oportunidad de conocer y aprender un tópico más del área de Cómputo Evolutivo.

Al Dr. Tapabrata Ray por la oportunidad de realizar una estancia académica en la Universidad de Nueva Gales del Sur. Asimismo, por sus sugerencias para mejorar la propuesta del presente trabajo.

A la Dra. Alicia Morales Reyes por la oportunidad de efectuar una estancia académica en el Instituto Nacional de Astrofísica Óptica y Electrónica, y por sus observaciones a la propuesta del presente trabajo.

A todas las personas que tuve la oportunidad de conocer en esta etapa de mi vida, compañeros y amigos del CIIA, de Reto, *roomies* y todas aquellas con las cuales entablé una amistad.

Al CONACyT el apoyo otorgado para cursar el doctorado.

A los revisores por sus observaciones y sugerencias para mejorar el presente documento.

Abstract

The number of evaluations used by an evolutionary algorithm (EA) in order to find a competitive solution is often high. Consequently, the computational time is high in most of the cases. Thus, this number can be reduced by the use of surrogate models, which can approximate the evaluation of a solution. Although there are many studies and applications of surrogate models in unconstrained optimization problems, their use in constrained numerical optimization problems (CNOPs) has been less explored. In this work, the incorporation of a surrogate model in the algorithm Differential Evolution Combined Variants (DECV) will allow obtaining competitive results on CNOPs, with using a lower number of evaluations. Then, two approaches are proposed: (1) A unique surrogate model is used to approximate the objective function value and the sum of constraint violation, whose aim is to study the performance of several constraint-handling techniques (CHTs). (2) Several surrogate local models are used to approximate the solutions on a global search and a limited search. According to the results, both approaches are able to obtain competitive results employing a lower number of evaluations. Furthermore, the most appropriate CHT is that preserves infeasible solutions with a good objective function value, and the direct-trajectory-based methods are more suitable in order to solve CNOPs with local surrogate models. Finally, an evolution control based on feasibility is a good option for EAs with generational replacement and a good exploratory ability.

Resumen

El número de evaluaciones usadas por un algoritmo evolutivo para encontrar una solución competitiva es frecuentemente alto. Como consecuencia, el tiempo de cómputo es alto en la mayoría de los casos. Así, este número puede ser reducido mediante el uso de modelos subrogados, los cuales pueden aproximar la evaluación de una solución. Aunque hay muchos estudios y aplicaciones de modelos subrogados en optimización sin restricciones, su uso en problemas de optimización numérica con restricciones ha sido menos explorado. En este trabajo, la incorporación de un modelo subrogado en el algoritmo Evolución Diferencial con Variantes Combinadas (DECV) permitirá obtener resultados competitivos en problemas de optimización con restricciones, usando un menor número de evaluaciones. Por tanto, dos enfoques se propusieron: (1) Un modelo subrogado único que se emplea para aproximar el valor de la función objetivo y la suma de violación de restricciones, cuyo objetivo es estudiar el desempeño de diferentes técnicas para manejo de restricciones. (2) Varios modelos subrogados locales se usaron para aproximar las soluciones en una búsqueda global y una búsqueda acotada. De acuerdo con los resultados, ambas propuestas son capaces de obtener resultados competitivos empleando un menor número de evaluaciones. Además, la técnica para manejo de restricciones más apropiada es aquella que conserva soluciones no factibles con un buen valor de la función objetivo, y los métodos basados en trayectoria son los más adecuados para resolver problemas con optimización restringida con modelos subrogados locales. Finalmente, un control de evolución basado en factibilidad es una buena opción para EAs con remplazo generacional y con una buena capacidad de exploración.

Índice

Agradecimientos v

Resumen ix

I Capítulos 1

1. Introducción 3
- 1.1. Motivación . . . 3
- 1.2. Hipótesis . . . 5
- 1.3. Objetivos . . . 5
- 1.4. Metodología . . . 6
- 1.5. Alcances y limitaciones . . . 7
- 1.6. Publicaciones . . . 7
- 1.7. Estructura del documento . . . 8

2. Optimización con restricciones 9
- 2.1. Conceptos básicos . . . 9
- 2.2. Optimización restringida . . . 11
- 2.3. Algoritmos evolutivos . . . 12
  - 2.3.1. Evolución Diferencial . . . 14
- 2.4. Técnicas para manejo de restricciones . . . 18
  - 2.4.1. Reglas de Factibilidad . . . 19
  - 2.4.2. Mecanismo de diversidad . . . 19
  - 2.4.3. Método ε-*constrained* . . . 20
  - 2.4.4. Jerarquización estocástica . . . 20
- 2.5. Resumen del capítulo . . . 21

3. Optimización asistida por subrogados **23**
3.1. Uso de modelos subrogados 23
3.2. Muestreo inicial 25
3.3. Métodos subrogados 27
3.3.1. k-vecinos más cercanos 27
3.3.2. Modelos polinomiales 28
3.3.3. Redes neuronales artificiales 29
3.3.4. Máquinas de soporte vectorial 31
3.3.5. Kriging 34
3.4. Criterios de muestreo de relleno 35
3.5. Métodos de evaluación de modelos subrogados 35
3.5.1. Error absoluto con respecto al óptimo 36
3.5.2. *Bias* o sesgo 36
3.5.3. Varianza 36
3.6. Resumen del capítulo 37

4. Algoritmos Evolutivos Asistidos por Subrogados **39**
4.1. Integración de un modelo subrogado en un algoritmo evolutivo 39
4.1.1. Conjunto de entrenamiento 41
4.1.2. Control de evolución 41
4.1.3. Medida de calidad del modelo 41
4.1.4. Modelos subrogados 42
4.1.5. Construcción de un modelo subrogado 42
4.2. Marco referencial 43
4.3. Resumen del capítulo 47

5. SA-DECV **49**
5.1. Propuesta 49
5.1.1. Población inicial 51
5.1.2. Conjunto de entrenamiento 51
5.1.3. Control de evolución 51
5.2. Diseño experimental 53
5.2.1. Medidas de desempeño 54
5.2.2. Comparación de SA-DECV con diferentes técnicas para manejo de restricciones 58
5.2.3. Comparación de SA-DECV$_{SR}$ contra algoritmos del estado del arte en SA-EA 72
5.3. Caso de estudio: problemas de optimización de ingeniería . . . 79

5.4. Resumen del capítulo 82

6. Estudio del control de evolución basado en factibilidad 83
6.1. Diseño experimental 83
6.1.1. Resultados estadísticos 84
6.2. Resumen del capítulo 90

7. SA-DECV con búsqueda acotada 91
7.1. Propuesta de SA-DECV con modelos subrogados locales . . . 91
7.2. Algoritmo 92
7.2.1. Población inicial 92
7.2.2. Evaluación y aproximación de soluciones 94
7.2.3. Conjunto de entrenamiento 94
7.2.4. Generación de hijos 94
7.2.5. Modelos subrogados locales 94
7.2.6. Distancia entre soluciones 95
7.2.7. Búsqueda acotada 95
7.2.8. Actualización 97
7.3. Diseño experimental 97
7.3.1. Análisis de convergencia 98
7.3.2. Resultados estadísticos 98
7.3.3. Medidas de desempeño 102
7.3.4. Restricciones de igualdad 105
7.3.5. Tolerancia dinámica para restricciones de igualdad . . 105
7.3.6. Método de Hooke-Jeeves 106
7.3.7. Análisis de convergencia 109
7.3.8. Discusión de resultados 116
7.4. Resumen del capítulo 116

8. Conclusiones 117
8.1. Trabajo futuro 118

II Apéndices 119

A. Problemas CEC 2006 121

B. Problemas de Diseño en Ingeniería 135

Índice de figuras

2.1. Problema 2D con dos restricciones. 12
2.2. Proceso de búsqueda de un EA. 13
2.3. Pseudocódigo de DECV. 17
2.4. Pseudocódigo de SR 21

3.1. Ejemplo de un LHS en dos dimensiones. 26
3.2. Clasificación lineal de dos muestras mediante un hiperplano. . 32
3.3. Pérdida de margen suave asignada por SVR. 33

4.1. Elementos de un SA-EA 40

5.1. Pseudocódigo de SA-DECV 50
5.2. Pseudocódigo del control de evolución. 52
5.3. Pseudocódigo de la función *Choose_ vectors*. 53
5.4. Soluciones seleccionadas por el control de evolución basado en factibilidad 54
5.5. Gráficas de convergencia de los problemas g14, g16 y g23. . . 60
5.6. Curvas de nivel de los problemas g08 y g11. 62
5.7. Convergencia de la población en el problema g08. 64
5.8. Convergencia de la población en el problema g11. 66
5.9. Valores de FP en las cuatro variantes de SA-DECV. 69
5.10. Valores de P en las cuatro variantes de SA-DECV. 70
5.11. Valores de AFES en las cuatro variantes de SA-DECV. 71
5.12. Valores de SP en las cuatro variantes de SA-DECV. 71
5.13. Valores de FP para SA-DECV$_{SR}$, ASRES y SBSM-GA. . . . 76
5.14. Valores de P para SA-DECV$_{SR}$, ASRES y SBSM-GA. 76
5.15. Valores de AFES para SA-DECV$_{SR}$, ASRES y SBSM-GA. . . 77
5.16. Valores de SP para SA-DECV$_{SR}$, ASRES y SBSM-GA. . . . 78

6.1. Valores de FP para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. 87
6.2. Valores de P para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. 88
6.3. Valores de AFES para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. 89
6.4. Valores de SP para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. 89

7.1. Búsqueda global aproximada en SA-DECV$_L$ 92
7.2. Búsqueda acotada aproximada en SA-DECV$_L$ 93
7.3. Pseudocódigo de SA-DECV$_L$. 93
7.4. Ubicación del mejor aproximado en P 96
7.5. Extensión de los límites de $\mathcal{S}_{\hat{x}_b}$ 97
7.6. Gráficas de convergencia de los problemas g03 y g10 99
7.7. Pseudocódigo del movimiento exploratorio de Hooke-Jeeves. . 106
7.8. Pseudocódigo del método de Hooke-Jeeves. 107
7.9. Pseudocódigo de SA-DECV$_L$ 108
7.10. Gráficas de convergencia de los problemas g07 y g13 110

Índice de Tablas

2.1. Diferencias entre reglas de factibilidad, mecanismo de diversidad, método ε-*constrained* y jerarquización estocástica. 22

5.1. Detalles de los 24 problemas de prueba. 55
5.2. Parámetros de SA-DECV. 57
5.3. Parámetros para cada CHT. 57
5.4. Resultados estadísticos de los problemas g01-g11 obtenidos por las variantes de SA-DECV. 67
5.5. Resultados estadísticos de los problemas g12-g24 obtenidos por las variantes de SA-DECV. 68
5.6. Resultados del test de suma de rangos de Wilcoxon para las variantes de SA-DECV. 69
5.7. Resultados estadísticos de los problemas g01-g11 para SA-DECV$_{SR}$, ASRES y SBSM-GA 73
5.8. Resultados estadísticos de los problemas g12-g24 para SA-DECV$_{SR}$, ASRES y SBSM-GA 74
5.9. Resultados del test de suma de rangos de Wilcoxon de SA-DECV, ASRES y SMSB-GA. 75
5.10. Detalles de los problemas de ingeniería. 79
5.11. Resultados estadísticos de los problemas de optimización de ingeniería. 80

6.1. Resultados estadísticos de los problemas g01-g11 obtenidos por SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. 85
6.2. Resultados estadísticos de los problemas g12-g24 obtenidos por SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. 86
6.3. Resultados del test de suma de rangos de Wilcoxon de SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. 87

7.1. Resultados estadísticos de los problemas g01-g09 obtenidos por las variantes de SA-DECV$_L$. 100
7.2. Resultados estadísticos en los problemas g10-g24 obtenidos por las variantes de SA-DECV$_L$. 101
7.3. Resultados del test de suma de rangos de Wilcoxon para las variantes de SA-DECV$_L$ 102
7.4. Valores de FP y P para las variantes de SA-DECV$_L$. 103
7.5. Valores de AFES y SP para las variantes de SA-DECV$_L$ . . . 104
7.6. Resultados estadísticos de los problemas g01-g09 obtenidos por SA-DECV$_L$ con diferente algoritmo de búsqueda en $\mathcal{S}_{\hat{x}_b}$. 112
7.7. Resultados estadísticos de los problemas g10-g24 obtenidos por SA-DECV$_L$ con diferente algoritmo de búsqueda en $\mathcal{S}_{\hat{x}_b}$. 113
7.8. Resultados del test de suma de rangos de Wilcoxon para variantes de SA-DECV$_L$ con diferente algoritmo de búsqueda en $\mathcal{S}_{\hat{x}_b}$ 114
7.9. Valores de FP y P para las variantes de SA-DECV$_L$ con diferente algoritmo de búsqueda en $\mathcal{S}_{\hat{x}_b}$ 114
7.10. Valores de AFES y SP para las variantes de SA-DECV$_L$ con diferente algoritmo de búsqueda en $\mathcal{S}_{\hat{x}_b}$ 115

A.1. Conjunto de datos para el problema g19. 130
A.2. Conjunto de datos para el problema g20. 132

Parte I

Capítulos

Capítulo 1

Introducción

En el presente capítulo se presenta la motivación para el desarrollo de este trabajo de tesis, además se definen su hipótesis y objetivos. También se explica la metodología que se sigue para desarrollarlo y se exponen sus alcances y limitaciones.

1.1. Motivación

Muchos problemas de optimización en ingeniería y ciencias aplicadas se definen mediante modelos costosos y complejos, es decir, un conjunto de funciones expresadas por un gran número de variables, lo cual se refleja en un mayor tiempo de procesamiento (costo computacional); una simulación o un experimento [28]. En cualquiera de los casos anteriores, la evaluación de una solución en uno de esos modelos implica varios minutos, horas o incluso días. Ejemplos de problemas de optimización que necesitan simulaciones son: el diseño de perfiles aerodinámicos [1], la gestión óptima de los campos petroleros [26], diseño óptimo de circuitos basados en electromagnetismo [24], entre otros. Con respecto a los problemas con un modelo costoso, se tiene un problema automotriz, denominado MOPTA08 [32], el cual está definido por 124 variables de decisión y 68 restricciones de desigualdad.

Existen diversos métodos para resolver este tipo de problemas, entre los que se encuentran los algoritmos evolutivos, los cuales necesitan de muchas evaluaciones para encontrar una solución competitiva [6]. Lo anterior se convierte en una desventaja para resolver un problema costoso o complejo debido al tiempo requerido para evaluar cada solución.

Una alternativa para reducir el tiempo requerido para evaluar una solución es el uso de modelos subrogados, los cuales aproximan la evaluación de

las soluciones mediante modelos menos costosos que el original [27]. Diversos modelos se han integrado en un algoritmo evolutivo, a tal combinación se le conoce como algoritmos evolutivos asistidos por subrogados. Un beneficio que se obtiene al aproximar las soluciones, es la reducción del número de evaluaciones en el modelo original, y por consiguiente el costo computacional del proceso de optimización empleado por un algoritmo evolutivo[28].

Los modelos que se han empleado para aproximar soluciones en un algoritmo evolutivo son [19, 72]: k-vecinos más cercanos, perceptrón multicapa, función de base radial, metodología de superficies de respuesta, máquinas de soporte vectorial y *kriging*. Estos modelos pueden ser globales o locales, es decir, si un modelo se construye a partir soluciones que están distribuidas en todo el espacio de búsqueda, el modelo es global, mientras que el modelo que se construye con soluciones dentro de una región del espacio, el modelo es local.

La tendencia principal del uso de modelos subrogados en algoritmos evolutivos es construir un modelo por cada función del problema, el cual se actualiza a lo largo del proceso de búsqueda, con un alcance global, local, o la combinación de ambos. También, hay propuestas que emplean varios modelos a lo largo del proceso de búsqueda [6, 71], para definir el modelo más adecuado se emplea un criterio de selección, generalmente, con base en la precisión de cada modelo.

Existe una gran diversidad de algoritmos evolutivos para resolver problemas de optimización, sin embargo uno de los algoritmos con menos parámetros, fácil de implementar y eficiente es evolución diferencial. La evolución diferencial es un algoritmo del cual se han derivado diferentes variantes, entre las que se encuentra la evolución diferencial con variantes combinadas (DECV, por sus siglas en inglés). Aunque la evolución diferencial no fue diseñada para resolver problemas de optimización numérica restringida, se le puede integrar una técnica para manejo de restricciones, la cual guiará la búsqueda hacia la zona factible.

La incorporación de un modelo subrogado en algoritmos evolutivos se ha empleado principalmente para resolver problemas sin restricciones. Por esta razón, la exploración y conocimiento del uso de modelos subrogados aplicados en problemas de optimización numérica restringida es menor. Además, la aproximación de soluciones en algoritmos evolutivos para evaluar problemas de optimización numérica con restricciones se ha enfocado principalmente en construir un modelo para cada función, por lo tanto, el uso de una técnica para manejo de restricciones considerando la suma de violación de restricciones aproximada se ha estudiado con menos detalle. Como consecuencia, en este

trabajo se estudia el uso de modelos aproximados para resolver problemas de optimización numérica con restricciones desde dos perspectivas: (1) la aproximación de soluciones mediante un único modelo, y (2) la aproximación de soluciones mediante varios modelos aproximados locales.

El propósito de este trabajo es aproximar soluciones mediante un modelo subrogado en el algoritmo DECV, lo que permitirá reducir el número de evaluaciones, dicho enfoque se denomina DECV asistido por subrogados (SA-DECV, por sus siglas en inglés). En un primer enfoque, SA-DECV aproxima tanto el valor de la función objetivo como la suma de violación de restricciones mediante un único modelo subrogado, el cual es kNN [64], con el objetivo de estudiar que técnica para manejo de restricciones es la más adecuada. No obstante, se realizan modificaciones a SA-DECV con el propósito de reducir más el número de evaluaciones, los cuales son: construir un modelo por cada función del problema, el cual es ϵ-SVR [8]. Además, se realiza una búsqueda acotada en cada generación, en donde se encuentra la solución más prometedora, es decir, la mejor solución aproximada.

1.2. Hipótesis

La integración de un modelo subrogado en el algoritmo de Evolución Diferencial con Variantes Combinadas logra obtener resultados competitivos con un menor número de evaluaciones en problemas de optimización con restricciones.

1.3. Objetivos

El objetivo principal de este trabajo es desarrollar un algoritmo basado en DECV, integrando un modelo subrogado para resolver problemas de optimización con restricciones, reduciendo el número de evaluaciones realizadas en el modelo original.

Los objetivos particulares de este trabajo son:

- Integrar un modelo subrogado en el algoritmo de DECV, denominado SA-DECV.
- Diseñar experimentos, probar y analizar resultados.
- Contar el número de evaluaciones en el modelo original de cada problema para verificar el ahorro en el número de evaluaciones.

- Comparar resultados con algoritmos del estado del arte.
- Mejorar la propuesta SA-DECV considerando un menor número de evaluaciones que el considerado en la primera propuesta.
- Diseñar experimentos, probar y analizar resultados.

1.4. Metodología

En lenguaje coloquial, la investigación es la búsqueda de nuevo conocimiento, ya que resuelve preguntas y da solución a problemas [35]. Existen dos tipos de metodología: cualitativa y cuantitativa, las cuales tienen sus raíces en la filosofía naturalista y positivista, respectivamente. La metodología cualitativa se relaciona a fenómenos sociales, es decir, se centra en comprender las acciones y el comportamiento de las personas. Su información se obtiene a través de entrevistas, cuestionarios y observaciones. Por otro lado, la metodología cuantitativa se aplica a fenómenos que se expresan en términos de cantidades [38]. Para analizar dicha información se emplea la estadística, el modelado matemático y simulaciones. Además, esta metodología se refiere de manera frecuente como la investigación de prueba de hipótesis [52].

Dada la naturaleza de este trabajo, la metodología adecuada es la cuantitativa, con la cual se podrá concluir si el uso de modelos subrogados en DECV reportará resultados competitivos en problemas de optimización con restricciones empleando menos evaluaciones.

El enfoque de investigación que se emplea en este trabajo es el deductivo, pues se parte de que el uso de modelos subrogados en DECV reducirá el número de evaluaciones reportando resultados competitivos. La información se genera mediante experimentos, para analizarla y compararla con otras propuestas del estado del arte. Como consecuencia se corroborará o refutará la hipótesis.

La recopilación de datos para poder validar la hipótesis se lleva a cabo realizando experimentos, los cuales consisten en probar la propuesta en un conjunto de problemas con restricciones, los cuales tienen diferentes características, tales como: el número de variables (dimensionalidad), tipo de restricciones (desigualdad, igualdad o ambas), tipo de función (lineal, no lineal, polinomial, entre otros), etc. Posteriormente los resultados se analizan mediante pruebas estadísticas, medidas específicas para problemas de opti-

mización con restricciones[1] y contando el número de evaluaciones realizadas en el modelo original, para determinar el ahorro de evaluaciones en cada problema.

Dada la naturaleza estocástica del algoritmo DECV y la aproximación de soluciones mediante los modelos subrogados, se realizan repeticiones de los experimentos por cada función de prueba. Lo anterior permite determinar la robustez y eficiencia de la propuesta.

1.5. Alcances y limitaciones

Los alcances del presente trabajo son:

1. El estudio del uso de modelos subrogados en DECV para resolver problemas de optimización con restricciones, ya sea mediante un único modelo para la suma de violación de restricciones y función objetivo o para cada restricción y función objetivo.

2. La variedad de los problemas de estudio abarca tanto problemas artificiales como problemas de ingeniería.

Actualmente no existe un conjunto de problemas de optimización con restricciones con las características necesarias para el uso de modelos subrogados, es decir, definido mediante modelos costosos computacionalmente. Sin embargo, hay una gran variedad de problemas con restricciones, los cuales son empleados por los algoritmos del estado del arte para definir las bases del uso de modelos subrogados en este tipo de problemas.

1.6. Publicaciones

Derivado de este trabajo de tesis se publicaron dos artículos: uno de congreso y uno de revista. En el primero se presenta un control de evolución basado en factibilidad, el cual decide que soluciones se van a aproximar y cuales se van a evaluar en el modelo original. Por otro lado, en el segundo se analiza la influencia positiva o negativa de una técnica para manejo de restricciones específica, considerando tanto valores aproximados como exactos para la suma de violación de restricciones.

[1] Estas medidas consideran sí la solución encontrada se encuentra en la zona factible y sí se encuentra en la vecindad del óptimo.

1. M.E. Miranda-Varela and E. Mezura-Montes. Surrogate-assisted differential evolution with an adaptive evolution control based on feasibility to solve constrained optimization problems. In M. Pant, K. Deep, J. C. Bansal, A. Nagar, and K. N. Das, editors, Proceedings of Fifth International Conference on Soft Computing for Problem Solving: SocProS 2015, Volume 1, pages 809-822. Springer Singapore, Singapore, 2016.

2. M.E. Miranda-Varela and E. Mezura-Montes. Constraint-Handling Techniques in Surrogate-Assisted Evolutionary Optimization. An empirical study. Applied Soft Computing.

1.7. Estructura del documento

La organización del resto del documento es la siguiente:

En el capítulo 2 se explican los conceptos de optimización y cómputo evolutivo de manera general. Además, se expone el problema que se resuelve en este trabajo.

En el capítulo 3 se expone el uso de modelos subrogados en optimización y los elementos que se deben considerar para su uso.

En el capítulo 4 se explica el uso de modelos subrogados en algoritmos evolutivos. Además, se listan todos los trabajos que emplean modelos subrogados para resolver problemas de optimización con restricciones.

En el capítulo 5 se describe la propuesta de SA-DECV con un único modelo subrogado, se presentan sus resultados y su análisis.

En el capítulo 6 se compara el mejor algoritmo del capítulo 5 contra algoritmos del estado del arte en optimización restringida, a los cuales se les incorpora un modelo subrogado.

En el capítulo 7 se describen las modificaciones que se realizan a SA-DECV para resolver problemas de optimización con restricciones, se muestran los resultados y su análisis.

En el capítulo 8 se presentan las conclusiones de la presente tesis y el trabajo futuro.

Capítulo 2

Optimización con restricciones

En este capítulo se presentan los conceptos básicos de la teoría de optimización y computación evolutiva, los cuales están relacionados con el presente trabajo. Además, se define el tipo de problema y se profundiza en el método de búsqueda que se emplea para resolverlo, en este caso un algoritmo evolutivo, denominado Evolución Diferencial.

2.1. Conceptos básicos

La teoría de optimización clásica, también denominada programación matemática, tiene como objetivo encontrar la solución más adecuada, dentro de un conjunto de posibles soluciones, de un problema de optimización, lo que se realiza mediante un método de búsqueda.

La definición de un problema de optimización puede involucrar diferentes aspectos: distintos tipos de datos, es decir, variables continuas, discretas o ambas, una o varias funciones objetivo, con o sin restricciones; inclusive, las características del problema pueden cambiar con el tiempo.

Desde el punto de vista del cómputo inteligente, un problema de optimización se define mediante [15]:

Función objetivo. Representa la cantidad que será optimizada. Se denota mediante f.

Variables de decisión. También se denominan como variables de diseño, o simplemente variables, y varían la cantidad de la función objetivo mediante los valores que se les asignan. Se representan mediante un vector $\vec{x}$ y son las variables independientes de la función objetivo $f(\vec{x})$.

Restricciones. Limitan los valores que pueden tomar las variables de decisión. Además, existen otro tipo de restricciones que están relacionados con características de diseño y que deben cumplirse.

Los problemas de optimización se pueden clasificar de acuerdo a las características de cada uno de los elementos previamente mencionados, excepto el último criterio (número de óptimos) [15]:

Número de variables. Si la definición del problema está en función de una variable es *univariable*. En caso de que sean más, el problema se conoce como *multivariable*.

Tipo de variables. Si el dominio de las variables son los reales, el problema es *continuo*. Si el dominio de las variables son los enteros, el problema es *discreto* o *entero*. Si las variables tienen un dominio real y entero, el problema es *mixto-entero*.

Grado de la función objetivo. El grado del problema está asociado al grado de la función objetivo, así, si la función objetivo es lineal, el problema es *lineal*. Si la función objetivo es cuadrática, el problema es *cuadrático*. Si la función objetivo es no-lineal, el problema es *no-lineal*.

Uso de restricciones. Cuando la definición del problema sólo involucra la restricción de límites, el *problema es sin restricciones*. En caso de que el problema tenga restricciones, de igualdad o desigualdad, el *problema es con restricciones*.

Número de funciones objetivo. Si la cantidad del problema está definida mediante una función objetivo, el problema es *mono-objetivo*. En caso de que se defina con más de una función, el problema es *multiobjetivo*. Una característica del último es que los valores de cada función objetivo deben estar en conflicto, es decir, en unas funciones el mejor valor debe ser el mínimo y en el resto, el mejor valor es el máximo.

Número de óptimos. Si el problema tiene un único óptimo, el problema es *unimodal*. En caso de que el problema tenga más de un óptimo, el problema es *multimodal*.

En el presente trabajo el objeto de estudio son los problemas de optimización con restricciones, cuya definición formal se presenta en la siguiente sección.

2.2. Optimización restringida

Un problema de optimización numérica restringida se define como:

Minimizar

$$f(\vec{x}), \vec{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \qquad (2.1)$$

sujeto a

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, m \qquad (2.2)$$
$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \qquad (2.3)$$

donde f es la función objetivo, m es el número de restricciones de desigualdad, p es el número de restricciones de igualdad, y $\mathcal{S}$ es el espacio de búsqueda ($\mathcal{S} \subseteq \mathbb{R}^n$) el cual está delimitado por los límites inferior y superior de las variables de diseño, denotados por $\vec{L}$ y $\vec{U}$, respectivamente. La región factible $\mathcal{F} \subseteq \mathcal{S}$ contiene aquellas soluciones que satisfacen todas las restricciones del problema. Para satisfacer las restricciones de igualdad (Ec. 2.3), éstas se transforman en restricciones de desigualdad (Ec. 2.4), donde ϵ es una tolerancia permitida, cuyo valor es 1E-4, generalmente.

$$|h_j(\vec{x})| - \epsilon \leq 0, \quad j = 1, \ldots, p \qquad (2.4)$$

Todas aquellas restricciones (igualdad o desigualdad) que toman un valor de cero cuando se alcanza el óptimo de la función, se denominan *restricciones activas* (las restricciones de igualdad son activas por definición). En la Fig. 2.1 se ilustra el espacio de búsqueda y zona factible de un problema con 2 variables de decisión, x_1 y x_2, cuyo rango es [0,3] y [0,4], respectivamente. La definición del problema es:

Minimizar:

$$f(\vec{x}) = -x_1 - x_2$$

sujeto a:

$$\begin{aligned} g_1 &= -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0 \\ g_2 &= -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0 \end{aligned} \qquad (2.5)$$

El problema tiene una región factible compuesta de 2 sub-regiones desconectadas. Además g_1 y g_2 son restricciones activas.

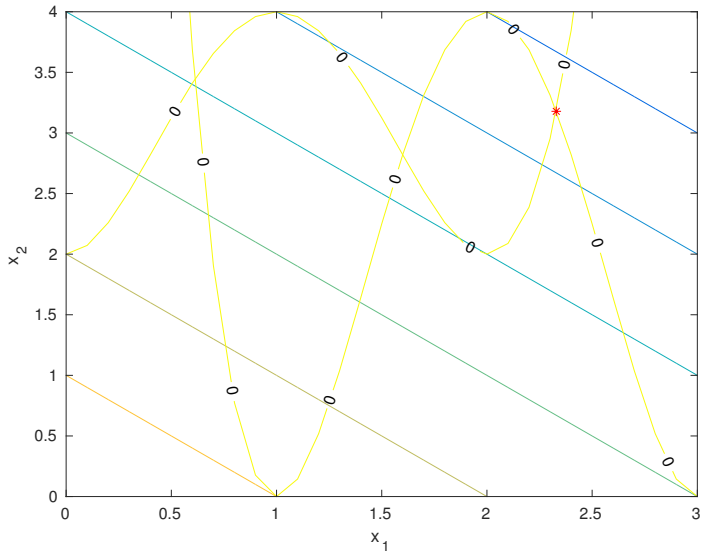


Figura 2.1: Problema 2D con dos restricciones. La región factible se compone de dos regiones disjuntas (intersección de líneas amarillas). El óptimo es el asterisco rojo y por su ubicación ambas restricciones son activas

2.3. Algoritmos evolutivos

Un algoritmo evolutivo es una búsqueda estocástica, que simula el proceso de evolución natural, donde el concepto principal es la supervivencia del más apto.

Los algoritmos evolutivos emplean un conjunto de soluciones, denominado población, por consiguiente, una solución es un individuo. Cada individuo se llama cromosoma y se representa por una cadena de símbolos, la cual generalmente se compone de números binarios, enteros o reales. A partir de la población actual (denominados padres), se producen nuevos individuos (denominados hijos) mediante los operadores de variación, cruza o recombinación, y/o mutación. En la cruza, los padres compiten para reproducirse, así los mejores combinan su información para generar nuevos individuos. Por otro lado, en la mutación se altera parcialmente el cromosoma. Al final de cada generación, se calcula el valor de aptitud de los hijos, mediante la función de aptitud, para determinar cuáles son más aptos para sobrevivir, y el resto se extingue. En la Fig. 2.2 se aprecia este proceso.

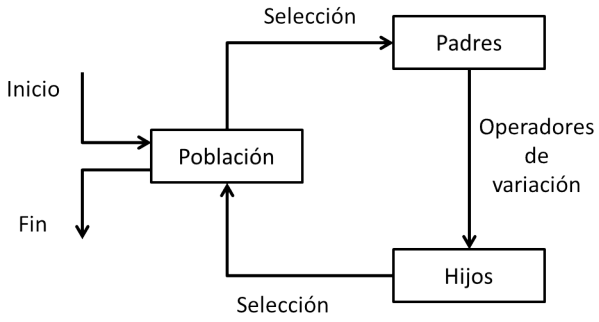


Figura 2.2: Proceso de búsqueda de un algoritmo evolutivo, el cual simula el proceso de evolución natural.

Existen diferentes clases de algoritmos evolutivos, los cuales son [15]:

Algoritmos genéticos Los algoritmos genéticos fueron propuestos por Holland [25] en los años 70s. Un algoritmo genético modela la evolución genética mediante los principios de la selección natural y la supervivencia del más fuerte. En sus orígenes, la representación de un individuo fue mediante un vector de números binarios de longitud fija, aunque actualmente también se emplean vectores de números enteros y reales. La cruza se considera el operador de variación más importante para crear la nueva generación.

Programación genética La programación genética tiene sus orígenes en los algoritmos genéticos. La diferencia entre ellos es la representación de los individuos, pues en la programación genética, un individuo se representa mediante árboles. Koza [39] desarrolló la programación genética para evolucionar programas de cómputo.

Programación evolutiva La programación evolutiva fue propuesta por Fogel [16] en 1962 y representa la evolución como un proceso de comportamiento adaptativo de especies. Además, su población está formada por un conjunto de máquinas de estados finitos, a la cual se le aplica el operador de mutación para generar una nueva población.

Estrategias Evolutivas Las estrategias evolutivas fueron propuestas por Rechenberg en los años 60s. Las estrategias evolutivas tienen como base la *evolución de la evolución*, por ello manejan la auto-adaptación

a nivel de individuo. La representación de un individuo es mediante un vector de números reales, y contiene la información de la solución y un conjunto de parámetros de la estrategia, los cuales se asocian al comportamiento del individuo en el ambiente.

Evolución Diferencial El algoritmo de evolución diferencial fue propuesto por Price y Storn [73] en 1995. Su proceso de búsqueda se guía mediante la distribución de la población a partir de la suma y diferencia de vectores. Las bondades de este algoritmo son su capacidad de encontrar el óptimo global, además es de fácil implementación, requiere de pocos parámetros y presenta una rápida convergencia [85].

Evolución cultural Los algoritmos culturales fueron propuestos por Reynolds [63], teniendo como esquema la evolución de un sistema cultural. Su objetivo es aprovechar el conocimiento generado por los individuos a lo largo del proceso de búsqueda. Por lo tanto, evoluciona el comportamiento y no las soluciones, entonces la selección se realiza de acuerdo a los mejores comportamientos de la búsqueda.

Coevolución En un algoritmo de coevolución competitiva, los individuos de una población evolucionan a través de la cooperación o competencia con individuos de otra población, con los cuales interactúan.

El algoritmo de búsqueda seleccionado para el desarrollo de este trabajo es el algoritmo de evolución diferencial, debido a sus bondades de implementación y convergencia. En la siguiente sección se explica de manera más detallada.

2.3.1. Evolución Diferencial

El algoritmo de evolución diferencial es estocástico y realiza una búsqueda directa, es decir, no requiere la derivada de las funciones del problema a optimizar. Además, la evolución diferencial requiere de pocos parámetros (NP, CR y F), por lo cual se ha empleado para resolver varios problemas del mundo real [10]. La diferencia con otros algoritmos evolutivos es cómo guía el proceso de búsqueda, pues la mutación se realiza con base en la dirección e información proporcionada por 3 vectores de la población actual.

El proceso de búsqueda que realiza el algoritmo de evolución diferencial consta de cuatro pasos [10]: inicialización de la población, mutación, recombinación o cruza y selección. El primero se realiza una vez y el resto se repite mientras no se cumple la condición de paro, la cual generalmente se define por un número de evaluaciones del problema.

2.3.1.1. Inicialización de la población

En la evolución diferencial una población está compuesta por un conjunto de soluciones, denominadas vectores, la cual se representa de la siguiente manera:

$$P = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{NP}\}$$

donde $\vec{x}_i$ es un vector n-dimensional [1] y NP es el número de vectores que conforman la población. Por consiguiente, P es una matriz de NP filas y n columnas. El acceso a una variable de diseño, en P, se indica por la posición del individuo, índice i y la posición de la variable, índice j, como $x_{i,j}$.

Existen varios métodos para inicializar la población, sin embargo en la evolución diferencial se genera de manera aleatoria con una distribución uniforme dentro del espacio de búsqueda, el cual está definido por los límites de las variables de diseño [$\vec{L}$, $\vec{U}$] [55]. La población se inicializa de la siguiente manera:

$$x_{i,j} = rand * (U_j - L_j) + L_j$$

donde *rand* genera un número aleatorio en el rango [0,1] y L_j y U_j representan el límite inferior y superior de la **j-ésima** variable de diseño, respectivamente.

2.3.1.2. Mutación

Es el primer operador de variación que se emplea en evolución diferencial. Por cada vector de la población (i), denominado *target*, se calcula un vector mutante, llamado *donor/mutant*. Existen diferentes tipos de mutaciones, el proceso para calcular el vector *mutant* consiste en sumar a un vector, seleccionado de manera aleatoria (r_0) o el mejor de la generación (*best*) y denominado vector base, una ($r1$, $r2$) o dos diferencias de vectores ($r1$-$r4$), las cuales se multiplican por un escalar F, el cual es un parámetro definido por el usuario. Los vectores que se emplean para las diferencias de vectores, se seleccionan de manera aleatoria y son diferentes entre ellos. Las ecuaciones para los diferentes tipos de mutación son:

[1] La dimensión del vector está dada por la definición del problema.

DE/rand/1	$\vec{v}_{G,i} = \vec{x}_{G,r_0} + F(\vec{x}_{G,r_1} - \vec{x}_{G,r_2})$
DE/best/1	$\vec{v}_{G,i} = \vec{x}_{G,best} + F(\vec{x}_{G,r_1} - \vec{x}_{G,r_2})$
DE/current-to-best/1	$\vec{v}_{G,i} = \vec{x}_{G,i} + F(\vec{x}_{G,best} - \vec{x}_{G,i}) + F(\vec{x}_{G,r_1} - \vec{x}_{G,r_2})$
DE/rand/2	$\vec{v}_{G,i} = \vec{x}_{G,r_0} + F(\vec{x}_{G,r_1} - \vec{x}_{G,r_2}) + F(\vec{x}_{G,r_3} - \vec{x}_{G,r_4})$
DE/best/2	$\vec{v}_{G,i} = \vec{x}_{G,best} + F(\vec{x}_{G,r_1} - \vec{x}_{G,r_2}) + F(\vec{x}_{G,r_3} - \vec{x}_{G,r_4})$

2.3.1.3. Cruza

La cruza combina los vectores *donor* y *target* para formar al vector hijo (o *trial*). La evolución diferencial emplea dos tipos de cruza: binomial (o uniforme) y exponencial (o modulo de 2 puntos). En la Ecuación 2.6 se muestra como se realiza la cruza binomial.

$$u_{G,i,j} = \begin{cases} v_{G,i,j} & \text{si } j = rand_j \text{ o } rand < CR \\ x_{G,i,j} & \text{en caso contrario} \end{cases} \tag{2.6}$$

donde CR representa el porcentaje de cruza y es un parámetro definido por el usuario, su valor se encuentra entre [0,1]. $rand_j$ es un número entero aleatorio entre [1, n], esto con el fin de garantizar que el vector *trial* tenga al menos un valor diferente al vector *target* (en caso de que CR tenga un valor cercano o igual a cero). *rand* es un valor real aleatorio entre [0, 1].

2.3.1.4. Selección

En este paso se determina cual de los dos vectores (*target* y *trial*) sobrevive para la siguiente generación. En la Ecuación 2.7 se representa este proceso.

$$\vec{x}_{G+1,i} = \begin{cases} \vec{u}_{G,i} & \text{si } f(\vec{u}_{G,i}) \leq f(\vec{x}_{G,i}) \\ \vec{x}_{G,i} & \text{en otro caso} \end{cases} \tag{2.7}$$

donde $f(\cdot)$ es la función objetivo que minimiza el problema. Por lo tanto, si el valor de la función objetivo del vector *trial* es mejor que el del *target*, entonces el segundo es reemplazado por su hijo. En caso contrario, el vector *target* sobrevive para la siguiente generación.

2.3.1.5. Evolución Diferencial con Variantes Combinadas

El proceso descrito en las subsecciones previas corresponde a la versión canónica DE/rand/1/bin. Las variantes más conocidas son: DE/rand/1/bin

y DE/best/1/bin, la diferencia entre ellas es el vector base que emplean, $\vec{x}_{G,r_0}$ y $\vec{x}_{G,best}$, respectivamente. Estas variantes se emplean en Evolución Diferencial con Variantes Combinadas (DECV, por sus siglas en inglés) [48]. DECV primero explora el espacio de búsqueda mediante DE/rand/1/bin, una vez que se alcanza la zona factible y se tiene el 10 % de soluciones factibles en la población, se explota la región cercana al mejor individuo de la población mediante la variante DE/best/1/bin. El pseudocódigo de DECV se muestra en la Figura 2.3.

1: Crear una población aleatoria inicial $P_0 = \{\vec{x}_{1,0}, \dots \vec{x}_{NP,0}\}$
2: Evaluar a P_0 en f
3: $flag\_B = false$
4: **for** $G = 1$ to MAX_GEN **do**
5: $nf =$ Número de soluciones factibles en P_{G-1}
6: **if** $nf \geq 10\,\%NP$ **then**
7: $flag\_B = true$
8: **end if**
9: **for** $i \leftarrow 1$ to NP **do**
10: **if** $flag\_B$ **then**
11: $r_0 =$ el mejor vector de P_{G-1}
12: Seleccionar de manera aleatoria r_1 y r_2 $(r_0 \neq r_1 \neq r_2 \neq i)$
13: **else**
14: Seleccionar de manera aleatoria r_0, r_1 y r_2 $(r_0 \neq r_1 \neq r_2 \neq i)$
15: **end if**
16: $J_{rand} = randint[1, n]$
17: **for** $j = 1$ to n **do**
18: **if** $rand_j \leq Cr$ O $j = J_{rand}$ **then**
19: $u_{i,j,G} = x_{r0,j,G} + F(x_{r1,j,G} - x_{r2,j,G})$
20: **else**
21: $u_{i,j,G} = x_{i,j,G}$
22: **end if**
23: **end for**
24: **if** $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$ **then**
25: $\vec{x}_{i,G+1} = \vec{u}_{i,G}$
26: **else**
27: $\vec{x}_{i,G+1} = \vec{x}_{i,G}$
28: **end if**
29: **end for**
30: **end for**

Figura 2.3: Pseudocódigo de DECV. El criterio para cambiar de DE/rand/1/bin a DE/best/1/bin se lleva a cabo en los pasos 5-8 mediante $flag\_B$. En los pasos 11-12 y 14 se seleccionan los vectores para DE/best/1/bin y DE/rand/1/bin, respectivamente.

2.4. Técnicas para manejo de restricciones

Los algoritmos evolutivos en sus inicios fueron diseñados para resolver problemas sin restricciones, por lo que al ser empleado en un problema con restricciones, el objetivo es encontrar la mejor solución factible, no tan sólo encontrar la mejor solución sin verificar si es factible o no. Por lo anterior se han desarrollado un conjunto de técnicas para manejo de restricciones, las cuales tienen como objetivo guiar los algoritmos evolutivos hacia la región factible del problema.

Las técnicas para manejo de restricciones se pueden clasificar de acuerdo a su taxonomía [9] y [50]. No obstante, en [46] se listan las técnicas para manejo de restricciones más empleados en los últimos años:

Reglas de factibilidad Es un conjunto de tres reglas (ver Sec. 2.4.1).

Jerarquización estocástica Es un método de ordenamiento de burbuja estocástico (ver Sec. 2.4.4).

Método ε-constrained Transforma un problema de optimización con restricciones en un problema sin restricciones (ver Sec. 2.4.3).

Funciones novedosas de penalización Transforman un problema de optimización restringida en un problema sin restricciones, $\phi(\vec{x}) = f(\vec{x}) + p(\vec{x})$, donde $\phi(\vec{x})$ es la nueva función a optimizar y está definida por la función objetivo ($f(\vec{x})$) y el valor de penalización ($p(\vec{x})$), que se calcula por $p(\vec{x}) = \sum_{i=1}^{m} r_i \cdot \text{máx}(0, g_i(\vec{x}))^2 + \sum_{j=1}^{p} c_j \cdot |h_j(\vec{x})|$, donde r_i y c_j son los factores de penalización. Por tanto, en un problema de minimización, una solución no factible tendrá un valor mayor a aquellas soluciones factibles, porque r_i y c_j incrementan el valor de la sumatoria que se calcula en $p(\vec{x})$.

Operadores especiales novedosos Transforman un individuo infactible en factible mediante estos operadores.

Conceptos de optimización multiobjetivo Transforman un problema de optimización con restricciones en un problema bi-objetivo (función objetivo y suma de violación de restricciones) o multiobjetivo (función objetivo y cada una de las restricciones).

Ensamble de técnicas para manejo de restricciones Integran varias técnicas para manejo de restricciones dentro de un algoritmo evolutivo para realizar el proceso de búsqueda.

Debido al tipo de problemas que se resuelve en este trabajo, es necesario considerar una técnica para manejo de restricciones para guiar la búsqueda hacia la zona factible. Por tanto, en las siguientes subsecciones se explican con mayor detalle las cuatro técnicas para manejo de restricciones, las cuales son los más empleadas en los algoritmos del estado del arte para solucionar problemas de optimización restringida.

2.4.1. Reglas de Factibilidad

Las reglas de Factibilidad fueron propuestas por Deb [12], pertenecen a la categoría de separación de las restricciones de la función objetivo. Es uno de los mecanismos más simples y más usado en varios algoritmos evolutivos [46]. El conjunto de reglas se define de la siguiente manera:

1. Si se comparan dos soluciones factibles, se selecciona aquella que tenga el mejor valor de la función objetivo.

2. Si se compara una solución factible contra una infactible, entonces la solución factible se selecciona.

3. Si se comparan dos soluciones infactibles, se selecciona aquella que tiene la menor suma de violación de restricciones.

La suma de violación de restricciones se calcula mediante $\phi(\vec{x}) = \sum_{i=1}^{m} \text{máx}(0, g_i(\vec{x}))^2 + \sum_{j=1}^{p} |h_j(\vec{x})|$. La ventaja de este conjunto de reglas es que no requiere ningún parámetro. Sin embargo, el proceso de búsqueda puede presentar convergencia prematura.

2.4.2. Mecanismo de diversidad

El mecanismo de diversidad (DM, por sus siglas en inglés) fue propuesto por Mezura y Coello [47] y es una variante de las reglas de factibilidad, a la cual se le agrega un mecanismo que permite mantener soluciones infactibles cercanas a la frontera de la región factible en la población de cada generación. Esta combinación proporciona resultados competitivos en aquellos problemas que tienen restricciones activas [46]. A diferencia de las reglas de factibilidad, esta variante requiere un parámetro para indicar el porcentaje de soluciones infactibles que permanece en la población (S_r). Estas soluciones se seleccionan de los padres e hijos.

2.4.3. Método ε-*constrained*

El método ε-*constrained* fue propuesto por Takahama y Sakai [75]. Este método transforma un problema de optimización restringida en un problema sin restricciones y se compone de un mecanismo de ordenamiento lexicográfico y un valor de relajación para soluciones no factibles. En el primero, la minimización de la suma de violación de restricciones precede a la minimización de la función objetivo, en el segundo, el valor de relajación permite que una solución se considere factible, por lo tanto el criterio de comparación es el valor de la función objetivo.

Este proceso es definido por ε-*level*, donde ε es un parámetro definido por el usuario:

$$(f(\vec{x_1}), \phi(\vec{x_1})) <_\varepsilon (f(\vec{x_2}), \phi(\vec{x_2})) \Leftrightarrow \begin{cases} f(\vec{x_1}) < f(\vec{x_2}), \text{ si } \phi(\vec{x_1}), \phi(\vec{x_2}) \leq \varepsilon \\ f(\vec{x_1}) < f(\vec{x_2}), \text{ si } \phi(\vec{x_1}) = \phi(\vec{x_2}) \\ \phi(\vec{x_1}) < \phi(\vec{x_2}), \text{ de otra manera} \end{cases}$$

$$(f(\vec{x_1}), \phi(\vec{x_1})) \leq_\varepsilon (f(\vec{x_2}), \phi(\vec{x_2})) \Leftrightarrow \begin{cases} f(\vec{x_1}) \leq f(\vec{x_2}), \text{ si } \phi(\vec{x_1}), \phi(\vec{x_2}) \leq \varepsilon \\ f(\vec{x_1}) \leq f(\vec{x_2}), \text{ si } \phi(\vec{x_1}) = \phi(\vec{x_2}) \\ \phi(\vec{x_1}) < \phi(\vec{x_2}), \text{ de otra manera} \end{cases}$$

donde $f(\cdot)$ es el valor de la función objetivo y $\phi(\cdot)$ es la suma de violación de restricciones, la cual se calcula con la Ecuación 2.8.

$$\phi(\vec{x}) = \sum_{i=1}^{m} \text{máx}\,(0, g_i(\vec{x})) + \sum_{j=1}^{p} |h_j(\vec{x})| \tag{2.8}$$

2.4.4. Jerarquización estocástica

La jerarquización estocástica (SR, por sus siglas en inglés) fue propuesta por Runarsson y Yao [66], está basado en el método de burbuja para ordenar las soluciones de la población de acuerdo a: (1) la suma de violación de restricciones y el valor de la función objetivo, (2) el valor de la función objetivo. Para ello requiere un parámetro (P_f) que indique la probabilidad de cómo se realizará la comparación. El pseudocódigo de la jerarquización estocástica se muestra en la Fig. 2.4.

```
1: for i = 1 to N do
2:    for j = 1 to N − 1 do
3:       u = random(0,1)
4:       if (φ(I_j) = φ(I_{j+1}) = 0) or (u < P_f) then
5:          if f(I_j) > f(I_{j+1}) then
6:             swap(I_j, I_{j+1})
7:          end if
8:       else
9:          if φ(I_j) > φ(I_{j+1}) then
10:            swap(I_j, I_{j+1})
11:         end if
12:      end if
13:   end for
14:   if no swap then
15:      break
16:   end if
17: end for
```

Figura 2.4: Pseudocódigo de SR. I_j es un individuo de la población, N es el tamaño de la población, $\phi(I_j)$ es la suma de violación de restricciones y $f(I_j)$ es el valor de la función objetivo del individuo I_j, respectivamente. La intensión de SR es favorecer soluciones no factibles con un buen valor de la función objetivo.

Comparación

En la tabla 2.1 se listan las diferencias de las cuatro técnicas para manejo de restricciones de las subsecciones anteriores.

2.5. Resumen del capítulo

En el presente capítulo se presentaron conceptos generales de optimización, tal como los elementos que definen un problema de optimización y su clasificación, profundizando en los problemas con restricciones, los cuales se resuelven en el presente trabajo. Además, se definieron de manera general los algoritmos evolutivos, explicando a mayor detalle la evolución diferencial, ya que una de sus variantes, denominada DECV, se empleará como método de búsqueda. Dado que los algoritmos evolutivos no se diseñaron para resolver problemas con restricciones, se finaliza el capítulo con la exposición de las técnicas para manejo de restricciones, las cuales guían el proceso de búsqueda hacia la región factible, enfocándose en cuatro técnicas: reglas de factibilidad, mecanismo de diversidad, método de ε-*constrained* y la jerarquización estocástica.

Tabla 2.1: Diferencias entre reglas de factibilidad, mecanismo de diversidad, método ε-*constrained* y jerarquización estocástica.

Técnica para manejo de restricciones	Diferencias
Reglas de factibilidad	▪ No requiere parámetros. ▪ Preferencia por soluciones factibles o con una baja suma de violación de restricciones.
Mecanismo de diversidad	▪ Requiere un parámetro, S_r. ▪ Permite soluciones no factibles cercanas a los límites de la región factible.
Método ε-*constrained*	▪ Requiere dos parámetros, T_c y cp. ▪ Relaja la región factible mediante ε, cuyo valor se actualiza a lo largo de la búsqueda.
Jerarquización estocástica	▪ Requiere un parámetro, P_f. ▪ Ordena estocásticamente la población actual y la anterior.

Capítulo 3

Optimización asistida por subrogados

Muchos problemas de optimización de la vida real están definidos mediante evaluaciones muy costosas, es decir, el cálculo de una solución puede requerir de minutos, horas o incluso días. Así, al resolverlos mediante un método de optimización, el cual necesita de varias evaluaciones, precisa que la tarea de encontrar el óptimo sea un proceso que demande mucho tiempo de cómputo. Una manera de reducir este costo es usando modelos subrogados, los cuales son más baratos que los modelos originales.

En este capítulo se explica el uso de modelos subrogados en optimización, así como los elementos necesarios para integrarlos en un método de optimización y los modelos que se pueden utilizar para aproximar las soluciones, además de los métodos para su evaluación.

3.1. Uso de modelos subrogados

La información que se genera a lo largo del proceso de búsqueda, a través de las soluciones y sus valores en los modelos originales, puede ser aprovechada mediante una técnica de aprendizaje automático para comprender el comportamiento de la búsqueda y poder guiarla hacia el óptimo global [88]. Las ventajas del uso de estas técnicas son: aumento en la velocidad de la convergencia y mejora en la calidad de las soluciones [57, 88]. Uno de los principales usos es el ajuste de curvas, el cual se produce a partir de los puntos que se han generado en el proceso de búsqueda mediante los modelos subrogados [18].

Un modelo subrogado (también llamado modelo aproximado o metamodelo) se emplea cuando: (1) la función objetivo y restricciones son muy costosas para evaluar, así los valores de las funciones para cada solución se aproximan mediante estos modelos, (2) no se cuenta con una función objetivo del problema y es el usuario quien determina el valor de una solución, a esto se le conoce como Cómputo Evolutivo Interactivo [28, 72], y se emplea en problemas de optimización de diseño, tal como diseño de productos, diseño de arte y composición de música, (3) en la evaluación en ambientes con ruido[1][19, 67], ya que algunas estrategias adoptadas para reducir el ruido, requieren de evaluaciones adicionales [27], y (4) en problemas multimodales, en los cuales un modelo subrogado se construye para suavizar los óptimos locales, conservando la ubicación del óptimo global [27].

El proceso que se lleva a cabo para usar un modelo subrogado en un proceso de optimización es el siguiente [19]:

1. Selección de las variables a ser optimizadas, generalmente se realiza por importancia y después de una experimentación preliminar.

2. Selección de una técnica de diseño de muestreo o de diseño de experimentos [27] para las soluciones iniciales.

3. Construir el modelo de la función.

4. Realizar una búsqueda para identificar nuevos puntos de diseño para análisis

5. Agregar los nuevos puntos a los existentes e ir al paso 3.

De acuerdo a lo anterior, la construcción de un modelo subrogado se lleva a cabo mediante un conjunto de soluciones y sus respectivos valores en la función objetivo, ya sea producto de una simulación o evaluación en el modelo original (pasos 2 y 5), a este conjunto se le conoce como muestra o conjunto de entrenamiento. El paso 2 se considera la primera etapa del modelo, debido a que se obtiene conocimiento del espacio de búsqueda para poder construirlo (paso 3). Por otro lado, los pasos 4 y 5 son la segunda etapa pues actualizan la muestra y el modelo. Para actualizar la muestra se emplean los criterios de muestreo de relleno en optimización asistida por

[1] El ruido en un problema de optimización puede tener diversas fuentes, tal como [2, 3]: limitaciones en mediciones físicas, uso de modelos de simulación estocástica, muestreo incompleto en espacios muy grandes, interacción humano-computadora, imprecisión en los cálculos realizados en una computadora, por mencionar algunos.

subrogados (SA-Op, por sus siglas en inglés), o un control de evolución, también llamado manejador de modelos, en algoritmos evolutivos asistidos por subrogados. El objetivo de esta etapa es acelerar la convergencia a una buena solución, además de poder modelar el problema de manera exacta y global [54]. Dicha actualización se repite hasta que se cumple un criterio de paro.

Los modelos subrogados se emplean de manera global y local [71]. Un modelo global permite tener un conocimiento general del espacio de búsqueda, además de prevenir los óptimos locales. Por otro lado, un modelo local se ajusta mejor al área que se está aproximando. Además, este tipo de modelos puede ser empleado para calcular de manera barata los gradientes o los patrones de búsqueda requeridos en los métodos de búsqueda indirecta [31].

En las siguientes secciones se explica cada uno de los elementos necesarios para integrar un modelo aproximado en un método de optimización.

3.2. Muestreo inicial

Para llevar a cabo esta etapa, es necesario el diseño y análisis de una experimentación, debido a que el muestreo inicial consiste en identificar el conjunto de variables de diseño que van a formar parte del proceso de SA-Op. Además, se ha demostrado que una selección eficiente de las muestras mejora la calidad del modelo de manera significativa [27].

Para construir un modelo global, del cual no se conoce su paisaje de aptitud, lo ideal es que los puntos estén distribuidos de manera uniforme sobre los ejes de cada variable y así evitar que se repitan valores para una variable [19].

El método más popular para el muestreo inicial es el diseño de experimentos, el cual es un plan de muestreo en el espacio de las variables y proporciona valores únicos para las variables de diseño [79]. Además, los diseños de experimentos permiten investigar los efectos de las variables de entrada sobre una variable de salida. Esto se realiza mediante simulaciones de computadora o experimentación [19].

Un diseño de experimentos se puede ver como una matriz de $n_p \times n_v$ [79], donde n_p es el número de puntos y n_v es el número de variables del problema (Ec. 3.1)

$$X = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{n_p}]^T \tag{3.1}$$

donde cada $\vec{x}_i$ representa una muestra ($\vec{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in_v}]$) y x_{ij} representa una variable. Los métodos más empleados se describen a continuación:

- Arreglos ortogonales. La matriz X de un diseño de experimentos es un arreglo ortogonal sí el producto interno de dos columnas cualesquiera es cero [27, 79].

- Muestreo hipercubo latino. Un muestreo hipecubo latino asegura la estratificación de una muestra en todo el espacio de búsqueda [79], es decir, cada una de las variables se distribuye a lo largo de su rango de valores. En la Fig. 3.1 se muestra un muestreo hipercubo latino de veinte puntos en dos dimensiones.

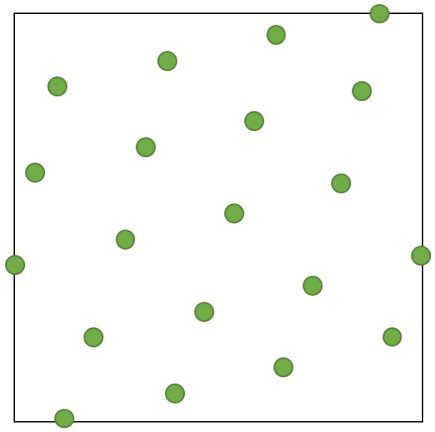


Figura 3.1: Ejemplo de un LHS de veinte puntos en dos dimensiones con una buena distribución de la muestra.

- Diseño central compuesto. Un diseño de cenrral compuesto [27] inicia con un diseño factorial[2], puntos centrales, extendido por un conjunto de puntos "axiales", los cuales se generan con direcciones positivas y negativas a partir del punto central.

Otra técnica que se puede emplear, para generar la muestra inicial, es el método Monte-Carlo [27, 84], el cual es totalmente aleatorio, por lo que no se garantiza que la muestra cubra todo el espacio de búsqueda. Además, este método es costoso computacionalmente pues requiere de muchos experimentos [84].

Todos los métodos anteriores se han empleado para generar la muestra inicial, no obstante, el método más usado en los enfoques propuestos es el muestreo hipercubo latino.

[2]Malla rectangular de puntos [18].

3.3. Métodos subrogados

La aproximación de funciones dentro de un proceso estocástico ha sido aplicada desde hace años en tres áreas [34]: geología matemática, optimización global y estadística. En geología matemática, el método de interpolación denominado *kriging* se ha empleado desde principios de los 60s. En optimización global, se encuentran la optimización global Bayesiana y el enfoque de función aleatoria. En estadística, a principios de los 70s comienza el interés por aproximar integrales.

Los modelos que se han empleado en optimización o algoritmos evolutivos para aproximar una función se explican en las siguientes subsecciones, los cuales tienen la capacidad de ajustar un conjunto de soluciones a un modelo para poder predecir un nuevo conjunto de soluciones.

3.3.1. k-vecinos más cercanos

k-vecinos más cercanos (kNN, por sus siglas en inglés) es un modelo que se basa en la similitud con los vecinos más cercanos al que se desea aproximar [17], donde la aproximación del valor de una solución se calcula por medio del promedio de sus k vecinos más cercanos [64], mediante la Ecuación 3.2.

$$\hat{f}(\vec{x}) = \frac{\sum_{i=1}^{k} f(\vec{y}_i)}{k}, \quad \vec{y}_i \in \mathcal{D} \tag{3.2}$$

donde $\vec{x}$ es la solución que se va a aproximar mediante $\hat{f}(\cdot)$, $\vec{y}_i$ es una solución evaluada en el modelo original, $f(\cdot)$, por lo tanto pertenece al conjunto de entrenamiento, denotado por $\mathcal{D}$ y es vecino cercano a $\vec{x}$, $i \leq k$. Si todas las soluciones de $\mathcal{D}$ se emplean para aproximar a $\vec{x}$, entonces el modelo es global, en caso contrario, $k < |\mathcal{D}|$, el modelo se considera local [64].

Las ventajas de kNN son: (1) su modelo es simple, (2) no se hacen suposiciones acerca del paisaje de la función que se quiere aproximar, y (3) no hay un proceso de *aprendizaje*, ya que la predicción de una solución se calcula con el promedio de k soluciones cercanas. Por otro lado, sus desventajas son: (1) no existe una descripción del modelo, (2) la definición de los k vecinos más cercanos a una solución se convierte en un proceso costoso cuando el tamaño de la muestra es muy grande, y (3) su desempeño depende del número de variables del problema.

3.3.2. Modelos polinomiales

Los modelos polinomiales o modelos de superficie de respuesta (RSM, por sus siglas en inglés) han sido ampliamente usados para resolver problemas de ingeniería [19]. El modelo más común es de segundo grado y se forma a partir de una muestra de soluciones, la cual tiene un tamaño mínimo de $n_t = (n+1)(n+2)/2$. La ecuación del modelo de segundo grado se define en la ecuación 3.3.

$$\hat{f}(\vec{x}) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1,j=i}^{n,n} \beta_{n-1+i+j} x_i x_j \tag{3.3}$$

donde n es la dimensionalidad del problema, β_i son los coeficientes que serán estimados y x_i es el valor de la i-ésima variable de diseño de la solución $\vec{x}$. Los métodos más comunes para calcular los coeficientes β_i son [27]:

Método de mínimos cuadrados [27, 71]. Este método requiere que el tamaño de la muestra sea mayor o igual a n_t, los coeficientes se estiman a partir la Ecuación 3.4.

$$\vec{y} = X\vec{\beta} \tag{3.4}$$

donde $\vec{y}$ es el vector de los valores de respuesta, X es la matriz de *Vandermonde* asociada a los valores de cada solución de la muestra (se asume que son linealmente independientes) y $\vec{\beta}$ es el vector de los coeficientes que se va a estimar. En las Ecuaciones 3.5 y 3.6 se ilustra el vector de respuesta y la matriz de *Vandermonde* para un polinomio de segundo grado de dimensión dos y con un tamaño de muestra n_t.

$$\vec{y} = [y_1, y_2, \ldots, y_{n_t}]^T \tag{3.5}$$

$$X = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,n}^2 \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,n}^2 \\ \vdots & & \ddots & & \vdots \\ 1 & x_{n_t,1} & x_{n_t,2} & \cdots & x_{n_t,n}^2 \end{pmatrix} \tag{3.6}$$

La estimación de los coeficientes esta dada por la Ecuación 3.7

$$\hat{\beta} = (X^T X)^{-1} X^T \vec{y} \tag{3.7}$$

El costo computacional de este método incrementa con la dimensionalidad del problema, lo cual es una desventaja [27, 71].

Método de gradiente [27]. El método del gradiente calcula las derivadas del error cuadrático para cada coeficiente a partir de la Ecuación 3.8.

$$E = \frac{1}{2}(y - \hat{y})^2 \tag{3.8}$$

donde y es el valor de respuesta y $\hat{y}$ es el valor que se obtiene de la Ecuación 3.3. La estimación de los coeficientes se realiza mediante:

$$\begin{aligned}
&\Delta\beta_0 = \xi(y - \hat{y}) \\
&\Delta\beta_i = \xi(y - \hat{y})x_i \\
&\Delta\beta_{n-1+i+j} = \xi(y - \hat{y})x_i x_j \quad 1 \leq i \leq j \leq n
\end{aligned}$$

Los modelos polinomiales se han empleado para resolver problemas reales [72]. Sin embargo, no son una buena opción para problemas multimodales o no lineales [19], debido a que el grado del polinomio para ajustar el modelo es de grado tres o mayor grado, el número de puntos y el costo computacional incrementan para construirlos.

3.3.3. Redes neuronales artificiales

Las redes neuronales artificiales (ANNs, por sus siglas en inglés) han sido ampliamente usadas para aproximar funciones. Tanto los perceptrones multicapa como las funciones de base radial han demostrado ser muy eficientes para esta tarea [27, 72].

3.3.3.1. Perceptrón multicapa

Un perceptrón multicapa (MLP, por sus siglas en inglés) es una red con alimentación hacia adelante, y se compone de [72]: (1) una capa de entrada, generalmente el número de neuronas es igual número de variables del problema, (2) una capa de salida y (3) una o varias capas ocultas. En la Ecuación 3.9 se muestra la fórmula de un perceptrón multicapa simple con una capa de entrada, una capa oculta y una neurona de salida.

$$y(x) = \sum_{k=1}^{K} w_j^{(2)} f(\sum_{i=1}^{n} w_{ki}^{(1)} x_i + w_{k0}^{(1)}) + w_0^{(2)} \tag{3.9}$$

donde n es el número de neuronas de entrada, K es el número de neuronas de la capa oculta, $\vec{w}$ es el vector de pesos, los superíndices indican la capa

a la que corresponden los pesos, (1) de entrada y (2) capa oculta, y f es la función de activación, la más empleada es la función logística (Ec. 3.10),

$$f(z) = \frac{1}{1 + \exp^{-cz}} \tag{3.10}$$

donde c es una constante y z es $\sum_{i=1}^{n} w_{ki}^{(1)} x_i + w_{k0}^{(1)}$.

Para que un perceptrón multicapa genere la salida esperada, este debe pasar por un proceso de *aprendizaje*, el cual se compone de dos fases: de entrenamiento y de prueba. Para realizar este proceso se requiere de un conjunto de variables de entrada y salida (muestra inicial). Durante el entrenamiento, los valores de $\vec{w}$ se ajustan para generar los valores de salida, y se realiza con un porcentaje de la muestra. Para determinar su precisión, el perceptrón multicapa se prueba con el resto de la muestra, una vez que se cumple la exactitud deseada, se considera que el perceptrón multicapa ya *aprendió*.

Las ventajas de un perceptrón multicapa son [77]: (1) pueden representar relaciones tanto lineales como no-lineales a partir de la información de la muestra, (2) no requieren un entrenamiento estadístico formal, y (3) son una buena opción para una aproximación global. Por otra parte, sus desventajas son: (1) pueden guiar la búsqueda hacia un falso óptimo cuando el tamaño de la muestra es pequeño y está mal distribuida en el espacio de búsqueda [30], (2) el proceso de *aprendizaje* es costoso computacionalmente, y (3) son predispuestas al sobreajuste o sobre-entrenamiento [77].

3.3.3.2. Funciones de base radial

Una función de base radial (RBF, por sus siglas en inglés) realiza una suma ponderada de funciones simples con el fin de emular un paisaje complicado [19]. Una función de base radial se compone de [72]: (1) una capa de entrada, cuyo tamaño es igual a la dimensionalidad del problema, (2) una capa oculta, cuyo tamaño es igual (dimensionalidad pequeña) o menor (dimensionalidad grande), para evitar exceso de cálculos, y (3) una capa de salida, que es una combinación lineal de un conjunto de funciones de base radial expresadas mediante la Ecuación 3.11,

$$\hat{f}(\vec{x}) = \vec{w}^T \vec{\psi} = \sum_{i=0}^{n_c} w_i \psi_i(\|\vec{x} - c_i\|) \tag{3.11}$$

donde w_i son los coeficientes desconocidos y que se aprenderán durante el entrenamiento, $\psi_i(\|\vec{x} - c_i\|)$ representa la i-ésima función de base radial,

también se le conoce como *kernel*, y evalúa la distancia entre $\vec{x}$ y el centro c_i de la función de base, estos son también desconocidos y se aprenden mediante otros métodos, tal como el método del *k-means*. $\vec{x}$ es la solución que se va a aproximar mediante ($\hat{f}$), a partir de una muestra, $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}^T$ y sus respuestas (obtenidas mediante una función f) $y = \{y_1, y_2, \ldots y_n\}^T$,

Los tipos de *kernels* que se emplean son: *splines* lineales, *splines* cúbicos, *spline* de placa delgada, multicuadrática, inverso multicuadrática y gaussiano, siendo el último el más común.

Esta técnica es muy empleada para aproximar los valores de funciones en algoritmos evolutivos [72]. Las ventajas de una RBF son[83]: (1) debido a que su arquitectura es más sencilla que la de un perceptrón multicapa, su proceso de entrenamiento es más rápido, (2) pueden aproximar funciones cuyo paisaje de aptitud está compuesto por picos o valles, y (3) son ideales para aproximaciones locales. Por otro lado, sus desventajas son: (1) la selección de la función del *kernel* es una tarea de prueba y error, y (2) el desempeño depende de los valores iniciales de los parámetros [83].

3.3.4. Máquinas de soporte vectorial

Las máquinas de soporte vectorial (SVM, por sus siglas en inglés) fueron propuestas por Vladimir Vapnik, su teoría está inspirada principalmente en la teoría del aprendizaje estadístico [27]. Las máquinas de soporte vectorial se emplean para resolver problemas de clasificación y regresión.

3.3.4.1. Clasificación mediante máquinas de soporte vectorial

Originalmente las máquinas de soporte vectorial fueron propuestas para resolver problemas de clasificación lineal, cuya idea es encontrar un hiperplano (H) que separa un conjunto de muestras positivas (+1) de las negativas (-1) mediante un margen máximo (Fig. 3.2), el cual se define por la distancia que existe entre H y los puntos más cercanos de las muestras positivas y negativas.

El hiperplano óptimo se define por

$$u = \vec{w} \cdot \vec{x} - b \tag{3.12}$$

donde $\vec{w}$ es la normal al hiperplano, $\vec{x}$ son los vectores de entrada y b es el sesgo (*bias*), el conjunto de vectores se representa por $\mathcal{D}$. Pueden existir varios hiperplanos que separen las clases. Sin embargo, existe un hiperplano separador óptimo o un hiperplano de margen máximo, sí $\mathcal{D}$ es separable sin

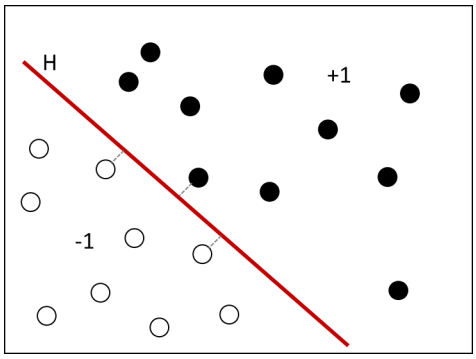


Figura 3.2: Clasificación lineal de dos muestras mediante SVM, donde H es el hiperplano que separa las clases.

error y la distancia entre el vector más cercano al hiperplano es máxima, lo anterior se indica matemáticamente en la Ecuación 3.13.

$$\begin{cases} \vec{w} \cdot \vec{x} - b \geq +1 \text{ sí } y_i = +1 \\ \vec{w} \cdot \vec{x} - b \leq -1 \text{ sí } y_i = -1 \end{cases} \tag{3.13}$$

La maximización del margen se puede expresar como un problema de optimización:

$$\begin{array}{l} \text{Minimizar}_w \ \ \frac{1}{2}\|\vec{w}\|^2 \\ \text{sujeta a:} \\ y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \forall i \end{array} \tag{3.14}$$

donde x_i es el i-ésimo dato de entrenamiento y y_i su salida, la cual puede tomar dos valores: $+1$ para muestras positivas y -1 para muestras negativas. El problema de minimización se puede resolver mediante multiplicadores de Lagrange [43, 56].

3.3.4.2. Regresión con máquinas de soporte vectorial

Considerando un conjunto de entrenamiento $\mathcal{D} = \{(\vec{x}_i, y_i), i = 1, \ldots, \ell\}$, donde $\vec{x}_i \in \mathbb{R}^n$ y $y_i \in \mathbb{R}$. La regresión en $\mathcal{D}$ se lleva a cabo mediante ϵ-SVR [78], que consiste en encontrar una función $\hat{f}_\epsilon$ que tiene al menos una desviación ϵ a partir de los valores de y_i y que además debe ser lo más suave posible. ϵ-SVR no considera el error de aquellas soluciones que tienen una desviación menor a ϵ, sin embargo no permite una desviación mayor a ϵ, lo cual se define

mediante la función de pérdida ϵ-insensible. Lo anterior se ilustra en la Fig. 3.3. La función de pérdida ϵ-insensible se describe matemáticamente como:

$$|\xi|_\epsilon = \begin{cases} 0 & \text{sí } |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{en otro caso} \end{cases} \tag{3.15}$$

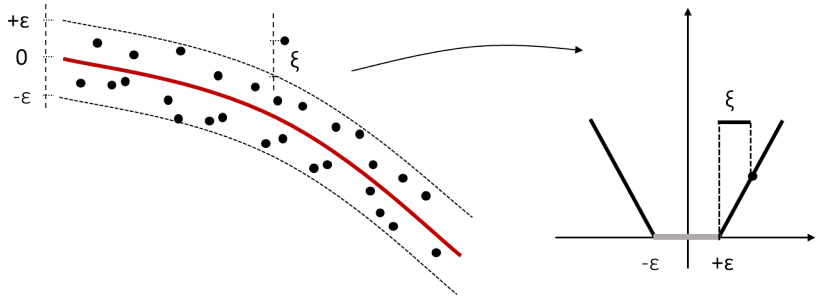


Figura 3.3: Pérdida de margen suave asignada por SVR.

La definición de un problema lineal mediante ϵ-SVR es:

$$\hat{f}_\epsilon = \vec{w} \cdot \vec{x} - b \tag{3.16}$$

donde $\vec{x}$ son los vectores del conjunto de entrenamiento y $\vec{w}$ es la norma. El problema anterior se reescribe como un problema de minimización:

$$\begin{array}{l} \text{Minimizar}_w \frac{1}{2}\|\vec{w}\|^2 \\ \text{sujeta a:} \\ y_i - (\vec{w} \cdot \vec{x}_i - b) \leq \epsilon \\ (\vec{w} \cdot \vec{x}_i + b) - y_i \leq \epsilon \end{array} \tag{3.17}$$

Al agregar la función de pérdida ϵ-insensible al problema anterior, queda

$$\begin{array}{ll} \text{Minimizar}_{\{w,\zeta\}} & \frac{1}{2}\|\vec{w}\|^2 + C\frac{1}{\ell}\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ \text{sujeta a:} & \\ & y_i - (\vec{w} \cdot \vec{x}_i + b) \leq \epsilon + \xi_i^* \\ & (\vec{w} \cdot \vec{x}_i + b) - y_i \leq \epsilon + \xi_i \\ & \xi_i, \xi_i^* \geq 0 \quad i = 1, \ldots, \ell \end{array} \tag{3.18}$$

El problema anterior puede resolverse mediante un método de programación cuadrática. Además de ϵ-SVR existe otro tipo de regresión en SVM, *nu*-SVR, aunque ϵ-SVR es el modelo más empleado para la aproximación de funciones en algoritmos evolutivos asistidos por subrogados [72].

Las ventajas de las máquinas de soporte vectorial son: (1) la generalización del error de una máquina de soporte vectorial no depende de la dimensión del espacio de búsqueda [27], y (2) no tiende a sobreajustarse a un problema [72]. Por otro lado, las desventajas son: (1) el tiempo requerido de entrenamiento para una muestra grande, y (2) elección de la función *kernel* adecuada.

3.3.5. Kriging

Kriging, también conocido como regresión de proceso Gaussiano, es una buena opción para problemas de baja dimensionalidad y con un espacio de búsqueda amplio [43]. En la Ecuación 3.19 se muestra su representación en su forma más simple, donde $g(\vec{x})$ es un modelo global de la función original, generalmente un polinomio y en algunos casos se reduce a una constante, β. Por otro lado, Z es una función Gaussiana (con media cero y covarianza diferente de cero) que representa una desviación "regionalizada" a partir del modelo global.

$$\hat{f}(\vec{x}) = g(\vec{x}) + Z(\vec{x}) \tag{3.19}$$

La covarianza ($Z(\vec{x})$) se expresa mediante la Ecuación 3.20.

$$\mathrm{Cov}[Z(\vec{x}_j), Z(\vec{x}_j)] = \sigma^2 \mathrm{R}[R(\vec{x}_j, \vec{x}_k)], \quad j,k = 1,\ldots,N \tag{3.20}$$

donde σ es la desviación estándar de la respuesta de las soluciones de la muestra, R es la matriz de correlación simétrica de dimensión $N \times N$ con unos en la diagonal y R es la función de correlación entre $\vec{x}_j$ y $\vec{x}_k$ (soluciones de la muestra), la cual puede ser indicada por el usuario. La función de correlación más usada es la función exponencial Gaussiana y se muestra en la ecuación 3.21.

$$R(\vec{x}_j, \vec{x}_k) = \exp\left[-\sum_{i=1}^{n} \theta_i |x_{j,i} - x_{k,i}|^2\right] \tag{3.21}$$

donde θ_i son los parámetros de correlación desconocidos, $x_{j,i}$ y $x_{k,i}$ son la i-ésima variable de las soluciones $\vec{x}_j$ y $\vec{x}_k$. Por lo tanto, la predicción ($\hat{y}$) de una solución ($\vec{x}$) se muestra en la Ecuación 3.22 y es una función en términos de β y $\theta_i = 1, 2, ..., n$:

$$\hat{y} = \hat{\beta} + \mathrm{r}^T(\vec{x})\mathrm{R}^{-1}(\vec{y} - \beta \mathrm{I}) \tag{3.22}$$

donde $\hat{\beta}$ es el valor estimado de β, $\vec{y}$ se define como en 3.5, I es un vector unitario y $\vec{r}$ representa la correlación entre $\vec{x}$ y las soluciones de la muestra $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N$, los dos vectores son de longitud N:

$$\mathrm{r}^T(\vec{x}) = [R(\vec{x}, \vec{x}_1), R(\vec{x}, \vec{x}_2), \ldots, R(\vec{x}, \vec{x}_N)]^T$$

La estimación de los parámetros puede llevarse a cabo mediante el método de máxima verosimilitud o el método de mínimos cuadrados.

Una ventaja del modelo *kriging* es que proporciona información de intervalos de confianza para los valores estimados [27, 71], los cuales se pueden obtener sin un costo adicional. Sin embargo, *kriging* calcula la inversa de una matriz y este procedimiento incrementa en función de la dimensionalidad del problema [27, 71]; además, es sensible a la dimensionalidad del problema [71] y para una muestra bastante pequeña *kriging* no es capaz de guiar la búsqueda hacia el óptimo global [36].

3.4. Criterios de muestreo de relleno

Cuando se habla de optimización restringida asistida por subrogados, donde se modela tanto la función objetivo como las restricciones, el objetivo de un criterio de muestreo de relleno es balancear tanto la exploración como la explotación de la aproximación en todas las funciones, así los puntos mejorarán todos los modelos.

Parr *et al.* [54] proponen tres criterios de muestreo de relleno para optimización restringida: (1) función de predicción con una función de penalización de un paso, transforma el problema de optimización restringida a un problema sin restricciones, mediante un término que representa la penalización de aquellas soluciones no factibles, (2) mejora esperada de una función de predicción con una función de penalización de paso, es ideal para aquellos modelos que se les puede calcular su incertidumbre, tal como *kriging* u otro modelo alternativo basado en un proceso Gaussiano. Así, se pueden detectar aquellas zonas en donde se requiere mejorar el modelo, mediante la medida de mejora esperada ($E(I(\vec{x}))$). Este criterio de muestreo de relleno se define por la suma de $E(I(\vec{x}))$ y la penalización para aquellas soluciones infactibles, (3) mejora esperada con probabilidad de factibilidad, se define mediante el producto de $E(I(\vec{x}))$ de la función objetivo y la probabilidad de factibilidad, la cual se calcula para cada restricción.

3.5. Métodos de evaluación de modelos subrogados

Un modelo aproximado debe mejorar a lo largo del proceso de búsqueda, es decir, la aproximación de las soluciones debe ser más precisa. Para deter-

minar esta mejora se emplean los valores de las soluciones iniciales que se evaluaron en el modelo original y aquellas soluciones que se van agregando a lo largo del proceso de búsqueda, seleccionadas mediante algún criterio de muestreo de relleno.

3.5.1. Error absoluto con respecto al óptimo

En caso de conocer el óptimo factible ($\vec{x}^*$ con valor y^*), los métodos más comunes son [54]: el error absoluto en el espacio de la función objetivo (Ec. 3.23), o en el espacio de las variables (Ec. 3.24) con respecto a la mejor solución encontrada ($\vec{x}^b$ con valor y^b), es decir, la distancia Euclideana entre $\vec{x}^*$ y $\vec{x}^b$.

$$|y^* - y^b| = \sqrt{(y^* - y^b)^2} \tag{3.23}$$

$$\|\vec{x}^* - \vec{x}^b\| = \sqrt{\sum_{i=1}^{n} (x_i^* - x_i^b)^2} \tag{3.24}$$

donde n es el número de variables de diseño del problema.

3.5.2. *Bias* o sesgo

Mide la diferencia que existe entre la salida del modelo aproximado $(\hat{f}(\cdot))$[3] de los valores reales $(f(\cdot))$, se calcula como un promedio sobre todos los datos del conjunto de entrenamiento [57], se muestra en la ecuación 3.25.

$$E_{bias} = \frac{1}{|D|} \sum_{\vec{x}_i \in \mathcal{D}} (\hat{f}(\vec{x}_i) - f(\vec{x}_i))^2 \tag{3.25}$$

3.5.3. Varianza

Mide la extensión para la cual el modelo subrogado $\hat{f}(\vec{x})$ es sensible a un conjunto particular de datos $\mathcal{D}$ [57]. Cada conjunto de datos $\mathcal{D}$ corresponde a una muestra aleatoria de la función de interés.

$$E_{var} = \hat{f}(\vec{x}_i - \hat{f}(\vec{x}_i))^2 \ \ \forall \ \vec{x}_i \in \mathcal{D} \tag{3.26}$$

[3]En regresión, a la salida del modelo aproximado también se le conoce como valor esperado, el cual se representa por $E(\vec{x})$.

Cuando un modelo subrogado se ajusta a un conjunto particular (bajo *bias*) proporcionará un valor grande para la varianza.

Para reducir el *bias*, se debe considerar un modelo más complejo. Mientras que, para reducir la varianza, el número de puntos debe incrementarse [57].

3.6. Resumen del capítulo

En este capítulo se explicó la integración de los modelos subrogados en optimización, los cuales son capaces de simular el paisaje de aptitud de una función a partir de la información que proporcionan las soluciones evaluadas en la función original. También se describieron las dos etapas de un modelo en el proceso de búsqueda, la primera consiste en la construcción del modelo mediante un muestreo inicial, mientras que la segunda es su actualización. Además, los modelos kNN, RSM, ANNs, SVM y Kriging se explicaron de manera general. Finalmente se definió la función de los criterios de muestreo de relleno y de los métodos para medir la calidad de los modelos, tales como: el error absoluto, sesgo o varianza.

Capítulo 4

Algoritmos Evolutivos Asistidos por Subrogados

El uso de modelos subrogados en un algoritmo evolutivo se puede considerar en diferentes etapas, desde la generación de la población inicial hasta la aproximación de soluciones a lo largo del proceso de búsqueda. En este capítulo se explica dónde y cómo se integra un modelo subrogado en un algoritmo evolutivo. Además, se listan los trabajos que han resuelto problemas de optimización con restricciones mediante la aproximación de soluciones.

4.1. Integración de un modelo subrogado en un algoritmo evolutivo

La integración de un modelo subrogado en un algoritmo evolutivo recibe el nombre de algoritmo evolutivo asistido por subrogado (SA-EA, por sus siglas en inglés). Las etapas donde puede integrarse un modelo subrogado en un algoritmo evolutivo son [27]: Población inicial, operadores de variación, evaluación y buscador local[1]. El uso de un modelo subrogado permite filtrar soluciones que de acuerdo a sus valores aproximados no son competitivas, es decir, no mejoran el proceso de búsqueda y por lo tanto son ignoradas. Con excepción de la etapa de evaluación, en donde algunos enfoques combinan soluciones aproximadas (evaluadas en el modelo subrogado) o soluciones exactas (evaluadas en el modelo original).

[1] Un buscador local realiza la búsqueda de una mejor solución dentro del vecindario de la solución de inicio.

El uso de modelos aproximados en la evaluación de soluciones se clasifica en [74]:

1. Todas las soluciones de la generación son aproximadas. En este enfoque se requiere de un modelo subrogado con alta precisión, pues la búsqueda se guía de acuerdo a la evaluación de las soluciones en este modelo, lo cual permite un ahorro de evaluaciones considerable.

2. Dentro de una generación existen soluciones aproximadas y soluciones exactas, es decir, evaluadas en el modelo original. La combinación de soluciones aproximadas y exactas, requiere de un control de evolución, también llamado manejador de modelo, el cual selecciona aquellas soluciones que se evalúan en el modelo original. En este enfoque se requiere un modelo subrogado preciso, ya que se comparan soluciones exactas contra soluciones aproximadas. En caso de que la precisión no sea buena, la búsqueda puede conducir a un falso óptimo.

3. Todas las soluciones son exactas. El modelo aproximado se emplea para encontrar la mejor solución a partir de un área local, la cual se evalúa en el modelo original y se integra en la población de la generación actual. El uso de un modelo subrogado en la población inicial, operadores de variación y buscador local entran en esta clasificación.

En el presente trabajo se integra el modelo subrogado en la etapa de evaluación, por lo que los elementos necesarios para incorporar un modelo subrogado en esta etapa son: conjunto de entrenamiento, control de evolución y medida de calidad del modelo (ver Fig. 4.1).

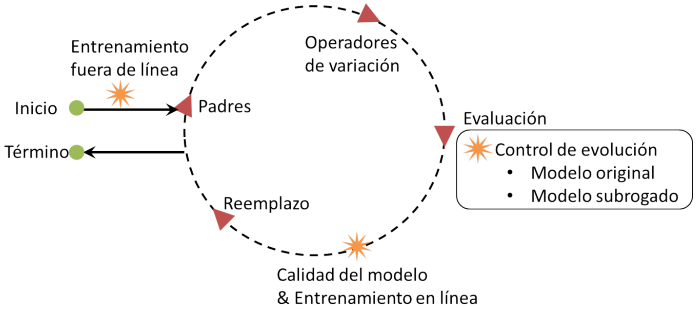


Figura 4.1: Los triángulos representan los elementos de un AE y las estrellas naranjas son los elementos necesarios para integrar un modelo subrogado en la etapa de evaluación

4.1.1. Conjunto de entrenamiento

El conjunto de entrenamiento se constituye de soluciones exactas, pues a partir de esta información se construye el modelo subrogado. Este conjunto se actualiza durante el proceso de búsqueda, por lo tanto el modelo también debe actualizarse.

4.1.2. Control de evolución

El control de evolución también se conoce como manejador de modelo y determina qué soluciones se evalúan o re-evalúan en el modelo original. Existen dos tipos: basado en individuo y basado en generación.

Basado en individuo Combina soluciones exactas y aproximadas dentro de una generación [29]. La metodología que se emplea generalmente es aproximar toda la población, posteriormente se selecciona un conjunto de soluciones para reevaluar en el modelo original. No existe un número definido de soluciones para reevaluar, sin embargo, se puede adaptar de acuerdo a la calidad del modelo [27]. Existen tres criterios de selección con base en el valor de aptitud: (1) reevaluar las mejores aproximadas, (2) reevaluar las peores aproximadas, y (3) evaluar soluciones aleatorias, donde no es necesario aproximar a toda la población. Otro enfoque es mediante un método de preselección, ya sea para seleccionar la población de hijos para las estrategias evolutivas [14] o para seleccionar las soluciones más representativas de acuerdo a su ubicación, lo anterior se realiza mediante *clusters*.

Basado en generación Consiste en evaluar a toda la población cada g generaciones o cuando la condición de uso se activa. Dicha condición puede estar en función de la calidad del modelo, es decir, de su precisión.

4.1.3. Medida de calidad del modelo

Las medidas comúnmente usadas son: el error cuadrático medio (Ec. 4.1) y el coeficiente de correlación (Ec. 4.2),

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{4.1}$$

$$r = \frac{n\sum \vec{x}_i y_i - \sum \vec{x}_i \sum y_i}{\sqrt{n\sum \vec{x}_i^2 - (\sum \vec{x}_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \tag{4.2}$$

donde $\hat{y}_i$ es el valor aproximado, y_i es el valor exacto y $\vec{x}_i$ es una solución.

4.1.4. Modelos subrogados

En [72] se realiza una clasificación de los modelos subrogados en algoritmos evolutivos, en la cual se encuentran modelos como kNN, perceptrón multicapa, redes neuronales artificiales, *kriging*, entre otros (ver Sec. 3.3). Adicionalmente, en los algoritmos evolutivos existe la *herencia de aptitud* (de padres a hijos), la *imitación de herencia* (clusters) y la *asignación de aptitud* (sistemas coevolutivos), cuyo objetivo es ahorrar evaluaciones mediante la estimación de aptitud a partir de soluciones similares [27].

4.1.5. Construcción de un modelo subrogado

Los modelos subrogados que se han integrado en un algoritmo evolutivo para aproximar el valor de aptitud de un problema de optimización son: modelos polinomiales (RSM), redes neuronales artificiales (ANNs), funciones de base radial (RBF), *kriging*, máquinas de soporte vectorial (SVM) y kNN. Se han realizado diversos estudios para determinar qué modelo es el más adecuado; sin embargo, sólo se han establecido ventajas y desventajas de cada uno [27]. También se han estudiado estos modelos como globales y locales, y se ha determinado que los segundos son mejores que los primeros, ya que son más precisos.

Debido a la aproximación de los valores de aptitud de una población, es necesario que el modelo seleccionado para llevar a cabo esta tarea sea el más adecuado. En consecuencia existen varias métricas para validar que modelo es el más apropiado [57]:

División de muestras La muestra se divide en un conjunto de entrenamiento y un conjunto de prueba, con el primero se construye el modelo y con el segundo se comprueba su precisión. La desventaja de este método es la estimación de error generalizado, ya que puede indicar una alta varianza. Además, para conjuntos de entrenamiento pequeños, limita la cantidad de datos para construir el modelo.

Cross-validation Los datos se dividen en k subconjuntos de igual tamaño. Un modelo se construye k veces, considerando cada uno de los subconjuntos para entrenamiento y el resto se puede considerar para prueba. Así, el error final es el promedio de los k errores. Su desventaja es que requiere construir k modelos.

Bootstrapping El conjunto de datos se considera como una población a partir de la cual se consideran las muestras. Existen diferentes variantes

mediante las cuales se puede identificar el mejor modelo y sus intervalos de confianza para las salidas. Su principal desventaja es el gran número de subconjuntos requeridos para esta métrica.

4.2. Marco referencial

Todas las propuestas que se mencionan a continuación surgieron con el objetivo de alcanzar la solución óptima o una solución competitiva, reduciendo el número de evaluaciones en la función objetivo y restricciones mediante el uso de modelos subrogados.

En SA-Op el modelo más empleado es *kriging*, las propuestas que usan este modelo aprovechan los intervalos de confianza de los valores estimados para balancear la exploración y explotación en todos los modelos. Una de las propuestas más conocidas es EGO [34, 70], el cual es un algoritmo de optimización global que emplea *kriging* para aproximar soluciones de problemas costosos. Aunque EGO se diseñó para resolver problemas de optimización sin restricciones, en [70] se resolvió un problema de optimización automotriz, que tiene una restricción de desigualdad. Debido al buen desempeño de EGO, éste se retomó en [68], donde se clasifican los problemas de acuerdo a su complejidad (con respecto al costo) de su función objetivo y restricciones, por tanto, se consideran ocho tipos de problemas. Aquellos problemas con funciones costosas se aproximan mediante EGO y los problemas con funciones no costosas se evalúan con DIRECT [33], un algoritmo de búsqueda global directo. Esta variante de EGO se enfocó en resolver aquellos problemas donde existen tanto funciones costosas y no costosas y se probó en cinco problemas de optimización con restricciones de desigualdad, con el objetivo de encontrar una solución factible cercana al óptimo (una región de 5 %) con el menor número de evaluaciones. En 2002, Sasena *et al.* realizan una extensión de EGO, denominada superEGO [69], la cual combina la evaluación en el modelo original de las restricciones no costosas, mientras que el resto las aproxima, esto lo combina en un criterio de muestreo de relleno. El conjunto de funciones de prueba para superEGO se compone de dos problemas con una región factible discontinua, la cual se forma mediante restricciones de desigualdad. Para medir la eficiencia de superEGO, se cuenta el número de evaluaciones requeridas para encontrar una solución óptima cercana al óptimo (una región de 1 %). En [54], Parr emplea *kriging* para resolver problemas de optimización restringida, además estudia la exploración y explotación de tres diferentes criterios de muestreo de relleno. Para definir su desempeño, se realizaron pruebas en dos problemas con restricciones de desigualdad y en el

problema de diseño del ala de un avión. Posteriormente, en [31] compara un enfoque simple contra un enfoque multiobjetivo para resolver problemas de optimización con restricciones mediante optimización asistida por subrogados, el modelo empleado es *kriging*. Las pruebas de este enfoque se realizaron en tres problemas con restricciones de desigualdad y el problema de diseño de una viga. Basudhar *et al.* [5] proponen un método de optimización basado en subrogados, el cual emplea *kriging* para aproximar la función objetivo y los límites de la región factible mediante SVM. El desempeño de la propuesta se analiza en cuatro problemas con restricciones de desigualdad.

RBF se ha integrado como modelo subrogado en: DYCORS [62], ORBIT[82], CONORBIT [61] y COBRA [59], todos propuestos por Regis. DYCORS emplea una estrategia de búsqueda coordinada dinámica para generar nuevas soluciones y resuelve problemas sin restricciones. Por otro lado, ORBIT y CONORBIT son algoritmos libres de derivadas pues trabajan con regiones de confianza. El primero resuelve problemas sin restricciones y el segundo con restricciones, los cuales son dos problemas de aplicación y veintisiete problemas de prueba, todos incluyen restricciones de desigualdad. Finalmente, COBRA se compone de dos etapas, la primera tiene como objetivo alcanzar la región factible, mientras que la segunda se enfoca en encontrar el óptimo factible y se prueba en un conjunto de 20 problemas de prueba y un problema de aplicación, todos tienen restricciones de desigualdad. Tang *et al.* [76] proponen un nuevo modelo híbrido, el cual combina RBF y RSM. Además, el gradiente de las restricciones y la función objetivo se emplean para guiar los puntos hacia la región factible. Dicha propuesta se probó en un conjunto de problemas de ingeniería, los cuales sólo contienen restricciones de desigualdad. Ong *et al.* [53] emplean RBF como modelos subrogados locales, los cuales trabajan en paralelo, el conjunto de prueba de esta propuesta se compone de 2 problemas sin restricciones y el problema de diseño del ala de un avión. Villanueva *et al.* [80] dividen el espacio de búsqueda en subregiones, las cuales son exploradas por agentes que emplean modelos subrogados para evaluar las soluciones y encontrar la mejor. Para determinar el funcionamiento de su propuesta, realizaron pruebas en problemas con restricciones de desigualdad.

En [37] y [4], se hacen mejoras en COBRA, en la primera se agrega un mecanismo de reparación de soluciones no factibles, el cual se basa en la aproximación de soluciones. Mientras que, en la segunda, la mejora consiste en que sus parámetros se autoajusten a lo largo del proceso de búsqueda. El conjunto de funciones de prueba de cada mejora fueron problemas con restricciones de desigualdad.

Los enfoques de SA-EA son: ASAGA [71], el cual emplea: i) el promedio de la población, ii) kNN, iii) modelos polinomiales (lineal y cuadrático) y iv) SVM, como modelos subrogados durante el proceso de búsqueda. Dichos modelos se combinan para una aproximación global y varias aproximaciones locales (mediante clusters), excepto SVM que sólo se emplea como modelo global. ASAGA se probó en problemas de optimización de ingeniería, los cuales contienen restricciones de desigualdad.

SBSM-GA [17] integra el modelo de regresión kNN para aproximar un porcentaje de la población en cada generación en un algoritmo genético. Esta propuesta sólo se prueba en problemas con restricciones de desigualdad. Runarsson integra el mismo modelo subrogado dentro de una estrategia evolutiva mejorada ($\hat{i}ES$) [66], como control de evolución emplea SR, esto es, aproxima todas las soluciones de la población, las ordena y la mejor solución aproximada la evalúa en el modelo original. El proceso anterior lo realiza desde 1 hasta el tamaño de la población (λ) [64] o hasta un cuarto del tamaño de la población ($\lambda/4$) [65] en cada iteración. Las propuestas previas se probaron en un conjunto de problemas bien definidos, los cuales tienen tanto restricciones de igualdad como desigualdad.

Regis ha propuesto varios enfoques usando RBF aumentadas por un modelo lineal, en la mayoría de ellos, como modelo subrogado global, las más recientes son: SA-EP [60] y TRICEPS [58]. En ambas propuestas, el algoritmo de búsqueda es programación evolutiva. Sin embargo, en TRICEPS refina la búsqueda de la mejor solución de cada iteración mediante regiones de confianza. El conjunto de prueba para SA-EP y TRICEPS está formado por problemas con restricciones de desigualdad y un problema de aplicación (MOPTA08). El mismo modelo se ha aplicado en [11, 61] para resolver problemas multiobjetivo con restricciones. Jin *et al* [29] aproximan cada restricción no lineal mediante ANNs, estos modelos se incorporaron en un algoritmo de estrategias evolutivas que emplea jerarquización estocástica (SRES, por sus siglas en inglés). Para determinar el desempeño de esta propuesta, se realizaron pruebas en un conjunto de problemas con restricciones de igualdad y/o desigualdad. Goh *et al.* [20] propusieron SCCMA, el cual emplea modelos subrogados en la búsqueda local. El desempeño de SCCMA se determinó mediante un conjunto de problemas que están definidos por restricciones de igualdad y/o desigualdad. Brownlee y Wright [7] emplean RBFs como modelos subrogados en NSGA-II. Kramer *et al.* [40] proponen un método novedoso para manejo de restricciones para el algoritmo de estrategias evolutivas mediante la adaptación de la matriz de covariancia (CMA-ES, por sus siglas en inglés), en el cual ajustan la matriz de covarianza mediante

los modelos subrogados de las restricciones. El desempeño de este enfoque se analizó probando el algoritmo en dos problemas con restricciones de igualdad y desigualdad.

En [6] se propone una nueva tendencia en SA-EA, que consiste en aproximar todas las soluciones de una población mediante modelos subrogados locales, para poder identificar una zona prometedora y posteriormente realizar una búsqueda dentro de esa región, esta propuesta se emplea para resolver problemas multiobjetivo y se denomina multiobjetivo asistido por subrogados (SAMO, por sus siglas en inglés). En la segunda parte de este trabajo se adopta el enfoque de SAMO, con el objetivo de reducir aún más el número de evaluaciones empleadas por SA-DECV para resolver problemas de optimización restringida

Dentro de los SA-EAs, los modelos subrogados se emplean para reducir evaluaciones en el modelo original. Por lo tanto, esta bondad se está aprovechando en aquellos algoritmos evolutivos con múltiples operadores de variación. En [21] se crea un conjunto de hijos a partir de diversos operadores de variación, seleccionando el mejor hijo de acuerdo a una función de densidad, el cual se evalúa en el modelo original. Mallipeddi [45] propone ESMDE, el cual es una integración de las cruzas y mutaciones de la evolución diferencial. ESMDE se auxilia de un modelo subrogado (*kriging*) construido a partir de la población actual, para generar hijos competitivos mediante los parámetros correctos a lo largo del proceso de búsqueda. Las propuestas anteriores se diseñaron para resolver problemas sin restricciones y sólo tienen soluciones exactas en cada generación, este enfoque también se encuentra en: Takahama y Sakai [74] estiman el valor de la función objetivo y el de la violación de restricciones de cada vector *trial* en DE mediante un *kernel* de regresión. Dicha propuesta se probó en un conjunto de problemas con restricciones de igualdad y/o desigualdad. Kong *et al.* predicen las restricciones mediante la condición de *Lipschitz* en una nueva versión adaptativa de evolución diferencial. Para determinar el desempeño de su propuesta, se empleó un conjunto de problemas sin restricciones y con restricciones.

En [26], Innocente *et al.* resuelven el problema de inundación de agua mediante *kriging*. En esta propuesta, el modelo subrogado sólo se entrena al inicio del proceso de búsqueda, posteriormente no se lleva a cabo ninguna simulación.

Como puede notarse a partir de la revisión de la literatura mencionada, el principal objetivo de estos enfoques consiste en agregar un modelo subrogado en un algoritmo evolutivo para reducir el número de evaluaciones en el modelo original. Además. la mayoría de los trabajos mencionados aproxima

cada una de las restricciones, siendo pocos los que aproximan la suma de violación de restricciones y que estudian el efecto de un modelo subrogado en la selección que se realiza a partir de una técnica para manejo de restricciones. Adicionalmente, la mayoría de estos trabajos se centra en resolver problemas con restricciones de desigualdad, ya que se pueden satisfacer con mayor facilidad que una restricción de igualdad.

Debido a lo anterior, en el presente trabajo se proponen dos enfoques: (1) mediante un único modelo subrogado que aproxima tanto el valor de la función objetivo como la suma de violación de restricciones. En esta propuesta se realiza un estudio sobre el efecto de diferentes técnicas para manejo de restricciones en una propuesta particular para resolver problemas de optimización restringida (ver Cap. 5), y (2) mediante modelos subrogados locales, los cuales se construyen para cada función del problema, se pretende alcanzar la zona factible tanto de problemas con restricciones de desigualdad, como con restricciones de igualdad (ver Cap. 7).

4.3. Resumen del capítulo

En este capítulo se explicó la integración de modelos subrogados en un algoritmo evolutivo. Además, se definieron los elementos necesarios para integrarlos, enfocándose en la evaluación de soluciones. Finalmente, se presentó el marco referencial de modelos subrogados empleados para resolver problemas de optimización con restricciones.

Capítulo 5

SA-DECV

En el presente capítulo se explica la propuesta de DECV asistido por un modelo subrogado (SA-DECV, por sus siglas en inglés). El objetivo es estudiar los efectos que sufre el proceso de búsqueda de DECV con diferentes técnicas para manejo de restricciones (CHT, por sus siglas en inglés), aproximando el valor de la función objetivo y la suma de violación de restricciones, resolviendo problemas de optimización restringida. El enfoque se prueba en un conjunto de problemas artificiales y de ingeniería, comparando los resultados contra los de algoritmos del estado del arte.

5.1. Propuesta

El esquema general para incluir un modelo subrogado en un algoritmo evolutivo es:

1: P_0 = población inicial (*entrenamiento fuera de línea*)
2: $train_{set} = P_0$
3: Construir el modelo
4: Generar la nueva población mediante los operadores de variación del EA
5: Aproximar y evaluar a la nueva población (*control de evolución*)
6: *Medir la calidad del modelo y entrenamiento en línea*
7: Reemplazo
8: Se cumple condición de paro, sí terminar, sino ir a paso 2

donde el texto en *cursivas* indica los elementos que se deben integrar dentro del algoritmo evolutivo para poder aproximar las soluciones de la población y $train_{set}$ representa el conjunto de entrenamiento.

En este enfoque se emplea DECV como algoritmo de búsqueda, un control de evolución basado en factibilidad, que establece el balance entre las soluciones aproximadas y exactas en cada generación, y kNN como modelo subrogado (ver Sec. 3.3.1), el cual aproxima tanto el valor de la función objetivo como la suma de violación de restricciones. En la Fig. 5.1 se muestra el pseudocódigo de esta propuesta.

1: P (Población inicial)
2: Evaluar (P)
3: P = Optimización_basada_Oposición
4: $train_{set}$ = *Off_line_training*
5: g = 1
6: **while** g < MAX_GEN **do**
7: u = Operadores_DECV
8: ind_{orig} = *evolution_control*
9: Evaluar(u)
10: MSE = *Measure_model_quality*
11: $train_{set}$ = *On_line_training*
12: P = Selección
13: g = g + 1
14: **end while**
15: Return best solution

Figura 5.1: SA-DECV emplea un control de evolución basado en factibilidad (línea 8), el cual selecciona un conjunto de soluciones para ser evaluadas en el modelo original. Los elementos que se agregaron para integrar un modelo subrogado en DECV se presentan en cursiva. En la función Selección (línea 12), se emplea el CHT.

Es preciso mencionar que tanto DECV como kNN son fáciles de implementar y emplean pocos parámetros para trabajar. Además, los parámetros adicionales para integrar kNN en DECV son tres: el tamaño del conjunto de entrenamiento ($|train_{set}|$), el porcentaje de soluciones para calcular la calidad del modelo (Fig. 5.1, línea 10), que es 5 %NP, y el número de vecinos más cercanos (k) requerido por kNN para aproximar las soluciones (Fig. 5.1, línea 9), el cual es igual a la dimensionalidad del problema.

En el resto de la sección se explica cada elemento de SA-DECV.

5.1.1. Población inicial

La población inicial (P_0) se genera de manera aleatoria con una distribución uniforme. Después, se calculan los puntos opuestos ($\breve{P}_0$) de P_0 y se realiza una selección entre P_0 y $\breve{P}_0$ de acuerdo a las reglas de Factibilidad, por consiguiente la población con la que se inicia el proceso de búsqueda está compuesta de las mejores soluciones entre P_0 y $\breve{P}_0$.

5.1.1.1. Puntos opuestos

Los puntos opuestos se calculan mediante la Ecuación 5.1,

$$\breve{x}_i = \vec{L} + \vec{U} - \vec{x}_i \tag{5.1}$$

donde $\breve{x}_i$ es el punto opuesto de $\vec{x}_i$ y $\vec{L}$ y $\vec{U}$ son los límites inferior y superior de las variables de diseño, respectivamente.

5.1.2. Conjunto de entrenamiento

El conjunto de entrenamiento, denotado por $train_{set}$, contiene las soluciones evaluadas en el modelo original, así P_0 y $\breve{P}_0$ se emplean para inicializarlo. Su tamaño es $(2 \times NP)$, valor sugerido en [17]. La actualización de $train_{set}$ se lleva a cabo con las soluciones que se evalúan en el modelo original en cada iteración.

5.1.3. Control de evolución

El compromiso entre las soluciones exactas (evaluadas en el modelo original) y las soluciones aproximadas (evaluadas con kNN) se establece mediante un control de evolución basado en factibilidad, el cual se presenta en la Fig. 5.2, donde se verifica si hay soluciones factibles en la población (P_{G-1}) para determinar qué soluciones de los hijos (u) se evalúan en el modelo original (ind_{orig}). El proceso de selección se lleva a cabo en función de la distribución de u, el conjunto de entrenamiento ($train_{set}$) y el porcentaje de generaciones (per_{gen}).

La evaluación de la población se realiza al principio del proceso de búsqueda (Fig. 5.1, Paso 2), una vez que se alcanza la región factible, se selecciona un conjunto de soluciones mediante la función *Choose_ vectors*. Dicha selección consiste en acotar el espacio de búsqueda a una región, así las soluciones que se encuentren dentro se evalúan con el modelo original.

Require: P_{G-1}, u, $train_{set}$, per_{gen}
Ensure: ind_{orig}
1: **if** there are feasible solutions in P_{G-1} **then**
2: $ind_{orig} = Choose\_vectors$
3: **else**
4: $ind_{orig} = u$
5: **end if**

Figura 5.2: Pseudocódigo del control de evolución.

Los criterios para acotar la región que se evaluará en el modelo original son: (1) un hiperespacio definido por la ubicación de las soluciones factibles y el extremo más cercano del espacio de búsqueda. Esta región es una buena opción principalmente para aquellos problemas que tienen su óptimo en los límites del espacio de búsqueda. Sin embargo, el número de soluciones que se evalúan en el modelo original puede ser grande. Además, no garantiza que la región factible se encuentre dentro de esa zona, (2) una hiperesfera con centro en la mejor solución de $train_{set}$ y cuyo radio está en función de la diferencia de los centroides de $train_{set}$ y u, y per_{gen}. La ubicación de la hiperesfera puede abarcar tanto la región factible como la no factible, además emplea menos evaluaciones en el modelo original que el hiperespacio. Por las razones anteriores, se optó por definir una hiperesfera para acotar la región de soluciones a evaluar en el modelo original.

El pseudocódigo de la función *Choose_vectors* se presenta en la Fig. 5.3. Cuando P_{G-1} converge a una región pequeña, es decir, el promedio de las desviaciones estándar de cada variable de diseño (SD) es menor a 1E-6 (valor definido a partir de una serie de experimentos), todas las soluciones se aproximarán. En caso contrario, se selecciona un conjunto de soluciones que se encuentra dentro de la hiperesfera con centro en la ubicación de la mejor solución de $train_{set}$ (b_{ts}).

Para definir la hiperesfera se calculan los centroides de $train_{set}$ y u, denotados por m_{ts} y m_u respectivamente. Posteriormente, la distancia entre b_{ts} y cada uno de los centroides se calcula (d_{ts} y d_u, respectivamente). De acuerdo al valor de per_{gen}, el radio (r) de la hiperesfera se define de acuerdo a:

$$r = \begin{cases} \text{mín}(d_{ts}, d_u) & \text{if } per_{gen} > 50\,\% \\ \text{máx}(d_{ts}, d_u) & \text{Otherwise} \end{cases}$$

Finalmente, las soluciones con una distancia menor o igual a r se seleccionan. El proceso anterior se muestra en la Fig. 5.4.

Require: u, P_{G-1}, $train_{set}$, per_gen
Ensure: ind_{orig}
1: **if** Desviación estándar (SD) de $P_{G-1} <$ 1E-06 **then**
2: ind$_{orig} = \emptyset$ {Todas las soluciones se aproximan}
3: **else**
4: b_{ts} = La mejor solución del conjunto de entrenamiento
5: m_{ts} = *centroid* ($train\_set$) {*centroid* se calcula con el promedio de la población con respecto a sus distancias}
6: m_u = *centroid* (u)
7: d_{ts} = b_{ts} - m_{ts}
8: d_u = b_{ts} - m_u)
9: **if** per_gen > 50 **then**
10: $r = \min(d_{ts}, d_u)$ {mín(x, y) Retorna el valor mínimo entre x y y}
11: **else**
12: $r = \max(d_{ts}, d_u)$ {máx(x, y) Retorna el valor máximo entre x y y}
13: **end if**
14: ind$_{orig}$ = Soluciones en u con una distancia menor o igual a r
15: **end if**

Figura 5.3: Pseudocódigo de la función *Choose_ vectors*.

5.2. Diseño experimental

En esta sección se analiza el desempeño de SA-DECV, el cual aproxima el valor de la función objetivo y la suma de violación de restricciones mediante un modelo subrogado simple y fácil de implementar, kNN. Para ello, SA-DECV se prueba en un conjunto de 24 funciones bien conocidas. En la Tabla 5.1 se muestran sus características, tales como: número de variables, tipo de función objetivo, tipo y número de restricciones, número de restricciones activas y una estimación de la zona factible, denotada por ρ, la cual se calcula generando un conjunto de soluciones aleatorias (S), $|S|$=1000000, se contabilizan sólo las soluciones factibles (F) y se divide F entre S. En aquellos problemas con ρ igual a 0, ninguna de las soluciones aleatorias fue factible, por lo tanto, la zona factible del problema es muy pequeña. Para un mayor detalle de las funciones ver el Apéndice A.

SA-DECV se prueba con cuatro técnicas para manejo de restricciones: reglas de factibilidad, mecanismo de diversidad, método de ε-*constrained* y jerarquización estocástica (ver sección 2.4). Además, la mejor variante se compara contra dos algoritmos del estado del arte en algoritmos evolutivos que emplean subrogados. Finalmente se hace un análisis del desempeño de la mejor variante en un conjunto de problemas de ingeniería.

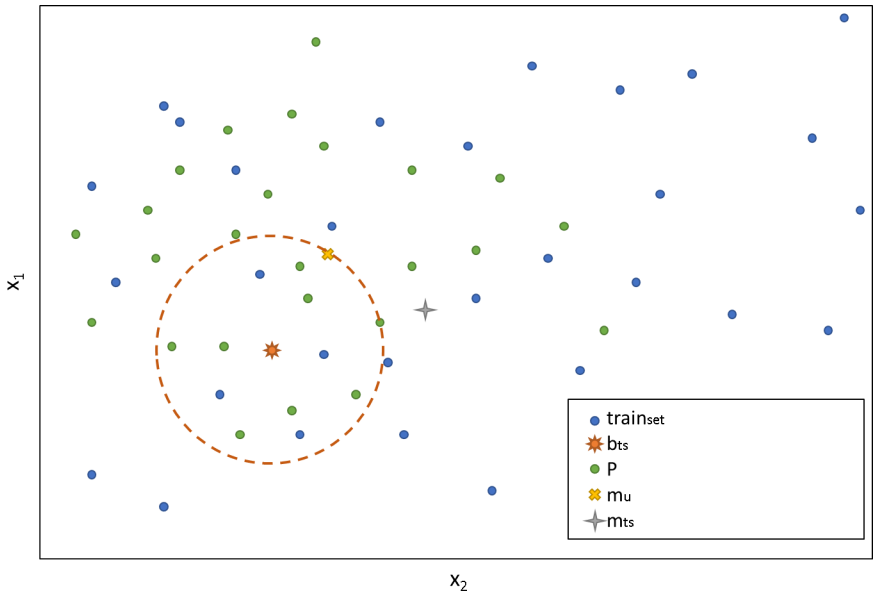


Figura 5.4: Definición de la esfera para el control de evolución basado en factibilidad, donde los puntos azules son el conjunto de entrenamiento ($train_{set}$), la estrella roja es la mejor solución de $train_{set}$, la población actual se representa por los puntos verdes P y los centroides de P y $train_{set}$ definen el radio de la esfera, considerando la menor diferencia r. Por lo tanto, todos aquellos puntos que tienen una distancia menor a r se evalúan en el modelo original.

5.2.1. Medidas de desempeño

Para medir el desempeño de SA-DECV con cada técnica para manejo de restricciones, se emplean cuatro medidas de desempeño [49], las cuales son:

Probabilidad de factibilidad (FP). Mide la razón entre el número de ejecuciones que alcanzan la zona factible (ft) y el número total de ejecuciones independientes (t), su fórmula se muestra en la Ecuación 5.2.

$$FP = \frac{ft}{t} \tag{5.2}$$

El valor de FP es un número real en el intervalo [0,1], donde uno indica que en todas las ejecuciones se encontró al menos una solución factible, cuando en ninguna ejecución se alcanza la región factible el valor es cero. De acuerdo a lo anterior, el valor ideal es uno.

Tabla 5.1: Detalles de los 24 problemas de prueba. n es el número de variables de decisión, $\rho = |F|/|S|$ es el relación estimada entre la región factible y el espacio de búsqueda, LI es el número de restricciones de desigualdad, NI es el número de restricciones de desigualdad no lineales, LE es el número de restricciones de igualdad lineales y NE es el número de restricciones de igualdad no lineales. a es el número de restricciones activas en el óptimo factible.

Prob.	n	Tipo de función	ρ	LI	NI	LE	NE	a
g01	13	cuadrática	0.0111 %	9	0	0	0	6
g02	20	no lineal	99.9971 %	0	2	0	0	1
g03	10	polinomial	0.0000 %	0	0	0	1	1
g04	5	cuadrática	52.1230 %	0	6	0	0	2
g05	4	cúbica	0.0000 %	2	0	0	3	3
g06	2	cúbica	0.0066 %	0	2	0	0	2
g07	10	cuadrática	0.0003 %	3	5	0	0	6
g08	2	no lineal	0.8560 %	0	2	0	0	0
g09	7	polinomial	0.5121 %	0	4	0	0	2
g10	8	lineal	0.0010 %	3	3	0	0	6
g11	2	cuadrática	0.0000 %	0	0	0	1	1
g12	3	cuadrática	4.7713 %	0	1	0	0	0
g13	5	no lineal	0.0000 %	0	0	0	3	3
g14	10	no lineal	0.0000 %	0	0	3	0	3
g15	3	cuadrática	0.0000 %	0	0	1	1	2
g16	5	no lineal	0.0204 %	4	34	0	0	4
g17	6	no lineal	0.0000 %	0	0	0	4	4
g18	9	cuadrática	0.0000 %	0	13	0	0	6
g19	15	no lineal	33.4761 %	0	5	0	0	0
g20	24	lineal	0.0000 %	0	6	2	12	16
g21	7	lineal	0.0000 %	0	1	0	5	6
g22	22	lineal	0.0000 %	0	1	8	11	19
g23	9	lineal	0.0000 %	0	2	3	1	6
g24	2	lineal	79.6556 %	0	2	0	0	2

En aquellos problemas donde ρ es cero, la región factible es muy pequeña con respecto al espacio de búsqueda.

Probabilidad de convergencia (P). Calcula la razón entre el número de ejecuciones exitosas (s) y el número total de ejecuciones independientes (t). En la Ecuación 5.3 se muestra su fórmula.

$$P = \frac{s}{t} \tag{5.3}$$

Una ejecución exitosa encuentra una solución factible $\vec{x}$ que cumple con lo siguiente: $f(\vec{x}) - f(\vec{x}^*)$ <1E-4. Los valores de P se encuentran en el mismo rango de FP, y al igual que FP, uno es el valor ideal.

Promedio de evaluaciones ($AFES$). Calcula el promedio de evaluaciones usadas ($EVAL$) en encontrar la primera solución que cumple los requisitos de una ejecución exitosa, mediante la Ec. 5.4.

$$AFES = \frac{1}{s}\sum_{i=1}^{s} EVAL_i \tag{5.4}$$

Para cualquier problema el valor de esta medida se prefiere bajo, principalmente en aquellos problemas donde la ejecución de una solución requiere de varios minutos, horas o días, es decir, con un alto costo computacional.

Rendimiento exitoso (SP). Mide la velocidad y confiabilidad de un algoritmo. Está definida por $AFES$ y P mediante la Ec. 5.5. Un valor bajo indica un mejor desempeño.

$$SP = \frac{AFES}{P} \tag{5.5}$$

5.2.1.1. Pruebas estadísticas

Para determinar si existe una diferencia entre los algoritmos de cada experimento del presente trabajo, se empleó la prueba de Kruskal-Wallis con un intervalo de confianza de 95 % usando las medianas de todos los problemas y la prueba post-hoc fue el método de Bonferroni. Los resultados de esta prueba, no mostraron una diferencia significativa entre los algoritmos de cada experimento. Por lo tanto, se aplicó el *test* de suma de rangos de Wilcoxon con un intervalo de confianza de 95 % para una comparación entre pares de algoritmos por cada problema, con excepción de las funciones g20 y g22, pues no se encontraron soluciones factibles con ningún algoritmo empleado en la parte experimental del trabajo. Los algoritmos base para el *test* estadístico en cada experimento se indican en cada capítulo en la sección de diseño experimental.

Notación

La notación empleada para identificar SA-DECV con cada técnica para manejo de restricciones es: SA-DECV$_{FR}$ para SA-DECV y las reglas de

factibilidad, SA-DECV$_\varepsilon$ para SA-DECV y el método de ε-*constrained*, SA-DECV$_{DM}$ para SA-DECV y el mecanismo de diversidad, y SA-DECV$_{SR}$ para SA-DECV y la jerarquización estocástica.

Los resultados de la mejor variante se comparan contra dos algoritmos del estado del arte, ASRES Y SBSM-GA, ya que ambas propuestas aproximan el valor de la función objetivo y la suma de violación de restricciones mediante el mismo modelo subrogado que SA-DECV. ASRES [65] es la versión mejorada de SRES, el cual emplea una estrategia evolutiva (μ, λ) y un control de evolución adaptativo basado en jerarquización estocástica. Por otro lado, SBSM-GA utiliza un algoritmo genético con representación binaria (código Gray), un control de evolución con selección aleatoria y un ordenamiento de burbuja basado en las reglas de factibilidad. La comparación con ASRES se realizó con los resultados que se reportan en [65].

Tabla 5.2: Parámetros de SA-DECV.

Parámetro	**Valor**	**Parámetro**	**Valor**
NP	90	$\|train_{set}\|$	$2\times NP$
Cr	1	avg_t	$5\,\%NP$
F	0.9	k	n
δ	1E-04		

Los parámetros de SA-DECV en cada experimento son los sugeridos en [48], para DECV (NP, Cr, F y δ), mientras que el resto de los parámetros se mencionan en la sección 5.1. En la Tabla 5.2 se presentan los parámetros y sus valores. Por otro lado, los parámetros de cada técnica para manejo de restricciones empleada en el primer experimento se presentan en la Tabla 5.3 y son los sugeridos para cada uno de ellos.

Tabla 5.3: Parámetros para cada CHT.

CHT	**Parámetros**
Reglas de factibilidad	-
Método ε-constrained	$cp = 10$, $T_c = 1500$ and $\theta = 1$ [75]
Jerarquización estocástica	$P_f = 0.45$ [66]
Mecanismo de diversidad	$S_r = 3\,\%$ [47]

Con respecto a los algoritmos del estado del arte, sus parámetros son los reportados en la literatura para cada uno de ellos. ASRES: μ=40, λ=400,

máximo número de evaluaciones por generación en el modelo original $\lambda/4$, tamaño del conjunto de entrenamiento $\ell=\lambda$ y el número de vecinos más cercanos $k=1$. Con respecto a SBSM-GA: tamaño de la población $\lambda=100$, probabilidad de cruza $p_c=0.72$ (2 puntos), porcentaje de mutación $p_m=0.10$, fracción de la población que se evalúa en el modelo original $p_{sm}=0.85$, tamaño del conjunto de entrenamiento $\eta=2\times\lambda$ y número de vecinos más cercanos $k=10$.

El número máximo de evaluaciones (MAX_FEs) en el modelo original para el primer experimento fue de 240000 y en el segundo 500000. Además, el número de ejecuciones fue de 30 para cada problema del primer experimento y 25 para cada problema del segundo experimento.

Loa algoritmos base para el *test* estadístico son: en el primero, aquella variante con el mejor desempeño de acuerdo a sus medianas y en el segundo, la mejor variante del primer experimento.

5.2.2. Comparación de SA-DECV con diferentes técnicas para manejo de restricciones

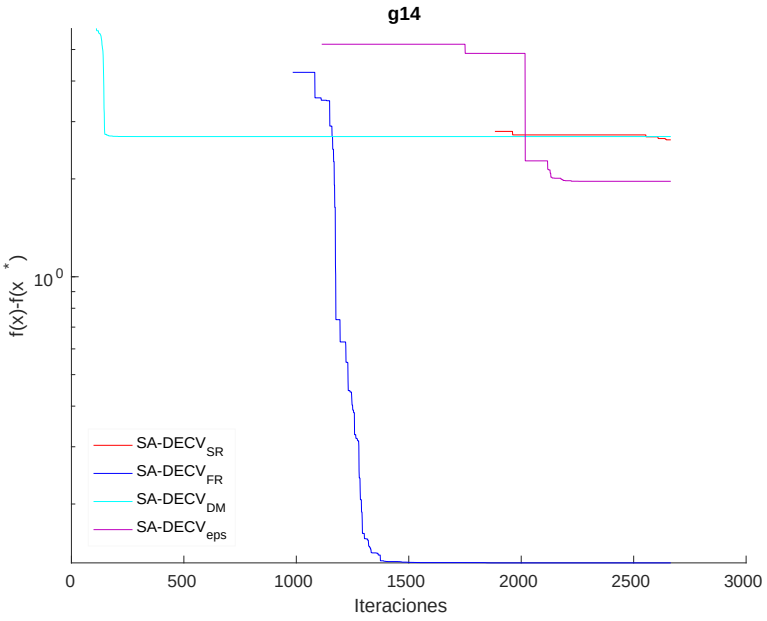
En esta subsección, cuatro técnicas para manejo de restricciones se emplearon en SA-DECV con el fin de estudiar sus efectos durante el proceso de búsqueda, por medio del análisis de convergencia, resultados estadísticos y medidas de desempeño.

5.2.2.1. Análisis de convergencia

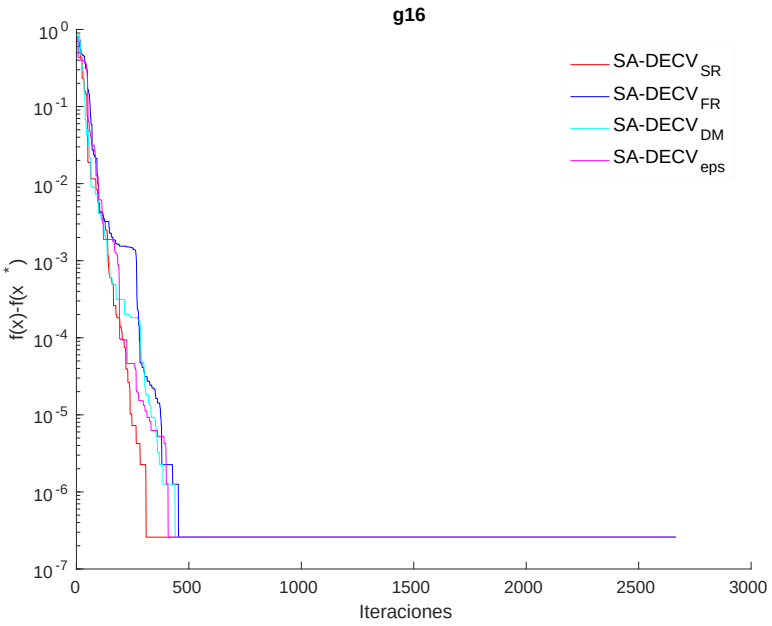
Para ilustrar el comportamiento del proceso de búsqueda, se presentan las gráficas de convergencia (de la ejecución localizada en la mediana del total de ejecuciones independientes realizadas) de tres funciones de prueba, las cuales se presentan en la Fig. 5.5. Cada gráfica presenta un tipo de problema: g16 (sólo restricciones de desigualdad), g14 (sólo restricciones de igualdad) y g23 (ambos tipos de restricciones).

De acuerdo a las gráficas de convergencia, para aquellos problemas con restricciones de desigualdad (Fig. 5.5 (b)), todas las variantes son capaces de encontrar una solución factible cerca de la vecindad de la mejor solución factible conocida. Sin embargo, en aquellos problemas con restricciones de igualdad (Fig. 5.5 (a) y 5.5 (c)) todas las variantes quedan atrapadas en un óptimo local.

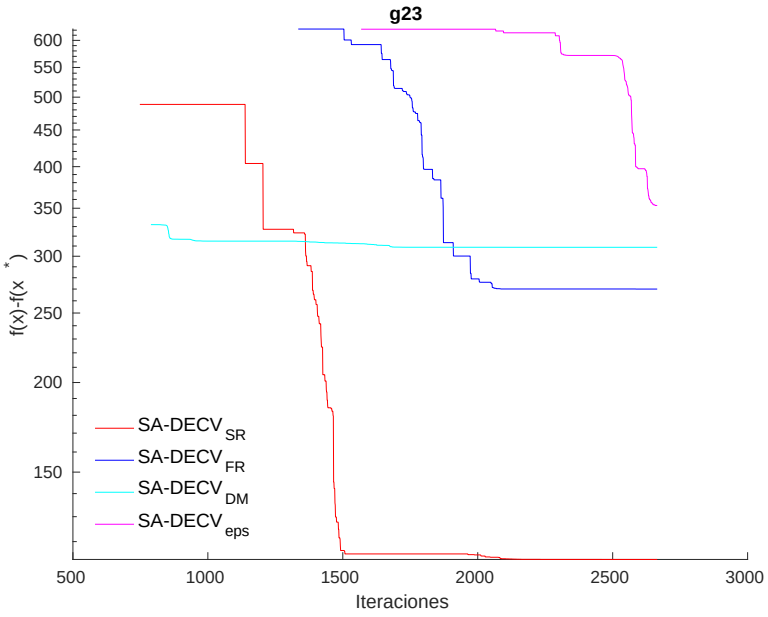
La convergencia de la población hacia la región factible se ilustra en los problemas g08 y g11, los cuales son de dimensionalidad dos. En la Fig. 5.6 se muestra la región factible y el óptimo de cada problema. Para el problema g08



(a) g14 tiene sólo restricciones de igualdad.



(b) g16 tiene sólo restricciones de desigualdad



(c) g23 tiene ambos tipos de restricciones

Figura 5.5: Gráficas de convergencia para cada clase de problema. El eje y está en escala logarítmica.

(Fig. 5.6 (a)) la región factible se forma por la intersección de las restricciones de desigualdad (líneas verdes), mientras que para g11 (Fig. 5.6 (b)) la región factible se define por una restricción de igualdad (línea azul marino). Los óptimos son los asteriscos rojos.

En las Fig. 5.7 y 5.8 se ilustra la convergencia de la población hacia la región factible en los problemas g08 y g11, respectivamente. En cada gráfica se muestra la población cada 5 iteraciones por las primeras 25 iteraciones del proceso de búsqueda.

En la Fig. 5.7 se aprecia que todas las variantes alcanzan la zona factible en las primeras 5 iteraciones, por lo tanto, se empieza a usar kNN para aproximar las soluciones que se encuentran fuera de la hiperesfera. En la iteración 10, la mayor parte de la población se encuentra en la zona factible, siendo el mecanismo de diversidad (Fig. 5.7 (a)) y las reglas de factibilidad (Fig. 5.7 (c)) las que tienen una mejor distribución de la población, lo cual permite una mejor exploración en la zona factible. Finalmente, en la iteración 25, ya todas las variantes convergieron a la vecindad del óptimo.

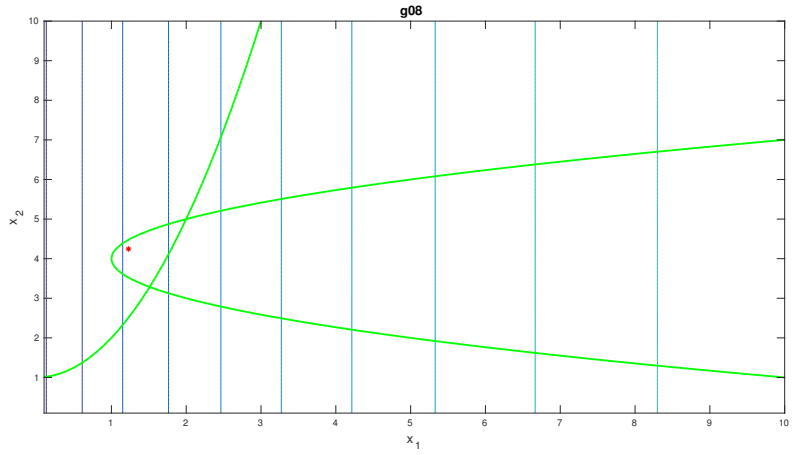
En la Fig. 5.8 se observa que todas las variantes se mueven hacia la región factible, sin embargo el mecanismo de diversidad (Fig. 5.8 (a)) muestra una convergencia prematura hacia una región lejana de la vecindad de la mejor solución factible conocida, mientras que las demás técnicas para manejo de restricciones conservan una mejor diversidad de la población, lo que permitirá mayor exploración de la región factible.

5.2.2.2. Resultados estadísticos

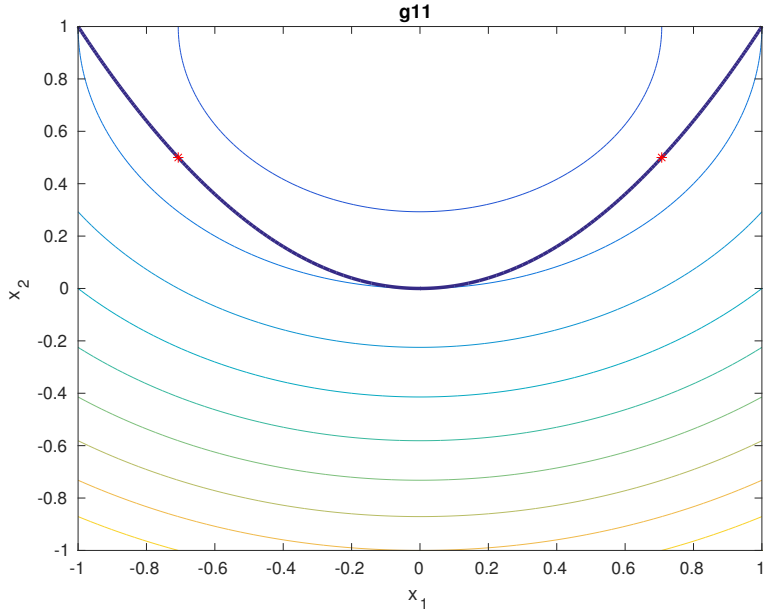
Los valores estadísticos (B:mejor, Md:mediana y SD:desviación estándar) de los resultados finales obtenidos por cada variante se presentan en las Tablas 5.4 y 5.5. Las mejores medianas en cada problema se muestran en **negrita**.

De acuerdo a las Tablas 5.4 y 5.5, el óptimo fue encontrado por las cuatro variantes de SA-DECV en 11 problemas (g01, g04-g06, g08-g09, g11-g12, g15-g16, y g24). Sin embargo, SA-DECV$_{SR}$ es la mejor variante ya que reporta la mejor mediana en 16 problemas. Por lo tanto, esta variante se tomó como el algoritmo base para el *test* de Wilcoxon. Los resultados se muestran en la tabla 5.6.

SA-DECV$_{SR}$ supera a SA-DECV$_{FR}$ en 8 problemas (g05, g07, g10, g13, g15, g17, g19, y g23). Por otro lado, SA-DECV$_{FR}$ tiene un mejor desempeño que SA-DECV$_{SR}$ en los problemas g02 y g14. En el resto de los problemas, se tiene el mismo comportamiento, de acuerdo al *test* de Wilcoxon.

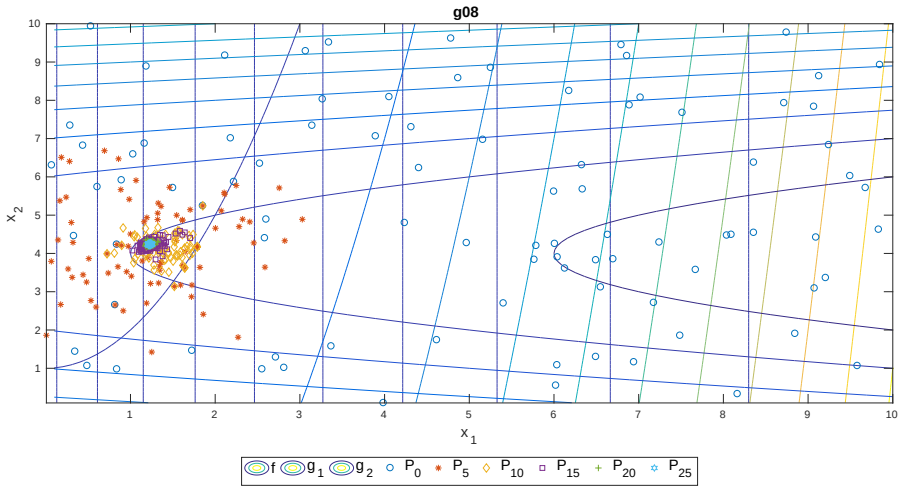


(a)

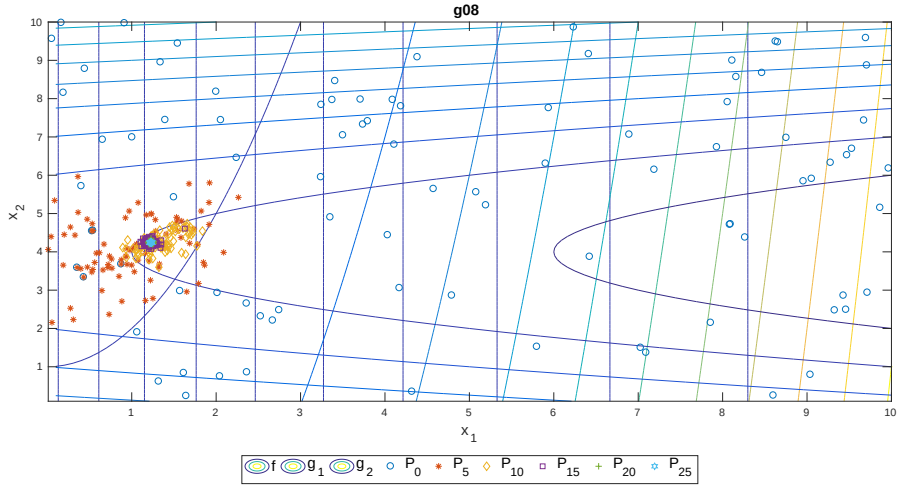


(b)

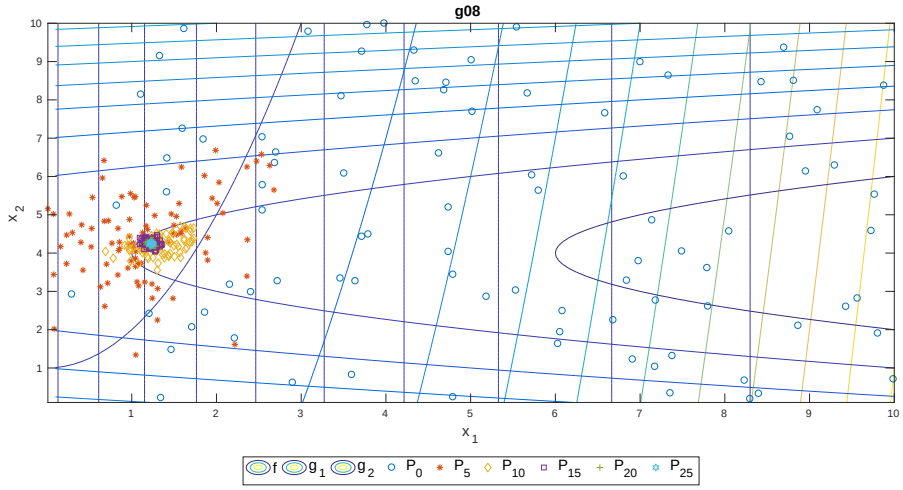
Figura 5.6: a) Curvas de nivel del problema g08, su región factible es la intersección de las líneas verdes. b) Curvas de nivel del problema g11, su región factible se encuentra sobre la línea azul marino. Los óptimos se representa por asteriscos rojos.



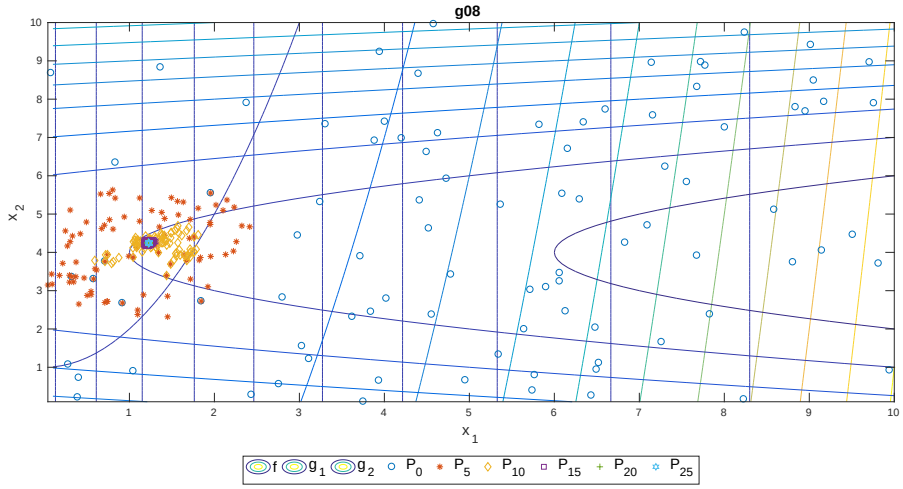
(a) Mecanismo de diversidad



(b) Método de ε-*constrained*

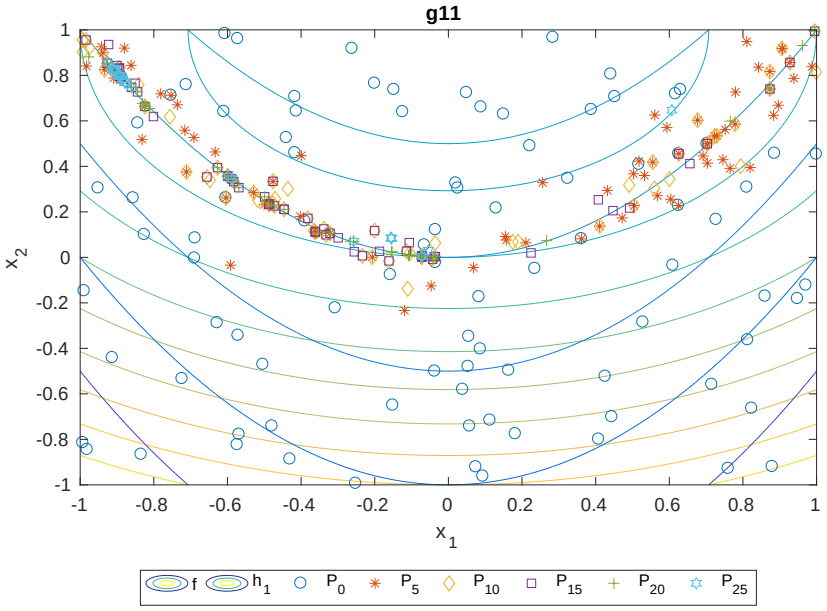


(c) Reglas de factibilidad

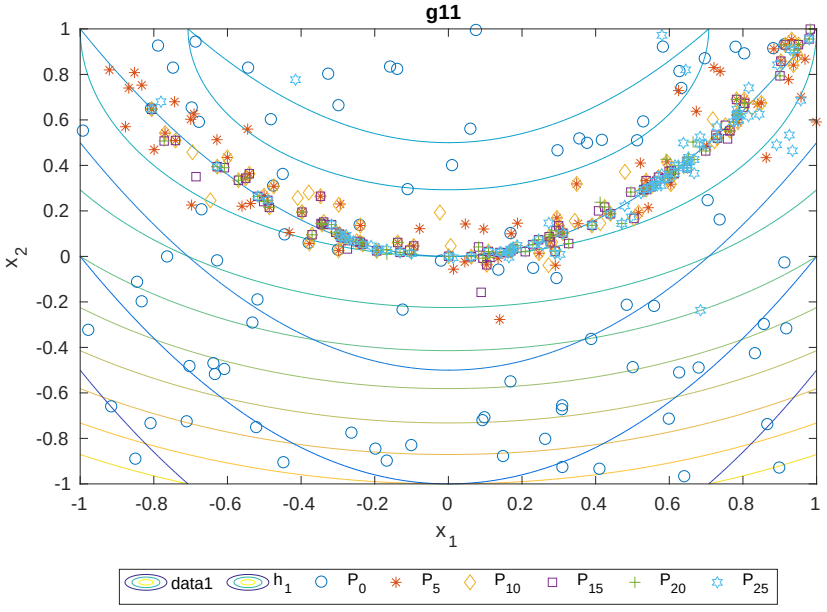


(d) Jerarquización estocástica

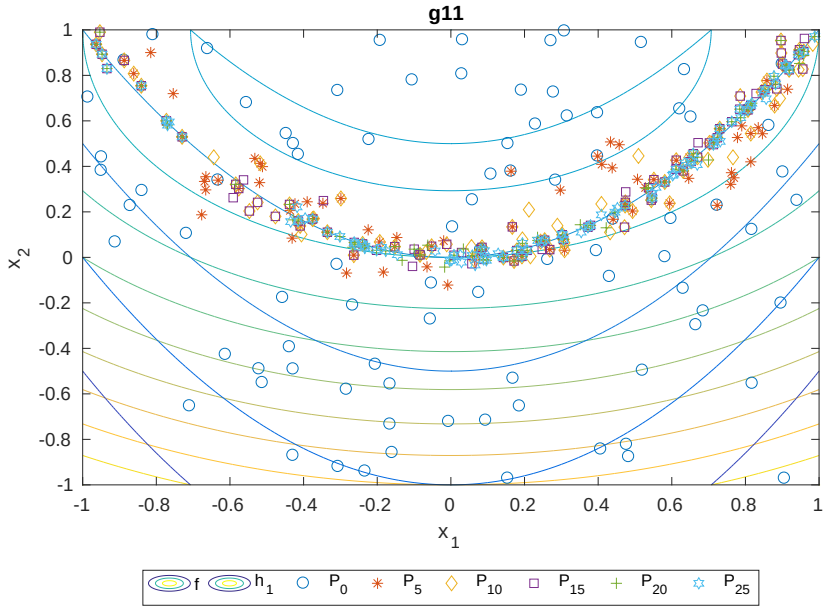
Figura 5.7: Convergencia de la población (desde la población inicial, P_0, hasta la población de la iteración 25, P_{25}) hacia la región factible para cada CHT en el problema g08.



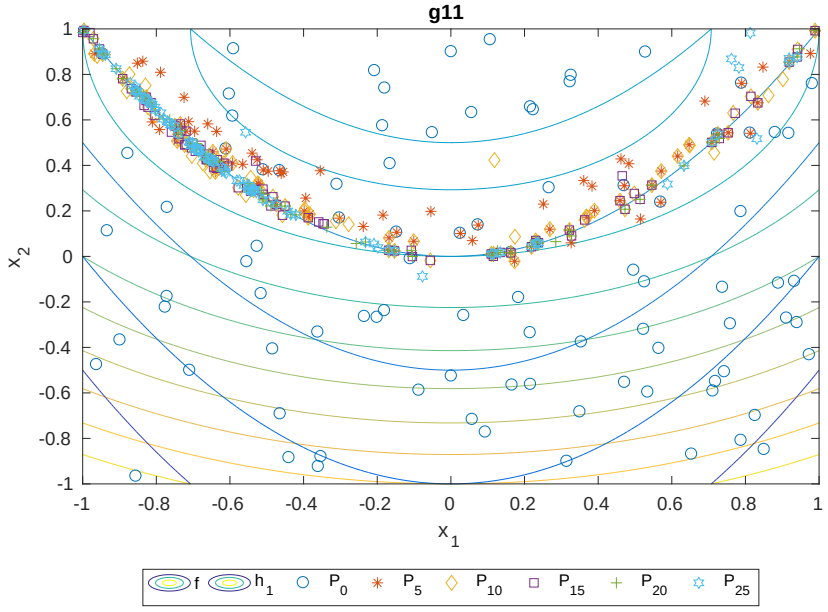
(a) Mecanismo de diversidad



(b) Método de ε-*constrained*



(c) Reglas de factibilidad



(d) Jerarquización estocástica

Figura 5.8: Convergencia de la población (desde la población inicial, P_0, hasta la población de la iteración 25, P_{25}) hacia la región factible para cada CHT en el problema g11.

Tabla 5.4: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 30 ejecuciones en los problemas g01-g11. Valores en **negrita** indican la mejor mediana.

Prob	Medida	SA-DECV$_{SR}$	SA-DECV$_{FR}$	SA-DECV$_{\varepsilon}$	SA-DECV$_{DM}$
g01	B	-15	-15	-15	-15
-15	Md	**-15**	**-15**	**-15**	**-15**
	SD	6.10E-01	7.77E-01	0.00E+00	1.12E+00
g02	B	-0.617282	-0.669849	-0.62949	-0.656218
-0.80362	Md	-0.44457	**-0.565742**	-0.488536	-0.547773
	SD	9.47E-02	8.03E-02	9.28E-02	8.93E-02
g03	B	-0.589924	-0.345011	-0.979946	-0.958986
-1.0005	Md	-0.067428	-0.041334	**-0.598731**	-0.255113
	SD	1.31E-01	7.09E-02	3.61E-01	3.61E-01
g04	B	-30665.53867	-30665.53867	-30665.53867	-30665.53867
-30665.53867	Md	**-30665.53867**	**-30665.53867**	**-30665.53867**	**-30665.53867**
	SD	1.61E-06	2.22E-11	2.22E-11	2.22E-11
g05	B	5126.496714	5126.496714	5126.496714	5126.496714
5126.496714	Md	**5126.496714**	5126.496731	5126.497144	5154.074046
	SD	7.25E-06	6.15E+01	8.22E-04	2.70E+02
g06	B	-6961.813876	-6961.813876	-6961.813876	-6961.813876
-6961.813876	Md	**-6961.813876**	**-6961.813876**	**-6961.813876**	**-6961.813876**
	SD	0.00E+00	0.00E+00	4.07E-03	4.37E-01
g07	B	24.306209	24.314683	24.420911	24.308817
24.306209	Md	**24.320205**	24.520898	28.062367	27.541391
	SD	2.32E+00	1.68E+00	1.05E+01	6.49E+01
g08	B	-0.095825	-0.095825	-0.095825	-0.095825
-0.095825	Md	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
	SD	4.23E-17	4.23E-17	4.23E-17	4.23E-17
g09	B	680.630057	680.630057	680.630057	680.630057
680.630057	Md	**680.630057**	680.630058	680.630058	680.630104
	SD	5.88E-03	1.22E-02	3.76E-01	6.84E+01
g10	B	7049.248021	7049.248251	7049.248327	7049.266398
7049.248021	Md	**7049.250155**	7052.582196	7052.478581	7086.875557
	SD	1.07E+01	1.12E+02	7.66E+02	9.29E+02
g11	B	0.7499	0.7499	0.7499	0.7499
0.7499	Md	**0.7499**	**0.7499**	**0.7499**	**0.7499**
	SD	2.00E-07	9.41E-04	2.99E-06	9.48E-02

SA-DECV$_{SR}$ supera a SA-DECV$_{\varepsilon}$ en los problemas g05, g07, g10, g13, g15, g18, g19, y g23. En los problemas g03 y g14, SA-DECV$_{\varepsilon}$ reportó mejores resultados. En el resto de problemas, ambos tiene el mismo desempeño, de acuerdo al *test* de Wilcoxon.

Tabla 5.5: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 30 ejecuciones en los problemas g12-g24. Valores en **negrita** indican la mejor mediana.

Prob	Medida	SA-DECV$_{SR}$	SA-DECV$_{FR}$	SA-DECV$_{\varepsilon}$	SA-DECV$_{DM}$
g12 -1	B	-1	-1	-1	-1
	Md	-0.994375	-0.994375	**-1**	-0.98425
	SD	6.23E-03	4.50E-03	3.93E-03	2.52E-02
g13 0.053942	B	0.05866	0.517177	0.053943	0.103814
	Md	**0.438803**	0.98007	0.960832	0.561962
	SD	1.68E-01	1.93E-01	9.07E-01	4.09E-01
g14 -47.764888	B	-47.209971	-47.763236	-47.719858	-47.742687
	Md	-44.972172	**-47.623693**	-45.798934	-45.223214
	SD	1.56E+00	1.33E+00	1.18E+00	1.24E+00
g15 961.715022	B	961.715022	961.715022	961.715022	961.715022
	Md	**961.715022**	961.715024	961.715023	961.715024
	SD	7.16E-07	2.42E-01	1.60E-05	4.89E-01
g16 -1.905155	B	-1.905155	-1.905155	-1.905155	-1.905155
	Md	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**
	SD	4.52E-16	4.52E-16	2.77E-05	4.52E-16
g17 8853.539675	B	8853.533913	8864.118199	8853.534178	8858.725207
	Md	8855.948306	8946.987414	**8855.772168**	8927.695637
	SD	1.36E+01	5.04E+01	2.51E+01	1.25E+02
g18 -0.866025	B	-0.866025	-0.866025	-0.866024	-0.866023
	Md	**-0.866023**	-0.865935	-0.789415	-0.525229
	SD	8.66E-02	9.40E-02	1.83E-01	2.03E-01
g19 32.655593	B	32.65595	32.656589	32.659044	32.657091
	Md	**32.658609**	32.768577	32.968456	32.746848
	SD	5.83E-01	7.45E+00	3.63E+00	4.77E+00
g21 193.72451	B	193.724556	193.724566	193.733162	193.724568
	Md	286.422437	220.539701	**194.240044**	297.760006
	SD	6.51E+01	6.37E+01	8.06E+01	1.61E+02
g23 -400.0551	B	-399.815595	-347.259392	-362.935788	-396.072448
	Md	**-290.90299**	-127.601304	-36.015652	-79.663474
	SD	1.42E+02	1.60E+02	1.80E+02	2.16E+02
g24 -5.508013	B	-5.508013	-5.508013	-5.508013	-5.508013
	Md	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**
	SD	2.71E-15	2.71E-15	2.71E-15	1.39E-03

SA-DECV$_{SR}$ y SA-DECV$_{DM}$ tienen el mismo desempeño en 9 problemas (g03-g04, g06, g08, g13- g14, g16, g21, y g24), de acuerdo al *test* de Wilcoxon. SA-DECV$_{DM}$ tuvo un mejor desempeño que SA-DECV$_{SR}$ en el problema g02. En el resto de los problemas es superado por SA-DECV$_{SR}$.

Tabla 5.6: Resultados del test de suma de rangos de Wilcoxon, con SA-DECV$_{SR}$ como algoritmo base comparado contra SA-DECV$_{FR}$, SA-DECV$_{\varepsilon}$ y SA-DECV$_{DM}$. "+" indica que hay diferencia significativa a favor del algoritmo base con respecto al algoritmo comparado. "-" indica que hay diferencia significativa a favor del algoritmo comparado con respecto al algoritmo base. "=" indica que no hay diferencia significativa entre los algoritmos.

	g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11
SA-DECV$_{FR}$	=	-	=	=	+	=	+	=	=	+	=
SA-DECV$_{\varepsilon}$	=	=	-	=	+	=	+	=	=	+	=
SA-DECV$_{DM}$	+	-	=	=	+	=	+	=	+	+	+

	g12	g13	g14	g15	g16	g17	g18	g19	g21	g23	g24
SA-DECV$_{FR}$	=	+	-	+	=	+	=	+	=	+	=
SA-DECV$_{\varepsilon}$	=	+	-	+	=	=	+	+	=	+	=
SA-DECV$_{DM}$	+	=	=	+	=	+	+	+	=	+	=

5.2.2.3. Medidas de desempeño

En las Figuras 5.9, 5.10, 5.11 y 5.12 se presentan los valores de las cuatro variantes de SA-DECV para las medidas de desempeño.

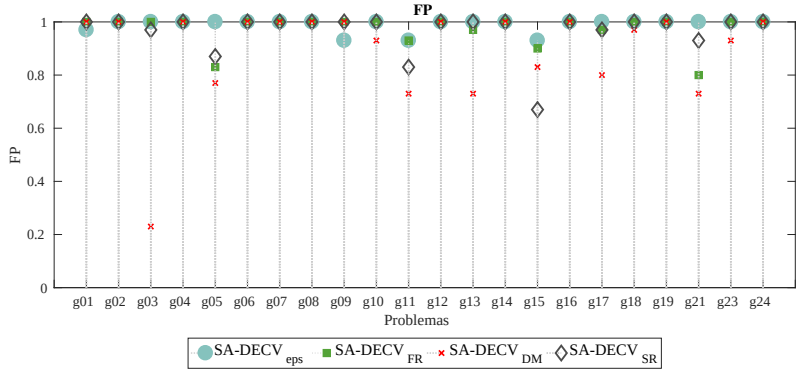


Figura 5.9: Valores para la medida FP por las cuatro variantes de SA-DECV.

De acuerdo a los valores de FP (Fig. 5.9), las cuatro variantes son capaces de reportar al menos una ejecución que alcanza la región factible en todos los problemas. Los valores de FP más bajos son reportados por la variante

SA-DECV$_{DM}$, la cual tiene dificultad en alcanzar la región factible en aquellos problemas que tienen restricciones de igualdad (g03, g05, g11, g13, g17, g21 y g23), principalmente. De acuerdo a sus promedios, la variante más competitiva es SA-DECV$_{\varepsilon}$ con un promedio de 0.98, SA-DECV$_{FR}$ con 0.97, SA-DECV$_{SR}$ con 0.96 y SA-DECV$_{DM}$ con 0.89.

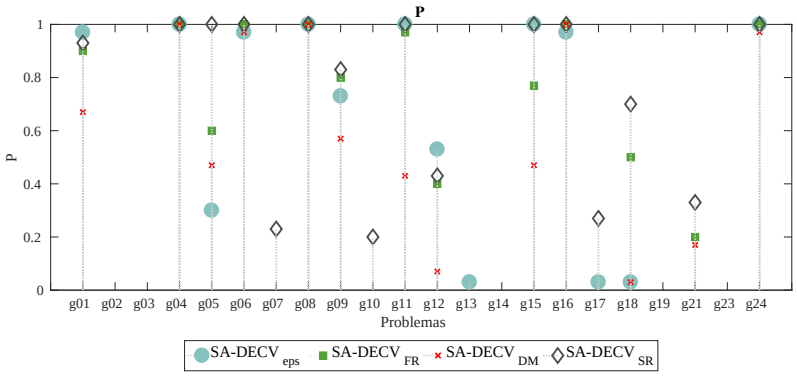


Figura 5.10: Valores para la medida P por las cuatro variantes de SA-DECV.

Aunque todas las variantes alcanzan en al menos una ejecución la región factible, los valores de P (Fig. 5.10) indican que las variantes tienen dificultad para moverse dentro de la región factible porque no son capaces de encontrar una solución factible dentro de la vecindad de la mejor solución factible conocida en varios problemas. Sin embargo, SA-DECV$_{SR}$ reporta los mejores valores en 14 problemas, con excepción de g01 y g12 que son menores a los reportados por SA-DECV$_{\varepsilon}$, la cual reporta al menos una ejecución exitosa en 14 problemas y el resto (SA-DECV$_{DM}$ y SA-DECV$_{FR}$) en 13 problemas.

De acuerdo a los valores de AFES (Fig. 5.11), SA-DECV$_{SR}$ fue la variante más competitiva porque reporta los mejores valores (valores mínimos) en 13 problemas, con un promedio de 2.33E+04 para los 16 problemas, seguido por SA-DECV$_{DM}$ con 2.34E+04 y SA-DECV$_{FR}$ con 2.37E+04 (13 problemas)y SA-DECV$_{\varepsilon}$ con 5.10E+04 (14 problemas).

Los valores de SP se visualizan en la Fig. 5.12, se puede apreciar un comportamiento similar a los valores de AFES (Fig. 5.11), es decir, SA-DECV$_{SR}$ reporta los mejores valores en la mayoría de los problemas que reportan al menos una ejecución exitosa (g01, g04-g11, g15-g18, g21 y g24).

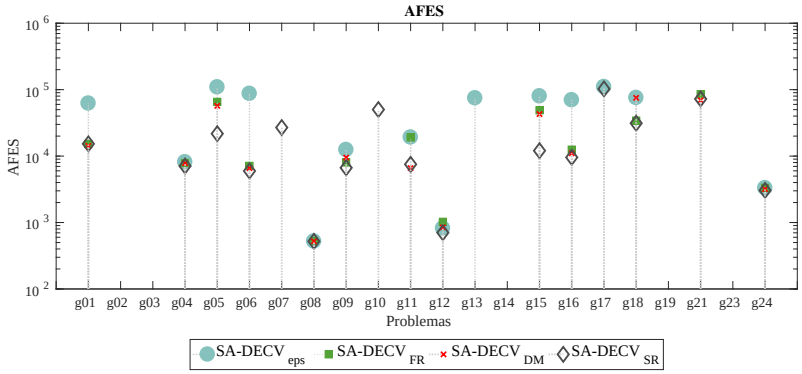


Figura 5.11: Valores para la medida AFES por las cuatro variantes de SA-DECV. El eje y está en escala logarítmica.

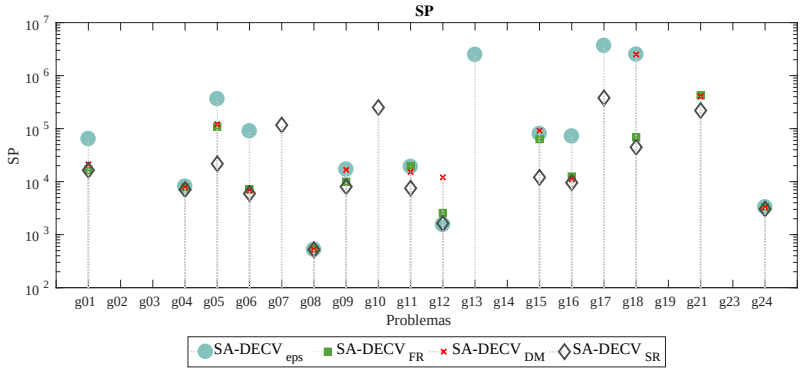


Figura 5.12: Valores para la medida SP por las cuatro variantes de SA-DECV. El eje y está en escala logarítmica.

Los resultados de este experimento llevan a las siguientes conclusiones:

1. El desempeño de SA-DECV fue parecido con las cuatro técnicas para manejo de restricciones debido a que todas las variantes alcanzaron la región factible. Sin embargo, todas presentaron dificultades para moverse dentro de la región factible, pues el número de ejecuciones exitosas decremento en la mayoría de los problemas.

2. SA-DECV$_{SR}$ fue la variante con la mejor combinación entre el menor número de evaluaciones requeridas en el modelo original (valores de AFES) y el porcentaje de ejecuciones exitosas (valores de P). Además, reporta los resultados más competitivos.

3. SA-DECV$_{\varepsilon}$ fue la variante con el mejor desempeño para alcanzar la región factible. Sin embargo, requiere un mayor número de evaluaciones para alcanzar la vecindad de la mejor solución factible conocida en la mayoría de los problemas.

4. SA-DECV$_{FR}$ y SA-DECV$_{DM}$ no fueron capaces de proporcionar un mejor desempeño con respecto a SA-DECV$_{SR}$.

5.2.3. Comparación de SA-DECV$_{SR}$ contra algoritmos del estado del arte en SA-EA

De acuerdo a la subsección anterior, SA-DECV$_{SR}$ fue la variante más competitiva, por lo tanto, se compara contra los algoritmos del estado del arte en SA-EA que resuelven problemas de optimización restringida: ASRES y SBSM-GA. Estos algoritmos aproximan el valor de la función objetivo y la suma de violación de restricciones. Además, usan como modelo subrogado kNN, sin embargo el control de evolución para cada uno es diferente, como se mencionó al inicio de esta sección.

5.2.3.1. Resultados estadísticos

Los valores estadísticos (B:mejor, Md:mediana y SD:desviación estándar) de los resultados finales obtenidos por SA-DECV$_{SR}$, ASRES y SBSM-GA se presentan en las Tablas 5.7 y 5.8. Las mejores medianas en cada problema se muestran en **negrita**. SA-DECV$_{SR}$ se consideró como el algoritmo base para el *test* de suma de rangos de Wilcoxon. Los resultados para dicho *test* se muestran en la tabla 5.9.

Los tres algoritmos reportan el óptimo en g08 y g12. Además, SA-DECV$_{SR}$ fue el único algoritmo en alcanzar la región factible en el problema g21.

SA-DECV$_{SR}$ y ASRES tienen el mismo desempeño en 12 problemas (g01, g04-g06, g08-g09, g11-g12, g15-g16, g18, y g24) de acuerdo al *test* de Wilcoxon. Además, SA-DECV$_{SR}$ supera a ASRES en 3 problemas (g10, g19 y

Tabla 5.7: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 25 ejecuciones en los problemas g01-g11. Valores en **negrita** indican la mejor mediana.

Prob	Medida	SA-DECV$_{SR}$	ASRES	SBSM-GA
g01	B	-15	-15	-11.506871
-15	Md	**-15**	**-15**	-9.422469
	SD	6.17E-01	0.00E+00	1.09E+00
g02	B	-0.625232	-0.739	-0.569676
-0.80362	Md	-0.479565	**-0.7**	-0.425451
	SD	9.34E-02	2.10E-02	3.93E-02
g03	B	-0.3127	-0.998	-0.348838
-1.0005	Md	-0.101432	**-0.995**	-0.020721
	SD	8.81E-02	1.60E-03	7.77E-02
g04	B	-30665.5387	-30665.539	-30640.3025
-30665.5387	Md	**-30665.5387**	**-30665.539**	-30530.0881
	SD	4.80E-04	0.00E+00	4.53E+01
g05	B	5126.49671	5126.497	-
5126.49671	Md	**5126.49671**	**5126.497**	-
	SD	2.82E-07	2.30E-12	-
g06	B	-6961.81388	-6961.814	-6859.05942
-6961.81387	Md	**-6961.81388**	**-6961.814**	-6489.64653
	SD	0.00E+00	1.90E-12	1.91E+02
g07	B	24.306233	24.306	42.441152
24.30621	Md	24.399145	**24.307**	81.575096
	SD	5.86E-01	1.60E-03	2.29E+01
g08	B	-0.095825	-0.096	-0.095825
-0.095825	Md	**-0.095825**	**-0.096**	**-0.095825**
	SD	1.33E-02	5.10E-17	4.25E-17
g09	B	680.630057	680.63	683.175546
680.630057	Md	**680.630057**	**680.63**	700.509571
	SD	8.93E-03	2.20E-06	1.78E+01
g10	B	7049.24804	7049.408	9052.29583
7049.24802	Md	**7049.28625**	7053.856	10363.0334
	SD	8.35E+00	1.90E+01	1.04E+03
g11	B	0.7499	0.75	0.749902
0.7499	Md	**0.7499**	**0.75**	0.74999
	SD	2.24E-07	1.10E-16	9.44E-04

Tabla 5.8: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 25 ejecuciones en los problemas g12-g24. Valores en **negrita** indican la mejor mediana.

Prob	Medida	SA-DECV$_{SR}$	ASRES	SBSM-GA
g12 -1	B	-1	-1	-1
	Md	**-1**	**-1**	**-1**
	SD	3.53E-03	0.00E+00	0.00E+00
g13 0.053942	B	0.337683	0.054	-
	Md	0.494733	**0.054**	-
	SD	1.27E-01	1.40E-01	-
g14 -47.76488	B	-47.483422	-47.765	-
	Md	-45.008422	**-47.763**	-
	SD	1.28E+00	3.30E-03	-
g15 961.71502	B	961.715022	961.715	-
	Md	**961.715022**	**961.715**	-
	SD	3.42E-07	4.60E-13	-
g16 -1.905155	B	-1.905155	-1.905	-1.882513
	Md	**-1.905155**	**-1.905**	-1.793759
	SD	4.53E-16	1.10E-15	5.18E-02
g17 8853.53967	B	8857.65497	8853.54	-
	Md	8863.10444	**8853.54**	-
	SD	1.53E+01	8.50E-01	-
g18 -0.866025	B	-0.866025	-0.866	-0.382358
	Md	**-0.86602**	**-0.866**	-0.359002
	SD	9.75E-02	5.30E-02	5.34E-02
g19 32.65559	B	32.655678	32.665	184.13232
	Md	**32.668844**	32.79	248.897427
	SD	7.07E+00	1.60E-01	4.43E+01
g21 193.72451	B	193.726764	-	-
	Md	**324.731834**	-	-
	SD	6.69E+01	-	-
g23 -400.0551	B	-365.298536	-348.816	-
	Md	**-196.173551**	-106.585	-
	SD	1.30E+02	1.40E+02	-
g24 -5.508013	B	-5.508013	-5.508	-5.507847
	Md	**-5.508013**	**-5.508**	-5.507109
	SD	1.81E-15	0.00E+00	2.40E-03

Tabla 5.9: Resultados del test de suma de rangos de Wilcoxon, con SA-DECV$_{SR}$ como algoritmo base comparado contra ASRES y SMSB-GA. "+" indica que hay diferencia significativa a favor del algoritmo base con respecto al algoritmo comparado. "-" indica que hay diferencia significativa a favor del algoritmo comparado con respecto al algoritmo base. "=" indica que no hay diferencia significativa entre los algoritmos. N/A indica que el resultado no esta disponible.

	g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11
ASRES	=	-	-	=	=	=	-	=	=	+	=
SMSB-GA	+	+	+	+	N/A	+	+	=	+	+	+
	g12	g13	g14	g15	g16	g17	g18	g19	g21	g23	g24
ASRES	=	-	-	=	=	-	=	+	N/A	+	=
SMSB-GA	-	N/A	N/A	N/A	+	N/A	+	+	N/A	N/A	+

g23). Por otro lado, ASRES reporta un mejor desempeño que SA-DECV$_{SR}$ en 6 problemas (g02-g03, g07, g13-g14 y g17).

De acuerdo al *test* de Wilcoxon, SA-DECV$_{SR}$ supera a SBSM-GA en 12 problemas (g01-g04, g06-g07, g09-g11, g16, g18-g19, y g24). En el problema g08, los tres algoritmos tienen el mismo desempeño. Con respecto al problema g12, SBSM-GA reporta un mejor desempeño que SA-DECV$_{SR}$. En el resto de los problemas, SBSM-GA no es capaz de alcanzar la región factible.

5.2.3.2. Medidas de desempeño

En las Figuras 5.13, 5.14, 5.15 y 5.16 se presentan los valores de SA-DECV$_{SR}$, ASRES y SBSM-GA para las medidas de desempeño.

Los valores de FP (Fig. 5.13) indican que SA-DECV$_{SR}$, ASRES y SBSM-GA encuentran la región factible en 13 problemas (g01-g04, g06-g12, g16, g18-g19 y g24), la mayoría de estos problemas tienen restricciones de desigualdad, con excepción del problema g11. El promedio de cada algoritmo en estos problemas fue: SA-DECV$_{SR}$ con 0.98, ASRES con 1 y SBSM-GA con 0.96. Aunque SA-DECV$_{SR}$ no reporta el promedio más alto, presenta un valor de FP mayor que ASRES en los problemas g21 y g23. Además, ambas propuestas reportan el mismo valor para los problemas g13 y g14. Todos los problemas anteriores (g13, g14, g21 y g23) incluyen restricciones de igualdad. En el resto de los problemas: ASRES fue mejor que SA-DECV$_{SR}$.

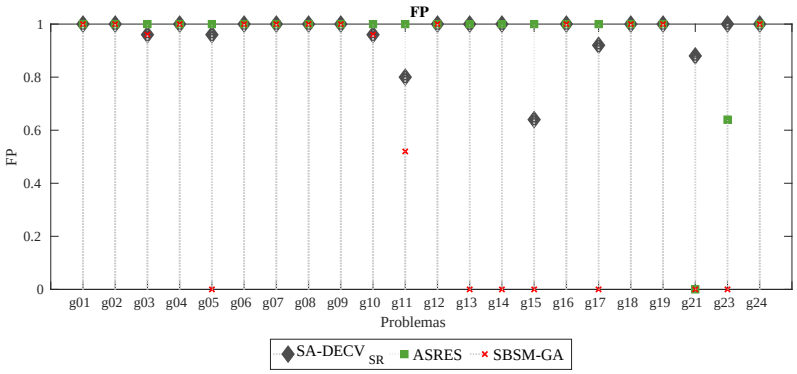


Figura 5.13: Valores para la medida FP para SA-DECV$_{SR}$, ASRES y SBSM-GA.

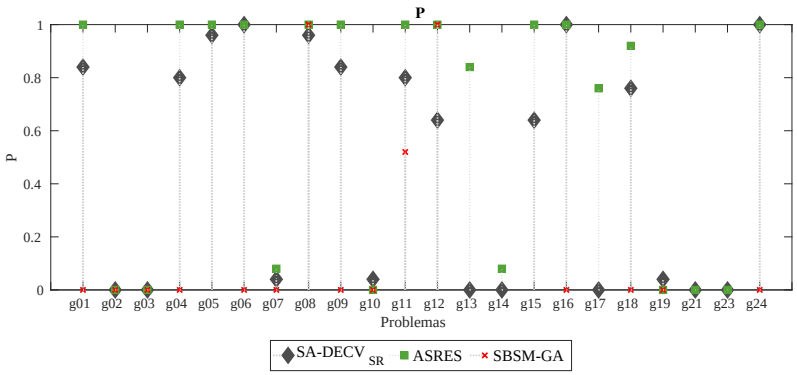


Figura 5.14: Valores para la medida P para SA-DECV$_{SR}$, ASRES y SBSM-GA.

Los valores de P (Fig. 5.14) sugieren que SA-DECV$_{SR}$, ASRES y SBSM-GA no son capaces de moverse dentro de la región factible, pues solo en 3 problemas (g08, g11 y g12), las 3 propuestas encuentran una solución dentro de la vecindad del óptimo. En los problemas g10 y g19, SA-DECV$_{SR}$ fue el único algoritmo que reporta una ejecución exitosa. Por otro lado, SA-DECV$_{SR}$ y ASRES tienen el mismo valor de P en 3 problemas (g06, g16 y g24). En el resto de los problemas, ASRES tienen un valor más alto de P

con respecto a SA-DECV$_{SR}$.

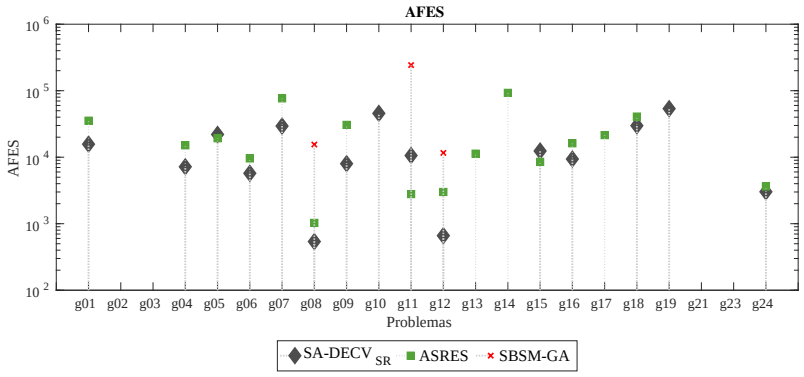


Figura 5.15: Valores para la medida AFES para SA-DECV$_{SR}$, ASRES y SBSM-GA. El eje y está en escala logarítmica.

De acuerdo a los valores de AFES (Fig. 5.15), SA-DECV$_{SR}$ fue la propuesta más competitiva en 12 problemas (g01, g04, g06-g10, g12, g16, g18, g19 y g24) al tener los mejores valores con respecto a los otros algoritmos. En el resto de los problemas (g05, g11, g13-g15 y g17), ASRES requiere de un menor número de evaluaciones de la función del problema que SA-DECV$_{SR}$ y SBSM-GA (en el problema 11). Con respecto a los valores de SP (Fig. 5.16), el comportamiento fue similar al correspondiente de los valores de AFES. Por lo tanto, SA-DECV$_{SR}$ tiene los mejores valores de SP en los mismos problemas.

Los resultados de este experimento lleva a las siguientes conclusiones:

1. Aunque ASRES reporta resultados más competitivos en varios problemas, también requiere un mayor número de evaluaciones que SA-DECV$_{SR}$ para alcanzarlos.

2. SBSM-GA presenta dificultades para encontrar una solución factible, principalmente en aquellos con restricciones de igualdad. Sin embargo, puede alcanzar la región factible en aquellos problemas con restricciones de desigualdad, pero no reporta una solución en la vecindad del óptimo.

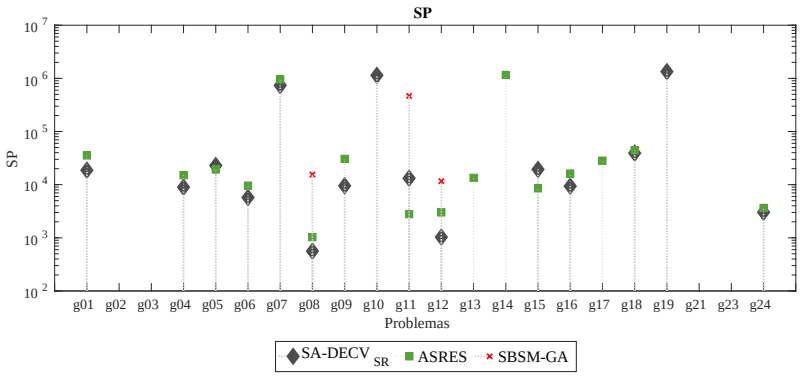


Figura 5.16: Valores para la medida SP para SA-DECV$_{SR}$, ASRES y SBSM-GA. El eje y está en escala logarítmica.

5.2.3.3. Discusión de resultados

Los resultados de los experimentos previos sugieren que el algoritmo SA-DECV fue capaz de encontrar la región factible de los problemas resueltos. Sin embargo, se presentan dificultades para reportar una solución dentro de la vecindad del óptimo. Este comportamiento se presentó principalmente en aquellos problemas con restricciones de igualdad o restricciones o funciones objetivo no lineales.

La combinación de un modelo de regresión de kNN y la jerarquización estocástica promovieron una mejor exploración durante el proceso de búsqueda, es decir, la jerarquización estocástica permite que soluciones no factibles con un buen valor de la función objetivo permanezca en la población, debido a que su mecanismo de selección es *lazy* y eso previene una convergencia prematura. Por lo tanto, SA-DECV$_{SR}$ reportó el mayor número de problemas con ejecuciones exitosas. Por otro lado, aquellas técnicas para manejo de restricciones que eliminan soluciones no factibles en la población (reglas de factibilidad, método de ε-*constrained* y mecanismo de diversidad) y el modelo de regresión de kNN afectan la calidad de los resultados finales. De acuerdo a los resultados de las medidas de desempeño, la variante más robusta fue aquella que empleó jerarquización estocástica como técnica para manejo de restricciones, porque ésta requiere el menor número de evaluaciones para alcanzar una solución dentro de la vecindad del óptimo. Además, reporta los mejores valores de ejecuciones exitosas.

El manejador de modelo más adecuado para resolver problemas con restricciones es aquel que re-evalúa la mejor solución aproximada o evalúa aquellas soluciones cercanas a la mejor solución, ya que evitan que el proceso de búsqueda se dirija hacia un falso óptimo.

5.3. Caso de estudio: problemas de optimización de ingeniería

En esta sección se presenta una comparación de la mejor variante de SA-DECV y otros algoritmos del estado del arte en problemas de optimización de ingeniería.

Se resolvieron cuatro problemas de optimización de ingeniería: la viga soldada (WB, por sus siglas en inglés), la vasija de presión (PV, por sus siglas en inglés), el resorte de tensión/compresión (TCS, por sus siglas en inglés) y el freno de embrague (CB, por sus siglas en inglés). Además, se consideró un problema de mecatrónica (MCP, por sus siglas en inglés), la optimización de un mecanismo de cuatro barras diseñado para seguir una trayectoria no alineada con una sincronización prescrita. Los características de los problemas se muestran en la Tabla 5.10 y los detalles se encuentran en el Apéndice B.

Tabla 5.10: Detalles de los problemas de ingeniería. n es el número de variables de decisión, $\rho = |F|/|S|$ es el relación estimada entre la región factible y el espacio de búsqueda, LI es el número de restricciones de desigualdad, NI es el número de restricciones de desigualdad no lineales, LE es el número de restricciones de igualdad lineales y NE es el número de restricciones de igualdad no lineales.

Prob.	n	Tipo de función	ρ	LI	NI	LE	NE
WB	4	cuadrática	2.6859 %	6	1	0	0
PV	4	cuadrática	39.6762 %	3	1	0	0
TCS	3	cuadrática	0.7537 %	1	3	0	0
CB	5	cuadrática	71.7837 %	8	0	0	0
MCP	6	cuadrática	2.0888 %	4	0	0	0

SA-DECV$_{SR}$ se comparó de manera directa con IAPSO [22] y C-LSHADE [86]. IAPSO es una variante de PSO, la cual se probó en varios

Tabla 5.11: Resultados estadísticos de 25 ejecuciones independientes en problemas de optimización de ingeniería.

Prob.	Óptimo	Algoritmo	B	M	W	SD	FEs
WB	1.7248523	IAPSO	1.7248523	1.7248528	1.7248624	2.02E-06	12500
		SA-DECV$_{SR}$	1.72486	1.72496	1.72595	2.39E-04	6971
PV	6059.71433	IAPSO	6059.7143	6068.7539	6090.5314	1.40E+01	7500
		SA-DECV$_{SR}$	6059.7	6296.0336	6820.44	2.65E+02	6139
TCS	0.01266523	IAPSO	0.01266523	0.013676527	0.01782864	1.57E-03	2000
		SA-DECV$_{SR}$	0.0127127	0.013047148	0.0143943	4.00E-04	1617
DC	0.313656	IAPSO	0.313656	0.313656	0.313656	1.13E-16	400
		SA-DECV$_{SR}$	0.313657	0.31491156	0.329339	3.87E-03	774
MCP*	2.6280E-03	C-LSHADE	2.6280E-03	2.6280E-03	2.6281E-03	8.55E-09	15000
		SA-DECV$_{SR}$	2.6281E-03	2.6381E-03	2.7439E-03	2.56E-05	13280

*30 ejecuciones independientes se llevaron a cabo.

problemas de optimización de ingeniería. Además, se comparó con varios algoritmos del estado del arte, siendo IAPSO el algoritmo que requiere un menor número de evaluaciones para encontrar una solución competitiva. Por otro lado, C-LSHADE es un algoritmo altamente competitivo para resolver problemas de diseño en mecatrónica.

Los valores para los parámetros de SA-DECV$_{SR}$ son los empleados en los experimentos anteriores. El número máximo de evaluaciones (Max_FEs) en el modelo original fue diferente para cada problema, para ello se consideró un valor cercano al número de evaluaciones reportados por [22]: WB 30000, PV 20000, TCS 5000, CB 2000 y MCP 60000. Se realizaron 25 ejecuciones independientes por cada problema de ingeniería (WB, PV, TCS y CB), mientras que para el problema de mecatrónica se efectuaron 30 ejecuciones independientes.

La Tabla 5.11 muestra el óptimo, los resultados estadísticos (B:mejor, M:promedio, W:peor y SD:desviación estándar) y el promedio del número de evaluaciones empleado por cada algoritmo (FEs) para cada problema.

SA-DECV$_{SR}$ encontró resultados competitivos al igual que IAPSO. Además, SA-DECV$_{SR}$ requirió un menor número de evaluaciones que IAPSO en la mayoría de los problemas. Con respecto al problema del mecanismo de cuatro barras (MCP), C-LSHADE y SA-DECV$_{SR}$ encontraron soluciones dentro de la vecindad del óptimo. Sin embargo, SA-DECV$_{SR}$ empleó un menor número de evaluaciones que las usadas por C-LSHADE.

SA-DECV$_{SR}$ fue capaz de encontrar una solución factible en cada uno de los problemas de ingeniería. Además en los problemas de la viga soldada (WB), el freno de embrague (CB) y el problema de mecatrónica (MCP), las soluciones reportadas están dentro de la vecindad de la mejor solución factible conocida y el número de evaluaciones fue menor con respecto al de los algoritmos con los que se comparó.

Conclusión general

Un modelo subrogado basado en similitud, tal como kNN, requiere que se mantenga un conjunto de soluciones no factibles pero con un buen valor de función objetivo para resolver problemas de optimización con restricciones, ya que mantiene diversidad dentro del conjunto de entrenamiento y esto a su vez evita que la búsqueda converja hacia un falso óptimo.

5.4. Resumen del capítulo

En el presente capítulo se explicó la propuesta SA-DECV, con el objetivo de estudiar los efectos que sufre el proceso de búsqueda de DECV con diferentes técnicas para manejo de restricciones, aproximando el valor de la función objetivo y la suma de violación de restricciones, para resolver problemas de optimización restringida. El enfoque se prueba en un conjunto de problemas conocido y en problemas de optimización de ingeniería, y se compara con algoritmos del estado del arte. En ambos experimentos, los resultados obtenidos son competitivos y el número de evaluaciones es reducido. Debido a que el control de evolución empleado en SA-DECV se propone en el presente trabajo, en el siguiente capítulo éste se integra en novedosos algoritmos del estado del arte en optimización restringida.

Capítulo 6

Estudio del control de evolución basado en factibilidad

En el capítulo anterior se presentó la propuesta de SA-DECV, la cual emplea kNN como modelo subrogado y un control de evolución basado en factibilidad. En el presente capítulo se realiza la comparación de la mejor variante del capítulo anterior (SA-DECV$_{SR}$) contra algoritmos del estado del arte en optimización con restricciones, ahora asistidos por subrogados.

6.1. Diseño experimental

Existe una gran variedad de algoritmos evolutivos que resuelven problemas de optimización con restricciones, de los cuales se seleccionaron dos algoritmos novedosos y recientes: CMODE [81] y NGLUPSO [42]. El primero es un algoritmo evolutivo de estado estacionario, el cual emplea DE y un mecanismo de reemplazo basado en optimización multiobjetivo. Por otro lado, NGLUPSO emplea un gradiente numérico (NG) para encontrar la región factible y una estructura dirigida de grupo (GDS), que consiste en dos grupos, el primero, denominado "toda información" comparte información con toda la población, mientras que el resto de soluciones pertenecen al grupo "información a medias" y sólo comparten información entre ellas. Lo anterior se integra en la variante de PSO Lu-Chen [44]. Dado que CMODE y NGLUPSO evalúan todas las soluciones en la función original, se integran tanto el control de evolución basado en factibilidad (el objeto de estudio del

presente capítulo) como kNN como modelo subrogado para aproximar tanto el valor de la función objetivo como la suma de violación de restricciones.

Los parámetros para SA-DECV$_{SR}$ fueron: NP=90, CR=1, F=0.9, tolerancia para las restricciones de igualdad ϵ=1E-4, tamaño del conjunto de entrenamiento $2\times NP$, número de elementos para calcular el error n_{err}=5 %NP, $k = n$ y P_f=0.45.

Con respecto a CMODE y NGLUPSO, los parámetros fueron los que se sugieren en [81] y [42], respectivamente, con excepción de NP y λ para CMODE, este ajuste se realizó con base en una serie de experimentos llevados a cabo. Así los parámetros para CMODE fueron: NP=180, F se seleccionó aleatoriamente entre [0.5, 0.6], CR se seleccionó aleatoriamente entre [0.9, 0.95], κ=22 y el número de soluciones que evolucionan por generación λ=90. Mientras que para NGLUPSO fueron: número de partículas 60, ω=1, $c_1 = c_2 = 1$, $r1$ and $r2$ son números reales generados de manera aleatoria entre [0, 1]. Para el método GDS, 10 partículas se usaron en el grupo "toda-información" y el resto en el grupo de "información a medias". Los parámetros para el modelo subrogado integrado en CMODE y NGLUPSO son los que se sugieren en [51].

El máximo número de evaluaciones en el modelo original fue Max_FES=240000, además se realizaron 30 ejecuciones independientes por problema.

6.1.1. Resultados estadísticos

Los valores estadísticos (B:mejor, Md:mediana y SD:desviación estándar) de los resultados finales obtenidos por SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO se muestran en las Tablas 6.1 y 6.2. Las mejores medianas en cada problema se muestran en **negrita**. SA-DECV$_{SR}$ se consideró como el algoritmo base para el *test* de Wilcoxon, usando las medianas de cada problema. El resultado de este *test* se presenta en la Tabla 6.3.

SA-DECV$_{SR}$ supera a SA-CMODE en once problemas (g01, g04-g06, g08-g11, g15-g16 y g19). Por otro lado, SA-DECV$_{SR}$ y SA-CMODE tienen el mismo desempeño en dos problemas (g03 y g24), de acuerdo al *test* de Wilcoxon. En el resto de los problemas, SA-CMODE reporta un mejor desempeño que SA-DECV$_{SR}$.

SA-DECV$_{SR}$ y SA-NGLUPSO tienen el mismo desempeño en los problemas g08, g11, g21 y g24. Además, SA-DECV$_{SR}$ reporta un mejor desempeño que SA-NGLUPSO en dieciséis problemas (g01-g02, g04-g07, g09-g10, g13-g19, y g23). En los problemas g03 y g12, SA-NGLUPSO supera a SA-DECV$_{SR}$.

Tabla 6.1: Resultados estadísticos (B:mejor, Md:mediana y SD:desviación estándar) de las 30 ejecuciones obtenidos por SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO en los problemas g01-g11. Los valores en **negrita** indican el mejor valor de la mediana.

Prob	Mea	Óptimo	SA-DECV$_{SR}$	SA-CMODE	SA-NGLUPSO
g01	B	-15	-15	-14.691035	-15
	Md		**-15**	-14.021593	-12.571476
	SD		6.10E-01	3.21E-01	1.58E+00
g02	B	-0.80362	-0.617282	-0.779503	-0.647076
	Md		-0.44457	**-0.7058675**	-0.38946
	SD		9.47E-02	3.60E-02	9.83E-02
g03	B	-1.0005	-0.589924	-0.521323	-1.00049
	Md		-0.070668	-0.317081	**-0.415783**
	SD		1.29E-01	2.63E-01	2.94E-01
g04	B	-30665.53867	-30665.53867	-30665.53863	-30665.53867
	Md		**-30665.53867**	-30665.53841	-30665.52713
	SD		1.61E-06	2.58E-04	7.61E+00
g05	B	5126.49671	5126.49671	5126.49673	5126.49674
	Md		**5126.49671**	5126.49679	5154.18773
	SD		6.75E-06	4.06E-05	9.28E+01
g06	B	-6961.81387	-6961.81387	-6961.78766	-6961.81387
	Md		**-6961.81387**	-6961.33051	**-6961.81387**
	SD		0.00E+00	8.84E-01	1.13E-02
g07	B	24.30621	24.30621	24.30964	27.42833
	Md		24.32021	**24.3124**	54.70839
	SD		2.32E+00	2.06E-03	5.96E+01
g08	B	-0.095825	-0.095825	-0.095825	-0.095825
	Md		**-0.095825**	**-0.095825**	**-0.095825**
	SD		4.23E-17	3.23E-05	4.23E-17
g09	B	680.630057	680.630057	680.630058	681.181933
	Md		**680.630057**	680.630059	701.137117
	SD		5.88E-03	1.69E-06	2.26E+01
g10	B	7049.24802	7049.248021	7079.202607	7061.397178
	Md		**7049.250155**	8585.900247	8610.05244
	SD		1.07E+01	2.18E+03	3.08E+03
g11	B	0.7499	0.7499	0.7499	0.7499
	Md		**0.7499**	**0.7499**	**0.7499**
	SD		1.83E-07	3.10E-06	1.13E-16

6.1.1.1. Medidas de desempeño

En las Figuras 6.1, 6.2, 6.3 y 6.4 se presentan los valores de las medidas de desempeño para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO.

Tabla 6.2: Resultados estadísticos (B:mejor, Md:mediana y SD:desviación estándar) de las 30 ejecuciones obtenidos por SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO en los problemas g12-g24. Los valores en **negrita** indican el mejor valor de la mediana.

Prob	Mea	Óptimo	SA-DECV$_{SR}$	SA-CMODE	SA-NGLUPSO
	B		-1	-1	-1
g12	Md	-1	-0.994375	**-1**	**-1**
	SD		6.23E-03	0.00E+00	0.00E+00
	B		0.05866	0.053942	0.12484
g13	Md	0.053942	0.4388	**0.053942**	0.72759
	SD		1.68E-01	3.05E-07	7.83E-01
	B		-47.20997	-47.76393	-46.02693
g14	Md	-47.76488	-44.97217	**-47.76228**	-38.82596
	SD		1.56E+00	1.33E-03	3.98E+00
	B		961.71502	961.71502	961.715026
g15	Md	961.71502	**961.71502**	961.71503	962.91178
	SD		8.55E-07	1.13E-06	1.69E+00
	B		-1.905155	-1.905147	-1.905155
g16	Md	-1.905155	**-1.905155**	-1.905093	-1.903975
	SD		4.52E-16	1.76E-03	2.42E-03
	B		8853.53391	8853.53415	8884.02774
g17	Md	8853.53967	8855.63366	**8853.53472**	8960.53094
	SD		1.34E+01	2.30E-04	1.62E+02
	B		-0.866025	-	-0.865511
g18	Md	-0.866025	**-0.866023**	-	-0.460797
	SD		8.66E-02	-	1.90E-01
	B		32.65595	32.76196	32.67879
g19	Md	32.65559	**32.65861**	32.85981	76.76793
	SD		5.83E-01	6.67E-02	3.98E+01
	B		193.72456	193.73407	193.73625
g21	Md	193.72451	229.26691	**193.74352**	396.35818
	SD		6.51E+01	3.72E+01	2.70E+02
	B		-399.81559	-	-23.25744
g23	Md	-400.0551	**-290.90299**	-	2.638291
	SD		1.42E+02	-	1.53E+02
	B		-5.508013	-5.508013	-5.508013
g24	Md	-5.508013	**-5.508013**	**-5.508013**	**-5.508013**
	SD		2.71E-15	2.71E-15	1.83E-07

Los valores de FP (Fig. 6.1) muestran que SA-DECV$_{SR}$ y SA-CMODE son capaces de encontrar siempre una solución factible en 12 problemas (g01-g02, g04, g06-g09, g12-g14, g19 y g24). En el resto de los problemas SA-

Tabla 6.3: Resultados del test de suma de rangos de Wilcoxon, con SA-DECV$_{SR}$ como algoritmo base comparado contra SA-CMODE y SA-NGLUPSO. "+" indica que hay diferencia significativa a favor del algoritmo base con respecto al algoritmo comparado. "-" indica que hay diferencia significativa a favor del algoritmo comparado con respecto al algoritmo base. "=" indica que no hay diferencia significativa entre los algoritmos. "NaN" indica que no se pudo aplicar el *test*.

	g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11
SA-CMODE	+	-	=	+	+	+	-	+	+	+	+
SA-NGLUPSO	+	+	-	+	+	+	+	=	+	+	=

	g12	g13	g14	g15	g16	g17	g18	g19	g21	g23	g24
SA-CMODE	-	-	-	+	+	-	NaN	+	-	NaN	=
SA-NGLUPSO	-	+	+	+	+	+	+	+	=	+	=

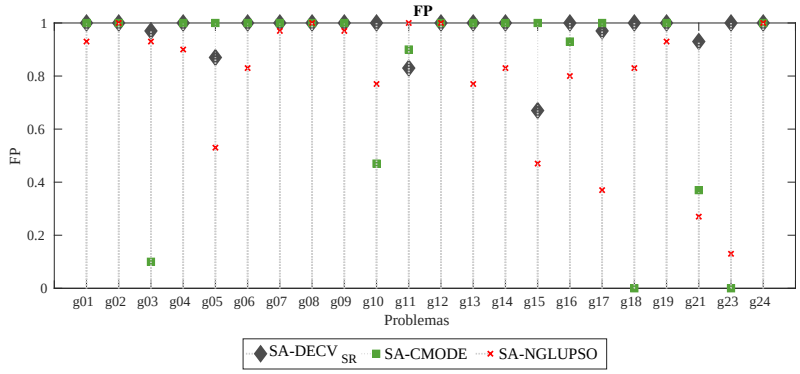


Figura 6.1: Valores para la medida FP para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO.

DECV$_{SR}$ es capaz de encontrar al menos una solución factible, mientras que SA-CMODE no encuentra soluciones factibles en los problemas g18 y g23. Por otro lado, SA-NGLUPSO encuentra una solución factible en todos los problemas de prueba, aunque no en todas las ejecuciones alcanza la región factible. El promedio de cada algoritmo fue: SA-DECV$_{SR}$ con 0.97, SA-CMODE con 0.89 y SA-GNLUPSO con 0.79.

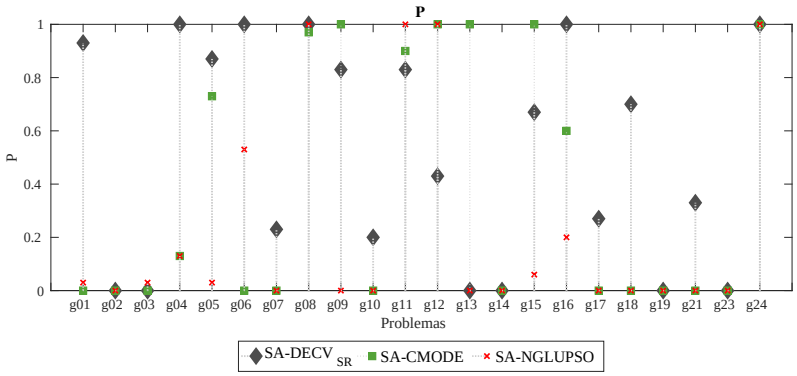


Figura 6.2: Valores para la medida P para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO.

Aunque los algoritmos alcanzaron la región factible en la mayoría de los problemas, los valores de P (Fig. 6.2) sugieren que SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO tuvieron problemas para moverse dentro de la región factible, debido a que sólo en ocho problemas (g04-g05, g08, g11-g12, g15-g16 y g24) todos los algoritmos encontraron una solución cercana a la vecindad del óptimo, con un promedio de 0.85 para SA-DECV$_{SR}$, 0.79 para SA-CMODE y 0.55 para SA-NGLUPSO. En los problemas g01, g06-g07, g10, g17-g18 y g21, SA-DECV$_{SR}$ reportó un valor alto para P en comparación de los otros algoritmos. Por otro lado, SA-CMODE tuvo mejores valores de P en los problemas g09 y g13 (en el último problema, fue el único algoritmo que tiene ejecuciones exitosas).

Con respecto a los valores de AFES (Fig. 6.3), SA-DECV$_{SR}$ fue el algoritmo más competitivo porque reportó los valores más bajos de AFES en quince problemas (g01, g04-g12, g15-g18 and g21). En los problemas g03 y g24, SA-NGLUPSO requirió un menor número de evaluaciones. Con respecto al problema g13, SA-CMODE fue el único algoritmo que reporta un valor para AFES. Los valores de SP (Fig. 6.4 presentan un comportamiento similar al de los valores de AFES. Por lo tanto, los tres algoritmos comparados tienen los mismos mejores valores para SP en los mismos problemas de la medida AFES.

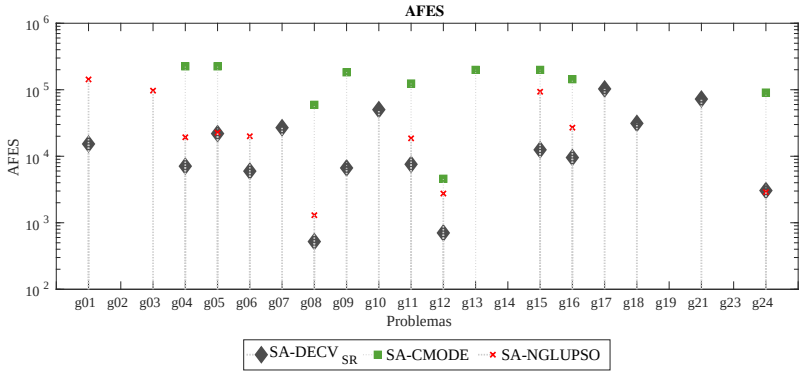


Figura 6.3: Valores para la medida AFES para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. El eje y está en escala logarítmica.

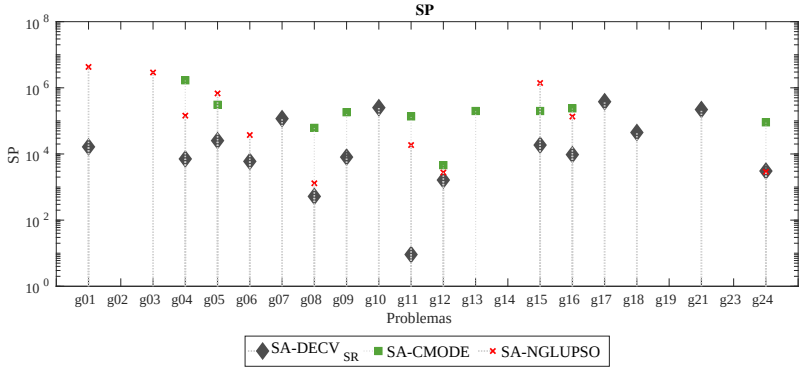


Figura 6.4: Valores para la medida SP para SA-DECV$_{SR}$, SA-CMODE y SA-NGLUPSO. El eje y está en escala logarítmica.

Las conclusiones de este experimento son:

1. En el algoritmo evolutivo de estado estacionario (SA-CMODE), la evolución parcial de la población y el uso de un modelo subrogado guían la búsqueda hacia falsos óptimos en varios problemas.

2. El uso en la etapa inicial del modelo aproximado decrementa el desempeño de SA-NGLUPSO.

Como conclusión general de este capítulo se tiene que el control de evolución basado en factibilidad es una buena opción en aquellos algoritmos que trabajan con toda la población en cada iteración, además que exploran el espacio de búsqueda antes de usar el modelo subrogado, lo anterior con el objetivo de garantizar que el conocimiento almacenado en el conjunto de entrenamiento sobre el espacio de búsqueda es suficiente para guiar la búsqueda hacia una solución competitiva.

6.2. Resumen del capítulo

En el presente capítulo se estudió el uso del control de evolución basado en factibilidad, el cual se integró en dos algoritmos del estado del arte en optimización restringida, CMODE y NGLUPSO, los cuales tienen diferencias con respecto a DECV que se reflejan en cómo realizan el proceso de búsqueda. De acuerdo a los resultados, el control de evolución es una buena opción en aquellos algoritmos donde hay una exploración al inicio del proceso de búsqueda, además, donde toda la población interviene en cada iteración.

Capítulo 7

SA-DECV con búsqueda acotada

En el capítulo 5 se presentó el estudio del modelo subrogado kNN, para aproximar tanto el valor de la función objetivo como la suma de restricciones, el cual reporta resultados competitivos y ahorra evaluaciones en el modelo original en la mayoría de los problemas. A pesar de que el número de evaluaciones se ha reducido, en problemas definidos mediante modelos costosos, este ahorro no puede ser suficiente.

En el presente capítulo se adopta el enfoque de SAMO [6], el cual se diseñó para resolver problemas multiobjetivo y emplea modelos subrogados locales por cada función del problema. Antes de explicar las mejoras a la nueva versión de SA-DECV, la cual se denomina SA-DECV$_L$, se proporciona una descripción general de SAMO. Posteriormente se presenta una comparación de dos variantes de SA-DECV$_L$.

7.1. Propuesta de SA-DECV con modelos subrogados locales

SAMO emplea una búsqueda global y una búsqueda acotada. Durante la búsqueda global se generan los hijos por medio de los operadores de variación de un algoritmo evolutivo, su evaluación (aproximación) se lleva a cabo mediante modelos subrogados locales, los cuales se construyen con las $4 \times n$ soluciones cercanas a la que se va aproximar. Cuando ya se han evaluado todos los hijos, se selecciona el mejor, para definir una región en donde se llevará a cabo un nuevo proceso de búsqueda, que puede ser para explorar

o explotar la región, ello depende del tamaño de la región (ver Fig. 7.1). La búsqueda en esta región se lleva a cabo mediante un algoritmo evolutivo (ver Fig. 7.2). La mejor solución encontrada se evalúa en el modelo original, sustituye a la peor solución de los padres y se agrega al conjunto de entrenamiento. El proceso anterior se realiza hasta cumplir la condición de paro, la cual está definida por un número de evaluaciones máximo. SAMO emplea el algoritmo genético tanto en la búsqueda global como en la acotada.

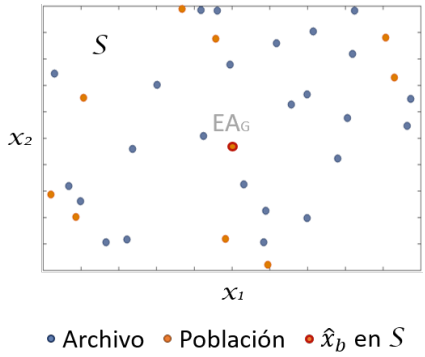


Figura 7.1: Búsqueda global ($\mathcal{S}$), donde todas las soluciones son aproximadas mediante modelos subrogados locales, definidos por las $4{\times}n$ soluciones cercanas a la que se va a aproximar. La solución resaltada por una línea roja gruesa es la mejor aproximada de la población.

7.2. Algoritmo

En la figura 7.3 se muestra el pseudocódigo de SA-DECV$_L$. En el resto de la sección se describen cada uno de los elementos de la propuesta.

7.2.1. Población inicial

La población inicial se genera mediante una técnica de muestreo llamada Hipercubo Latino (LHS) (ver Sec. 3.2). Dicha técnica se emplea para que las soluciones iniciales estén distribuidas de manera uniforme dentro del espacio de búsqueda.

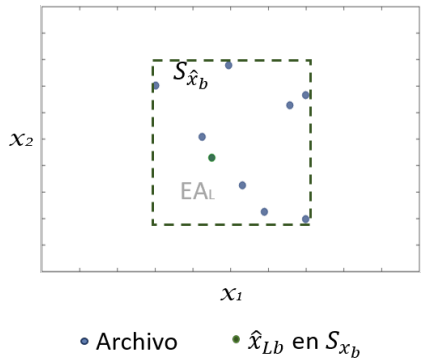


Figura 7.2: Búsqueda acotada aproximada ($\mathcal{S}_{\hat{x}_b}$, recuadro morado punteado). Los modelos para cada función se construyen con $4\times n$ soluciones definidas en la búsqueda global. La solución verde es la mejor aproximada de esta búsqueda, la cual se evaluará en el modelo original y agregará al conjunto de entrenamiento.

1: $P_0 = \text{LHS}(NP)$
2: **Evaluar** (P_0)
3: $FEs = |P_0|$
4: $\mathcal{A} = P_0$ {$\mathcal{A}$ contiene las soluciones evaluadas en el modelo original.}
5: $c_g = 1$
6: **while** $FEs < MAX\_FEs$ **do**
7: $O = \textbf{Generar\_hijos}()$
8: $no_{nn} = \text{mín}(|\mathcal{A}|, 4 \times n)$
9: $O_u = O - (O \cap \mathcal{A})$
10: $[\hat{f}, \hat{g}] = \textbf{Aproximar\_soluciones}(O_u, no_{nn})$
11: $\textbf{Ordenar}(O_u)$
12: $\hat{x}_b = \textbf{Búsqueda\_acotada}\ (O_u^1)$
13: **Evaluar** $(\hat{x}_b)$
14: $\mathcal{A} = \textbf{Actualizar}_{\mathcal{A}}\ (\mathcal{A}, \hat{x}_b)$
15: $P_{c_g} = \textbf{Actualizar}_{P}\ (P_{c_g-1}, \hat{x}_b)$
16: $FEs = FEs + 1 + mo_{nn}$
17: $c_g = c_g + 1$
18: **end while**
19: Return $\hat{x}_b$

Figura 7.3: Pseudocódigo de SA-DECV$_L$.

7.2.2. Evaluación y aproximación de soluciones

Evaluación se lleva a cabo cuando el valor de la función objetivo y de cada restricción del problema se calcula en el modelo original.

Aproximación se requiere de un modelo subrogado o aproximado, construido a partir de un conjunto de soluciones *evaluadas* en el modelo original. El modelo subrogado que aproxima las soluciones es ϵ-SVR.

7.2.3. Conjunto de entrenamiento

Las soluciones *evaluadas* se almacenan en el conjunto de entrenamiento, denominado archivo y representado por ($\mathcal{A}$), el cual se inicializa con la población inicial y se actualiza en cada iteración con la mejor solución aproximada, de una región del espacio de búsqueda, la cual es *evaluada*. El tamaño máximo del archivo se definió con base en una experimentación previa, con un número de 500 soluciones.

7.2.4. Generación de hijos

La población de la búsqueda global genera sus hijos a partir de los operadores de variación de DECV (ver Sec. 2.3.1.5).

7.2.5. Modelos subrogados locales

La construcción del modelo subrogado local se efectúa con las $4 \times n$ soluciones más cercanas a la solución que se aproximará. El conjunto de soluciones cercanas pertenece a $\mathcal{A}$ y se denota por $\mathcal{B}$. En caso de que $|\mathcal{A}| < (4 \times n)$ se considera toda la población. Así, en aquellos problemas de alta dimensionalidad se usan modelos globales al inicio del proceso de búsqueda y se finaliza con modelos locales.

En SAMO [6] se generan varios modelos subrogados para cada función que compone el problema multiobjetivo: RBF, Kriging, MLP, RSM de primer y segundo orden, seleccionando el más preciso. Sin embargo, en el presente enfoque no se consideran todos los modelos, pues el número de modelos construidos para el proceso de búsqueda completo es $(MAX\_FEs - NP) \times NP \times 5$, donde MAX_FEs es el número máximo de evaluaciones y NP es el tamaño de la población. Por lo anterior, se considera un modelo de los menos estudiados en la literatura, ϵ-SVR, considerando dos *kernels*: lineal

y gaussiano, seleccionando el modelo con el *kernel* que tiene el menor error cuadrático medio.

7.2.6. Distancia entre soluciones

Para lograr un mejor desempeño y una mejor distribución de las soluciones durante el proceso de búsqueda, se considera una restricción de distancia entre las soluciones. El objetivo de esta restricción es evitar la convergencia prematura y se indica en la Ecuación 7.1,

$$Ed_{tol} - \|\vec{x} - \vec{x}_{\in\mathcal{A}}\| \leq 0 \tag{7.1}$$

donde Ed_{tol} se inicializa con la distancia Euclidiana mínima entre todas las soluciones de la población inicial y se actualiza durante el proceso de búsqueda con la ecuación:

$$Ed_{tol} = P_0 * (a^{c_g}); \tag{7.2}$$

donde c_g es el número de iteraciones realizadas, a y P_0 son constantes, las cuales se inicializan con las Ecuaciones 7.3 y 7.4, respectivamente.

$$a = \left(\frac{0.1}{Ed_{tol}}\right)^{\frac{1}{|\mathcal{A}|-NP}} \tag{7.3}$$

$$P_0 = \frac{Ed_{tol}}{a} \tag{7.4}$$

7.2.7. Búsqueda acotada

Al finalizar la aproximación de todas las soluciones de la población, se puede identificar la mejor solución aproximada, denominada $\hat{x}_b$, para realizar una búsqueda en la región donde se ubica, la cual se denota como $\mathcal{S}_{\hat{x}_b}$. El proceso de búsqueda se puede realizar con cualquier algoritmo evolutivo o un método de optimización clásico, además en esta etapa no se evaluará ninguna solución en el modelo original, ya que todas se aproximan. A continuación se explican cada uno de los elementos involucrados.

7.2.7.1. Definición de $\mathcal{S}_{\hat{x}_b}$

La definición de los límites ($\vec{L}_{\mathcal{S}_{\hat{x}_b}}$ y $\vec{U}_{\mathcal{S}_{\hat{x}_b}}$) de $\mathcal{S}_{\hat{x}_b}$ se puede llevar a cabo considerando los elementos de $\mathcal{B}$. Sin embargo, esta definición puede limitar el proceso de búsqueda en aquellos problemas que tienen su óptimo factible

en un extremo de $\mathcal{S}$ y que no existe una solución en $\mathcal{A}$ que dirija la búsqueda hacia esa zona. Por lo anterior, se extendieron los límites de acuerdo a la diferencia entre $\hat{x}_b$ y los límites $\vec{L}_{\mathcal{S}_{\hat{x}_b}}$ y $\vec{U}_{\mathcal{S}_{\hat{x}_b}}$, denotados por dif_L y dif_U, mediante las siguientes opciones:

1. La coordenada cuya diferencia es la menor, representada por i:

$$\begin{array}{ll} L^i_{\mathcal{S}_{\hat{x}_b}} = \text{máx}(L^i_{\mathcal{S}_{\hat{x}_b}} - \text{abs}(L^i_{\mathcal{S}_{\hat{x}_b}}), L^i) & \text{if } \min_i \in dif_L \\ U^i_{\mathcal{S}_{\hat{x}_b}} = \text{mín}(U^i_{\mathcal{S}_{\hat{x}_b}} + U^i_{\mathcal{S}_{\hat{x}_b}}), U^i) & \text{if } \min_i \in dif_U \end{array} \tag{7.5}$$

2. La menor diferencia de vectores.

$$\begin{array}{l} \vec{L}_{\mathcal{S}_{\hat{x}_b}} = \text{máx}(\vec{L}_{\mathcal{S}_{\hat{x}_b}} - \text{abs}(dif_L), \vec{L}) \\ \vec{U}_{\mathcal{S}_{\hat{x}_b}} = \text{mín}(\vec{U}_{\mathcal{S}_{\hat{x}_b}} + dif_U, \vec{U}) \end{array} \tag{7.6}$$

3. Sí $\hat{x}_b \in \mathcal{S}_{\hat{x}_b}$ entonces extender igual que en 2, en caso contrario extender como en 1.

En la Fig. 7.4 y 7.5 se ilustra el proceso de extensión y generación de puntos opuestos en el caso de que el mejor aproximado de $\mathcal{S}$ se encuentra fuera de los límites de $\mathcal{S}_{\hat{x}_b}$ (Fig. 7.4), por lo tanto, se extiende hacia la menor diferencia (Fig. 7.5).

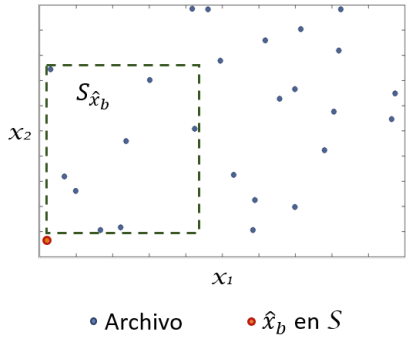


Figura 7.4: El mejor aproximado de la población (solución roja) se encuentra fuera de los límites de $\mathcal{S}_{\hat{x}_b}$.

Para garantizar una mejor precisión en el modelo que aproxima las soluciones dentro de $\vec{L}_{\mathcal{S}_{\hat{x}_b}}$ y $\vec{U}_{\mathcal{S}_{\hat{x}_b}}$ extendidos, se calcularon los puntos opuestos a $\mathcal{B}$ (ver Sec. 5.1.1.1). Sin embargo, esta mejora tiene un costo en evaluaciones, pues los puntos opuestos deben evaluarse antes de construir los modelos. Los puntos opuestos se emplean para actualizar $\mathcal{A}$ (ver Sec. 7.2.8).

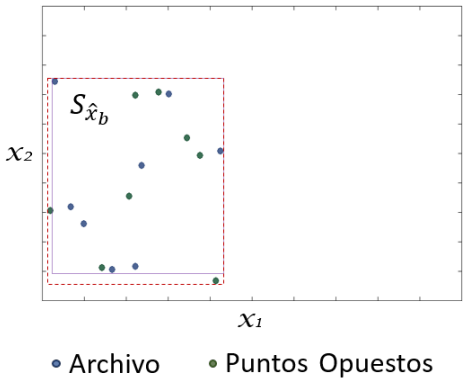


Figura 7.5: Extensión de los límites de $\mathcal{S}_{\hat{x}_b}$ hacia la menor diferencia entre $\hat{x}_b$ y los límites de la región acotada.

7.2.7.2. Algoritmo de búsqueda

La búsqueda en $\mathcal{S}_{\hat{x}_b}$ puede realizarse mediante un algoritmo evolutivo o un método de optimización clásica.

7.2.8. Actualización

Población

$$P_{c_g} = (P_{c_g} - x_{w \in P_{c_g-1}}) \cup \hat{x}_b \tag{7.7}$$

donde x_w es la peor solución de P_{c_g-1}.

Archivo

$$\mathcal{A} = \begin{cases} \mathcal{A} \cup \hat{x}_b & \text{if } |\mathcal{A}| < 500 \\ (\mathcal{A} - x_{w \in \mathcal{A}}) \cup \hat{x}_b & \text{otherwise} \end{cases} \tag{7.8}$$

donde x_w es la peor solución de $\mathcal{A}$ y $\hat{x}_b$ puede ser el mejor aproximado o un punto opuesto.

7.3. Diseño experimental

En esta sección se estudia el desempeño de 3 variantes de SA-DECV$_L$, las cuales se diferencian en cómo extienden los límites de $\mathcal{S}_{\hat{x}_b}$. En todas se generan puntos opuestos dentro de $\mathcal{S}_{\hat{x}_b}$. Además, se construyen 2 modelos subrogados locales, ϵ-SVR con *kernel* lineal y gaussiano por cada función (función objetivo y restricciones), seleccionando el más preciso. Para ello, SA-DECV$_L$ se prueba en un conjunto de 24 funciones bien conocidas (ver

Sec. 5.2 y Apéndice A). El desempeño de las versiones se determina mediante sus resultados estadísticos y las medidas de desempeño explicadas en la Sec. 5.2.1.

Los nombres para cada una de las variantes son: SA-DECV$_{(L,1)}$ que extiende $\mathcal{S}_{\hat{x}_b}$ hacia la coordenada cuya diferencia es menor, SA-DECV$_{(L,A)}$ que extiende $\mathcal{S}_{\hat{x}_b}$ hacia la diferencia menor y SA-DECV$_{(L,M)}$ si $\hat{x}_b \notin \mathcal{S}_{\hat{x}_b}$, se extiende hacia la diferencia menor, en caso contrario se extiende hacia la coordenada cuya diferencia es menor.

Los parámetros para SA-DECV$_L$ fueron definidos mediante una experimentación previa: NP=40, CR=1 y F=0.9. Los parámetros para DECV, algoritmo empleado en la búsqueda acotada, fueron: DECV: NP=100, CR=1, F=0.9 y MAX_{gen}=50. Finalmente, el número máximo de evaluaciones es 100,000. Dado el número de modelos que se genera por función, el número de ejecuciones por problema es 10.

Para el *test* de Wilcoxon no se consideraron las funciones g05, g13, g14, g17, g20-g23, en donde no se reportan soluciones factibles con ninguna de las variantes.

7.3.1. Análisis de convergencia

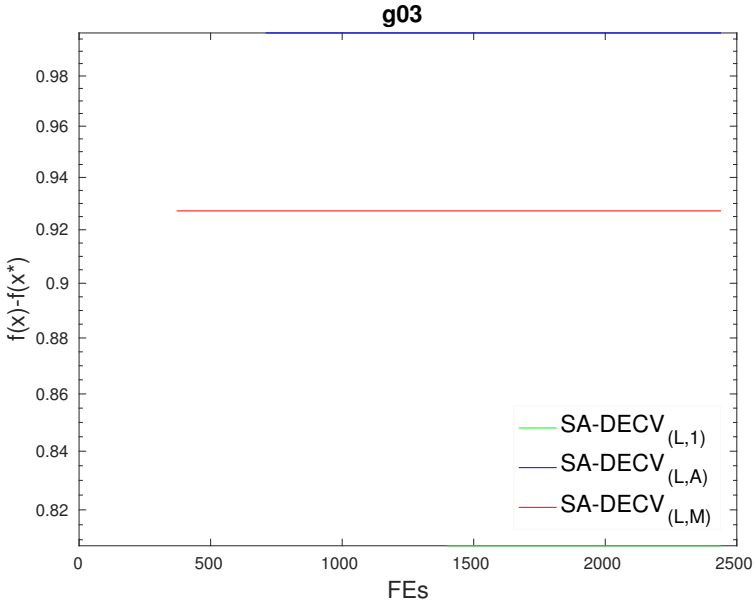
Para ilustrar el comportamiento del proceso de búsqueda, se presentan las gráficas de convergencia (de la ejecución localizada en la mediana) de dos funciones de prueba, las cuales se presentan en la Fig. 7.6. Cada gráfica presenta un problema: g03 (una restricción de igualdad) y g10 (sólo restricciones de desigualdad).

De acuerdo a las gráficas, en aquellos problemas con restricciones de igualdad (Fig. 7.6 (a)), todas las variantes alcanzan la región factible, sin embargo, se quedan atrapadas en un óptimo local. Con respecto a los problemas con restricciones de desigualdad (Fig. 7.6 (b)), cuando se extienden los límites en una sola coordenada (SA-DECV$_{(L,1)}$) el proceso de búsqueda no queda atrapado en un óptimo local, como sucede con las otras dos variantes.

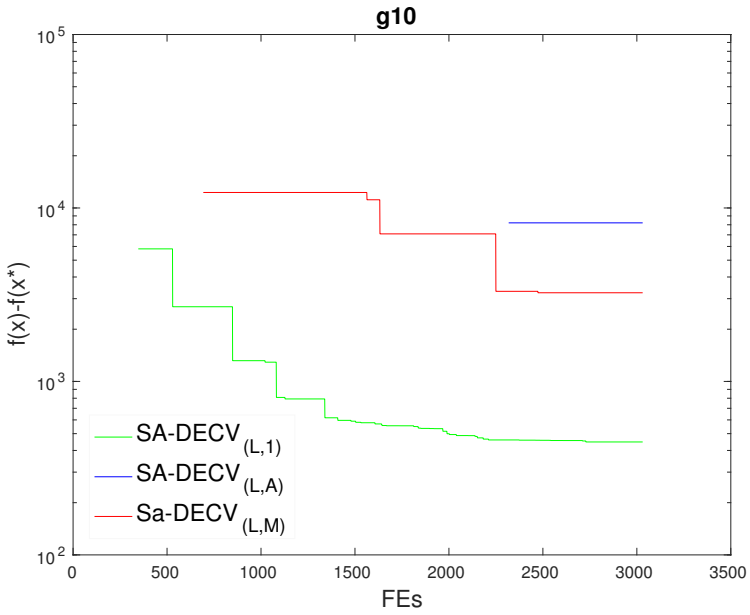
7.3.2. Resultados estadísticos

Los valores estadísticos (B:mejor, Md:mediana y SD:desviación estándar) de los resultados finales obtenidos por cada variante se muestran en las Tablas 7.1 y 7.2. Las mejores medianas en cada problema se muestran en **negrita**.

Todas las variantes encuentran el óptimo global factible en dos problemas (g08 y g12). Sin embargo, SA-DECV$_{(L,M)}$ es la mejor variante pues tiene las



(a)



(b)

Figura 7.6: Gráficas de convergencia. El eje y está en escala logarítmica.

Tabla 7.1: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 10 ejecuciones en los problemas g01-g09. Valores en **negrita** indican la mejor mediana.

Prob.	Medida	SA-DECV$_{(L,M)}$	SA-DECV$_{(L,1)}$	SA-DECV$_{(L,A)}$
g01 -15	B	-14.9987	-14.9998	-14.686
	Md	-14.9878	**-14.9994**	-14.50315
	SD	8.33E-03	1.31E-03	1.52E-01
g02 -0.80362	B	-0.278254	-0.371304	-0.237213
	Md	-0.263907	**-0.327352**	-0.218633
	SD	1.39E-02	2.20E-02	1.24E-02
g03 -1.0005	B	-0.387771	-0.452605	-0.519953
	Md	-0.0944167	**-0.322555**	-0.1141452
	SD	1.47E-01	1.84E-01	1.98E-01
g04 -30665.53867	B	-30665.5	-30640.5	-30665.5
	Md	**-30665.5**	-30594	-30665.25
	SD	0	2.90E+01	2.16E-01
g06 -6961.813876	B	-6961.81	-6954.86	-6961.81
	Md	**-6961.81**	-6944.71	**-6961.81**
	SD	9.59E-13	8.04E+00	9.59E-13
g07 24.306209	B	33.1761	24.552	73.7595
	Md	35.6861	**24.6501**	242.0475
	SD	1.53E+00	1.46E-01	1.35E+02
g08 -0.095825	B	-0.095825	-0.095825	-0.095825
	Md	**-0.095825**	**-0.095825**	**-0.095825**
	SD	1.46E-17	5.03E-07	1.46E-17
g09 680.630057	B	685.083	681.065	744.258
	Md	690.143	**683.9715**	1249.785
	SD	3.76E+00	3.29E+00	3.44E+02

Tabla 7.2: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 10 ejecuciones en los problemas g10-g24. Valores en **negrita** indican la mejor mediana.

Prob.	Medida	SA-DECV$_{(L,M)}$	SA-DECV$_{(L,1)}$	SA-DECV$_{(L,A)}$
g10 7049.248021	B	9094.6	7425.89	13227.5
	Md	10293.8	**7464.985**	15368.95
	SD	5.04E+03	1.79E+02	2.54E+03
g11 0.7499	B	0.779964	0.763506	0.770655
	Md	**0.8343995**	0.89503	0.8855865
	SD	7.70E-02	8.99E-02	9.13E-02
g12 -1	B	-1	-1	-1
	Md	**-1**	**-1**	**-1**
	SD	0.00E+00	2.21E-06	0.00E+00
g15 961.715022	B	961.715	962.305	965.685
	Md	**962.404**	963.229	965.685
	SD	1.35E+00	2.20E+00	0.00E+00
g16 -1.905155	B	-1.90515	-1.90467	-1.9038
	Md	**-1.904845**	-1.89635	-1.902645
	SD	9.05E-04	1.35E-02	1.05E-03
g18 -0.866025	B	-0.714943	-0.819137	-
	Md	**-0.682289**	-0.7990465	-
	SD	6.88E-02	5.12E-02	-
g19 32.655593	B	67.8632	35.5992	49.1214
	Md	115.715	**58.30855**	198.183
	SD	3.19E+01	1.96E+01	2.29E+02
g24 -5.508013	B	-5.50801	-5.50661	-5.50801
	Md	**-5.50801**	-5.50087	**-5.50801**
	SD	8.49E-04	3.62E-03	1.26E-05

Tabla 7.3: Resultados del test de suma de rangos de Wilcoxon, con SA-DECV$_{(L,M)}$ como algoritmo base comparado contra SA-DECV$_{(L,1)}$ y SA-DECV$_{(L,A)}$. "+" indica que hay diferencia significativa a favor del algoritmo base con respecto al algoritmo comparado. "-" indica que hay diferencia significativa a favor del algoritmo comparado con respecto al algoritmo base. "=" indica que no hay diferencia significativa entre los algoritmos. "NaN" indica que no se puede aplicar la prueba.

	g01	g02	g03	g04	g06	g07	g08	g09
SA-DECV$_{(L,1)}$	-	-	=	+	+	-	=	-
SA-DECV$_{(L,A)}$	+	+	=	+	=	+	=	+
	g10	g11	g12	g15	g16	g18	g19	g24
SA-DECV$_{(L,1)}$	-	=	=	=	+	+	-	+
SA-DECV$_{(L,A)}$	+	=	=	=	+	NaN	=	=

mejores medianas en nueve problemas y reporta el óptimo en seis problemas: por lo tanto, esta variante se considera como el algoritmo base para el *test* de Wilcoxon. Los resultados se muestran en la Tabla 7.3.

SA-DECV$_{(L,M)}$ supera a SA-DECV$_{(L,1)}$ en cinco problemas (g04, g06, g16, g18 y g24) y tiene el mismo desempeño en cinco problemas (g03, g08, g11, g12 y g15). En el resto de los problemas, SA-DECV$_{(L,1)}$ supera a SA-DECV$_{(L,M)}$ de acuerdo al *test* de Wilcoxon.

SA-DECV$_{(L,M)}$ supera a SA-DECV$_{(L_A)}$ en siete problemas (g01, g02, g04, g07, g09, g10 y g16), y tiene el mismo desempeño en el resto de los problemas de acuerdo al *test* de Wilcoxon, con excepción de g18, donde SA-DECV$_{(L,A)}$ no reporta soluciones factibles.

7.3.3. Medidas de desempeño

En las Tablas 7.4 y 7.5 se presentan los valores para las medidas de desempeño.

De acuerdo a los valores de FP (Tabla 7.4), las tres variantes alcanzan la región factible en la mayoría de los problemas con restricciones de desigualdad, con excepción del problema g18. Con respecto a aquellos problemas con restricciones de igualdad (g03, g11 y g15), todas las variantes presentan dificultades para reportar soluciones factibles. De acuerdo al promedio de los valores de FP la mejor variante es SA-DECV$_{(L,M)}$ con 0.93, seguida de

Tabla 7.4: Valores de FP y P para las variantes de SA-DECV$_L$. Los mejores valores para FP y P se muestran en **negrita.**

	FP			P		
Prob	SA-DECV$_{(L,M)}$	SA-DECV$_{(L,1)}$	SA-DECV$_{(L,A)}$	SA-DECV$_{(L,M)}$	SA-DECV$_{(L,1)}$	SA-DECV$_{(L,A)}$
g01	**1**	**1**	**1**	0	0	0
g02	**1**	**1**	**1**	0	0	0
g03	**0.8**	0.2	0.6	0	0	0
g04	**1**	**1**	**1**	**1**	0	0
g06	**1**	**1**	**1**	**1**	0	**1**
g07	**1**	**1**	**1**	0	0	0
g08	**1**	**1**	**1**	**1**	**1**	**1**
g09	**1**	**1**	**1**	0	0	0
g10	0.9	**1**	0.6	0	0	0
g11	0.2	**0.9**	0.6	0	0	0
g12	**1**	**1**	**1**	**1**	**1**	**1**
g15	**0.9**	0.6	0.1	0	0	0
g16	**1**	**1**	**1**	**0.3**	0	0
g18	**1**	**1**	0	0	0	0
g19	**1**	**1**	**1**	0	0	0
g24	**1**	**1**	**1**	0.9	0	**1**

Tabla 7.5: Valores de AFES y SP. Los mejores valores de AFES y SP se muestran en **negrita**. "-" indica que no existe valor en ese problema.

	AFES			SP		
Prob	SA-DECV$_{(L,M)}$	SA-DECV$_{(L,1)}$	SA-DECV$_{(L,A)}$	SA-DECV$_{(L,M)}$	SA-DECV$_{(L,1)}$	SA-DECV$_{(L,A)}$
g01	-	-	-	-	-	-
g02	-	-	-	-	-	-
g03	-	-	-	-	-	-
g04	**76915.7**	-	-	**76915.7**	-	-
g06	36539.6	-	**27068.9**	36539.6	-	**27068.9**
g07	-	-	-	-	-	-
g08	12437.6	**9585.5**	12787.7	12437.6	**9585.5**	12787.7
g09	-	-	-	-	-	-
g10	-	-	-	-	-	-
g11	-	-	-	-	-	-
g12	6361.6	5876.7	**5511.4**	6361.6	5876.7	**5511.4**
g15	-	-	-	-	-	-
g16	**61956**	-	-	**206520**	-	-
g18	-	-	-	-	-	-
g19	-	-	-	-	-	-
g24	67528	-	**58312.4**	75031.1	-	**58312.4**

SA-DECV$_{(L,1)}$ con 0.92, ambas con dieciséis problemas. SA-DECV$_{(L,A)}$ con 0.9 con quince problemas.

Aunque las variantes alcanzan la región factible, de acuerdo a los valores de P (Tabla 7.4) no son capaces de alcanzar la vecindad de la mejor solución factible conocida, con excepción de aquellos problemas con restricciones de desigualdad y dimensionalidad baja (menor o igual a cinco). La variante más competitiva de acuerdo al número de problemas que alcanza la vecindad de la mejor solución factible conocida es SA-DECV$_{(L,M)}$ con seis problemas (g04, g06, g08, g12, g16 y g24), seguida de SA-DECV$_{(L,A)}$ con cuatro problemas (g04, g08, g12 y g24) y SA-DECV$_{(L,1)}$ con dos problemas (g08 y g12).

Con respecto a los valores de AFES (Tabla 7.5), la variante más competitiva, pues tiene los valores más bajos de AFES en tres de cuatro problemas es SA-DECV$_{(L,A)}$ con un promedio de 25920.1 para los cuatro problemas. Sin embargo, el promedio más bajo es para SA-DECV$_{(L,1)}$ de 7731.1 para dos problemas y SA-DECV$_{(L,M)}$ tiene un promedio de 43623.08 para seis problemas.

El comportamiento de las variantes con respecto a la medida SP (Tabla 7.5) es semejante al de AFES. Así, los promedios son SA-DECV$_{(L,1)}$ con 7731.1 (dos problemas), seguida de SA-DECV$_{(L,A)}$ con 25920.1 (cuatro problemas) y SA-DECV$_{(L,M)}$ con 68967.6 (seis problemas).

Los resultados del experimento anterior llevan a las siguientes conclusiones: aunque todas las variantes alcanzan la zona factible en aquellos pro-

blemas con restricciones de desigualdad, no siempre reportan una solución cercana al mejor óptimo factible. Con respecto a los problemas que incluyen restricciones de igualdad, no todas las variantes son capaces de alcanzar la región factible y en aquellos que la alcanzan, no reportan una solución dentro de la vecindad del mejor óptimo factible conocido. Asimismo, la extensión de límites en $\mathcal{S}_{\hat{x}_b}$ en la menor diferencia reporta mejores resultados que en aquellos que se extiende hacia la coordenada de menor diferencia.

7.3.4. Restricciones de igualdad

En el experimento anterior las variantes muestran un pobre desempeño en aquellos problemas con restricciones de igualdad. Además, el número de evaluaciones se incrementa al evaluar los puntos opuestos para aproximar cada solución de la población. En este experimento se reduce el número de evaluaciones y el número de modelos construido para cada función.

A partir del desempeño de las variantes del primer experimento, se realizan los siguientes cambios en SA-DECV$_L$: (1) se emplea un ϵ-SVR con kernel RBF, (2) se extienden los límites de $\mathcal{S}_{\hat{x}_b}$ sólo cuando el mejor aproximado no está en $\mathcal{S}_{\hat{x}_b}$. (3) Se emplea un método de optimización clásico (Hooke-Jeeves) para realizar la búsqueda en $\mathcal{S}_{\hat{x}_b}$. (4) se emplea una tolerancia dinámica para las restricciones de igualdad.

A continuación se explican la tolerancia dinámica y el método de optimización clásico empleados.

7.3.5. Tolerancia dinámica para restricciones de igualdad

Las restricciones de igualdad son difíciles de satisfacer debido a que la región factible definida por ellas es muy pequeña [89]. Por esta razón, las restricciones de igualdad se transforman en restricciones de desigualdad (ver Sec. 2.2) agregando una tolerancia ϵ. El valor de la tolerancia extiende artificialmente la región factible, lo que facilita que las soluciones la alcancen.

Existen diversas alternativas para asignar un valor a ϵ, las cuales son: (1) Asignar un valor fijo durante todo el proceso de búsqueda [41]. (2) Un ajuste dinámico con respecto a un parámetro definido por el usuario [23, 49]. (3) Un ajuste adaptativo por ambos lados de la restricción de igualdad [23] (4) Un ajuste adaptativo con base en el porcentaje de soluciones factibles (prc_f) [87].

El enfoque adaptado en este trabajo es el último, el cual consiste en asignar una tolerancia inicial, ϵ=1, la cual se decrementa en 10 % hasta llegar a la tolerancia final ϵ_f (1E-4). ϵ se decrementa sí el porcentaje de soluciones

factibles (prc_f) se mantiene. El valor inicial de PFP es 100 % y se actualiza cada generación mediante la Ecuación 7.9,

$$PFP = \left(1 - \frac{it}{MAX_{it}}\right) \% \tag{7.9}$$

donde it es el número de iteración actual y MAX_{it} es el máximo de número de iteraciones.

El decremento de ϵ no siempre puede alcanzar el valor de ϵ_f al final del proceso de búsqueda, por lo tanto, en la última iteración se considera $\epsilon = \epsilon_f$.

7.3.6. Método de Hooke-Jeeves

El método de Hooke-Jeeves es un método de programación matemática que se compone de dos tipos de movimientos: movimiento exploratorio y movimiento de patrones. El primero permite calcular direcciones de descenso en la vecindad del punto actual, mientras que el segundo calcula un nuevo punto con base en la dirección del punto actual y el anterior.

Require: $\vec{x_c}$, $\vec{\Delta}$
Ensure: $\vec{x}$, *success*
1: $\vec{x_c} = \vec{x}$
2: **for** i<n **do**
3: $f = f(\vec{x})$, $f^+ = f(x_i + \Delta\_i)$, $f^- = f(x_i - \Delta\_i)$
4: $f_{min} = \text{mín}(f, f^+, f^-)$, $\vec{x}$ corresponde a f_{min}
5: **end for**
6: **if** $\vec{x} \neq \vec{x_c}$ **then**
7: *success* = *true*
8: **else**
9: *success* = *false*
10: **end if**

Figura 7.7: Pseudocódigo del movimiento exploratorio de Hooke-Jeeves.

Movimiento exploratorio. Perturba el punto actual (x_c) con una dirección positiva y negativa (Δ) en cada variable, guardando la mejor posición en x. Si al final del movimiento exploratorio x_c es diferente de x, se considera un éxito (*success*), de otra manera es un fracaso. En la Fig. 7.7 se muestra el pseudocódigo de este movimiento.

Movimiento de patrones. Encuentra un nuevo punto mediante un salto del punto actual x_c a lo largo de la dirección que existe entre el punto anterior x_{k-1} y el punto base actual x_k, mediante la Ecuación 7.10.

$$\vec{x}_p = \vec{x}_k + (\vec{x}_k - \vec{x}_{k-1}) \tag{7.10}$$

Finalmente, para no evaluar más de una solución por iteración sólo se extienden $\vec{L}_{\mathcal{S}_{\hat{x}_b}}$ y $\vec{U}_{\mathcal{S}_{\hat{x}_b}}$ cuando $\hat{x}_b$ está fuera de $\mathcal{S}_{\hat{x}_b}$, es decir, $\vec{L}_{\mathcal{S}_{\hat{x}_b}}$ y $\vec{U}_{\mathcal{S}_{\hat{x}_b}}$ están en función de $\mathcal{B} \cap \hat{x}_b$

En la Fig. 7.8 se muestra el pseudocódigo del método de Hooke-Jeeves.

Require: $\vec{x}_0$, $\vec{\Delta}$, α, ϵ
Ensure: $\vec{x}$
1: $\vec{x}_k = \vec{x}_0$
2: $[\vec{x}_{k1}, band]$ = movimiento exploratorio($\vec{x}_k$, $\vec{\Delta}$) {Fig. 7.7}
3: **repeat**
4: **if** *success* **then**
5: **repeat**
6: $\vec{x}_p$ = Movimiento de patrones {Ec. 7.10}
7: $[\vec{x}, band]$ = movimiento exploratorio($\vec{x_p}$, $\vec{\Delta}$) {Fig. 7.7}
8: $\vec{x}_{k1} = \vec{x}$
9: **until** $f(\vec{x}_{k1}) > f(\vec{x}_k)$
10: **end if**
11: $\vec{\Delta} = \vec{\Delta}/\alpha$
12: $[\vec{x}, band]$ = movimiento exploratorio($\vec{x_k}$, $\vec{\Delta}$) {Fig. 7.7}
13: **until** $\|\Delta\| < \epsilon$

Figura 7.8: Pseudocódigo del método de Hooke-Jeeves.

El método de Hooke-Jeeves es simple y sencillo [13]. Sin embargo, el número de evaluaciones que se realiza en el movimiento exploratorio está en función del número de variables de diseño, por lo que para problemas de alta dimensionalidad, este método es costoso por el número de evaluaciones. Esta desventaja no afecta en la propuesta de SA-DECV$_L$, ya que la evaluación de las soluciones se lleva a cabo mediante el modelo subrogado, ϵ-SVR en este caso.

Dado que este método se emplea para realizar una búsqueda acotada, la condición de paro se hizo más robusta considerando los siguientes aspectos: (1) Sí $\vec{x} \in \mathcal{A}$, entonces terminar. Esta condición evita que no se pierda la diversidad en $\mathcal{A}$. (2) Sí $\vec{x} < \vec{L}_{\mathcal{S}_{\hat{x}_b}}$ o $\vec{x} > \vec{U}_{\mathcal{S}_{\hat{x}_b}}$, entonces terminar. Al

terminar la búsqueda cuando la solución está fuera del espacio acotado, se da cierto grado de libertad para encontrar una solución fuera de esa región. Sin embargo, no debe estar muy lejos porque el modelo subrogado está entrenado sólo para esa área.

En la figura 7.9 se muestra el pseudocódigo final para SA-DECV$_L$.

1: **if** nh >0 **then**
2: $\epsilon = 1$
3: $\epsilon_f = 1E-4$
4: $PFP = 100\,\%$
5: **end if**
6: $P_0 = \text{LHS}(2*(n+1))$
7: **Evaluar** (P_0)
8: $FEs = |P_0|$
9: $\mathcal{A} = P_0$ {$\mathcal{A}$ contiene las soluciones evaluadas en el modelo original.}
10: $c_g = 1$
11: **while** $FEs < MAX\_FEs$ **do**
12: $O =$ **Generar_hijos**()
13: $no_{nn} = \text{mín}(|\mathcal{A}|, 4 \times n)$
14: $O_u = O - (O \cap \mathcal{A})$
15: $[\hat{f}, \hat{g}] =$ **Aproximar_soluciones**(O_u, no_{nn})
16: **Ordenar**(O_u)
17: $\hat{x}_b =$ **Búsqueda_acotada** (O_u^1)
18: **Evaluar** $(\hat{x}_b)$
19: $\mathcal{A} =$ **Actualizar**$_{\mathcal{A}}$ $(\mathcal{A}, \hat{x}_b)$
20: $P_{c_g} =$ **Actualizar**$_P$ $(P_{c_g-1}, \hat{x}_b)$
21: $FEs = FEs + 1$
22: n_{fac} = Número de soluciones factibles
23: **if** $FEs < 0.99 * MAX\_FEs$ **and** $n_{fac} > PFP$ **then**
24: $\epsilon = \epsilon * 0{,}9$
25: **else**
26: $\epsilon = \epsilon_f$
27: **end if**
28: Actualizar la suma de violación de restricciones
29: $PFP = (1 - (it/MAX_{it}))\,\%$
30: $c_g = c_g + 1$
31: **end while**
32: Return $\hat{x}_b$

Figura 7.9: Pseudocódigo de SA-DECV$_L$. Las líneas 1-5 y 22-29 integran la tolerancia adaptativa para los problemas con restricciones de igualdad. En la línea 17 se integra Hooke-Jeees como algoritmo de búsqueda.

El desempeño de dos variantes de SA-DECV$_L$, las cuales difieren del algoritmo de búsqueda empleado en $\mathcal{S}_{\hat{x}_b}$: una emplea DECV y la otra Hooke-Jeeves, se presenta a continuación. Cada variante se prueba en un conjunto de 24 funciones bien conocidas. Además su desempeño se determina con base en sus resultados estadísticos y las medidas de desempeño FP, P, AFES y SP.

Los nombres de cada una de las variantes son: SA-DECV$_{(L,HJ)}$ que emplea Hooke-Jeeves como algoritmo de búsqueda en la región acotada, mientras que SA-DECV$_{(L,DECV)}$ emplea DECV como algoritmo de búsqueda.

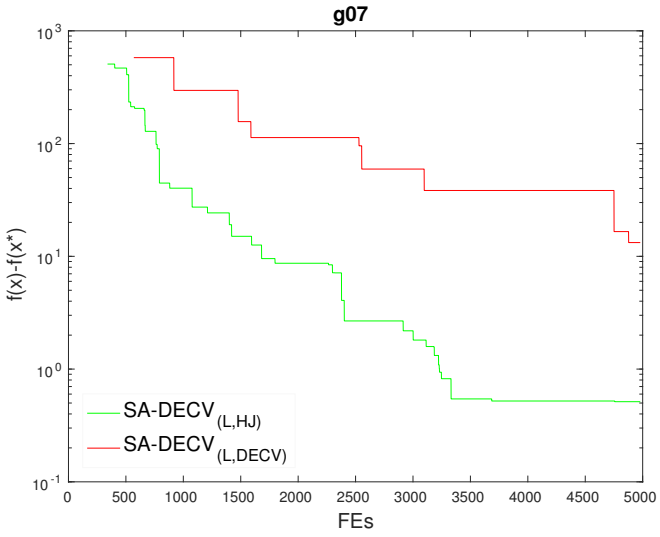
Los parámetros para SA-DECV$_L$ son: $NP = 2(n+1)$, $CR = 1$ y $F = 0{,}9$. Los parámetros para cada algoritmo que se emplea en la región acotada son: (1) DECV: $NP = 100$, $CR = 1$, $F = 0{,}9$ y $MAX_{gen} = 50$. (2) Hooke-Jeeves: δ=$0.4\left(\frac{\vec{U}_{\mathcal{S}_{\hat{x}_b}} - \vec{L}_{\mathcal{S}_{\hat{x}_b}}}{\sqrt{n}}\right)$, $\alpha = 2$ y $\varepsilon_{HJ} = \text{mín}\left(0.1\left(\frac{\vec{U}_{\mathcal{S}_{\hat{x}_b}} - \vec{L}_{\mathcal{S}_{\hat{x}_b}}}{\sqrt{n}}\right)\right)$. El número máximo de evaluaciones es 5,000 y el número de ejecuciones por problema es 30.

El *test* de Wilcoxon no se aplicó en los problemas g05, g20-g23 en donde no se reportan soluciones factibles en ninguna de las variantes. Se tomó como algoritmo base el que reporta las mejores medianas de las variantes.

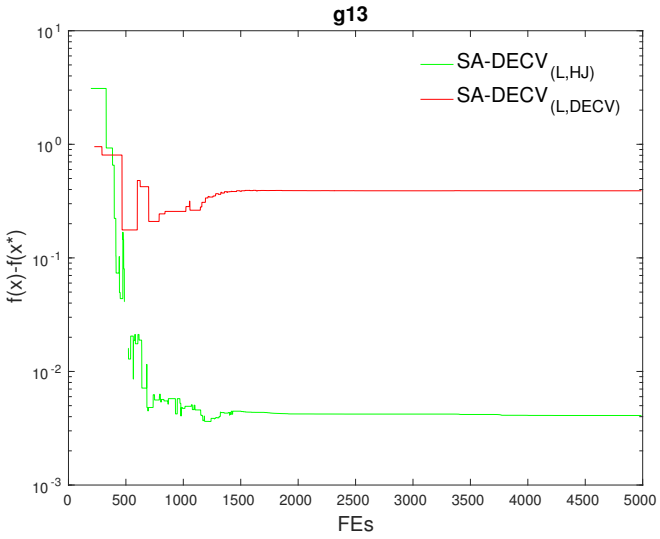
7.3.7. Análisis de convergencia

Para ilustrar el comportamiento del proceso de búsqueda, se presentan las gráficas de convergencia (de la ejecución localizada en la mediana) de dos funciones de prueba, las cuales se presentan en la Fig. 7.10. Cada gráfica presenta un problema: g07 (sólo restricciones de desigualdad) y g13 (sólo restricciones de igualdad).

De acuerdo a las gráficas, en aquellos problemas con restricciones de desigualdad (Fig. 7.10 (a)), las dos variantes alcanzan la región factible antes de las 1,000 evaluaciones y SA-DECV$_{(L,HJ)}$ realiza una mejor exploración dentro de la región factible que SA-DECV$_{(L,DECV)}$. Con respecto a los problemas con restricciones de igualdad (Fig. 7.10 (b)) las dos variantes quedan atrapadas en óptimos locales, sin embargo, SA-DECV$_{(L,HJ)}$ presenta mayor exploración antes de quedar atrapada en un óptimo local.



(a)



(b)

Figura 7.10: Gráficas de convergencia para los problemas: (a) g07 tiene sólo restricciones de desigualdad y (b) g13 tiene sólo restricciones de igualdad. El eje y está en escala logarítmica.

7.3.7.1. Resultados estadísticos

Los valores estadísticos (B:mejor, Md:mediana y SD:desviación estándar) de los resultados finales obtenidos por cada algoritmo se muestran en las Tablas 7.6 y 7.7. Las mejores medianas en cada problema se muestran en **negrita**.

De acuerdo a las Tablas 7.6 y 7.7, el óptimo global factible fue encontrado por los dos algoritmos en 5 problemas (g08, g11, g12, g15 y g24). Sin embargo, SA-DECV$_{(L,HJ)}$ es la mejor variante ya que reporta la mejor mediana en todos los problemas. Por lo tanto, esta variante se tomó como algoritmo base para el *test* de Wilcoxon. Los resultados se muestran en la tabla 5.6.

SA-DECV$_{(L,DECV)}$ tiene el mismo desempeño que SA-DECV$_{(L,HJ)}$ en 5 problemas (g01, g03, g08, g12 y g24) de acuerdo al *test* de Wilcoxon, en el problema g17 no se pudo aplicar la prueba porque SA-DECV$_{(L,DECV)}$ no reporta soluciones factibles y en el resto de los problemas SA-DECV$_{(L,HJ)}$ es superior a SA-DECV$_{(L,DECV)}$.

7.3.7.2. Medidas de desempeño

En las Tablas 7.9 y 7.10 se muestran los resultados de las medidas de desempeño.

De acuerdo a los valores de FP (Tabla 7.9) las 2 variantes son capaces de encontrar al menos una solución factible en aquellos problemas con restricciones de desigualdad. Sin embargo, SA-DECV$_{(L,DECV)}$ no reporta soluciones factibles en 2 problemas con restricciones de igualdad (g14 y g17). El promedio de cada variante es: SA-DECV$_{(L,HJ)}$ con 0.9 en 19 problemas y SA-DECV$_{(L,DECV)}$ con 0.89 en 17 problemas. Por otro lado, la variante que reporta soluciones cerca de la vecindad del óptimo, de acuerdo a los valores de P (Tabla 7.9) es SA-DECV$_{(L,HJ)}$ con 10 problemas (g01, g04, g06, g08, g11-g13, g15, g16 y g24), mientras que SA-DECV$_{(L,DECV)}$ reporta una solución cercana al óptimo en 5 problemas (g08, g11, g12, g15 y g24).

La variante que requiere menos evaluaciones (valores de AFES, Tabla 7.10) cuando las 2 variantes alcanzan la vecindad del mejor óptimo factible conocido es SA-DECV$_{(L,HJ)}$, pues reporta los mejores valores en 4 de 5 problemas (g08, g11, g15 y g24) con un promedio de 524.96 y SA-DECV$_{(L,DECV)}$ tiene un promedio de 925.74, ambas en los 5 problemas. Con respecto a los valores de SP, el promedio de SA-DECV$_{(L,HJ)}$ es 756.42 y SA-DECV$_{(L,DECV)}$ tiene 2912.62, en los 5 problemas.

Tabla 7.6: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 30 ejecuciones en los problemas g01-g09. Valores en **negrita** indican la mejor mediana.

Prob.	Medida	SA-DECV$_{(L,HJ)}$	SA-DECV$_{(L,DECV)}$
g01 -15	B	-15	-14.9692
	Md	**-14.99825**	-14.92935
	SD	1.26E+00	7.61E-02
g02 -0.80362	B	-0.52033	-0.251542
	Md	**-0.406084**	-0.2023045
	SD	6.32E-02	1.68E-02
g03 -1.0005	B	-0.872306	-0.516083
	Md	**-0.40303**	-0.309591
	SD	2.75E-01	1.30E-01
g04 -30665.53867	B	-30665.5	-30665
	Md	**-30665.5**	-30636.85
	SD	1.22E-01	5.82E+01
g06 -6961.813876	B	-6961.81	-6961.81
	Md	**-6961.81**	-6961.79
	SD	1.85E-12	1.51E-01
g07 24.306209	B	24.4694	30.8769
	Md	**24.81**	37.75975
	SD	7.94E-01	6.79E+00
g08 -0.095825	B	-0.095825	-0.095825
	Md	**-0.095825**	**-0.095825**
	SD	4.23E-17	4.23E-17
g09 680.630057	B	680.643	681.023
	Md	**680.974**	684.623
	SD	1.66E+00	1.14E+01
g10 7049.248021	B	7684.28	8443.62
	Md	**8238.355**	12038.45
	SD	1.31E+03	3.17E+03

Tabla 7.7: Resultados estadísticos (B: mejor, Md: mediana y SD: desviación estándar) de las 30 ejecuciones en los problemas g10-g24. Valores en **negrita** indican la mejor mediana.

Prob.	Medida	SA-DECV$_{(L,HJ)}$	SA-DECV$_{(L,DECV)}$
g11 0.7499	B	0.7499	0.749905
	Md	**0.749906**	**0.749906**
	SD	2.02E-06	1.43E-02
g12 -1	B	-1	-1
	Md	**-0.994375**	-0.9864455
	SD	2.06E-02	1.70E-02
g13 0.053942	B	0.0539498	0.0580865
	Md	**0.0575785**	0.4444515
	SD	2.36E-01	1.94E-01
g14 -47.764888	B	-47.4358	0
	Md	**-46.90265**	0
	SD	6.12E-01	0.00E+00
g15 961.715022	B	961.715	961.715
	Md	**961.715**	961.784
	SD	9.46E-03	6.12E-01
g16 -1.905155	B	-1.90512	-1.87538
	Md	**-1.899395**	-1.62832
	SD	1.86E-02	1.05E-01
g17 8853.539675	B	9005.57	0
	Md	**9005.57**	0
	SD	0.00E+00	0.00E+00
g18 -0.866025	B	-0.864521	-0.778215
	Md	**-0.844113**	-0.597824
	SD	7.45E-02	7.98E-02
g19 32.655593	B	34.3971	52.9184
	Md	**45.79325**	145.398
	SD	1.43E+01	2.26E+02
g24 -5.508013	B	-5.50801	-5.50801
	Md	**-5.50801**	**-5.50801**
	SD	2.71E-15	2.71E-15

Tabla 7.8: Resultados del *test* de suma de rangos de Wilcoxon, con SA-DECV$_{(L,HJ)}$ como algoritmo base comparado contra SA-DECV$_{(L,DECV)}$. "+" indica que hay diferencia significativa a favor del algoritmo base con respecto al algoritmo comparado. "-" indica que hay diferencia significativa a favor del algoritmo comparado con respecto al algoritmo base. "=" indica que no hay diferencia significativa entre los algoritmos. "NaN" indica que no se pudo aplicar el *test*.

	g01	g02	g03	g04	g06	g07	g08	g09	g10	g11
SA-DECV$_{(L,DECV)}$	=	+	=	+	+	+	=	+	+	+
	g12	g13	g14	g15	g16	g17	g18	g19	g24	
SA-DECV$_{(L,DECV)}$	=	+	+	+	+	NaN	+	+	=	

Tabla 7.9: Valores de FP y P. Los mejores valores para FP y P se muestran en **negrita.**

	FP		P	
Prob	SA-DECV$_{(L,HJ)}$	SA-DECV$_{(L,DECV)}$	SA-DECV$_{(L,HJ)}$	SA-DECV$_{(L,DECV)}$
g01	**1**	**1**	**0.2**	0
g02	**1**	**1**	0	0
g03	**0.6**	0.33	0	0
g04	**1**	**1**	**0.3**	0
g06	**1**	**1**	**1**	0
g07	**1**	**1**	0	0
g08	**1**	**1**	**1**	**1**
g09	**1**	**1**	0	0
g10	**1**	**1**	0	0
g11	**1**	**1**	**1**	0.83
g12	**1**	**1**	**0.3**	0.13
g13	**1**	0.13	**0.43**	0
g14	**0.93**	0	0	0
g15	**1**	0.63	**0.97**	0.23
g16	**1**	**1**	**0.03**	0
g17	**0.03**	0	0	0
g18	**1**	**1**	0	0
g19	**1**	**1**	0	0
g24	**1**	**1**	**1**	**1**

Tabla 7.10: Valores de AFES y SP. Los mejores valores para AFES y SP se muestran en **negrita.**

	AFES		SP	
Prob	SA-DECV$_{(L,HJ)}$	SA-DECV$_{(L,DECV)}$	SA-DECV$_{(L,HJ)}$	SA-DECV$_{(L,DECV)}$
g01	**4271.3**	-	**21356.5**	-
g02	-	-	-	-
g03	-	-	-	-
g04	**2499.7**	-	**8332.3**	-
g06	**1452.1**	-	**1452.1**	-
g07	-	-	-	-
g08	**212.7**	339.7	**212.7**	339.7
g09	-	-	-	-
g10	-	-	-	-
g11	**396.8**	1132.7	**396.8**	1364.7
g12	480	**469.5**	**1600**	3611.5
g13	**635.7**	-	**1478.4**	-
g14	-	-	-	-
g15	**1206.1**	1959.6	**1243.4**	8520
g16	**820**	-	**27333.3**	-
g17	-	-	-	-
g18	-	-	-	-
g19	-	-	-	-
g24	**329.2**	727.2	**329.2**	727.2

Los resultados de este experimento llevan a las siguientes conclusiones: la tolerancia dinámica mejoró el desempeño en aquellos problemas que tienen sólo restricciones de igualdad, pues en problemas con ambos tipos de restricciones ninguna variante alcanza la región factible. A pesar de que el movimiento exploratorio se evalúa con la ayuda de modelos aproximados, este logra guiar la búsqueda de manera eficiente en la mayoría de los problemas.

7.3.8. Discusión de resultados

Los resultados de los experimentos previos sugieren que la aproximación de la población mediante modelos subrogados locales, los cuales aproximan la función objetivo y cada una de las restricciones, permiten guiar el proceso de búsqueda hacia la zona factible en aquellos problemas con restricciones de desigualdad principalmente. Este comportamiento no se logra en los problemas con restricciones de igualdad, pues en los problemas que tienen los dos tipos de restricciones, ninguna variante de los experimentos anteriores alcanzó la región factible.

La construcción de modelos con diferente *kernel* por función (experimento 1) o un sólo modelo (experimento 2), no muestra una diferencia significativa en los resultados, pues ninguna de las variantes es robusta. Esta reducción de modelos, permitió que las variantes del experimento 2 fueran más rápidas.

ϵ-SVR permitió guiar mejor la búsqueda con un algoritmo de optimización clásico, Hooke-Jeeves. Además, la tolerancia dinámica en los problemas de igualdad dio mejores resultados. Lo anterior se fundamenta con los resultados estadísticos y las medidas de desempeño SA-DECV$_{(L,HJ)}$.

7.4. Resumen del capítulo

En este capítulo se presentó una mejora a SA-DECV empleando las características de la propuesta SAMO, con el objetivo de encontrar una solución competitiva empleando un menor número de evaluaciones que las usadas en la versión del capítulo 5. Además, se integra una tolerancia dinámica para resolver problemas con restricciones de igualdad y se realiza una búsqueda acotada con un método de optimización clásica. De acuerdo a resultados, la tolerancia dinámica y el método de optimización obtienen los resultados más competitivos.

Capítulo 8

Conclusiones

En el presente trabajo se ha incorporado un modelo subrogado en el algoritmo evolutivo DECV, para resolver problemas de optimización con restricciones, con el objetivo de obtener resultados competitivos empleando un menor número de evaluaciones en el modelo original (hipótesis). Para lograr dicho objetivo, se resolvieron las siguientes preguntas: ¿en qué etapa del algoritmo evolutivo se van a aproximar las soluciones, en la población inicial, en los operadores de variación, en la evaluación o en un buscador local, ¿qué alcance tendrá el modelo subrogado, global o local?, ¿se van a considerar sólo soluciones exactas en cada generación, sólo aproximadas o se van a combinar?, ¿qué modelo subrogado es el más adecuado?, ¿qué medida de desempeño me indicará la precisión del modelo?, entre otras.

Debido a que el uso de estos modelos en algoritmos evolutivos está en crecimiento, muchas de las preguntas se resuelven revisando la literatura o de manera empírica. Así, en el presente trabajo se realiza un estudio de los modelos subrogados empleados en la etapa de evaluación de las soluciones (individuos) desde dos perspectivas: la primera combina tanto soluciones exactas, evaluadas en el modelo original, como soluciones aproximadas, evaluadas en el modelo subrogado en cada generación, empleando un control de evolución basado en factibilidad. Y la segunda, sólo considera soluciones exactas en cada generación.

De acuerdo al análisis de resultados se puede establecer lo siguiente, cuando se combinan soluciones exactas y aproximadas en una generación (primer enfoque), la probabilidad de requerir más evaluaciones para alcanzar una solución competitiva es mayor que cuando se emplean sólo soluciones exactas en el proceso de búsqueda (segundo enfoque). Además, si se requiere aproximar la suma de violación de restricciones en lugar de generar un modelo

para cada restricción, una buena opción es emplear como técnica para manejo de restricciones aquella que conserve tanto soluciones factibles como no factibles, tal como la jerarquización estocástica. Asimismo, dada la complejidad que representa el alcanzar la zona factible y encontrar el óptimo en problemas con restricciones de igualdad, el uso de una tolerancia dinámica mejora el desempeño de las propuestas. Finalmente, los modelos subrogados son una buena opción en aquellos métodos de optimización que realizan una exploración en cada eje del espacio de búsqueda, pues se pueden aproximar las soluciones que realizan la exploración sin gastar evaluaciones en el modelo original.

Con respecto a las versiones de SA-DECV la conclusión es que ambas variantes, son capaces de encontrar soluciones competitivas, cumpliendo el objetivo de reducir el número de evaluaciones efectuadas en el modelo original. Por tanto, la hipótesis del trabajo se corrobora. Además, es competitivo con otros algoritmos del estado del arte, tanto de aquellos que integran subrogados, como los que resuelven problemas de ingeniería.

De acuerdo a las métricas de desempeño, SA-DECV alcanza la región factible (valores de FP) en la mayoría de los problemas, sin embargo, encontrar una solución cercana al óptimo se complica (valores de P). Los valores de AFES y SP varían para los dos enfoques, pues el número máximo de evaluaciones para cada uno es diferente.

8.1. Trabajo futuro

SA-DECV es una buena opción para resolver problemas de optimización con restricciones, sin embargo, se puede mejorar el control de evolución basado en factibilidad, para ahorrar más evaluaciones en el algoritmo, considerando una hiperesfera con un radio menor al considerado actualmente. Con base en los resultados del segundo enfoque, se puede integrar la tolerancia adaptativa en el primer enfoque para estudiar su desempeño en problemas con restricciones de igualdad.

En el segundo enfoque se puede integrar un algoritmo libre de derivadas, tal como regiones de confianza, para realizar la búsqueda en la región acotada, además de probar otros modelos, tal como kriging o MLP. Finalmente, SA-DECV se puede probar en los problemas del CEC 2010 o en un problema real.

Parte II

Apéndices

Apéndice A

Problemas CEC 2006

Definición matemática de los 24 problemas utilizados en este trabajo.

g01

Minimizar:

$$f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{A.1}$$

sujeto a:

$$\begin{aligned}
g_1(\vec{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(\vec{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(\vec{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(\vec{x}) &= -8x_1 + x_{10} \leq 0 \\
g_5(\vec{x}) &= -8x_2 + x_{11} \leq 0 \\
g_6(\vec{x}) &= -8x_3 + x_{12} \leq 0 \\
g_7(\vec{x}) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(\vec{x}) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(\vec{x}) &= -2x_8 - x_9 + x_{12} \leq 0
\end{aligned} \tag{A.2}$$

donde $0 \leq x_i \leq 1$ (i=1,...,9), $0 \leq x_i \leq 100$ (i=10, 11, 12), y $0 \leq x_{13} \leq 1$. El óptimo global factible se localiza en $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, con $f(x^*) = -15$. g_1, g_2, g_3, g_7, g_8, g_9 son restricciones activas.

g02

Minimizar:

$$f(\vec{x}) = -\left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{A.3}$$

sujeto a:

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$
$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0 \tag{A.4}$$

donde n=20 y $0 < x_i \leq 10$ $(i = 1, \ldots, n)$. La mejor solución conocida se localiza en x^*= (3.16246061572185, 3.12833142812967, 3.09479212988791, 3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285, 2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865, 0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199, 0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760, 0.44038285956317), con $f(x^*)$=-0.80361910412559. g_1 es cercana a una restricción activa.

g03

Minimizar:

$$f(\vec{x}) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i \tag{A.5}$$

sujeto a:

$$h(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \tag{A.6}$$

donde n=10 y $0 \leq x_i \leq 1$ $(i = 1, \ldots, n)$. El mínimo global factible se localiza en x^* = (0.31624357647283069, 0.31624357741433833 9, 0.31624357801234592 7, 0.31624357566401789 5, 0.31624357820552606 6, 0.31624357738855069, 0.31624357547294951 2, 0.31624357716488393 8, 0.31624357815592030 2, 0.31624357614737491 6), con $f(x^*)$=-1.00050010001000.

g04

Minimizar:

$$f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{A.7}$$

sujeto a:

$$\begin{aligned}
g_1(\vec{x}) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\
g_2(\vec{x}) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\
g_3(\vec{x}) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \\
g_4(\vec{x}) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \\
g_5(\vec{x}) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \\
g_6(\vec{x}) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0
\end{aligned} \tag{A.8}$$

donde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ y $27 \leq x_i \leq 45$ (i=3, 4, 5). El óptimo global factible se localiza en x^*=(78, 33, 29.9952560256815985, 45, 36.7758129057882073), con $f(x^*)$= -3.066553867178332E+04. g_1 y g_6 son restricciones activas.

g05

Minimizar:

$$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3 \tag{A.9}$$

sujeto a:

$$\begin{aligned}
g_1(\vec{x}) &= -x_4 + x_3 - 0.55 \leq 0 \\
g_2(\vec{x}) &= -x_3 + x_4 - 0.55 \leq 0 \\
h_3(\vec{x}) &= 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\
h_4(\vec{x}) &= 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\
h_5(\vec{x}) &= 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0
\end{aligned} \tag{A.10}$$

donde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ y $-0.55 \leq x_4 \leq 0.55$. La mejor solución conocida se localiza en x^*=(679.945148297028709, 1026.06697600004691, 0.118876369094410433, -0.39623348521517826), con $f(x^*)$=5126.4967140071.

g06

Minimizar:

$$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{A.11}$$

sujeto a:

$$\begin{aligned}
g_1(\vec{x}) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\
g_2(\vec{x}) &= (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0
\end{aligned} \tag{A.12}$$

donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$. El óptimo global factible se localiza en x^*=(14.09500000000000064, 0.8429607892154795668), con $f(x^*)$=-6961.81387558015. Ambas restricciones son activas.

g07

Minimizar:

$$\begin{aligned}
f(\vec{x}) = {} & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\
& + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45
\end{aligned} \tag{A.13}$$

sujeto a:

$$\begin{aligned}
g_1(\vec{x}) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\
g_2(\vec{x}) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\
g_3(\vec{x}) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\
g_4(\vec{x}) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
g_5(\vec{x}) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
g_6(\vec{x}) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\
g_7(\vec{x}) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\
g_8(\vec{x}) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0
\end{aligned} \tag{A.14}$$

donde $-10 \leq x_i \leq 10$ (i = 1,..., 10). El óptimo global factible se localiza en x^*=(2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173,

0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347), con $f(x^*)$=24.30620906818. g_1, g_2, g_3, g_4, g_5 y g_6 son restricciones activas.

g08

Minimizar:

$$f(\vec{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)} \tag{A.15}$$

sujeto a:

$$\begin{aligned} g_1(\vec{x}) &= x_1^2 - x_2 + 1 \leq 0 \\ g_2(\vec{x}) &= 1 - x_1 + (x_2-4)^2 \leq 0 \end{aligned} \tag{A.16}$$

donde $0 < x_1 \leq 10$ y $0 < x_2 \leq 10$. El óptimo global factible se localiza en x^*=(1.22797135260752599, 4.24537336612274885) con $f(x^*)$=-0.0958250414180359.

g09

Minimizar:

$$\begin{aligned} f(\vec{x}) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 \\ + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \end{aligned} \tag{A.17}$$

sujeto a:

$$\begin{aligned} g_1(\vec{x}) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\ g_2(\vec{x}) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\ g_3(\vec{x}) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0 \\ g_4(\vec{x}) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \end{aligned} \tag{A.18}$$

donde -10 $\leq x_i \leq$ 10 for (i = 1,..., 7). El óptimo global factible se localiza en: x^*=(2.33049935147405174, 1.95137236847114592, -0.477541399510615805, 4.36572624923625874, -0.624486959100388983, 1.03813099410962173, 1.5942266780671519) con $f(x^*)$=680.630057374402. g_1 y g_4 son restricciones activas

g10

Minimizar:

$$f(\vec{x}) = x_1 + x_2 + x_3 \tag{A.19}$$

sujeto a:

$$\begin{aligned} g_1(\vec{x}) &= -1 + 0.0025(x_4 + x_6) \leq 0 \\ g_2(\vec{x}) &= -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\ g_3(\vec{x}) &= -1 + 0.01(x_8 - x_5) \leq 0 \\ g_4(\vec{x}) &= -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0 \\ g_5(\vec{x}) &= -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \\ g_6(\vec{x}) &= -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0 \end{aligned} \tag{A.20}$$

donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ (i=2, 3) y $10 \leq x_i \leq 1000$ (i=4,..., 8). El óptimo global factible se localiza en: x^*=(579.306685017979589, 1359.97067807935605,

5109.97065743133317, 182.01769963061534, 295.601173702746792, 217.982300369384632, 286.41652592786852, 395.601173702746735) con $f(x^*)$=7049.24802052867. g_1, g_2 y g_3 son restricciones activas..

g11

Minimizar:

$$f(\vec{x}) = x_1^2 + (x_2 - 1)^2 \tag{A.21}$$

sujeto a:

$$h(\vec{x}) = x_2 - x_1^2 = 0 \tag{A.22}$$

donde -1 $\leq x_1 \leq$ 1 y -1 $\leq x_2 \leq$ 1. El óptimo global factible se localiza en: x^*=($\pm$ 0.707036070037170616, 0.500000004333606807) con $f(x^*)$=0.7499.

g12

Minimizar:

$$f(\vec{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100 \tag{A.23}$$

sujeto a:

$$g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0 \tag{A.24}$$

donde $0 \leq x_i \leq 10$ (i=1, 2, 3) y p, q, r=1, 2,..., 9. La región factible del espacio de búsqueda se compone de 9^3 esferas disjuntas. Un punto (x_1, x_2, x_3) es factible sí y sólo sí existen p, q, r tal que la igualdad de arriba se cumple. El óptimo global factible se localiza en: x^*=(5, 5, 5) con $f(x^*)$= -1.

g13

Minimizar:

$$f(\vec{x}) = \exp^{x_1 x_2 x_3 x_4 x_5} \tag{A.25}$$

sujeto a:

$$\begin{aligned} h_1(\vec{x}) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ h_2(\vec{x}) &= x_2 x_3 - 5 x_4 x_5 = 0 \\ h_3(\vec{x}) &= x_1^3 + x_2^3 + 1 = 0 \end{aligned} \tag{A.26}$$

donde -2.3$\leq x_i \leq$2.3 (i=1, 2) y -3.2$\leq x_i \leq$3.2 (i=3, 4, 5). El óptimo global factible está en: x^*=(-1.71714224003, 1.59572124049468, 1.8272502406271, -0.763659881912867, -0.76365986736498) con $f(x^*)$=0.053941514041898.

g14

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right) \tag{A.27}$$

sujeto a:

$$\begin{aligned} h_1(\vec{x}) &= x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0 \\ h_2(\vec{x}) &= x_4 + 2x_5 + x_6 + x_7 - 1 = 0 \\ h_3(\vec{x}) &= x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0 \end{aligned} \tag{A.28}$$

donde $0< x_i \leq 10$ (i=1,...,10), y c=(-6.089, -17.164, -34.054, -5.914, -24.721, -14.986, -24.1, -10.708, -26.662, -22.179). La mejor solución conocida está en x^*=(0.0406684113216282, 0.147721240492452, 0.783205732104114, 0.00141433931889084, 0.485293636780388, 0.000693183051556082, 0.0274052040687766, 0.0179509660214818, 0.0373268186859717, 0.0968844604336845) con $f(x^*)$=-47.7648884594915.

g15

Minimizar:

$$f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \tag{A.29}$$

sujeto a:

$$\begin{aligned} h_1(\vec{x}) &= x_1^2 + x_2^2 + x_3^2 - 25 = 0 \\ h_2(\vec{x}) &= 8x_1 + 14x_2 + 7x_3 - 56 = 0 \end{aligned} \tag{A.30}$$

donde $0\leq \ x_i \ \leq 10$ (i = 1, 2, 3). La mejor solución conocida está en x^*=(3.51212812611795133, 0.216987510429556135, 3.55217854929179921) con $f(x^*)$=961.715022289961.

g16

Minimizar:

$$\begin{aligned} f(\vec{x}) =& 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} \\ &+ 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17} \end{aligned} \tag{A.31}$$

sujeto a:

$$\begin{aligned} g_1(\vec{x}) =& \frac{0.28}{0.72}y_5 - y_4 \leq 0 \\ g_2(\vec{x}) =& x_3 - 1.5x_2 \leq 0 \\ g_3(\vec{x}) =& 3496\frac{y_2}{c_{12}} - 21 \leq 0 \\ g_4(\vec{x}) =& 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0 \\ g_5(\vec{x}) =& 213.1 - y_1 \leq 0 \\ g_6(\vec{x}) =& y_1 - 405.23 \leq 0 \\ g_7(\vec{x}) =& 17.505 - y_2 \leq 0 \\ g_8(\vec{x}) =& y_2 - 1053.6667 \leq 0 \\ g_9(\vec{x}) =& 11.275 - y_3 \leq 0 \\ g_{10}(\vec{x}) =& y_3 - 35.03 \leq 0 \\ g_{11}(\vec{x}) =& 214.228 - y_4 \leq 0 \\ g_{12}(\vec{x}) =& y_4 - 665.585 \leq 0 \\ g_{13}(\vec{x}) =& 7.458 - y_5 \leq 0 \\ g_{14}(\vec{x}) =& y_5 - 584.463 \leq 0 \\ g_{15}(\vec{x}) =& 0.961 - y_6 \leq 0 \\ g_{16}(\vec{x}) =& y_6 - 265.916 \leq 0 \\ g_{17}(\vec{x}) =& 1.612 - y_7 \leq 0 \end{aligned}$$

$$
\begin{aligned}
g_{18}(\vec{x}) =& y_7 - 7.046 \leq 0 \\
g_{19}(\vec{x}) =& 0.146 - y_8 \leq 0 \\
g_{20}(\vec{x}) =& y_8 - 0.222 \leq 0 \\
g_{21}(\vec{x}) =& 107.99 - y_9 \leq 0 \\
g_{22}(\vec{x}) =& y_9 - 273.366 \leq 0 \\
g_{23}(\vec{x}) =& 922.693 - y_{10} \leq 0 \\
g_{24}(\vec{x}) =& y_{10} - 1286.105 \leq 0 \\
g_{25}(\vec{x}) =& 926.832 - y_{11} \leq 0 \\
g_{26}(\vec{x}) =& y_{11} - 1444.046 \leq 0 \\
g_{27}(\vec{x}) =& 18.766 - y_{12} \leq 0 \\
g_{28}(\vec{x}) =& y_{12} - 537.141 \leq 0 \\
g_{29}(\vec{x}) =& 1072.163 - y_{13} \leq 0 \\
g_{30}(\vec{x}) =& y_{13} - 3247.039 \leq 0 \\
g_{31}(\vec{x}) =& 8961.448 - y_{14} \leq 0 \\
g_{32}(\vec{x}) =& y_{14} - 26844.086 \leq 0 \\
g_{33}(\vec{x}) =& 0.063 - y_{15} \leq 0 \\
g_{34}(\vec{x}) =& y_{15} - 0.386 \leq 0 \\
g_{35}(\vec{x}) =& 71084.33 - y_{16} \leq 0 \\
g_{36}(\vec{x}) =& -140000 + y_{16} \leq 0 \\
g_{37}(\vec{x}) =& 2802713 - y_{17} \leq 0 \\
g_{38}(\vec{x}) =& y_{17} - 12146108 \leq 0
\end{aligned}
\tag{A.32}
$$

donde:

$$
\begin{aligned}
y_1 =& x_2 + x_3 + 41.6 \\
c_1 =& 0.024x_4 - 4.62 \\
y_2 =& \frac{12.5}{c_1} + 12 \\
c_2 =& 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1 \\
c_3 =& 0.052x_1 + 78 + 0.002377y_2x_1 \\
y_3 =& \frac{c_2}{c_3} \\
y_4 =& 19y_3 \\
c_4 =& 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3 \\
c_5 =& 100x_2 \\
c_6 =& x_1 - y_3 - y_4 \\
c_7 =& 0.950 - \frac{c_4}{c_5} \\
y_5 =& c_6c_7 \\
y_6 =& x_1 - y_5 - y_4 - y_3 \\
c_8 =& (y_5 + y_4)0.995
\end{aligned}
$$

$$
\begin{aligned}
y_7 =& \frac{c_8}{y_1} \\
y_8 =& \frac{c_8}{3798} \\
c_9 =& y_7 - \frac{0.0663 y_7}{y_8} - 0.3153 \\
y_9 =& \frac{96.82}{c_9} + 0.321 y_1 \\
y_{10} =& 1.29 y_5 + 1.258 y_4 + 2.29 y_3 + 1.71 y_6 \\
y_{11} =& 1.71 x_1 - 0.452 y_4 + 0.580 y_3 \\
c_{10} =& \frac{12.3}{752.3} \\
c_{11} =& (1.75 y_2)(0.995 x_1) \\
c_{12} =& 0.995 y_{10} + 1998 \\
y_{12} =& c_{10} x_1 + \frac{c_{11}}{c_{12}} \\
y_{13} =& c_{12} - 1.75 y_2 \\
y_{14} =& 3623 + 64.4 x_2 + 58.4 x_3 + \frac{146312}{y_9 + x_5} \\
c_{13} =& 0.995 y_{10} + 60.8 x_2 + 48 x_4 - 0.1121 y_{14} - 5095 \\
y_{15} =& \frac{y_{13}}{c_{13}} \\
y_{16} =& 148000 - 331000 y_{15} + 40 y_{13} - 61 y_{15} y_{13} \\
c_{14} =& 2324 y_{10} - 28740000 y_2 \\
y_{17} =& 14130000 - 1328 y_{10} - 531 y_{11} + \frac{c_{14}}{c_{12}} \\
c_{15} =& \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52} \\
c_{16} =& 1.104 - 0.72 y_{15} \\
c_{17} =& y_9 + x_5
\end{aligned}
$$

y donde $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ y $25 \leq x_5 \leq 84.1988$. La mejor solución conocida está en x^*=(705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832) con $f(x^*)$=-1.90515525853479.

g17

Minimizar:

$$f(\vec{x}) = f_1(x_1) + f_2(x_2) \tag{A.33}$$

donde

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

sujeto a:

$$\begin{aligned}
h_1(\vec{x}) =& -x_1 + 300 - \frac{x_3 x_4}{131{,}078}\cos(1{,}48477 - x_6) + \frac{0{,}90798 x_3^2}{131{,}078}\cos(1{,}47588) = 0 \\
h_2(\vec{x}) =& -x_2 - \frac{x_3 x_4}{131{,}078}\cos(1{,}48477 + x_6) + \frac{0{,}90798 x_4^2}{131{,}078}\cos(1{,}47588) = 0 \\
h_3(\vec{x}) =& -x_5 - \frac{x_3 x_4}{131{,}078}\sin(1{,}48477 + x_6) + \frac{0{,}90798 x_4^2}{131{,}078}\sin(1{,}47588) = 0 \\
h_4(\vec{x}) =& 200 - \frac{x_3 x_4}{131{,}078}\sin(1{,}48477 - x_6) + \frac{0{,}90798 x_3^2}{131{,}078}\sin(1{,}47588) = 0
\end{aligned} \tag{A.34}$$

donde $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$, y $0 \leq x_6 \leq 0.5236$. La mejor solución conocida está en x^* = (201.784467214523659, 99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.0731482312084287128) con $f(x^*)$=8853.53967480648.

g18

Minimizar:

$$f(\vec{x}) = -0{,}5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 * x_8 - x_6 x_7) \tag{A.35}$$

sujeto a:

$$\begin{aligned}
g_1(\vec{x}) =& x_3^2 + x_4^2 - 1 \leq 0 \\
g_2(\vec{x}) =& x_9^2 - 1 \leq 0 \\
g_3(\vec{x}) =& x_5^2 + x_6^2 - 1 \leq 0 \\
g_4(\vec{x}) =& x_1^2 + (x_2 - x_9)^2 - 1 \leq 0 \\
g_5(\vec{x}) =& (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0 \\
g_6(\vec{x}) =& (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0 \\
g_7(\vec{x}) =& (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0 \\
g_8(\vec{x}) =& (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0 \\
g_9(\vec{x}) =& x_7^2 + (x_8 - x_9)^2 - 1 \leq 0 \\
g_{10}(\vec{x}) =& x_2 x_3 - x_1 x_4 \leq 0 \\
g_{11}(\vec{x}) =& -x_3 x_9 \leq 0 \\
g_{12}(\vec{x}) =& x_5 x_9 \leq 0 \\
g_{13}(\vec{x}) =& x_6 x_7 - x_5 x_8 \leq 0
\end{aligned} \tag{A.36}$$

donde $-10 \leq x_i \leq 10$ (i=1,...,8) y $0 \leq x_9 \leq 20$. La mejor solución conocida está en x^*= (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542) con $f(x^*)$=-0.866025403784439.

g19

Minimizar:

$$f(\vec{x}) = \sum_{j=1}^{5}\sum_{i=1}^{5} c_{ij}x_{(10+i)}x_{(10+j)} + 2\sum_{j=1}^{5} d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i \tag{A.37}$$

sujeto a:

$$g_j(\vec{x}) = -2\sum_{i=1}^{5} c_{ij}x_{(10+i)} - 3d_j x_{(10+j)}^2 - e_j + \sum_{i=1}^{10} a_{ij}x_i \leq 0 \quad j = 1,\ldots,5 \tag{A.38}$$

donde $\vec{b}$=[-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1] y los datos faltantes están en la Tabla A.1. $0 \leq x_i \leq 10$ (i=1,...,15). La mejor solución conocida está en x^*=(1.6699134132629134 4E-17, 3.95378229282456509E-16, 3.94599045143233784, 1.06036597479721211E-16, 3.2831773458454161, 9.99999999999999822, 1.12829414671605333E-17, 1.2026194599794709E-17, 2.50706276000769697E-15, 2.24624122987970677E-15, 0.370764847417013987, 0.278456024942955571, 0.523838487672241171, 0.388620152510322781, 0.298156764974678579) con $f(x^*) = 32.6555929502463$.

Tabla A.1: Conjunto de datos para el problema g19.

j	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
c_{1j}	30	-20	-10	32	-10
c_{2j}	-20	39	-6	-31	32
c_{3j}	-10	-6	10	-6	-10
c_{4j}	32	-31	-6	39	-20
c_{1j}	-10	32	-10	-20	30
d_j	4	8	10	6	2
a_{1j}	-16	2	0	1	0
a_{2j}	0	-2	0	0.4	2
a_{3j}	-3.5	0	2	0	0
a_{4j}	0	-2	0	-4	-1
a_{5j}	0	-9	-2	1	-2.8
a_{6j}	2	0	-4	0	0
a_{7j}	-1	-1	-1	-1	-1
a_{8j}	-1	-2	-3	-2	-1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

g20

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{24} a_i x_i \tag{A.39}$$

sujeto a:

$$
\begin{aligned}
g_i(\vec{x}) &= \frac{x_i + x_{(i+12)}}{\sum_{j=1}^{24} x_j + e_i} \leq 0 && i = 1,2,3 \\
g_i(\vec{x}) &= \frac{x_{(i+3)} + x_{(i+15)}}{\sum_{j=1}^{24} x_j + e_i} \leq 0 && i = 4,5,6 \\
h_i(\vec{x}) &= \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_{(i+12)} \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0 \quad i = 1,\ldots,12 \\
h_{13}(\vec{x}) &= \sum_{i=1}^{24} x_i - 1 = 0 \\
h_{14}i(\vec{x}) &= \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1{,}671 = 0
\end{aligned}
\tag{A.40}
$$

donde k=(0.7302)(503)($\frac{14,7}{40}$) y el conjunto de datos se detalla en la Tabla A.2. 0$\leq x_i \leq$10 (i=1,...,24). La mejor solución conocida está en x^*=(1.28582343498528086E-18, 4.83460302526130664E-34, 0, 0, 6.30459929660781851E-18, 7.57192526201145068E-34, 5.03350698372840437E-34, 9.28268079616618064E-34, 0, 1.76723384525547359E-17, 3.55686101822965701E-34, 2.99413850083471346E-34, 0.158143376337580827, 2.29601774161699833E-19, 1.06106938611042947E-18, 1.31968344319506391E-18, 0.530902525044209539, 0, 2.89148310257773535E-18, 3.34892126180666159E-18, 0, 0.310999974151577319, 5.41244666317833561E-05, 4.84993165246959553E-16). Esta solución es un poco infactible y ninguna solución factible se ha reportado hasta ahora.

g21

Minimizar:

$$f(\vec{x}) = x_1 \tag{A.41}$$

sujeto a:

$$
\begin{aligned}
g_1(\vec{x}) &= -x_1 + 35x_2^{0,6} + 35x_3^{0,6} \leq 0 \\
h_1(\vec{x}) &= -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0 \\
h_2(\vec{x}) &= 100x_2 + 155{,}365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536{,}5 = 0 \\
h_3(\vec{x}) &= -x_5 + \ln(-x_4 + 900) = 0 \\
h_4(\vec{x}) &= -x_6 + \ln(x_4 + 300) = 0 \\
h_5(\vec{x}) &= -x_7 + \ln(-2x_4 + 700) = 0
\end{aligned}
\tag{A.42}
$$

donde 0$\leq x_1 \leq$1000, 0$\leq x_2, x_3 \leq$40, 100$\leq x_4 \leq$300, 6.3$\leq x_5 \leq$6.7, 5.9$\leq x_6 \leq$6.4 y 4.5$\leq x_7 \leq$6.25. La mejor solución conocida está en x^*=(193.724510070034967, 5.569441315533684 33E-27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451) con $f(x^*)$=193.724510070035.

g22

Minimizar:

$$f(\vec{x}) = x_1 \tag{A.43}$$

Tabla A.2: Conjunto de datos para el problema g20.

i	a_i	b_i	c_i	d_i	e_i
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

sujeto a:

$$
\begin{aligned}
g_1(\vec{x}) &= -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0 \\
h_1(\vec{x}) &= x_5 - 100000x_8 + 1E7 = 0 \\
h_2(\vec{x}) &= x_6 + 100000x_8 - 100000x_9 = 0 \\
h_3(\vec{x}) &= x_7 + 100000x_9 - 5E7 = 0 \\
h_4(\vec{x}) &= x_5 + 100000x_{10} - 3.3E7 = 0 \\
h_5(\vec{x}) &= x_6 + 100000x_{11} - 4.4E7 = 0 \\
h_6(\vec{x}) &= x_7 + 100000x_{12} - 6.6E7 = 0 \\
h_7(\vec{x}) &= x_5 - 120x_2x_{13} = 0 \\
h_8(\vec{x}) &= x_6 - 80x_3x_{14} = 0 \\
h_9(\vec{x}) &= x_7 - 40x_4x_{15} = 0 \\
h_{10}(\vec{x}) &= x_8 - x_{11} + x_{16} = 0 \\
h_{11}(\vec{x}) &= x_9 - x_{12} + x_{17} = 0 \\
h_{12}(\vec{x}) &= -x_{18} + \ln(x_{10} - 100) = 0 \\
h_{13}(\vec{x}) &= -x_{19} + \ln(-x_8 + 300) = 0 \\
h_{14}(\vec{x}) &= -x_{20} + \ln(x_{16}) = 0 \\
h_{15}(\vec{x}) &= -x_{21} + \ln(-x_9 + 400) = 0
\end{aligned}
\tag{A.44}
$$

$$h_{16}(\vec{x}) = -x_{22} + \ln(x_{17}) = 0$$
$$h_{17}(\vec{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$
$$h_{18}(\vec{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$
$$h_{19}(\vec{x}) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$$

donde $0 \leq x_1 \leq 20000$, $0 \leq x_2, x_3, x_4 \leq 1^6$, $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13}, x_{14}, x_{15} \leq 500$, $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$. La mejor solución conocida está en x^*=(236.430975504001054, 135.82847151732463, 204.818152544824585, 6446.54654059436416, 3007540.83940215595, 4074188.65771341929, 32918270.5028952882, 130.075408394314167, 170.817294970528621, 299.924591605478554, 399.258113423595205, 330.817294971142758, 184.51831230897065, 248.64670239647424, 127.658546694545862, 269.182627528746707, 160.000016724090955, 5.29788288102680571, 5.13529735903945728, 5.59531526444068827, 5.43444479314453499, 5.07517453535834395) con $f(x^*)$=236.430975504001.

g23

Minimizar:

$$f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7) \tag{A.45}$$

sujeto a:

$$\begin{aligned} g_1(\vec{x}) = &= x_9x_3 + 0{,}02x_6 - 0{,}025x_5 \leq 0 \\ g_2(\vec{x}) = &x_9x_4 + 0{,}02x_7 - 0{,}015x_8 \leq 0 \\ h_1(\vec{x}) = &x_1 + x_2 - x_3 - x_4 = 0 \\ h_2(\vec{x}) = &0{,}03x_1 + 0{,}01x_2 - x_9(x_3 + x_4) = 0 \\ h_3(\vec{x}) = &x_3 + x_6 - x_5 = 0 \\ h_4(\vec{x}) = &x_4 + x_7 - x_8 = 0 \end{aligned} \tag{A.46}$$

donde $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ y $0.01 \leq x_9 \leq 0.03$. La mejor solución conocida está en x^*=(0.00510000000000259465, 99.9947000000000514, 9.01920162996045897E-18, 99.9999000000000535, 0.000100000000027086086, 2.75700683389584542E-14, 99.9999999999999574, 2000.0100000100000100008) con $f(x^*)$ = -400.055099999999584.

g24

Minimizar:

$$f(\vec{x}) = -x_1 - x_2 \tag{A.47}$$

sujeto a:

$$\begin{aligned} g_1 = &- 2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0 \\ g_2 = &- 4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0 \end{aligned} \tag{A.48}$$

donde $0 \leq x_1 \leq 3$ y $0 \leq x_2 \leq 4$. El mínimo global factible está en x^*= (2.32952019747762, 3.17849307411774) con $f(x^*)$=-5.50801327159536. Este problema tiene una región factible compuesta de 2 sub-regiones desconectadas.

Apéndice B

Problemas de Diseño en Ingeniería

Problema de la viga soldada

Minimizar:

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2); \tag{B.1}$$

sujeto a:

$$\begin{aligned}
&g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leq 0\\
&g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leq 0\\
&g_3(\vec{x}) = x_1 - x_4 \leq 0\\
&g_4(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5.0 \leq 0\\
&g_5(\vec{x}) = 0.125 - x_1 \leq 0\\
&g_6(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leq 0\\
&g_7(\vec{x}) = P - P_c \leq 0\\
&0.1 \leq x_1, x_4 \leq 2\\
&0.1 \leq x_2, x_3 \leq 10
\end{aligned} \tag{B.2}$$

donde:

$$\begin{aligned}
\tau(\vec{x}) =& \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}\\
P_c(\vec{x}) =& \frac{4.013E}{L^2}\sqrt{\frac{x_3^2x_4^6}{36}}\left(1 - \left(\frac{x_3}{2L}\right)\sqrt{\frac{E}{4G}}\right)\\
\sigma(\vec{x}) =& \frac{6PL}{x_4x_3^2}, \quad \delta(\vec{x}) = \frac{4PL^3}{Ex_3^3x_4}\\
J =& 2\left(\sqrt{2}x_1x_2\left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right)\right)\\
R =& \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}
\end{aligned}$$

$$
\begin{aligned}
M =& P\left(L+\frac{x_2}{2}\right)\\
\tau' =& \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}, \quad \delta_{max} = 0{,}25in\\
\tau_{max} =& 13600\ psi, \quad \sigma_{max} = 30000\ psi, \quad P = 6000\ lb\\
L =& 14\ in \quad E = 30\text{E}6\ psi \quad G = 12\text{E}6\ psi
\end{aligned}
$$

Diseño de una vasija de presión

Minimizar:

$$f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{B.3}$$

sujeto a:

$$
\begin{aligned}
g_1(\vec{x}) &= \text{-}x_1 + 0.0193x_3 \leq 0\\
g_2(\vec{x}) &= \text{-}x_2 + 0.00954x_3 \leq 0\\
g_3(\vec{x}) &= \pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0\\
g_4(\vec{x}) &= x_4 - 240 \leq 0
\end{aligned}
\tag{B.4}
$$

donde $1\times0.0625 \leq x_1, x_2 \leq 99\times0.0625$ y $10.0 \leq x_3, x_4 \leq 200.0$

Problema del resorte de tensión/compresión

Minimizar:

$$f(\vec{x}) = (x_3+2)x_2x_1^2 \tag{B.5}$$

sujeto a:

$$
\begin{aligned}
g_1(\vec{x}) &= 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0\\
g_2(\vec{x}) &= \frac{4x_2^2 - x_1x_2}{12566x_2x_1^3 - x_1^4} + \frac{1}{5108x_1^2} - 1 \leq 0\\
g_3(\vec{x}) &= 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0\\
g_4(\vec{x}) &= \frac{x_2+x_1}{1.5} - 1 \leq 0
\end{aligned}
\tag{B.6}
$$

donde $0.05 \leq x_1 \leq 2.0$, $0.25 \leq x_2 \leq 1.3$ y $2 \leq x_3 \leq 15$.

Problema del freno de embrague

Minimizar:

$$f(\vec{x}) = \pi(x_2^2 - x_1^2)x_3(x_5+1)\rho \tag{B.7}$$

sujeto a:

$$\begin{aligned} g_1(\vec{x}) &= x_1 + \Delta R - x_2 \leq 0 \\ g_2(\vec{x}) &= (x_5 + 1)(x_3 + \delta) - L_{max} \leq 0 \\ g_3(\vec{x}) &= p_{rz} - p_{max} \leq 0 \\ g_4(\vec{x}) &= ((p_{rz} \times V_{sr})/1000) - (p_{max} \times V_{srmax}) \leq 0 \\ g_5(\vec{x}) &= (V_{sr}/1000) - V_{srmax} \leq 0 \\ g_6(\vec{x}) &= T - T_{max} \leq 0 \\ g_7(\vec{x}) &= (s \times M_s) - M_h \leq 0 \\ g_8(\vec{x}) &= -T \leq 0 \end{aligned} \tag{B.8}$$

donde

$$\begin{aligned} \rho &= 7.8E-6 & \Delta R &= 20 & n &= 250 \\ L_{max} &= 30 & \delta &= 0.5 & T_{max} &= 15 \\ p_{max} &= 1 & V_{srmax} &= 10 & \mu &= 0.5 \\ M_s &= 40 & I_z &= 55 & s &= 1.5 \\ A &= \pi(x_2^2 - x_1^2) & p_{rz} &= \frac{x_4}{A} & R_{sr} &= \frac{2}{3}\frac{x_2^3 - x_1^3}{x_2^2 - x_1^2} \\ w &= \frac{\pi n}{30} & M_f &= 3 & M_h &= \frac{2}{3}\mu x_4 x_5 \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2}\frac{1}{1000} \\ V_{sr} &= \frac{\pi R_{sr} n}{30} & T &= \frac{I_z w}{M_h + M_f} \end{aligned}$$

donde $60 \leq x_1 \leq 80$, $90 \leq x_2 \leq 110$, $1 \leq x_3 \leq 3$, $0 \leq x_4 \leq 1000$ y $2 \leq x_5 \leq 9$. Además x_1, x_2 y $x_3 \in \mathbb{N}$, $x_3 = 0.5n_1$ y $x_4 = 10n_2$, donde n_1 y $n_2 \in \mathbb{N}$.

Problema de seguimiento de una trayectoria no alineada, con sincronizacion previa

Minimizar:

$$f(\vec{p}) = \sum_{i=1}^{N} [(C_{xd}^i - C_x^i)^2 (C_{yd}^i - C_y^i)^2] \tag{B.9}$$

sujeto a:

$$\begin{aligned} g_1(\vec{p}) &= r_1 + r_2 - r_3 - r_4 \leq 0 \\ g_2(\vec{p}) &= r_2 - r_3 \leq 0 \\ g_3(\vec{p}) &= r_3 - r_4 \leq 0 \\ g_4(\vec{p}) &= r_4 - r_1 \leq 0 \end{aligned} \tag{B.10}$$

donde la trayectoria es no alineada y está dada por los siguientes puntos;

$$\Omega = \{(3,3), (2.759, 3.363), (2.372, 3.663), (1.890, 3.862), (1.355, 3.943)\}$$

la sincronización prescrita está dada por los siguientes ángulos:

$$\theta_2^i = \left\{ \frac{2\pi}{12}, \frac{3\pi}{12}, \frac{4\pi}{12}, \frac{5\pi}{12}, \frac{6\pi}{12} \right\}$$

y los valores de las variables de diseño son θ_0, x_0 y y_0 valen cero, $0 \leq r_1, r_2, r_3, r_4 \leq 50$ y $-50 \leq r_{cx}, r_{cy} \leq 50$.

Referencias

[1] A. Arias, C. Coello, and E. Mezura. Multi-objective airfoil shape optimization using a multiple-surrogate approach. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[2] D. V. Arnold. *Evolution Strategies in Noisy Environments - A Survey of Existing Work*, pages 239–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[3] D. V. Arnold and H.-G. Beyer. Noisy optimization with evolution strategies. Technical report, SIAM Journal on Optimization, 2002.

[4] S. Bagheri, W. Konen, M. Emmerich, and T. Bäck. Solving the g-problems in less than 500 iterations: Improved efficient constrained optimization by surrogate modeling and adaptive parameter control. *CoRR*, 2015.

[5] A. Basudhar, C. Dribusch, S. Lacaze, and S. Missoum. Constrained efficient global optimization with support vector machines. *Structural and Multidisciplinary Optimization*, 46(2):201–221, 2012.

[6] K.S. Bhattacharjee, A. Isaacs, and T. Ray. Multiobjective optimization using an evolutionary algorithm embedded with multiple spatially distributed surrogates. *Multi-objective Optimization: Techniques and Applications in Chemical Engineering*, pages 135–152, 2016.

[7] A.E.I. Brownlee and J.A. Wright. Constrained, mixed-integer and multi-objective optimisation of building designs by nsga-ii with fitness approximation. *Applied Soft Computing*, 33:114–126, 2015.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[9] C. A. Coello-Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11):1245 – 1287, 2002.

[10] S. Das, S. S. Mullick, and P.N. Suganthan. Recent advances in differential evolution - an updated survey. *Swarm and Evolutionary Computation*, 27(Supplement C):1 – 30, 2016.

[11] R. Datta and R.G. Regis. A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Syst. Appl.*, 57:270–284, 2016.

[12] K. Deb. An efficient constraint handling method for genetic algorithms. In *Computer Methods in Applied Mechanics and Engineering*, pages 311–338, 1998.

[13] K. Deb. *Optimization for Engineering Design: Algorithms and Examples*. Prentice-Hall of India, 2000.

[14] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel–assisted evolution strategies. In J. J. M. Guervós, P. Adamidis, H-G Beyer, H-P Schwefel, and J-L Fernández-Villacañas, editors, *Parallel Problem Solving from Nature — PPSN VII: 7th International Conference Granada, Spain, September 7–11, 2002 Proceedings*, pages 361–370, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[15] A. P. Engelbrecht. *Computational Intelligence: An Introduction.* Wiley Publishing, 2nd edition, 2007.

[16] L.J. Fogel. Autonomous automata. (4):14–19, 1962.

[17] L.G. Fonseca, H.J.C. Barbosa, and A.C.C. Lemonge. A similarity-based surrogate model for enhanced performance in genetic algorithms. *OPSEARCH*, 46(1):89–107, 2009.

[18] A. Forrester, A.Ã. Sobester, and A. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide.* Wiley, 2008.

[19] A.I.J. Forrester and A.J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3):50–79, January 2009.

[20] C.K. Goh, D. Lim, L. Ma, Y.S. Ong, and P.S. Dutta. A surrogate-assisted memetic co-evolutionary algorithm for expensive constrained optimization problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 744–749, June 2011.

[21] W. Gong, A. Zhou, and Z. Cai. A multioperator search strategy based on cheap surrogate models for evolutionary optimization. *IEEE Trans. Evolutionary Computation*, 19(5):746–758, 2015.

[22] N. B. Guedria. Improved accelerated pso algorithm for mechanical engineering optimization problems. *Appl. Soft. Comput.*, 40:455–467, 2016.

[23] S. B. Hamida and M. Schoenauer. Aschea: new results using adaptive segregational constraint handling. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 884–889, May 2002.

[24] Abdel-Karim S.O. Hassan, Ahmed S.A. Mohamed, Azza A. Rabie, and Ahmed S. Etman. A novel surrogate-based approach for optimal design of electromagnetic-based circuits. *Engineering Optimization*, 48(2):185–198, 2016.

[25] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[26] M.S. Innocente, S.M. Bastos, J. Sienz, and H.M. Davies. Particle swarm algorithm with adaptive constraint handling and integrated surrogate model for the management of petroleum fields. *Appl. Soft Comput.*, 34(C):463–484, sep 2015.

[27] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[28] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(1):61–70, 2011.

[29] Y. Jin, S. Oh, and M. Jeon. Incremental approximation of nonlinear constraint functions for evolutionary constrained optimization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, July 2010.

[30] Y. Jin, M. Olhofer, and B. SendhoffS. A framework for evolutionary optimization with approximate fitness functions. In *Evolutionary Computation, IEEE Transactions on*, volume 6, pages 481–494, October 2002.

[31] J.M.Parr, A.I.J. Forrester, A.J.Keane, and C.M.E. Holden. Enhancing infill sampling criteria for surrogate-based constrained optimization. *J. Comput. Meth. in Science and Engineering*, 12(1-2):25–45, 2012.

[32] D. R. Jones. Large-scale multi-disciplinary mass optimization in the auto industry. August 2008. Presented at the Modeling and Optimization: Theory and Applications Conference (MOPTA 2008).

[33] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, Oct 1993.

[34] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, Dec 1998.

[35] J. A. Khan. *Research Methodology*. APH Publishing Corporation, 2011.

[36] J. P. C. Kleijnen, W. van Beers, and I. van Nieuwenhuyse. Expected improvement in efficient global optimization through bootstrapped kriging. *Journal of Global Optimization*, 54(1):59–73, Sep 2012.

[37] P. Koch, S. Bagheri, W. Konen, C. Foussette, P. Krause, and T. Bäck. A new repair method for constrained optimization. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 273–280, New York, NY, USA, 2015. ACM.

[38] C. R. Kothari. *Research Methodology: Methods and Techniques*. New Age International (P) Limited, 2004.

[39] J. R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'89, pages 768–774, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[40] O. Kramer, A. Barthelmes, and G. Rudolph. Surrogate constraint functions for cma evolution strategies. In Bärbel Mertsching, Marcus Hund, and Zaheer Aziz, editors, *KI 2009: Advances in Artificial Intelligence*, volume 5803 of *Lecture Notes in Computer Science*, pages 169–176. Springer Berlin Heidelberg, 2009.

[41] J.J. Liang, T. Runarsson, E. Mezura, M. Clerc, P. Suganthan, C. A. Coello, and K. Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapure, December 2005.

[42] Z. Liu and Q. Hui. A numerical gradient based technique and directed neighborhood structure for constrained particle swarm optimization. In *2013 IEEE American Control Conference*, pages 4783–4788, 2013.

[43] I. G. Loshchilov. *Surrogate-Assisted Evolutionary Algorithms*. PhD thesis, Université Paris-Sud 11, 2013.

[44] H. Lu and W. Chen. Dynamic-objective particle swarm optimization for constrained optimization problems. *Journal of Combinatorial Optimization*, 12(4):409–419, 2006.

[45] R. Mallipeddi and M. Lee. An evolving surrogate model-based differential evolution algorithm. *Applied Soft Computing*, 34(Supplement C):770 – 787, 2015.

[46] E. Mezura-Montes and C. A. Coello-Coello. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[47] E. Mezura-Montes and C.A. Coello-Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *Evolutionary Computation, IEEE Transactions on*, 9(1):1–17, Feb 2005.

[48] E. Mezura-Montes, M.E. Miranda-Varela, and R.C. Gómez-Ramón. Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences*, 180(22):4223–4262, 2010.

[49] Efrén Mezura-Montes and Omar Cetina-Domínguez. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. *Applied Mathematics and Computation*, 218(22):10943–10973, 2012.

[50] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, March 1996.

[51] M.E. Miranda-Varela and E. Mezura-Montes. Surrogate-assisted differential evolution with an adaptive evolution control based on feasibility to solve constrained optimization problems. In M. Pant, K. Deep, J. C. Bansal, A. Nagar, and K. N. Das, editors, *Proceedings of Fifth International Conference on Soft Computing for Problem Solving: SocProS 2015, Volume 1*, pages 809–822. Springer Singapore, Singapore, 2016.

[52] I. Newman and C.R. Benz. *Qualitative-quantitative Research Methodology: Exploring the Interactive Continuum*. Southern Illinois University Press, 1998.

[53] Y.S. Ong, P.B. Nair, and A.J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.

[54] J. Parr, C.M.E. Holden, A.I.J. Forrester, and A.J. Keane. Review of efficient surrogate infill sampling criteria with constraint handling. In *2nd International Conference on Engineering Optimization*, pages 1–10, September 2010.

[55] A. P. Piotrowski. Adaptive memetic differential evolution with global and local neighborhood-based mutation operators. *Inf. Sci.*, 241:164–194, August 2013.

[56] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

[57] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucher. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41:1–28, 2005.

[58] R. G. Regis. Trust regions in surrogate-assisted evolutionary programming for constrained expensive black-box optimization. In Rituparna Datta and Kalyanmoy Deb, editors, *Evolutionary Constrained Optimization*, Infosys Science Foundation Series, pages 51–94. Springer India, 2015.

[59] R.G. Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014.

[60] R.G. Regis. Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. *Evolutionary Computation, IEEE Transactions on*, 18(3):326–347, June 2014.

[61] R.G. Regis. Multi-objective constrained black-box optimization using radial basis function surrogates. *J. Comput. Sci.*, 16:140–155, 2016.

[62] R.G. Regis and C.A. Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

[63] R. G. Reynolds. An introduction to cultural algorithms. In *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific Publishing Company, September 1994.

[64] T.P. Runarsson. Constrained evolutionary optimization by approximate ranking and surrogate models. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 401–410. Springer Berlin Heidelberg, 2004.

[65] T.P. Runarsson. Approximate evolution strategy using stochastic ranking. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 745–752, 2006.

[66] T.P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294, Sep 2000.

[67] M. Sasena, P. Papalambros, and P. Goovaerts. *Global optimization of problems with disconnected feasible regions via surrogate modeling.* American Institute of Aeronautics and Astronautics Inc., 2002.

[68] M.J. Sasena, P. Papalambros, and P. Goovaerts. The use of surrogate modeling algorithms to exploit disparities in function computation time within simulation-based optimization. *Dalian, China.*, pages 1–6, 2001.

[69] M.J. Sasena, P. Papalambros, and P. Goovaerts. Global optimization of problems with disconnected feasible regions via surrogate modeling. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 1–8, Sep 2002.

[70] M. Schonlau. *Computer experiments and global optimization.* PhD thesis, 1997.

[71] L. Shi and K. Rasheed. Asaga: An adaptive surrogate-assisted genetic algorithm. In Conor Ryan and Maarten Keijzer, editors, *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 1049–1056. ACM, 2008.

[72] L. Shi and K. Rasheed. A survey of fitness approximation methods applied in evolutionary algorithms. In Lim e. n. g. Hiot, Yew o. o. n. Ong, Yoel Tenne, and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation Learning and Optimization*, chapter 1, pages 3–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[73] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, 1995.

[74] T. Takahama and S. Sakai. Efficient constrained optimization by the ε constrained differential evolution with rough approximation using kernel regression. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1334–1341, June 2013.

[75] T. Takahama, S. Sakai, and N. Iwane. Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm. In S. Zhang and R. Jarvis, editors, *AI 2005: Advances in Artificial Intelligence*, volume 3809 of *Lecture Notes in Computer Science*, pages 389–400. Springer Berlin Heidelberg, 2005.

[76] Y. Tang, J. Chen, and J. Wei. A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions. *Engineering Optimization*, 45(5):557–576, 2013.

[77] J. V. Tu. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49(11):1225–1231, 1996.

[78] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[79] F. A. C. Viana, G. Venter, and V. Balabanov. An algorithm for fast optimal latin hypercube design of experiments. *International Journal for Numerical Methods in Engineering*, 82(2):135–156, 2010.

[80] D. Villanueva, R. Riche, R. Picard, and R.T. Haftka. Surrogate-based agents for constrained optimization. In *Proceedings of 14th AIAA Non-Deterministic Approaches Conference*, pages 1–17, Honolulu, États-Unis, 2012. AIAA.

[81] Y. Wang and Z. Cai. Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 16(1):117–134, Feb 2012.

[82] Sx . M. Wild, R. G. Regis, and C. A. Shoemaker. Orbit: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219, 2008.

[83] T. Xie, H. Yu, and B. Wilamowski. Comparison between traditional neural networks and radial basis function networks. In *2011 IEEE International Symposium on Industrial Electronics*, pages 1194–1199, June 2011.

[84] L. Xinhua, L. Yongzhi, and L. Hao. Theory and application of monte carlo method. In Y. Wu, editor, *Software Engineering and Knowledge Engineering: Theory and Practice*, pages 841–848, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[85] M. Husainy Yar, V. Rahmati, and H. R. Dalili Oskouei. A survey on evolutionary computation: Methods and their applications in engineering. *Modern Applied Science*, 10(11):131–139, november 2016.

[86] M.F. Zapata-Zapata, E. Mezura-Montes, and E.A. Portilla-Flores. Evolución diferencial con memoria de parámetros para la optimización de mecanismos de cuatro barras. In *Congreso Mexicano de Inteligencia Artificial (COMIA 2017). Research in Computer Science*, 2017. In press.

[87] A. E. Muñoz Zavala, A. Hernández Aguirre, and E. R. Villa Diharce. Continuous constrained optimization with dynamic tolerance using the copso algorithm. In Efrén Mezura-Montes, editor, *Constraint-Handling in Evolutionary Optimization*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[88] J. Zhang, Z. h. Zhan, Y. Lin, N. Chen, Y. j. Gong, J. h. Zhong, H. S. H. Chung, Y. Li, and Y. h. Shi. Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine*, 6(4):68–75, Nov 2011.

[89] K. Zielinski, S. P. Vudathu, and R. Laur. Influence of different deviations allowed for equality constraints on particle swarm optimization and differential evolution. In N. Krasnogor, G. Nicosia, M. Pavone, and D. Pelta, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, pages 249–259. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.