Centro de Investigación en Inteligencia Artificial
Universidad Veracruzana

"Incorporación de control y calibración de parámetros en un algoritmo de evolución diferencial para problemas de optimización con restricciones"

Tesis para obtención del grado de Maestro en Inteligencia Artificial

Autor: Octavio Ramos Figueroa

Director: Dr. Efrén Mezura Montes

Asesora: Dra. María Margarita Reyes Sierra

Veracruz-México, agosto de 2017

Resumen

Se diseñó una variante del algoritmo Evolución Diferencial (ED), al cual se le incorporó un mecanismo para controlar los parámetros factor de escalamiento (F) y probabilidad de cruza (CR). El algoritmo ED originalmente no fue diseñado para tratar con problemas de optimización con restricciones (PORs), de tal forma que esta investigación no sólo consistió en identificar un mecanismo para el control de parámetros que se adaptara bien a ED, sino en encontrar la mejor combinación: manejador de restricciones, variante del algoritmo ED y mecanismo para el control de parámetros. Se estudiaron los manejadores de restricciones: ε-constrained method y las reglas de factibilidad. Los mecanismos para controlar los parámetros fueron planteados bajo un enfoque determinista, específicamente utilizando funciones matemáticas. Se propusieron diversos algoritmos, que fueron evaluados a través de funciones de prueba tomadas de la literatura especializada y medidas de comparación de algoritmos, donde la mejor variante resolvió las 18 funciones estudiadas en 10 y 30 dimensiones. Una vez identificada la mejor versión, se llevó a cabo un estudio algorítmico para discernir la forma en que abordó cada caso de estudio, análisis que permitió justificar cada uno de los cambios efectuados sobre el algoritmo de ED original.

Agradecimiento

En estas líneas expreso mi más profundo y sincero agradecimiento:

.

- Al Centro de Investigación en Inteligencia Artificial de la Universidad Veracruzana (CIIA-UV), que me brindó todo el apoyo durante mi estancia.
- Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado durante la realización de este proyecto de investigación.
- A mi director de tesis el Dr. Efrén Mezura Montes, por sus consejos, paciencia, conocimientos, compromiso y confianza a lo largo de estos dos años en la maestría, parte fundamental para que este trabajo llegara a su término en tiempo y forma.
- A mi asesora la Dra. María Margarita Reyes Sierra, por todos sus grandes aportes a este trabajo de investigación.
- A los doctores del CIIA por sus atenciones y gratas enseñanzas que nunca olvidaré.
- A los Drs. Arturo Hernández Aguirre y Carlos Segura González por su apoyo y atención durante la estancia en el Centro de Investigación en Matemáticas (CIMAT), en la que compartieron su conocimiento e hicieron grandes aportes a este trabajo.
- Agradezco a todos mis compañeros de la Maestría y del Doctorado por compartir su experiencia y conocimiento, pero sobre todo su amistad. Gracias por hacerme sentir en un segundo hogar.
- A mis amigos por su apoyo emocional, consejos y palabras de aliento.
- Pero sobre todo a mi familia y en especial a mi madre por su apoyo incondicional quien sigue siendo un soporte fundamental y me ha dado las mejores enseñanzas de la vida.

Índice general

Resumen…….……..…….. I

Agradecimiento…….. II

Índice de tablas…...……... VI

Índice de figuras……..... IX

Índice de algoritmos……... X

1. Introducción……….……..…….……. 1
1.1 Antecedentes…….…….…….... 1
1.2 Planteamiento del problema……..……..……..…. 4
1.3 Objetivos de la investigación……...……..... 5
1.3.1 Objetivo general…..…..…….…..... 5
1.3.2 Objetivos específicos……..…….…… 5
1.4 Contribución……. 6
1.5 Hipótesis…….…….…….. 6
1.6 Justificación…….... 6
1.7 Estructura del documento……..……..……. 7

2. Optimización clásica……... 8
2.1 Introducción a la programación matemática……...…. 8
2.2 Definición de optimización…….…..... 9
2.3 Definición de un problema de optimización……...… 9
2.4 Algoritmos de optimización clásica de una sola variable…….……..... 13
2.5 Algoritmos de optimización clásica multivariables…….. 15
2.6 Ventajas y desventajas de la optimización clásica……..….. 16

3. Algoritmos evolutivos…… 17
3.1 Modelo general de un algoritmo evolutivo……. 19
3.2 Algoritmo de Evolución Diferencial…… 21
3.2.1 Análisis de los parámetros de ED……..... 24
3.3 Clasificación de las técnicas para el manejo de restricciones……..... 25
3.3.1 Funciones de penalización…… 26
3.3.2 Operadores especiales…….. 29
3.3.3 Separación de restricciones y objetivos…….**.....** 29
3.3.4 Métodos híbridos……**...**…….. 29
3.4 ED para problemas de optimización con restricciones……... 30

4. Configuración de parámetros……**...** 36
4.1 Calibración de parámetros…….…….. 36
4.2 Control de parámetros…….. 37

5. Trabajo propuesto…….**.....** 41
5.1 Diseño experimental……... 41
5.1.1 Diseño e implementación de algoritmos adaptativos de DE para PORs…….……..…….**.....** 41
5.1.2 Experimentación y pruebas de los algoritmos de ED adaptativos para PORs diseñados…… …….…….... 47
5.1.3 Estudio del comportamiento algorítmico del mejor algoritmo diseñado….**.....**…… 49
5.1.4 Rediseño del mejor algoritmo desarrollado...……..…….**.....** 50
5.1.5 Análisis de diversidad del mejor algoritmo rediseñado**.....**…… 51
5.2 Resultados…… 52
5.2.1 Resultados de la experimentación de los algoritmos de ED adaptativos para PORs diseñados.…….…….... 52
5.2.2 Resultados del comportamiento algorítmico del mejor algoritmo diseñado.…….…..… 57
5.2.3 Resultados del rediseño del mejor algoritmo desarrollado..….**.....** 58

5.2.4 Resultados del análisis de diversidad del mejor algoritmo rediseñado…….……...…… 61
5.3 Discusión de resultados……... 63
5.3.1 Discusión de resultados de la experimentación de los algoritmos de ED adaptativos para PORs diseñados…….. 63
5.3.2 Discusión de resultados del comportamiento algorítmico del mejor algoritmo diseñado ……..... 65
5.3.3 Discusión de resultados del rediseño del mejor algoritmo desarrollado……...……... 66
5.3.4 Discusión de resultados del análisis de diversidad del mejor algoritmo rediseñado……..…….....**.....** 68

6. Conclusiones y trabajo futuro……... 69
6.1 Conclusiones…… 69
6.2 Trabajo Futuro…… 72

Anexos……..… 73
Anexo I. Detalle de las funciones del CEC2010…….. 74
Anexo II. Tablas comparativas de la experimentación en 10 dimensiones…..**.** 80
Anexo III. Gráficas de factibilidad……... 81
Anexo IV. Tablas comparativas del algoritmo ε-ED-S ran-Conf1 ajustado en 10 dimensiones……...……...…….. 94
Anexo V. Estudio de diversidad en 10 dimensiones……... 98

Referencias……..…..**.** 117

Índice de tablas

3.1 Descripción de variantes del algoritmo de ED para PORs …….. 30

3.2 Descripción de algunas variantes de los algoritmos Rf-ED, ε-ED, Rf-EDVC y ε-EDVC……..…. 31

4.1 Valores propuestos para F y CR…….…... 37

5.1 Conjunto 1: Mecanismos para el control adaptativo dinámico de parámetros basados en una función Senoidal (S)…..…….. 43

5.2 Conjunto 2: Funciones Lineales, Exponenciales y Senoidales (LES)….. 43

5.3 Descripción de los esquemas generados con las funciones del Conjunto 2……. 45

5.4 Detalle de los algoritmos diseñados…….. 46

5.5 Sintaxis para los nombres de los algoritmos diseñados…… 47

5.6 Parámetros utilizados en los algoritmos que manejaron las restricciones con las reglas de factibilidad……..… 49

5.7 Parámetros utilizados en los algoritmos que manejaron las restricciones con ε-constrained method…….…..... 49

5.8 Comparativa entre los algoritmos: ε-ED-S best-Conf1, ε-ED-S best-Conf2, ε-ED-S best-Conf3, ε-ED-S best-Conf4, ε-ED-S best-Conf5 y ε-ED-S best-Conf6 en 10 dimensiones…….. 52

5.9 Resumen de la medida P en los algoritmos: ε-ED-S best-Conf1, ε-ED-S best-Conf4 y ε-ED-S best-Conf6 en 10 dimensiones……... 53

5.10 Comparativa entre los algoritmos: ε-ED-S ran-Conf1, ε-ED-S ran-Conf2, ε-ED-S ran-Conf3, ε-ED-S ran-Conf4, ε-ED-S ran-Conf5 y ε-DE-S ran-Conf6 en 10 dimensiones…….. 53

5.11 Resumen de la medida P en los algoritmos: ε-ED-S ran-Conf1, ε-ED-S ran-Conf4 y ε-ED-S ran-Conf6 en 10 dimensiones…….. 54

5.12 Comparativa entre los algoritmos: ε-ED-S best-Conf1, ε-ED-S best-Conf2, ε-ED-S best-Conf3, ε-ED-S best-Conf4, ε-ED-S best-Conf5 y ε-ED-S best-Conf6 en 30 dimensiones……. 55

5.13 Resumen de la medida P en los algoritmos: ε-ED-S best-Conf2, ε-ED-S best-Conf2, ε-ED-S best-Conf3, ε-ED-S best-Conf4, ε-ED-S best-Conf5 y ε-ED-S best-Conf6 en 30 dimensiones…….. 56
5.14 Comparativa entre los algoritmos: ε-ED-S ran-Conf1, ε-ED-S ran-Conf2, ε-ED-S ran-Conf3, ε-ED-S ran-Conf4, ε-ED-S ran-Conf5 y ε-ED-S ran-Conf6 en 30 dimensiones…….. 56
5.15 Resumen de la medida P en los algoritmos: ε-ED-S ran-Conf2, ε-ED-S ran-Conf2, ε-ED-S ran-Conf3, ε-ED-S ran-Conf4, ε-ED-S ran-Conf5 y ε-ED-S ran-Conf6 en 30 dimensiones…….. 57
5.16 Comportamiento algorítmico de ε-ED-S ran-Conf1 en las funciones más representativas…….. 58
5.17 Comparativa entre los algoritmos: ε-ED-S ran-Conf1 (Tc=0.2), ε-ED-S ran-Conf1 (Tc=0.3), ε-ED-S ran-Conf1 (Tc=0.4), ε-ED-S ran-Conf1 (Tc=0.5), ε-ED-S ran-Conf1 (Tc=0.6), ε-ED-S ran-Conf1 (Tc=0.7), ε-ED-S ran-Conf1 (Tc=0.8) y ε-ED-S ran-Conf1 (Tc=0.9) en 10 dimensiones…….. 59
5.18 Resumen de la medida P en los algoritmos: ε-ED-S ran-Conf1 (Tc=0.2), ε-ED-S ran-Conf1 (Tc=0.3), ε-ED-S ran-Conf1 (Tc=0.4), ε-ED-S ran-Conf1 (Tc=0.5), ε-ED-S ran-Conf1 (Tc=0.6), ε-ED-S ran-Conf1 (Tc=0.7), ε-DE-S ran-Conf1 (Tc=0.8) y ε-ED-S ran-Conf1 (Tc=0.9) en 10 dimensiones……. 60
5.19 Comparativa entre las 10 versiones del algoritmo ε-ED-S ran-Conf1 (diferentes en sus parámetros Tc y cp), con mejor desempeño en 10 dimensiones…….. 60
5.20 Resumen de la medida P en las mejores versiones del algoritmo: ε-ED-S ran-Conf1 con diferentes valores para los parámetros Tc y cp, en 10 dimensiones…….. 60
5.21 Comparativa entre las 10 mejores versiones del algoritmo ε-ED-S ran-Conf1 en 30 dimensiones. P: probabilidad de convergencia…… 61
5.22 Resumen de la medida P en las 10 mejores versiones del algoritmo: ε-ED-S ran-Conf1 con diferentes valores para los parámetros Tc y cp, en 30 dimensiones……..….. 61

5.23 Comparativa entre algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función 4 en 10 dimensiones…… 62
5.24 Comparativa entre algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función 4 en 30 dimensiones…….. 63

Índice de figuras

2.1 Características que puede tener un POR..... 12
3.1 Modelo general de un AE……..……. 19
3.2 Operadores de cruza y mutación de ED aplicados al *trial i* en la generación *g*…… 23
4.1 Taxonomía de las técnicas para la configuración de parámetros…...…. 36
5.1 Comportamiento esperado del esquema Conf1……..…… 43
5.2 Comportamiento esperado de las funciones lineales, exponenciales y senoidales del conjunto 2……...…… 44

Índice de algoritmos

3.1 ED…….…….... 23
3.2 RF-ED……..…… 32
3.3 ε –ED……..…….…….. 32
3.4 RF-EDVC…….……... 33
3.5 ε-EDVC……..…….…….. 34

Capítulo 1

Introducción

A lo largo de la historia, los investigadores han buscado en la naturaleza formas de resolver problemas computacionales con alto grado de complejidad, lo que ha dado lugar al surgimiento de los Algoritmos Bio-Inspirados (ABIs), rama de la Inteligencia Artificial (IA) en la que se diseñan métodos metaheurísticos (estrategias inteligentes compuestas por un conjunto de heurísticas que pueden ser iniciadas o detenidas de acuerdo a las necesidades del problema a resolver), la mayoría no determinísticos (algoritmos que con una misma entrada pueden obtener diferentes resultados), de búsqueda y optimización que pretenden emular la forma en que la naturaleza enfrenta los problemas [1]. A través de los años, han emergido diversos enfoques entre los que destacan los Algoritmos Evolutivos (AEs) y la Inteligencia Colectiva (IC) [2]. El surgimiento de la Computación Evolutiva (CE) se remonta a principios de los años 50s. Desde sus inicios, este tipo de técnicas fueron bien recibidas gracias a su amplia aplicación en problemas del mundo real, por ejemplo: en el diseño de tácticas militares, diseño de estructuras de avión, diseño de fármacos, sistemas de agentes inteligentes y minería de datos [3]. Por otro lado, han sido fuertemente criticados por la fina configuración de parámetros que requieren, lo que hasta hoy sigue siendo una de sus principales debilidades, por lo que la comunidad científica del área ha llevado a cabo grandes esfuerzos con el fin de minimizar el impacto de dicha calibración [4].

1.1 Antecedentes

A la fecha, se han realizado diversos estudios sobre la configuración de los parámetros de un AE, estudios que han sido clasificados en calibración y control de parámetros.

Definición 1. La calibración o ajuste de parámetros toma lugar durante la etapa de diseño, consiste en identificar los valores óptimos para los parámetros que requiere determinado AE a través de una experimentación exhaustiva, valores que permanecen fijos durante todo el proceso de búsqueda.

Definición 2. El control de parámetros es una forma alternativa para configurar los parámetros de un AE, la cual se presenta durante la etapa de ejecución, donde a diferencia del ajuste de parámetros el algoritmo parte con determinados valores de parámetros que se van ajustando por medio de diversas reglas. Las técnicas de control de parámetros se pueden clasificar en: deterministas, adaptativas y auto-adaptativas [5].

Definición 3. El control de parámetros determinista es un enfoque donde el o los parámetros en cuestión son alterados con base en una regla en particular que modifica los valores de los parámetros de manera determinista, no usa retroalimentación y usualmente utiliza la variación del tiempo.

Definición 4. El control de parámetros adaptativo es una técnica donde a diferencia del enfoque determinista, se utiliza algún tipo de retroalimentación para reestablecer la dirección o la magnitud de las soluciones.

Definición 5. El control de parámetros auto-adaptativo es muy similar al control adaptativo, en este enfoque los parámetros que se van a auto-adaptar, se agregan al vector solución y se van cambiando de acuerdo a los operadores de cruza y mutación, está basado en la idea de que mejores parámetros llevan a mejores individuos, por lo que los mejores parámetros persisten generación tras generación.

La principal ventaja de la calibración de parámetros es su simplicidad, sin embargo tiene el inconveniente de que requiere una experimentación exhaustiva para identificar la calibración que mejor se adecúe a los parámetros en cuestión, proceso que toma demasiado tiempo. Por otro lado, la principal ventaja del control de parámetros es que puede cambiar los valores de los parámetros en diferentes etapas del proceso evolutivo, sin embargo algunas de estas

estrategias pueden llegar a ser demasiado complejas, lo que se ve reflejado en un incremento en el costo computacional.

Desde que surgieron los primeros AEs, se han propuesto diversos esquemas para localizar valores de parámetros que lleven a buenos resultados en diversos problemas, empleando el ajuste de parámetros. En los últimos años se ha demostrado teórica y empíricamente que actualizar los valores de los parámetros en diferentes etapas del proceso evolutivo permite obtener mejores resultados [4]. Debido a esto, en la última década la comunidad científica del área se ha encauzado al desarrollo de algoritmos que utilicen mecanismos para el control de parámetros. Por ejemplo: SaDE y jDE, algoritmos desarrollados para problemas de optimización sin restricciones [4]. En esta investigación se pretende desarrollar un algoritmo adaptativo (algoritmo capaz de modificar los valores de sus parámetros utilizando algún enfoque para el control de parámetros) capaz de solucionar problemas de optimización con restricciones (PORs), donde la localización de la solución óptima es a menudo difícil, ya que sus características y propiedades matemáticas no siguen ningún patrón [6].

El algoritmo a desarrollar tendrá como base el algoritmo de Evolución Diferencial (ED). Creado por *Storn* y *Price* en 1995 [7], ED surge como una forma competitiva de la CE, sus principales característica son: eficiencia, simplicidad y uso de una representación real en lugar de una binaria.

La ED utiliza tres parámetros: tamaño de la población (NP por sus siglas en inglés), factor de escalamiento (F por sus siglas en inglés) y porcentaje de cruza (CR por sus siglas en inglés).

NP controla el número de individuos dentro de la población, F se encarga de controlar el tamaño de paso, mientras que CR dirige el proceso de recombinación o cruza. Cabe mencionar que los tres parámetros están mutuamente relacionados, por lo que la mala calibración de uno se ve directamente reflejada en los demás. De acuerdo a lo antes mencionado, si se le quiere brindar a ED la capacidad de explorar en la medida de lo posible, los

valores para F y CR deben ser lo más altos permisible y viceversa. Dado que la calidad de la calibración de los parámetros de un algoritmo está estrechamente relacionada con el desempeño que éste pueda alcanzar [2], los tres parámetros deben ser configurados buscando mantener un buen compromiso exploración-explotación.

Ahora bien, cuando se desarrolla un AE para PORs es necesario utilizar mecanismos especiales para manejar las restricciones. Existen diferentes formas de tratar a los individuos no factibles (soluciones que no satisfacen todas las restricciones), tal es el caso de las reglas de factibilidad (Rf) y el ε-constrained method (ε) [8, 9].

A la fecha se han desarrollado diversas variantes de ED adaptativas para PORs [6, 10-20]. Sin embargo, de acuerdo a los alcances de la revisión de la literatura, hasta ahora no existen estudios sobre el efecto del uso de funciones matemáticas aplicadas al control de los parámetros de ED para PORs.

1.2 Planteamiento del problema

Desde su surgimiento los AEs han sido fuertemente criticados por la fina configuración que requieren sus parámetros para obtener buenos resultados. Después de haberse llevado a cabo diversos estudios sobre el ajuste de parámetros, en la última década la comunidad científica del área ha optado por la implementación del control de parámetros. Actualmente existen diversos mecanismos para el control de parámetros que han sido incorporados en algoritmos diseñados para resolver problemas de optimización con y sin restricciones. Sin embargo, es importante destacar que el número de algoritmos adaptativos para PORs es mucho menor, debido en gran medida a su complejidad.

Con base en el algoritmo ED, el estudio de los mecanismos de control de parámetros enunciados en [1, 4-6, 10-20] y las técnicas para el manejo de restricciones ε-constrained method y reglas de factibilidad [8,9], se pretende

identificar las tendencias actuales para guiar la implementación de un nuevo algoritmo que controle los parámetros F y/o CR bajo un enfoque determinista a través de diversas funciones matemáticas y que maneje las restricciones utilizando una técnica competitiva. De acuerdo a la revisión de la literatura especializada, a la fecha no se han diseñado variantes de ED adaptativas para PORs que controlen sus parámetros utilizando funciones matemáticas, lo que se ha identificado como un nicho de oportunidad.

1.3 Objetivos de la investigación

1.3.1 Objetivo general

Diseñar un algoritmo basado en evolución diferencial (ED) que contenga un mecanismo para el control de los parámetros factor de escalamiento (F) y porcentaje de cruza (CR), capaz de resolver problemas de optimización con restricciones (PORs) de forma eficiente, utilizando una técnica competitiva para el manejo de restricciones.

1.3.2 Objetivos específicos

- Analizar la literatura especializada para identificar algoritmos basados en ED para PORs.
- Implementar y probar los algoritmos seleccionados.
- Diseñar nuevos algoritmos con base en la combinación de algunas variantes de los algoritmos de ED para PORs estudiados, mecanismos para el control de los parámetros F y/o CR basados en funciones matemáticas y los manejadores de restricciones ε-constrained method y reglas de factibilidad.
- Analizar y comparar el desempeño de las variantes adaptativas de ED para PORs diseñadas utilizando problemas tomados de la literatura especializada [21], los cuales se pueden observar en el Anexo I.
- Identificar el algoritmo diseñado con mejor desempeño, que mejore el rendimiento de ED disminuyendo el número de evaluaciones requeridas y/o mejorando el resultado final de su aplicación a los casos de estudio.

- Analizar el comportamiento algorítmico de la variante diseñada con el mejor desempeño para identificar sus debilidades y fortalezas.
- Rediseñar el algoritmo con base en las conclusiones obtenidas del estudio algorítmico, con el cometido de mejorar su desempeño.

1.4 Contribución

Diseño de una nueva variante adaptativa del algoritmo de evolución diferencial capaz de resolver problemas de optimización con restricciones, que utilice una técnica competitiva para el manejo de restricciones y un mecanismo para el control de los parámetros F y/o CR basado en funciones matemáticas.

1.5 Hipótesis

Es posible diseñar un algoritmo adaptativo de ED para PORs que sea capaz de reducir el número de evaluaciones requeridas y/o mejore los resultados obtenidos de su aplicación en problemas de la literatura especializada, utilizando un mecanismo para el control de los parámetros F y/o CR basado en funciones matemáticas y una técnica competitiva para el manejo de restricciones.

1.6 Justificación

La eficiencia de los AEs varía de acuerdo a las características específicas de cada problema, sin embargo la calidad de la configuración de los parámetros se ve reflejada directamente en los resultados que éstos pueden alcanzar [2]. Como se detalló en los antecedentes, el primer enfoque que se utilizó para minimizar el impacto de la calibración de los parámetros fue el ajuste de parámetros, hasta que se demostró teórica y empíricamente que modificar los valores de los parámetros durante el proceso evolutivo permite llegar a mejores resultados. Actualmente, los esfuerzos de la comunidad científica se han dirigido hacia el control de parámetros, buscando pasar del estado de exploración al de

explotación en diferentes etapas de la búsqueda, lo que en principio debe llevar a mejores resultados y/o una reducción en el número de evaluaciones requeridas. Este enfoque pretende avanzar en el camino hacia una búsqueda autónoma. Es importante remarcar que para lograrlo aún falta mucho por hacer, sin embargo, eso convierte esta área en un gran nicho de oportunidad. Según [2], hasta el día de hoy no hay ningún algoritmo en la literatura especializada que sea dominante ante los demás. Cabe resaltar que, si bien llevar a cabo el ajuste de los valores de los parámetros de un AE es difícil, controlarlos a través de funciones es aún más complicado, por lo que éste sigue siendo un problema abierto a la investigación.

1.7 Estructura del documento

Este documento de tesis continúa de la siguiente forma: en el Capítulo 2 se detallan las generalidades de la optimización clásica (programación matemática), en el Capítulo 3 se describe a detalle el algoritmo de evolución diferencial y el funcionamiento de sus parámetros, así como una clasificación de los tipos de manejadores de restricciones que existen, el Capítulo 4 habla sobre la taxonomía de las técnicas para la configuración de parámetros, el Capítulo 5 contiene el trabajo propuesto, donde se describe el diseño experimental, así como los resultados y su análisis, por último, el Capítulo 6 contiene las conclusiones generales del proyecto de investigación y el trabajo futuro.

Capítulo 2
Optimización Clásica

2.1 Introducción a la programación matemática

La programación matemática (PM) es ampliamente utilizada en los campos de la IA y la Investigación de Operaciones (IO). Surge con el objetivo de resolver problemas de alta complejidad haciendo uso de modelos matemáticos, particularmente modelos de optimización. El término programación hace referencia a la necesidad de diseñar algoritmos (conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permiten desarrollar actividades específicas a través de una secuencia de pasos que llevan a un estado final y/o una solución), base de la programación computacional.

Una de las características de la PM es que encuentra la mejor solución a un modelo. De tal forma que no siempre se encuentra la mejor solución al problema del mundo real, si no la mejor solución al modelo generado.

Para encontrar una solución con base en un proceso de optimización se requiere un modelo (definido en términos de variables de decisión, restricciones y una función objetivo), datos (instancia del modelo) y un motor de búsqueda (algoritmo que resuelva las instancias del modelo). De tal forma que un algoritmo de optimización tiene por objetivo identificar los valores para las variables de decisión que mejor satisfagan todas las restricciones y optimicen la función objetivo. Los elementos y características principales de la PM se describen a continuación con base en el libro: *"Optimization for engineering design: algorithms and examples",* propuesto por Kalyanmoy Deb en el 2000 [22].

2.2 Definición de optimización.

Técnica matemática altamente utilizada en campos como la IA y IO que hace referencia a la selección del mejor (con respecto a algún criterio) de un conjunto de elementos disponibles. Por ejemplo maximizar o minimizar alguna función relativa a algún conjunto de valores de entrada, que a menudo representa un rango de opciones disponibles en una cierta situación. La función permite medir el valor de las diferentes opciones para compararlas e identificar la mejor. Esta herramienta ha sido utilizada para resolver una amplia gama de problemas de gran interés teórico y práctico. Por ejemplo en IO, identificar: dónde, cuándo y cuántos artículos fabricar en una empresa; o cómo transportar artículos de forma eficiente, personas o materias primas [22].

2.3 Definición de un problema de optimización

Cada problema tiene determinadas características específicas, lo que implica generar un modelo particular para cada caso de estudio abordado, sin embargo se puede tomar en cuenta un modelo general, que permite guiar el proceso de formulación y resolución de un problema dado. Los componentes y/o etapas principales que deben tomarse en cuenta cuando se requiere resolver un problema de optimización son: (1) conocer y entender a fondo el problema a resolver, (2) elegir las variables de decisión, (3) formular las restricciones, (4) diseñar la función objetivo, (5) fijar los límites de las variables, (6) elegir un algoritmo de optimización y (7) obtener una solución [22].

Las variables de decisión que conforman un modelo de PM representan valores o decisiones. De tal manera que el objetivo de un algoritmo de optimización es identificar el conjunto de valores que al ser fijados en las variables del problema estudiado resuelvan de mejor forma la función objetivo. Algunos ejemplos que pueden ser utilizados como variables de decisión de un problema son [22]:

- Número de unidades a fabricar de determinado producto.

- Número de personas requerido para llevar a cabo una tarea dada.
- Tiempo de inicio o culminación de una tarea.

El valor de cada variable es calculado mediante el proceso de resolución. Al diseñar un modelo de PM, los valores de las variables son usualmente desconocidas, aunque en ocasiones es posible generar pistas iniciales que ayuden en el proceso de optimización. Las variables deben tener un dominio (conjunto de valores que puede tomar una variable), un límite (impuesto por los límites de su dominio) y un tipo (por ejemplo: real, entero, booleano o intervalo). El proceso de selección de las variables de decisión es muy importante porque impactan directamente en el cumplimiento de las restricciones y la minimización o maximización de la función objetivo [22].

Las restricciones permiten definir cómo se relacionan entre si las variables de decisión y los valores que éstas pueden tomar. Representan los límites en los que se encuentra la solución. A continuación se describe como se podrían restringir los ejemplos de variables de decisión antes mencionados [22]:

- Cuántas unidades de determinado producto se deben generar de acuerdo a las capacidades de producción de una empresa.
- Habilidades mínimas requeridas por una persona para llevar a cabo un trabajo determinado.
- Determinada tarea comienza si y sólo si otra tarea relacionada ha culminado.

La función objetivo de un modelo de optimización representa matemáticamente el problema que se está abordando, puede ser de maximización o minimización y puede involucrar un objetivo único (optimización global) o varios objetivos (optimización multiobjetivo). Por ejemplo: maximizar las ganancias de una empresa, minimizar los costos de una construcción, minimizar los retrasos en la ejecución de determinadas tareas, maximizar la calidad de un servicio prestado, entre otros. En el proceso de solución de un problema, la finalidad de la función objetivo es medir qué tan

buenas o malas son las soluciones potenciales al problema e identificar a la que mejor optimice el caso de estudio [22].

La última etapa de la formulación de un problema consiste en establecer los límites inferior y superior de las variables, lo que permite restringir la búsqueda. De tal forma que no todos los posibles valores de las variables sean aceptados. Para hacer dicha validación, se utiliza la ecuación 2.1 [22]:

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \; para \; i = 1, 2, \dots, N \tag{2.1}$$

Cuando un problema de optimización no tiene restricciones, la búsqueda de la solución se lleva a cabo en todo el espacio de búsqueda, respetando los límites inferior y superior de cada variable de decisión. Por otro lado, existe una gran diversidad de PORs, que requieren de un proceso de búsqueda más complejo, debido a que la localización de la solución óptima es a menudo difícil, ya que sus características y propiedades matemáticas no siguen ningún patrón [5]. Un POR normalmente se escribe como un problema de programación no lineal de la siguiente manera [6]:

$$\text{Minimizar: } f(x), x = (x_1, x_2, \dots, x_N) \; y \; x \; \epsilon \; S \tag{2.2}$$

$$\text{Sujeto a: } g_i(x) \leq 0, i = 1, \dots, p \tag{2.3}$$

$$h_i(x) = 0, j = p + 1, \dots, m \tag{2.4}$$

La función *f* no tiene que ser continua pero si tiene que estar limitada. S es el espacio de búsqueda, p es el número de restricciones de desigualdad y $(m - p)$ el número de restricciones de igualdad. Las restricciones de igualdad y desigualdad deben satisfacer las funciones $h_i(x) = 0$ y $g_i(x) \leq 0$ respectivamente en la solución óptima global. Las restricciones de igualdad pueden ser transformadas en restricciones de desigualdad con base en una tolerancia $1\text{x}10^{-4}$. De tal forma que el objetivo es minimizar la función $f(x)$ satisfaciendo todas las restricciones $g_i(x)$ y $h_i(x)$. La no factibilidad de los individuos puede ser medida con base en la suma de la cantidad de restricciones

no satisfechas. La Figura 2.1 muestra gráficamente algunas características que puede tener un problema restringido [6]. Cuando se está tratando de resolver un POR, puede haber algunas soluciones factibles (a, c, d, e) que conforman la o las zonas factibles, individuos no factibles (b, d, f) y la solución óptima (x), que forma parte de la zona factible. El problema de cómo tratar a los individuos no factibles es muy trivial. En general, se tienen que diseñar dos funciones de evaluación, una para el dominio de las soluciones factibles y otra para el de las no factibles. Algunos puntos importantes a considerar son: ¿cómo comparar dos individuos factibles?, ¿cómo comparar dos individuos no factibles? y ¿cómo comparar un individuo factible contra uno no factible? Partiendo del hecho de que una solución factible es mejor que una no factible, un algoritmo diseñado para resolver PORs debe buscar eliminar los individuos no factibles de la población, para llevar a cabo dicha tarea, primero debe utilizar los individuos no factibles para explorar de forma más eficiente el espacio de búsqueda, para posteriormente proceder a eliminarlos (cuando la zona factible ha sido identificada). Se han creado diferentes enfoques para tratar las restricciones de un problema, los cuales se describirán más adelante [23].

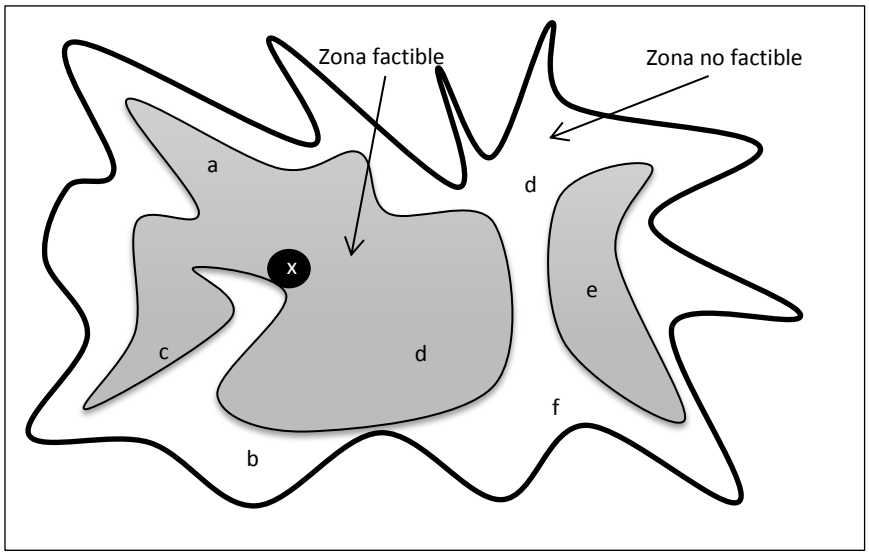


Figura 2.1 Características que puede tener un POR.

Ahora bien, cuando se intenta resolver un problema de optimización con o sin restricciones, es importante conocer los tipos de óptimos que existen, los cuales se describen a continuación [22]:

- Óptimo local. Se denomina óptimo local a un punto x^* si no existe en su vecindario otra solución x que sea mejor. De esta forma, en un problema de minimización un punto x^* puede ser considerado un óptimo local si ninguna solución en su vecindario tiene un valor menor que $f(x^*)$.
- Óptimo global. Se conoce como óptimo global a la solución x^{**} si no existe otro punto en todo el espacio de búsqueda mejor que la solución x^{**}. Ésto es, una solución x^{**} es un punto mínimo global si no existe otro punto x en todo el espacio de búsqueda tal que $f(x) < f(x^{**})$.
- Punto de inflexión. Un punto $x^{«»}$ es llamado punto de inflexión si el valor de la función aumenta localmente cuando $x^{«»}$ aumenta y disminuye localmente cuando $x^{«»}$ se reduce o si el valor de la función disminuye localmente a medida que $x^{«»}$ aumenta e incrementa localmente cuando $x^{«»}$ disminuye.

2.4 Algoritmos de optimización clásica de una sola variable

Métodos diseñados para resolver problemas que involucran una sola variable de decisión basados en procesos simples y fáciles de entender. Estos algoritmos son comúnmente utilizados como sub-tareas en algoritmos multivariables (algoritmos diseñados para resolver problemas con varias variables). Los problemas de minimización o maximización de una sola variable pueden ser definidos de la siguiente forma [22]:

$$Minimizar\ f(x) \tag{2.5}$$

Donde $f(x)$ representa la función objetivo y x una variable real. De tal manera que el propósito de un algoritmo de optimización de una variable radica

en encontrar el valor de x que minimice (o maximice) de mejor forma la función objetivo $f(x)$. Este tipo de algoritmos se pueden clasificar en: (1) métodos de búsqueda directos (se basan únicamente en los valores de la función objetivo para encontrar la solución óptima) y (2) métodos basados en gradiente (hacen uso de la primera y la segunda derivada de la función objetivo para localizar la solución óptima). A continuación se presentan algunos ejemplos de métodos de búsqueda directos [22].

En los métodos de tipo bracking, se divide el proceso de optimización en dos fases. En la primera, los límites inferior y superior entre los que se puede encontrar la solución son identificados mediante el uso de una técnica simple. Posteriormente, utiliza un método más sofisticado para buscar dentro de los límites acotados hasta encontrar la solución óptima de acuerdo a la exactitud deseada. Tal es el caso de los métodos de búsqueda exhaustiva y de fase de limitación [22].

Los métodos de eliminación de regiones se basan en el principio de la eliminación de regiones. Dados dos valores $x_1\ y\ x_2$ tales que $a < x_1 < x_2 < b$ donde (a, b) representan el espacio de búsqueda. De acuerdo a los valores evaluados y asumiendo que la función es unimodal en el espacio de búsqueda, se puede concluir que el mínimo no puede estar en una de las dos porciones del espacio de búsqueda. La regla fundamental para determinar en qué porción se encuentra el mínimo consiste en: (1) si $f(x_1) > f(x_2)$ el mínimo no pertenece a la porción (a, x_1), (2) si $f(x_1) < f(x_2)$ el mínimo no pertenece a la porción (x_2, b) y (3) si $f(x_1) = f(x_2)$ el mínimo no pertenece a la porción (a, x_1) ni a la porción (x_2, b). Algunos algoritmos basados en esta regla son: el método de intervalos a la mitad y el método de búsqueda de Fibonacci [22].

Por último, los métodos de estimación de puntos utilizan las magnitudes y los signos de los valores de la función objetivo para guiar la búsqueda. Tal es el caso del método de estimación cuadrática sucesiva [22].

Por otro lado, los métodos indirectos, también conocidos como métodos basados en gradiente, trabajan con la información de la derivada. Si bien en muchos problemas del mundo real es difícil obtener esta información, estos métodos son muy utilizados gracias a su efectividad. Entre los métodos más conocidos están: el método Newton-Raphson, el método de bisección, el método secante y el método de búsqueda cúbico [22].

2.5 Algoritmos de optimización clásica multivariables

En esta sección se describen algunos algoritmos de optimización para problemas con múltiples variables de diseño o de decisión, donde los criterios de optimalidad son similares a cuando se trabaja con una sola variable. De tal manera que en ocasiones se utilizan algoritmos de una sola variable para dirigir la búsqueda en una dirección deseada [22].

Cuando se trabaja con un problema de una variable existen sólo dos direcciones de búsqueda que se pueden tomar a partir de un punto determinado, es decir, las direcciones positiva y negativa de dicho punto. En las funciones con múltiples variables de diseño se pueden tomar 2^N direcciones. Existen algoritmos que van resolviendo dimensión por dimensión del problema abordado, conocidos como métodos de una variable a la vez. Sin embargo ese tipo estrategias sólo son efectivas cuando la función es linealmente separable. De tal forma que para resolver una función no lineal, se requiere cambiar el concepto de dirección de búsqueda por el de manipulación de un conjunto de puntos, a partir de los cuales se debe crear un nuevo conjunto de puntos en principio mejores o utilizar una búsqueda compleja de direcciones. Algunos ejemplos de métodos directos son: el método de optimización evolutiva, el método de búsqueda *Simplex y* el método de búsqueda de patrones Hooke-jeeves [22].

Los métodos directos requieren de muchas evaluaciones de la función objetivo para encontrar la solución óptima de un problema con múltiples

variables, mientras que los indirectos explotan la información que brinda la derivada, por lo que son usualmente más rápidos. Sin embargo tienen la debilidad de que no se pueden aplicar a la mayoría de los problemas del mundo real. Por ejemplo, los métodos basados en gradiente no pueden ser aplicados a funciones discretas o continuas. Este tipo de métodos pueden trabajar con la primera derivada o con la primera y la segunda, las cuales se obtienen del valor de la función objetivo de sólo dos o tres puntos vecinos. La mayoría de estos métodos trabajan buscando diversas soluciones iterativamente, de tal manera que estos algoritmos se diferencian de acuerdo a la forma en la que se definen las direcciones de búsqueda. Algunas técnicas bajo este enfoque son: el método de Cauchy y el método de Newton [22].

2.6 Ventajas y desventajas de la optimización clásica

- Ventajas**:** Los enfoques tradicionales siempre deben ser considerados como la primera opción para resolver un problema de búsqueda y optimización, de tal forma que si el problema a resolver se ajusta a las condiciones de un método clásico, se puede garantizar encontrar la mejor solución con un costo computacional considerablemente bajo.
- Desventajas**:** En algunos problemas, este tipo de técnicas no pueden ser aplicadas o pueden demorar demasiado tiempo en devolver una solución aceptable, debido a la complejidad de los casos de estudio tratados. Además, algunos métodos distan de ser sencillos y fáciles de entender y/o aplicar.

Capítulo 3
Algoritmos Evolutivos

Debido a que cada día surgen nuevos y más complejos problemas de búsqueda y optimización, las técnicas clásicas de programación dejaron de ser suficientemente efectivas para resolverlos. Esto dio paso al surgimiento del cómputo inteligente, que se puede definir de manera general como "el estudio de mecanismos adaptativos para generar o facilitar el comportamiento inteligente en ambientes complejos, inciertos y cambiantes" [3]. Los paradigmas que lo conforman son: las redes neuronales (RNs), los AEs, la IC, los sistemas inmunes artificiales (SIAs) y los sistemas difusos (SD).

Una de las estrategias más utilizadas para abordar problemas de alta complejidad son los AEs, que esencialmente emulan la evolución de las especies y el principio de supervivencia del más apto. Dentro de los AEs existen varios paradigmas que se diferencian por los esquemas de representación para codificar y manipular las soluciones, así como los operadores (cruza, copia, mutación, selección y remplazo) que utilizan. Su aplicación se focaliza en problemas de optimización. Estas estrategias más que reflejar fielmente los procesos evolutivos, son modelos ideales de evolución orientados a la resolución de problemas del mundo real. El objetivo general que se ha perseguido durante muchos años es encontrar solucionadores aplicables a una amplia gama de problemas, que con ajustes finos devuelvan buenas soluciones en tiempos de búsqueda razonables. Entre sus principales características están: (1) son poblacionales "manejan varias soluciones simultáneamente", (2) utilizan la combinación de soluciones para mezclar su información generando nuevas soluciones y (3) son estocásticos "utilizan algo de aleatoriedad" [3]. Los paradigmas de los AEs se describen a continuación:

- Las Estrategias Evolutivas (EE) surgieron en los años 60´s en la U. T de Berlín gracias a *Rechenberg* y *Schwefel*, quienes formularon ideas de cómo un proceso evolutivo puede ser utilizado para resolver problemas

numéricos de optimización complejos. En sus inicios se centró en la optimización numérica (números reales) mediante los modelos $(1+1)-\text{ES}$ y $(1+\lambda)-\text{ES}$, la reproducción sexual se lleva a cabo utilizando una distribución gaussiana y remplazo determinístico. Su principal característica es la capacidad que tienen para auto-adaptar sus parámetros relativos a la mutación [24].

- De forma casi simultánea, durante la misma década en la UCLA, *Foguel* vio en la evolución el medio ideal para alcanzar las metas de la IA, los primeros intentos constaron de la evolución de agentes inteligentes representados como máquinas de estado finito, lo que dio origen al paradigma conocido como Programación Evolutiva (PE), la cual se concentra principalmente en modelos con reproducción asexual y mecanismos de remplazo probabilístico. Sus principales parámetros son el tamaño de la población y el porcentaje de reproducción sexual [25-26].
- Por otro lado en la misma década en la U. de Michigan, *Holland* vio en el proceso evolutivo la clave para diseñar e implementar sistemas adaptativos robustos (algoritmos capaces de resolver una amplia gama de problemas), con la capacidad de lidiar con ambientes cambiantes e inciertos. Resaltó la auto-adaptación de estas estrategias como una necesidad al interaccionar con el ambiente. Las ideas antes mencionadas, dieron origen a los planes reproductivos, actualmente conocidos como Algoritmos Genéticos (AGs). Se caracterizaron por la promoción de algoritmos independientes de la aplicación mediante una representación de cadenas de bits, uso intensivo de reproducción sexual, reemplazo generacional y selección basada en aptitud [27].
- En sus inicios la Programación Genética (PG) fue utilizada principalmente para resolver problemas relativamente simples, ya que tiene un costo computacional alto. Sin embargo, en los últimos años gracias a los avances tecnológicos ha producido resultados novedosos y excelentes en áreas como la computación cuántica y el diseño

electrónico. La principal característica de este paradigma es su forma particular de representar y manipular las soluciones (mediante árboles) [28].

3.1 Modelo general de un algoritmo evolutivo

Gracias al impacto que han tenido los AEs, ha surgido un sin número de algoritmos y a la vez variaciones de los mismos, siempre con el objetivo de generar esquemas cada vez más robustos, todos ellos basados en el siguiente modelo general: un AE parte de una población de individuos (soluciones potenciales al problema), la influencia del ambiente origina la selección natural (función de calidad o aptitud), de tal forma que los individuos mejor adaptados a su ambiente son seleccionados para su reproducción (mediante operadores de cruza y mutación principalmente) y tienen altas probabilidades de sobrevivir para la siguiente generación. La descendencia se inserta en la población mediante operadores de remplazo que introducen a los descendientes (en principio a los mejores) en la generación previa utilizando diversos criterios (por ejemplo: los peores, los padres o aleatoriamente). El modelo se puede observar gráficamente en la Figura 3.1 [3].

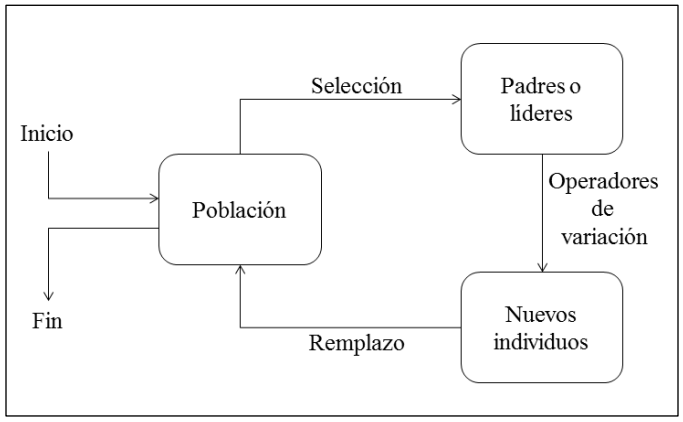


Figura 3.1 Modelo general de un AE.

Los componentes que conforman un AE son [3]:

- Esquema de representación. Es uno de los puntos más importantes a considerar al diseñar un algoritmo de optimización. Es la liga entre el modelo generado y el problema del mundo real. La representación puede hacerse a nivel genotípico y fenotípico, espacios considerados muy diferentes. De esta manera una solución, un fenotipo y un individuo, son considerados sinónimos en este ámbito.
- Función de aptitud. También conocida como función objetivo, representa la tarea a resolver mediante una función matemática definida usualmente en el espacio fenotípico, su trabajo consiste en evaluar qué tan buenas o malas son las soluciones así como identificar la solución óptima a determinado problema. Es decir, simula el ambiente en el que se mueven los individuos.
- Población de soluciones. Conjunto de individuos (soluciones potenciales al problema) diseñados en el espacio genotípico, generalmente son creados a partir de una heurística sencilla que genera soluciones aleatorias. La población es considerada la unidad de la evolución, donde la relación 1 genotipo = 1 fenotipo = 1 valor de aptitud debe prevalecer. Una población es considerada buena cuando tiene una alta diversidad (población de soluciones con características lo más diferentes posible), que debe prevalecer después de la aplicación de los operadores que trabajan sobre ella (operadores de selección y remplazo).
- Mecanismo de selección de padres. Tiene como propósito principal elegir a los individuos que participarán en el proceso de reproducción, para lo cual, tiene la capacidad de distinguir entre individuos considerando su calidad, prefiriendo en principio a los mejores. Entonces un individuo es padre si ha sido seleccionado para aplicársele operadores de variación (para generar descendencia), con el objetivo de promover mejoras en la calidad de las soluciones. Algunas estrategias utilizadas para elegir a los padres son la selección por torneo y la selección por ruleta.

- Operadores de variación. Su cometido es generar nuevos individuos a partir de los ya existentes, hay diferentes tipos clasificados de acuerdo a su aridad. La mutación y la copia son unarios, la primera provoca que alguno de los genes de un individuo sea perturbado aleatoriamente bajo cierta probabilidad (Pm), generalmente menor al 1%, mientras que el segundo consiste en la copia de ciertos individuos para que permanezcan en generaciones posteriores siguiendo una estrategia asexual. Por otro lado está la cruza, donde el objetivo es intercambiar material genético entre los padres seleccionados para generar la descendencia para la siguiente generación siguiendo una estrategia sexual.
- Mecanismos de remplazo. Al igual que un operador de selección, tienen la capacidad de distinguir entre individuos de acuerdo a su calidad, se emplean para mantener el tamaño de la población fijo. Se utilizan después de los operadores de variación y usualmente son deterministas. Su función principal es remplazar individuos de la generación anterior (aleatoriamente, los peores o sus padres) por algunos de los descendientes (en principio los mejores).
- Formas de detener un AE. Existen diferentes formas de parar un AE, algunos ejemplos se enlistan a continuación: (1) cuando se encuentra la solución buscada (se asume que se conoce el óptimo y por lo tanto puede utilizarse una pequeña tolerancia $\varepsilon = 0.1\text{x}10^{-4}$, (2) tiempo máximo de uso del CPU, (3) número máximo de evaluaciones, (4) límite de generaciones, (5) cuando no mejora el valor de aptitud en un determinado periodo de tiempo y (6) cuando la diversidad de la población está por debajo de un límite determinado [3].

3.2 Algoritmo de Evolución Diferencial

ED fue creado por *Storn* y *Price* en 1995 [7]. Surgió como una forma competitiva de la CE, sus principales características son: eficiencia, simplicidad y uso de valores decimales en lugar de números binarios.

Como muchos de los AEs, ED inicia con una población aleatoria que posteriormente se mejora a partir de los operadores de selección, cruza y mutación. Los parámetros del algoritmo ED son: CR, F, y NP. El principal cambio ante los demás AEs radica en los operadores de cruza y mutación. Con la finalidad de distinguir las diferentes versiones de ED, las cuales pueden diferir en las estrategias de mutación y/o los esquemas de cruza que utilizan, se introdujo la notación *ED/x/y/z*, propuesta en [7]. Donde *x* representa el vector que va a ser mutado, *y* denota el número de vectores diferencia utilizados y *z* indica el esquema de cruza empleado. La versión *ED/rand/1/bin* es la más utilizada, su funcionamiento se describe a continuación.

El proceso de reproducción de la versión *ED/rand/1/bin* parte de una población inicial de targets (conjunto de soluciones potenciales al problema), de tal forma que por cada target se seleccionan 3 vectores de forma aleatoria (línea 6 del Algoritmo 3.1), de los cuales se utilizan dos para hacer una diferencia que posteriormente es escalada por el parámetro F (línea 10 del Algoritmo 3.1), el vector resultante es sumado al tercer vector (vector base), operaciones que permiten generar el vector mutante, aunado a esto se lleva a cabo de forma simultanea el proceso de cruza, el cual es controlado por el parámetro CR (línea 9 del Algoritmo 3.1), que determina si cada gen del vector trial (vector descendiente) se tomará del vector target o del vector mutado, cabe resaltar que la variable *jrand* (línea 7 del Algoritmo 3.1) toma un valor aleatorio entre las diferentes dimensiones del vector mutante, con el objetivo de asegurar que al menos un gen del vector trial sea diferente del target. El funcionamiento de *ED/rand/1/bin* se detalla en la Figura 3.3 y el Algoritmo 3.1.

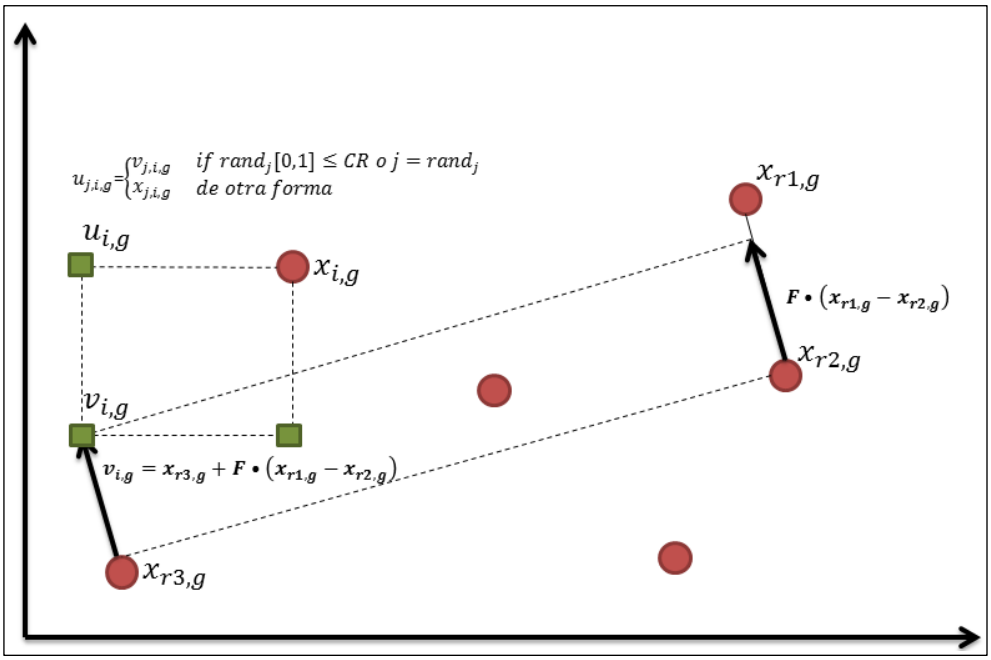


Figura 3.3 Operadores de cruza y mutación de ED aplicados al trial *i* en la generación *g*. $x_{i,g}$: target, $x_{r1,g} \neq x_{r2,g} \neq x_{r3,g} \neq x_{i,g}$: tres vectores elegidos aleatoriamente diferentes entre sí y diferentes al target, v_{ig}: vector base mutado, u_{ig}: *trial*.

Algoritmo 3.1 ED

1: **Inicio**
2: Crear población inicial aleatoria $\boldsymbol{x_{0,i}}\ \boldsymbol{i = 1, \dots, NP}$
3: Evaluación de la población $\boldsymbol{f}\ (\boldsymbol{x_{0,i}})\ \boldsymbol{i = 1, \dots, NP}$
4: **for** $\boldsymbol{g = 0}$ hasta itMax
5: **for** $\boldsymbol{i = 1}$ hasta NP:
6: Seleccionar aleatoriamente r1 ≠ r2 ≠ r3 ≠ i
7: $\boldsymbol{rand_j}$**= rand[1, D]**
8: **for** j = 1 hasta D:
9: **if** $\boldsymbol{rand_j}$ [0, 1) < $\boldsymbol{CR}$ || j = $\boldsymbol{rand_j}$:
10: $\boldsymbol{u_{g,i,j} = x_{g,r3,j} + F(x_{g,r1,j} - x_{g,r2,j})}$
11: **else:**
12: $\boldsymbol{u_{g,i,j} = x_{g,i,j}}$
13: **termina if**
14: **termina for**
15: **if** $\boldsymbol{f(u_{g,i}) \le f(x_{g,i})}$:

16: $x_{g+1,i} = u_{g,i}$
17: **else:**
18: $x_{g+1,i} = x_{g,i}$
19: **termina if**
20: **termina for**
21: **termina for**
22: **fin**

Por otro lado, la versión ED/best/1/bin trabaja de forma muy similar a la versión ED/rand/1/bin. La diferencia entre estas dos variantes radica en el vector mutado, de tal forma que en ED/rand/1/bin por cada target se muta un vector seleccionado de forma aleatoria, mientras que en ED/best/1/bin el vector mutado para cada target es el mejor vector de la población en la generación actual.

3.2.1 Análisis de los parámetros de ED

Como ya se ha descrito en secciones anteriores, la calidad de la configuración de los parámetros de un AE está estrechamente relacionada con el desempeño que éste pueda alcanzar. A continuación se describe el comportamiento esperado de los parámetros del algoritmo ED:

- NP: es el parámetro menos adaptado, sin embargo, su correcta calibración es crucial para que ED alcance buenos resultados. De tal forma que si NP es demasiado pequeño, el algoritmo tiene altas posibilidades de converger prematuramente y/o estancarse en óptimos locales, por el contrario, cuando NP es considerablemente grande, se reducen las posibilidades de que ED converja de forma eficiente. Es decir, el parámetro NP debe ser configurado de tal forma que se mantenga un compromiso exploración-explotación (no ser demasiado grande ni demasiado pequeño).
- F: parámetro sumamente importante durante el proceso de mutación, su función principal es establecer el tamaño de paso. De tal forma que si F es muy pequeño, los vectores mutantes se generarán cerca del vector

base (explotación) y entre mayor sea su valor, los vectores se generará más lejos (exploración).

- CR: su función es controlar el proceso de recombinación o cruza, de tal forma que entre mayor sea su valor, el descendiente (vector trial) tendrá más genes del vector mutante, con lo que se promueve la exploración al generar vectores lo más diferentes posible a los targets y viceversa.

Es importante resaltar que los tres parámetros están mutuamente relacionados, de tal forma que la mala calibración de uno se ve reflejada directamente en los demás. Su configuración debe efectuarse de tal manera que se mantenga un eficiente compromiso exploración-explotación. Como se mencionó anteriormente, se ha demostrado teórica y empíricamente que un algoritmo alcanza mejores resultados cuando sus parámetros son adaptados en diferentes etapas del proceso de búsqueda. De tal forma que si se le quiere brindar a ED la capacidad de explorar en la medida de lo posible, los valores para F y CR deben ser lo más altos permisible y viceversa.

3.3 Clasificación de las técnicas para el manejo de restricciones

Las técnicas tradicionales de la PM son muy limitadas, lo que les prohíbe tratar con problemas de alta complejidad. Esta y otras debilidades de la programación tradicional dieron paso al surgimiento de los AEs, que han tenido gran éxito en una amplia gama de problemas con y sin restricciones.

Para brindarle a un algoritmo la capacidad de resolver problemas restringidos, se han diseñado diversas técnicas, las cuales han sido clasificadas de diferentes formas. En [30], fueron organizadas en las siguientes cuatro categorías: (1) funciones de penalización, (2) operadores especiales, (3) separación de restricciones y objetivos y (4) métodos híbridos.

3.3.1 Funciones de penalización

Es el enfoque más utilizado para el desarrollo de manejadores de restricciones, originalmente propuesto por *Richard Courant* en 1940, expandido por *Carroll* y *Fiacco* y posteriormente por *McCormick*. La idea principal de esta técnica radica en la transformación de un POR en uno sin restricciones, para lo cual, se agrega o sustrae cierto valor de la función objetivo de acuerdo al grado de violación de restricciones que presenta determinada solución. Existen dos tipos de funciones de penalización: exteriores e interiores. Las primeras parten de una solución no factible, donde el objetivo es llegar a la zona factible a partir de dicha solución. Mientras que los métodos internos tienen la característica de que cuando se genera una solución a partir de una que ya es factible, la solución resultante debe pertenecer siempre a la zona factible. El umbral de penalización debe estar entre los límites de las restricciones y debe tender a infinito. Las funciones de penalización pueden tratar con restricciones de igualdad y desigualdad, para lo cual, las restricciones de igualdad son transformadas en restricciones de desigualdad utilizando la siguiente ecuación [30]:

$$|h_j(\vec{x})| - \varepsilon \leq 0 \qquad \textbf{(3.1)}$$

Donde ε es la tolerancia permitida (un valor muy pequeño).

Existen diferentes tipos de funciones de penalización entre los que se encuentran: la pena de muerte, la penalización estática, la penalización dinámica y la penalización adaptativa; las cuales se describen a continuación.

La pena de muerte tiene como principal fundamento el rechazo de los individuos no factibles, es considerada una de las formas más fáciles para manejar las restricciones, donde una solución factible es siempre mejor que una no factible. Es importante calcular el grado de violación de restricciones que tiene cada solución, lo que permite seleccionar la mejor entre dos soluciones no factibles. Esta técnica es normalmente utilizada cuando la zona factible es muy grande, debido a que cuando ésta es muy pequeña la búsqueda se puede

estancar, ya que las capacidades exploratorias de los algoritmos se ven limitadas al no utilizar información de los individuos no factibles. Un ejemplo de manejador de restricciones en esta categoría es el esquema basado en las reglas de factibilidad, propuesto por *Deb* en [8]. Cuando se habla de manejadores de restricciones, éste es sin lugar a duda uno de los referentes. Considerado uno de los más sencillos gracias a su fácil entendimiento e implementación, su funcionamiento está basado en las siguientes tres reglas:

1. Si dos soluciones son factibles, sobresale la solución con mejor valor de aptitud.
2. Si una solución es factible y la otra no, la solución factible es la mejor.
3. Cuando las dos soluciones son no factibles, la mejor es la solución con menor grado de violación de restricciones.

Por otro lado, en la penalización estática se utiliza un factor de penalización, el cual permanece constante durante todo el proceso de búsqueda. Actualmente es poco utilizada, debido a que en los últimos estudios se ha demostrado que mantener los valores estáticos durante toda la búsqueda no lleva a buenos resultados, además de que no todos los PORs pueden ser abordados con los mismos factores de penalización [30].

Al contrario de la penalización estática, en la penalización dinámica los valores de penalización se van adaptando conforme avanza la búsqueda. Algunos investigadores señalan que esta técnica es mejor que la penalización estática, principalmente porque trabaja con PORs arbitrarios. Uno de los manejadores más populares que cabe dentro de esta categoría es ε-constrained method. Método que se basa esencialmente en la transformación de un problema restringido en uno no restringido [9]. Al formar parte de los métodos de penalización, requiere del cálculo del grado de violación de restricciones de cada solución, el cual puede ser obtenido de las siguientes formas:

$$\Phi = max\ (max\ (0,\ g_j(x)),\ max|h_j(x)|) \quad \textbf{(3.2)}$$

$$\Phi = \sum_j ||max\ (0, g_j(x))||^p + \sum_j ||h_j(x)||^p \quad \textbf{(3.3)}$$

Donde p es un número positivo, h y g representan las restricciones de igualdad y desigualdad respectivamente. x_1 y x_2 son dos soluciones a comparar, mientras que $f(x_1)$ y $f(x_2)$ son sus valores de aptitud y $\varphi(x_1)$, $\varphi(x_2)$ representan su grado de violación de restricciones respectivamente. El nivel ε de comparación entre dos soluciones se calcula de la siguiente forma:

$$[f(x_1),\varphi(x_1)] <_{\varepsilon} f(x_2),\varphi(x_2)) \iff \begin{cases} f(x_1) < f(x_2), if\ \varphi(x_1),\varphi(x_2) \leq \varepsilon; \\ f(x_1) < f(x_2), if\ \varphi(x_1),\varphi(x_2) = \varepsilon; \\ \varphi(x_1) < \varphi(x_2), De\ otra\ forma \end{cases} \quad \textbf{(3.4)}$$

Donde $\varepsilon = 0$, las soluciones no factibles se comparan con base en la violación de restricciones. Por otro lado, cuando $\varepsilon = \infty$ las soluciones son comparadas de acuerdo a su valor en la función objetivo. El nivel de ε es controlado con base en las Ecuaciones 3.5 y3.6.

$$\varepsilon(0) = \varphi(x_\Phi) \quad \textbf{(3.5)}$$

$$\varepsilon(t) \quad \begin{cases} \varepsilon(0)(1-\frac{t}{Tc})^{cp} & 0 < t < Tc; \\ 0 & t \leq Tc \end{cases} \quad \textbf{(3.6)}$$

Donde *t* es la iteración actual; *Tc* = número máximo de iteraciones, $x_\Phi$es la *φ-esima* solución, $\theta = 0.2 * N$ y cp es el parámetro que controla la velocidad con la que se reduce la tolerancia de las restricciones ε.

Por último, en la penalización adaptativa uno de los métodos más conocidos es el desarrollado por *Bean* y *Hadj-Alouane* [1992, 1997], donde la función de penalización toma una retroalimentación del proceso de búsqueda. Con este tipo de mecanismos la búsqueda se lleva cabo de una forma más inteligente, tiene la característica de que no permite que toda la población se vuelva factible ni no factible. Su principal desventaja radica en que agrega más parámetros al algoritmo, que requieren de una fina calibración [30].

3.3.2 Operadores especiales

Algunos investigadores han optado por desarrollar esquemas de representación y operadores especiales debido a que los tradicionales no tienen el desempeño requerido cuando los problemas a resolver tienen una alta complejidad. Los decodificadores, son una clase importante dentro de los operadores especiales, se encargan de mapear los cromosomas de la región no factible en la región factible del problema, incluso algunos operadores se diseñan con el propósito de generar descendencia en los límites entre la región factible y la región no factible. Otra idea consiste en transformar la región factible en una forma diferente que sea más fácil de explorar [30].

3.3.3 Separación de restricciones y objetivos

A diferencia de los métodos basados en la pena de muerte, donde se combina el valor de la función objetivo y la violación de restricciones, en esta categoría se manejan las restricciones y los objetivos por separado. Por ejemplo: la coevolución (consiste en dos poblaciones que interactúan entre sí), la superioridad de puntos factibles (la idea base es asignar una mayor aptitud a las soluciones factibles), memoria del comportamiento (donde se busca satisfacer secuencialmente las restricciones de un problema) y el uso del concepto de la optimización multiobjetivo (consiste en convertir un problema mono-objetivo con restricciones en uno multiobjetivo sin restricciones, donde las restricciones son convertidas en las nuevas funciones objetivo) [30].

3.3.4 Métodos híbridos

Dentro de esta categoría están todos los métodos que se combinan con otras técnicas (ya sea heurística o enfoque de la PM). Por ejemplo: (1) *Kim* y *Myung* en 1997 combinaron un método evolutivo con una función lagrangiana aumentada, (2) la optimización restringida por evolución aleatoria (combina una búsqueda de evolución aleatoria con el método de Nelder y Mead), (3) el

sistema hormiga (donde el algoritmo principal es un sistema multi-agente), (4) recocido simulado (desarollado por *Wah* y *Chen* en 2001 donde se combina el algoritmo de recocido simulado con un algoritmo genético), (5) el sistema inmune artificial (basado en el enfoque de selección negativa) y (6) los algoritmos culturales (enfoque micro-evolutivo) [30].

3.4 ED para problemas de optimización con restricciones

ED es un algoritmo que originalmente no fue pensado para abordar PORs, sin embargo, a la fecha se han diseñado múltiples variantes del mismo para tratar este tipo de problemas, a los cuales se les han incorporado mecanismos para el manejo de restricciones, como se puede observar en la Tabla 3.1.

Tabla 3.1 Descripción de variantes del algoritmo de ED para PORs [7-9, 29].

Algoritmo	**Descripción**
Rf-ED	Algoritmo de Evolución Diferencial para PORs, que distingue entre individuos factibles y no factibles a través de las reglas de factibilidad.
ε-ED	Algoritmos de Evolución Diferencial para PORs, que distinguen entre individuos factibles y no factibles a través ε-constrained method.
Rf-EDVC	Algoritmo de Evolución Diferencial con Variantes Combinadas para PORs, que distingue entre individuos factibles y no factibles a través de las reglas de factibilidad y tiene la capacidad de cambiar de una variante (por ejemplo, pasar de ED/ran/1/bin a ED/best/1/bin y viceversa) a otra de acuerdo al porcentaje de individuos factibles (acción controlada por el parámetro PCV) en la generación actual del proceso evolutivo.
ε-EDVC	Algoritmo de Evolución Diferencial con Variantes Combinadas para PORs, que distingue entre individuos factibles y no factibles a través de ε-constrained method y tiene la capacidad de cambiar de una variante (por ejemplo, pasar de ED/ran/1/bin a ED/best/1/bin y

	viceversa) a otra de acuerdo al porcentaje de individuos factibles (acción controlada por el parámetro PCV) en la generación actual del proceso evolutivo.

Ahora bien, algunas variantes de los algoritmos Rf-ED, ε-ED, Rf-EDVC y ε-EDVC se describen en la Tabla 3.2.

Tabla 3.2 Descripción de algunas variantes de los algoritmos Rf-ED, ε-ED, Rf-EDVC y ε-EDVC.

Algoritmo	**Variantes**
Rf-ED	Rf-ED/ran/1/bin
	Rf-ED/best/1/bin
ε-ED	ε-ED/ran/1/bin
	ε-ED/best/1/bin
Rf-EDVC	Rf-EDVC/ran-best (algoritmo que cambia de la variante ED/ran/1/bin a la ED/best/1/bin cuando cierto porcentaje de la población es factible).
	Rf-EDVC/best-ran (algoritmo que cambia de la variante ED/best/1/bin a la ED/ran/1/bin cuando cierto porcentaje de la población es factible).
	Rf-EDVC/ran-best-ran (algoritmo que cambia de la variante ED/ran/1/bin a la ED/best/1/bin y posteriormente regresa a la variante ED/ran/1/bin con base en el porcentaje de individuos factibles).
ε-EDVC	ε-EDVC/ran-best (algoritmo que cambia de la variante ED/ran/1/bin a la ED/best/1/bin cuando cierto porcentaje de la población es factible).
	ε-EDVC/best-ran (algoritmo que cambia de la variante ED/best/1/bin a la ED/ran/1/bin cuando cierto porcentaje de la población es factible).
	ε-EDVC/ran-best-ran (algoritmo que cambia de la variante ED/ran/1/bin a la ED/best/1/bin y posteriormente regresa a la variante ED/ran/1/bin con base en el porcentaje de individuos factibles).

El funcionamiento de los algoritmos Rf-ED, ε-ED, Rf-EDVC y ε-EDVC se detalla en los Algoritmos 3.2-3.5, donde se describen las variantes Rf-ED/ran/1/bin, ε-ED/ran/1/bin, Rf-EDVC/ran-best y ε-EDVC/ran-best respectivamente.

Algoritmo 3.2 RF-ED

1: **Inicio**
2: Crear población inicial aleatoria $\boldsymbol{x_{0,i}}$ i=1,…, NP
3: Evaluación de la población $\boldsymbol{f}\ (\boldsymbol{x_{0,i}})\ \boldsymbol{i = 1, \ldots, NP}$
4: **for** $\boldsymbol{g = 0}$ hasta itMax
5: **for** $\boldsymbol{i = 1}$ hasta NP:
6: Seleccionar aleatoriamente $\mathbf{r1}$, $\mathbf{r2}$, $\mathbf{r3}$, $\mathbf{i}$
7: $\boldsymbol{rand_j = randint[1, D]}$
8: **for** $\boldsymbol{j = 1}$ hasta D:
9: **if** $\boldsymbol{rand_j}$ [0, 1) < $\boldsymbol{CR_{it}}$ || j = $\boldsymbol{rand_j}$:
10: $\boldsymbol{u_{g,i,j} = x_{g,r3,j} + F(x_{g,r1,j} - x_{g,r2,j})}$
11: **else:**
12: $\boldsymbol{u_{g,i,j} = x_{g,i,j}}$
13: **termina if**
14: **termina for**
15: **if** $\boldsymbol{f}(\boldsymbol{u_{g,i}})$ es menor en las 3 reglas que $\boldsymbol{f}(\boldsymbol{x_{g,i}})$:
16: $\boldsymbol{x_{g+1,i} = u_{g,i}}$
17: **else:**
18: $\boldsymbol{x_{g+1,i} = x_{g,i}}$
19: **termina if**
20: **termina for**
21: **termina for**
22: **fin**

Algoritmo 3.3 ε –ED

1: **Inicio**
2: Crear población inicial aleatoria $\boldsymbol{x_{0,i}\ i = 1, \ldots, NP}$
3: Evaluación de la población $\boldsymbol{f}\ (\boldsymbol{x_{0,i}})\ \boldsymbol{i = 1, \ldots, NP}$
4: Inicializar ε
5: **for** $\boldsymbol{g = 0}$ hasta itMax
6: Actualizar el valor de ε
7: **for** $\boldsymbol{i = 1}$ hasta NP:

8: Seleccionar aleatoriamente **r1** , **r2** , **r3** , **i**
9: $\boldsymbol{rand_j}$ = **randint**[**1**, $\boldsymbol{D}$]
10: **for** $\boldsymbol{j = 1}$ hasta D:
11: **if** $\boldsymbol{rand_j}$ [0, 1) < $\boldsymbol{CR_{it}}$ || j = $\boldsymbol{rand_j}$:
12: $\boldsymbol{u_{g,i,j}} = \boldsymbol{x_{g,r3,j}} + \boldsymbol{F}(\boldsymbol{x_{g,r1,j}} - \boldsymbol{x_{g,r2,j}})$
13: **else:**
14: $\boldsymbol{u_{g,i,j}} = \boldsymbol{u_{g,i,j}}$
15: **termina if**
16: **termina for**
17: **if** (Q($\boldsymbol{u_{g,i}}$) && Q($\boldsymbol{x_{g,i}}$)) ≤ ε:
18: **if** f($\boldsymbol{u_{g,i}}$) ≤ f($\boldsymbol{x_{g,i}}$))
19: $\boldsymbol{x_{g+1,i}} = \boldsymbol{u_{g,i}}$
20: **else:**
21: $\boldsymbol{x_{g+1,i}} = \boldsymbol{x_{g,i}}$
22: **if** $\boldsymbol{f}(\boldsymbol{u_{g,i}})$ es menor en las 3 reglas que $\boldsymbol{f}(\boldsymbol{x_{g,i}})$:
23: $\boldsymbol{x_{g+1,i}} = \boldsymbol{u_{g,i}}$
24: **else:**
25: $\boldsymbol{x_{g+1,i}} = \boldsymbol{x_{g,i}}$
26: **termina if**
27: **termina for**
28: **termina for**
29: **fin**

Algoritmo 3.4 RF-EDVC

1: **Inicio**
2: Crear población inicial aleatoria $\boldsymbol{x_{0,i}}$ $\boldsymbol{i = 1, \ldots, NP}$
3: Evaluación de la población $\boldsymbol{f}$ ($\boldsymbol{x_{0,i}}$) $\boldsymbol{i = 1, \ldots, NP}$
4: **for** $\boldsymbol{g = 0}$ hasta itMax
5: Calcular el porcentajeFactibilidad
6: **for** $\boldsymbol{i = 1}$ hasta NP:
7: **if** porcentajeFactibilidad ≥ PCV:
8: Seleccionar aleatoriamente **r1** , **r2** , **i**
9: **else**:
10: Seleccionar aleatoriamente **r1** , **r2** , **r3** , **i**
11: **termina if**
12: $\boldsymbol{rand_j}$ = **randint**[**1**, $\boldsymbol{D}$]

13: **for** j=1 hasta D:
14: **if** $\boldsymbol{rand_j}$ [0, 1) < $\boldsymbol{CR_{it}}$ || j = $\boldsymbol{rand_j}$:
15: **if** porcentajeFactibilidad ≥ PCV:
16: $\boldsymbol{u_{g,i,j}} = \boldsymbol{x_{g,best,j}} + \boldsymbol{F}(\boldsymbol{x_{g,r1,j}} - \boldsymbol{x_{g,r2,j}})$
17: **else:**
18: $\boldsymbol{u_{g,i,j}} = \boldsymbol{x_{g,r3,j}} + \boldsymbol{F}(\boldsymbol{x_{g,r1,j}} - \boldsymbol{x_{g,r2,j}})$
19: **termina if**
20: **else:**
21: $\boldsymbol{u_{g,i,j}} = \boldsymbol{x_{g,i,j}}$
22: **termina if**
23: **termina for**
24: **if** $\boldsymbol{f}(\boldsymbol{u_{g,i}})$ es menor en las 3 reglas que $\boldsymbol{f}(\boldsymbol{x_{g,i}})$:
25: $\boldsymbol{x_{g+1,i}} = \boldsymbol{u_{g,i}}$
26: **else:**
27: $\boldsymbol{x_{g+1,i}} = \boldsymbol{x_{g,i}}$
28: **termina if**
29: **termina for**
30: **termina for**
31: **fin**

Algoritmo 3.5 ε –EDVC

1: **Inicio**
2: Crear población inicial aleatoria $\boldsymbol{x_{0,i}}\ \boldsymbol{i} = \boldsymbol{1}, \dots, \boldsymbol{NP}$
3: Evaluación de la población $\boldsymbol{f}\ (\boldsymbol{x_{0,i}})\ \boldsymbol{i} = \boldsymbol{1}, \dots, \boldsymbol{NP}$
4: Inicializar ε
5: **for** $\boldsymbol{g} = \boldsymbol{0}$ hasta itMax
6: Actualizar el valor de ε
7: Calcular el porcentajeFactibilidad
8: **for** $\boldsymbol{i} = \boldsymbol{1}$ hasta NP:
9: **if** porcentajeFactibilidad ≥ PCV:
10: Seleccionar aleatoriamente **r1** , **r2** , **i**
11: **else**:
12: Seleccionar aleatoriamente **r1** , **r2** , **r3** , **i**
13: **termina if**
14: jrand = randint[$\boldsymbol{1}, \boldsymbol{D}$]
15: **for** j=1 hasta D:
16: **if** $\boldsymbol{rand_j}$ [0, 1) < $\boldsymbol{CR_{it}}$ || j = $\boldsymbol{rand_j}$:

17: **if** porcentajeFactibilidad ≥ PCV:
18: $u_{g,i,j} = x_{g,best,j} + F(x_{g,r1,j} - x_{g,r2,j})$
19: **else:**
20: $u_{g,i,j} = x_{g,r3,j} + F(x_{g,r1,j} - x_{g,r2,j})$
21: **termina if**
22: **else:**
23: $u_{g,i,j} = u_{g,i,j}$
24: **termina if**
25: **termina for**
26: **if** (Q($u_{g,i}$) && Q($x_{g,i}$)) ≤ ε:
27: **if** f($u_{g,i}$) ≤ f($x_{g,i}$))
28: $x_{g+1,i} = u_{g,i}$
29: **else:**
30: $x_{g+1,i} = x_{g,i}$
31: **if** $f(u_{g,i})$ es menor en las 3 reglas que $f(x_{g,i})$:
32: $x_{g+1,i} = u_{g,i}$
33: **else:**
34: $x_{g+1,i} = x_{g,i}$
35: **termina if**
36: **termina for**
37: **termina for**
38: **fin**

Capítulo 4
Configuración de parámetros

Desde que surgieron los primeros AEs fueron fuertemente criticados por la fina configuración de parámetros que requieren, lo que se convirtió en una de sus principales debilidades. Actualmente, los investigadores del área han llevado a cabo grandes esfuerzos con el fin de minimizar el impacto de dicha configuración, estudios que se clasifican en: calibración de parámetros y control de parámetros, como se puede observar en la Figura 4.1 [4], donde se puede observar que el enfoque determinista está sombreado en gris, esto debido a que los parámetros del algoritmo diseñado en esta investigación fueron controlados bajo dicho enfoque.

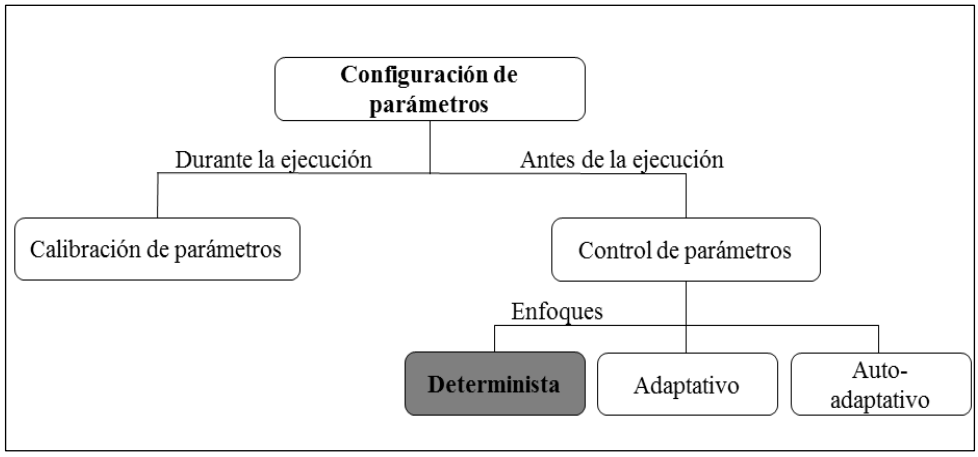


Figura 4.1 Taxonomía de las técnicas para la configuración de parámetros.

4.1 Calibración de parámetros

La calibración de parámetros se da durante la etapa de diseño, consiste en encontrar los valores óptimos para los parámetros que requiere un AE a través de una experimentación exhaustiva, dichos valores se establecen antes de ejecutar el algoritmo y se mantienen fijos durante la búsqueda. Cabe resaltar

que probar todos los posibles valores que puede tomar cada parámetro y sus combinaciones, es un proceso que requiere de mucho tiempo. De tal manera que si por ejemplo se requiere ajustar 4 parámetros y cada parámetro puede tomar cuatro diferentes valores, se tendrían que probar $4^4 = 256$ diferentes configuraciones. Si se realizan 100 ejecuciones independientes con cada configuración para validar los resultados obtenidos, implicaría llevar a cabo 25, 600 corridas sólo para establecer un buen diseño del algoritmo.

Desde que surgieron los primeros AEs, se han propuesto diversos esquemas para localizar valores de parámetros, que lleven a buenos resultados en diversos problemas empleando el ajuste de parámetros. Por ejemplo, algunos valores y/o rangos de valores propuestos para los parámetros F y/o CR del algoritmo ED se pueden observar en la Tabla 4.1 [4].

Tabla 4.1 Valores propuestos para F y CR.

F	**CR**	**Referencia**
[0.4, 1]	0.1, 0.9	Storn, Price 1997
0.9	0.9	Liu, Lampinen 2002
	0.5	Ali, Törn 2004
0.9	0.9	Rönkkönen et al. 2005
0.5	0.1, 0.5	Kaelo, Ali 2006
≥ 0.6	≥ 0.6	Zielinski et al. 2006
[0.35, 0.37]	0.5	Salman et al. 2007

4.2 Control de parámetros

El control de parámetros es una forma alternativa de tratar la configuración de parámetros, técnica que se presenta durante la etapa de ejecución. De tal forma que el algoritmo arranca con valores iniciales para cada parámetro que se van actualizando en cada ciclo del proceso evolutivo utilizando diversas reglas. Dicha modificación se lleva a cabo a partir de una retroalimentación basada en el estado actual de la búsqueda o del simple paso de los ciclos.

De acuerdo a [6], al implementar un mecanismo de control de parámetros se deben de tomar en cuenta algunos factores como son: qué se va a cambiar (cuál o cuáles parámetros), cómo se va a llevar cabo el cambio (enfoque a utilizar), base del cambio (diversidad de la población, dispersión de las soluciones, etc.) y el nivel del cambio (poblacional, por individuo, etc.). De tal forma que los enfoques para el control de parámetros se pueden clasificar en: deterministas, adaptativos y auto-adaptativos.

A la fecha se han diseñado diversos esquemas para controlar los parámetros de un algoritmo, por ejemplo, en [31] se propusieron 6 diferentes esquemas bajo un enfoque determinista para adaptar los parámetros F y/o CR del algoritmo ED para problemas de optimización sin restricciones. Estos esquemas tienen como base una función senoidal, donde los parámetros adaptados pueden iniciar con valores cercanos a 0.5 y comenzar a crecer conforme avanza el proceso evolutivo, o pueden llevar a cabo el comportamiento contrario, es decir, comenzar con valores alejados a 0.5 e ir acercándose poco a poco con forme avanza la búsqueda.

Por otro lado, en [32] fue propuesto un conjunto de funciones para adaptar un parámetro de un algoritmo PSO (particle swarm optimization) bajo un enfoque determinista. Este algoritmo fue desarrollado para resolver problemas de optimización sin restricciones. Las funciones propuestas tienen un comportamiento creciente, las cuales se enlistan a continuación.

- x
- $\sqrt[2]{x}$
- $\sqrt[4]{x}$
- x^2
- x^4
- $\mathrm{X} - \frac{SIN(2\pi\, x)}{6.3}$

En lo que respecta al ámbito restringido, en la literatura especializada se pueden encontrar diversos algoritmos que utilizan mecanismos para el control de parámetros para resolver PORs, por ejemplo: Saber M. et al. propusieron en [6] un framework que trabaja con los algoritmos: ED, AG, EE y PE, cada uno de ellos tiene su propia sub-población y cada sub-población utiliza sus propios parámetros de búsqueda, donde todos los operadores y los tamaños de población se actualizan de forma adaptativa. Mallipeddi et al. [10] proponen el uso de una mezcla de cuatro técnicas para el manejo de restricciones (CHTs) con base en ED para resolver PORs, donde se inicializan poblaciones diferentes, cada población utiliza una determinada CHT, además utilizan mezclas de las estrategias de mutación y valores para los parámetros F y CR entre los rangos: [0.1, 0.9] y [0.4, 0.9] respectivamente. V. L. Huang y A. K. Qin propusieron una variante del algoritmo SaDE para resolver PORs donde el criterio de remplazo fue modificado para diferenciar entre individuos factibles y no factibles utilizando las reglas de Deb [11].

Otros algoritmos relevantes en la literatura especializada son: EM que simula la teoría física del electromagnetismo [12], jDE-2 que es una mejora a jDE donde se utilizó un mecanismo para el control de individuos factibles basado en una técnica de penalización [13] y ASCHEA que utiliza un mecanismo adaptativo a nivel poblacional [14]. Por otro lado, Farmani, R. y Wright, J. propusieron en [15] una formula auto-adaptativa con base en el valor de la función objetivo. En [16] se propuso un esquema auto-adaptativo donde a cada individuo se le asigna una secuencia aleatoria de restricciones a evaluar; secuencias que evolucionan conforme avanza el proceso de búsqueda. Otro algoritmo importante es DCDE que básicamente consiste en establecer un rango de valores para los parámetros F y CR [17]. Saber M. et al. propusieron en [18] una variante del algoritmo ED que usa múltiples operadores de variación y técnicas para el manejo de restricciones y en [19] dos algoritmos auto-adaptativos incorporados a una heurística mixta de operadores (DE/HMO). Caponio et al. propusieron el algoritmo memético veloz (FAMA) en [20]. Por

último, Tanabe y Fukunaga propusieron LSHADE en [33], que utiliza una memoria para almacenar valores que fueron buenos en generaciones pasadas.

Capítulo 5
Trabajo propuesto

5.1 Diseño experimental

La presente investigación puede ser catalogada como experimental o de laboratorio, debido a que se hicieron varios estudios (exploratorios, descriptivos y explicativos) siguiendo el método inductivo, de tal forma que se analizaron casos particulares (PORs sintéticos) para extraer conclusiones de carácter general. Este trabajo fue realizado bajo la motivación de diseñar un nuevo algoritmo para PORs, que utilice un mecanismo para el control de los parámetros F y/o CR basado en funciones matemáticas y una técnica competitiva para el manejo de restricciones. El diseño experimental se describe en las siguientes secciones.

5.1.1 Diseño e implementación de algoritmos adaptativos de ED para PORs

Las nuevas variantes fueron diseñadas con base en los algoritmos ED y EDVC, las cuales pueden utilizar el manejador de restricciones ε-constrained method (ε) o las reglas de factibilidad (Rf), lo que les permite resolver PORs. Además se les incorporaron mecanismos para el control de los parámetros F y/o CR basados en funciones matemáticas.

Se utilizaron 2 conjuntos de funciones, el primero está compuesto por 6 esquemas (Conf1,…, Conf6), los cuales están basados en dos versiones de una función Senoidal (S), una con comportamiento creciente y otra con decreciente. Estos esquemas fueron propuestos en [31] para adaptar los parámetros del algoritmo ED en problemas de optimización sin restricciones. La Tabla 5.1 contiene los esquemas del conjunto 1, donde se puede observar que algunos esquemas adaptan ambos parámetros (F y CR) y otros solo uno. Como ejemplo, el comportamiento esperado del esquema Conf1 se muestra en la Figura 5.1, donde el eje *x* indica los valores que pueden tomar los parámetros F y CR en

cada generación *y*. Como se puede apreciar, el parámetro CR se mantiene fijo en 0.9, mientras que F va tomando valores que oscilan entre 0 y 1, los cuales son generados utilizando la función senoidal. El segundo conjunto está compuesto por funciones Lineales, Exponenciales y Senoidales (LES), las cuales fueron propuestas originalmente con un comportamiento creciente en [30] para adaptar un parámetro de un algoritmo PSO. Para efectos de esta investigación se agregó una nueva versión por cada una de las ya existentes con un comportamiento decreciente, de igual forma se agregó la función senoidal del primer conjunto en ambas versiones (creciente y decreciente), así como los valores de parámetro 0.5 y 0.9 para F y CR respectivamente. Las funciones se pueden observar en la Tabla 5.2 y su comportamiento esperado en la Figura 5.2, donde el eje *x* representa el valor que pueden tomar los parámetros en cada generación *y*. De tal forma que los valores de *x* son generados a través de los mecanismos para el control de parámetros del conjunto 2, donde las funciones precedidas por el signo (-) representan el comportamiento descendiente. En las funciones del conjunto 2, la diferencia entre una función con comportamiento creciente y una decreciente radica en la forma de generar el valor de *x*, para lo cual se utiliza la generación actual (*Generación*) y el número máximo de generaciones (*NumMaxGeneraciones*). Las Ecuaciones 5.1 y 5.2 se utilizan para generar el valor de *x* de una función creciente y una decreciente respectivamente.

$$\frac{Generación}{NumMaxGeneraciones} \quad \textbf{(5.1)}$$

$$\frac{NumMAxGeneraciones - Generación}{NumMaxGeneraciones} \quad \textbf{(5.2)}$$

Tabla 5.1 Conjunto 1: Mecanismos para el control adaptativo dinámico de parámetros basados en una función Senoidal (S). *freq*: 0.5, it: generación actual y $itMax$:número máximo de evaluaciones.

Conf1	**Conf2**
$\mathrm{F}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{it}{itMax} + 1\right)$ $\mathrm{CR}_{it} = 0.9$	$\mathrm{F}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{it}{itMax} + 1\right)$ $\mathrm{CR}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{it}{itMax} + 1\right)$
Conf3	**Conf4**
$\mathrm{F}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{itMax - it}{itMax} + 1\right)$ $\mathrm{CR}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{itMax - it}{itMax} + 1\right)$	$\mathrm{F}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{itMax - it}{itMax} + 1\right)$ $\mathrm{CR}_{it} = 0.9$
Conf5	**Conf6**
$\mathrm{F}_{it} = 0.5$ $\mathrm{CR}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{it}{itMax} + 1\right)$	$\mathrm{F}_{it} = 0.5$ $\mathrm{CR}_{it} = \frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{itMax - it}{itMax} + 1\right)$

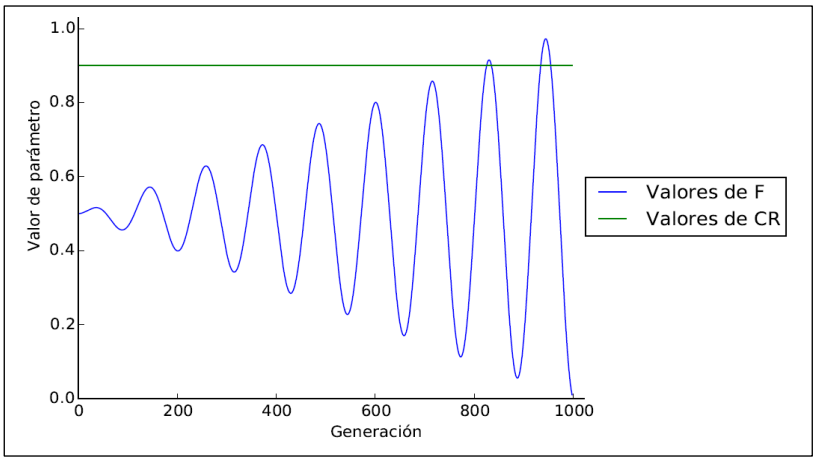


Figura 5.1 Comportamiento esperado del esquema Conf1.

Tabla 5.2 Conjunto 2: Funciones Lineales, Exponenciales y Senoidales (LES). *freq*: 0.5, it: generación actual e $itMax$: número máximo de evaluaciones.

Parámetro F	**Parámetro CR**
x	x
$\sqrt[2]{x}$	$\sqrt[2]{x}$

$\sqrt[4]{x}$	$\sqrt[4]{x}$
x^2	x^2
x^4	x^4
$x - \frac{sin(2\pi\, x)}{6.3}$	$x - \frac{sin(2\pi\, x)}{6.3}$
$\frac{1}{2} * \left(\sin(2\pi * freq * it) * \frac{it}{itMax} + 1\right)$	$\frac{1}{2} * \left(\sin(2\pi * freq * it) * \frac{it}{itMax} + 1\right)$
0.5	0.9
$-x$	$-x$
$-\sqrt[2]{x}$	$-\sqrt[2]{x}$
$-\sqrt[4]{x}$	$-\sqrt[4]{x}$
$-x^2$	$-x^2$
$-x^4$	$-x^4$
$-x - \frac{sin(2\pi\, x)}{6.3}$	$-x - \frac{sin(2\pi\, x)}{6.3}$
$-\frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{itMax - it}{itMax} + 1\right)$	$\frac{1}{2} * \left(sin(2\pi * freq * it) * \frac{itMax - it}{itMax} + 1\right)$

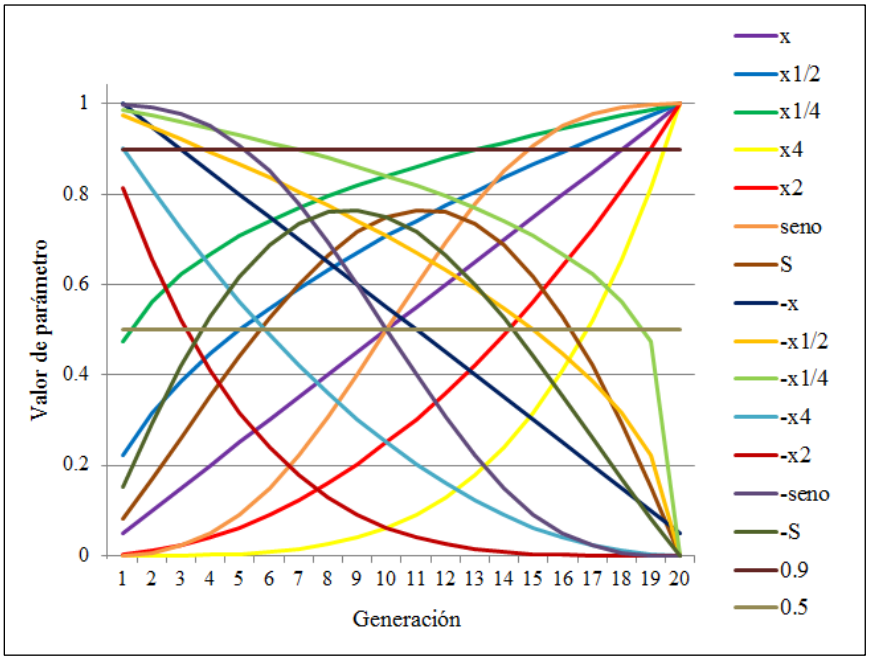


Figura 5.2 Comportamiento esperado de las funciones lineales, exponenciales y senoidales del conjunto 2.

Cabe resaltar que se generaron 224 esquemas para el control de parámetros con las funciones del conjunto 2, debido a que se hicieron todas las combinaciones posibles entre dichas funciones, de tal forma que en la Tabla 5.3 cada cuadro de color blanco puede representar la intersección de dos funciones (donde cada función adapta un parámetro) o de una función con un valor de parámetro fijo (donde sólo se adapta uno de los parámetro). El cuadro negro representa la combinación de los valores 0.5 y 0.9 para los parámetros F y CR respectivamente, esta combinación no está permitida, ya que se generaría un esquema de calibración y no uno de control.

Tabla 5.3 Descripción de los esquemas generados con las funciones del Conjunto 2. $S=\frac{1}{2}*\left(\sin(2\pi*freq*it)*\frac{it}{itMax}+1\right)$, $-S=\frac{1}{2}*\left(\sin(2\pi*\text{freq}*\text{it})*\frac{\text{itMax}-\text{it}}{\text{itMax}}+1\right)$.

F / CR	x	√x	∜x	x^2	x^4	Sen	S	0.5	-x	-√x	-∜x	-x^2	-x^4	-Sen	-S
x															
√x															
∜x															
x^2															
x^4															
Sen															
S															
0.9															
-x															
-√x															
-∜x															
-x^2															

-x^4															
-Sen															
-S															

Como resultado de la combinación de algunas variantes de los algoritmos estudiados (ED y EDVC), las técnicas para el manejo de restricciones ε y Rf, así como los mecanismos para el control de parámetros previamente señalados (Conjunto 1 y Conjunto 2), se diseñaron 284 variantes. En la Tabla 5.4 se puede observar el detalle de los algoritmos generados.

Tabla 5.4 Detalle de los algoritmos diseñados. ran: ED/ran/1/bin, best: ED/best/1/bin.

Manejador de restricciones	**Algoritmo**	**Variante**	**Mecanismos para el control de parámetros**	**Versiones diseñadas**
Rf	ED	ran	Conjunto 1	6
		best	Conjunto 1	6
		best	Conjunto 2	224
ε	ED	ran	Conjunto 1	6
		best	Conjunto 1	6
Rf	EDVC	ran-best	Conjunto 1	6
		best-ran	Conjunto 1	6
ε	EDVC	ran-best	Conjunto 1	6
		best-ran	Conjunto 1	6
		ran-best-ran	Conjunto 1	6
		best-ran-best	Conjunto 1	6
Total de variantes generadas				**284**

Los algoritmos diseñados fueron nombrados de acuerdo a la sintaxis detallada en la Tabla 5.5. De tal forma que el algoritmo ED/ran/1/bin con el manejador de restricciones ε-constrained method y el mecanismo para el control de parámetros basado en las funciones F= x y CR= $\sqrt[2]{x}$ lleva por nombre ε-ED-LES ran-x-$\sqrt[2]{x}$, mientras que el mismo algoritmo con el mecanismo para el

control de parámetros Conf3 se puede identificar con el nombre ε-DE-S ran-Conf3.

Tabla 5.5 Sintaxis para los nombres de los algoritmos diseñados. Fr: reglas de factibilidad, ε: ε-constrained method, ED: evolución diferencial, EDVC: evolución diferencial con variantes combinadas, S: función senoidal, LES: funciones lineales, exponenciales y senoidales, ran: ED/ran/1/bin, best: ED/best/1/bin.

Manejador de restricciones	Algoritmo de ED	Conjunto de funciones	Variante	Función del parámetro F	Función del parámetro CR
○ Rf ○ ε	ED	S	○ ran ○ best	Conf1, …, Conf6	
		LES		0.5 x, $\sqrt[2]{x}$, $\sqrt[4]{x}$, x^2, x^4, etc.	0.9 x, $\sqrt[2]{x}$, $\sqrt[4]{x}$, x^2, x^4, etc.
○ Rf ○ ε	EDVC	S	○ ran-best ○ best-ran ○ ran-best-ran ○ best-ran-best	Conf1, …, Conf6	
		LES		0.5 x, $\sqrt[2]{x}$, $\sqrt[4]{x}$, x^2, x^4, etc.	0.9 x, $\sqrt[2]{x}$, $\sqrt[4]{x}$, x^2, x^4, etc.

5.1.2 Experimentación y pruebas de los algoritmos de ED adaptativos para PORs diseñados

El desempeño de los algoritmos diseñados fue medido bajo un enfoque experimental, utilizando un conjunto de funciones tomadas de la literatura especializada, específicamente los problemas del benchmark del CEC2010 [21], funciones que están definidas de tal forma que se puede cambiar su dimensión (número de variables de decisión). Primero se llevó a cabo la experimentación en 10 dimensiones con todas las variantes diseñadas, lo que permitió identificar a los algoritmos con el mejor rendimiento. Posteriormente, dichos algoritmos fueron probados con las funciones en 30 dimensiones, para verificar si se podía

generalizar su desempeño, experimentación que permitió detectar la versión más robusta en ambas dimensiones. Los resultados experimentales se presentan en la Sección 5.2.1 Resultados de la experimentación de los algoritmos de ED adaptativos para PORs diseñados.

Para medir el rendimiento de cada algoritmo, se llevaron a cabo 25 ejecuciones independientes por cada función en 10 y 30 dimensiones, para las cuales se calcularon dos medidas de calidad comúnmente usadas: probabilidad de convergencia (P) y velocidad de convergencia (AFES) [34]. La primera medida permite conocer el porcentaje de ejecuciones en las que el algoritmo alcanzó la mejor solución conocida de cada problema. De tal forma que el valor de P se obtiene dividiendo el número de ejecuciones exitosas (ejecuciones en las que se encontró la mejor solución que se conoce hasta ahora de cada problema) entre el número de ejecuciones realizadas. Por otro lado, la medida AFES devuelve el promedio de evaluaciones necesarias para encontrar la mejor solución de cada problema, para lo cual se hace una suma del número de evaluaciones utilizadas para encontrar la solución (donde sólo se cuentan las evaluaciones de las ejecuciones exitosas), suma que se divide por el número de ejecuciones exitosas. Para efectos de esta investigación, un problema se considera resuelto cuando al ser abordado por determinado algoritmo, éste tiene por lo menos una ejecución exitosa. Por otro lado, un algoritmo es robusto cuando tiene la capacidad de resolver una amplia gama de problemas con una alta probabilidad de convergencia.

Los parámetros que no fueron adaptados, se calibraron conforme a la literatura especializada, de tal forma que la Tabla 5.6 contiene los parámetros de los algoritmos que utilizaron las reglas de factibilidad para manejar las restricciones, mientras que los parámetros de las variantes que trataron las restricciones con ε-constrained method se detallan en la Tabla 5.7. Los demás parámetros (F y CR) fueron adaptados de acuerdo a los mecanismos detallados en las Tablas 5.1 y 5.2.

Tabla 5.6 Parámetros utilizados en los algoritmos que manejaron las restricciones con las reglas de factibilidad. NP: Tamaño de población, MaxEval: número de evaluaciones, freq: frecuencia de oscilación de la función seno.

Dimensión	NP	MaxEval	freq
10D	90	200000	0.5
30D	90	600000	0.5

Tabla 5.7 Parámetros utilizados en los algoritmos que manejaron las restricciones con ε-constrained method. NP: Tamaño de población MaxEval: número de evaluaciones, freq: frecuencia de oscilación de la función seno, cp: control de la velocidad de reducción de la tolerancia de la restricción, Tc: número de iteraciones de ε.

Dimensión	NP	MaxEval	freq	cp	Tc
10D	90	200000	0.5	5	0.2 * itMax
30D	90	600000	0.5	5	0.2 * itMax

5.1.3 Estudio del comportamiento algorítmico del mejor algoritmo diseñado

La segunda etapa de la experimentación consistió en llevar a cabo un estudio algorítmico de la variante con mejor desempeño en 10 y 30 dimensiones, donde el algoritmo más robusto fue ε-ED-S ran-Conf1. Dicho algoritmo tiene las siguientes características: (1) utiliza la variante ED/ran/1/bin, (2) el manejador de restricciones ε-constrained method y (3) el mecanismo para el control de parámetros Conf1, donde F se adapta con la función senoidal de forma creciente y CR se mantiene fijo en 0.9. El análisis del proceso algorítmico fue realizado con base en la generación de gráficas de factibilidad (donde se plasmó el porcentaje de individuos factibles generación tras generación de la mejor, media y peor ejecución respecto a la calidad de la solución alcanzada) de las funciones en 10 dimensiones, lo que permitió conocer la velocidad y las etapas de la búsqueda en las que la población llegó a la zona factible. Los resultados

experimentales se presentan en la Sección 5.2.2 Resultados del comportamiento algorítmico del mejor algoritmo diseñado.

5.1.4 Rediseño del mejor algoritmo desarrollado

El análisis de los resultados del estudio del comportamiento algorítmico de ε-DE-S ran-Conf1 en 10 dimensiones, permitió identificar sus debilidades y fortalezas, donde principalmente se detectó que la calibración de los parámetros del manejador de restricciones Tc y cp, estaba limitando sus capacidades de exploración y explotación (detallado en las Secciones 5.2.2 y 5.3.2). Con base en dicho análisis, se llevó a cabo el rediseño del algoritmo ε-DE-S ran-Conf1, mejora que básicamente consistió en calibrar los parámetros Tc y cp. Se diseñaron 63 nuevas variantes, las cuales difieren por la calibración de los parámetros del manejador de restricciones. Esta etapa se dividió en dos fases, en la primera sólo se calibró el parámetro Tc con los valores: 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 y 0.9. Los algoritmos generados fueron probados en 10 dimensiones bajo las condiciones señaladas en la primera experimentación. Posteriormente, en la segunda fase se procedió a calibrar el parámetro cp, donde se utilizaron los valores: 1, 2, 3, 4, 5, 6, 7, 8 y 9. De tal forma que por cada valor de Tc se generaron 9 variantes. Los algoritmos diseñados fueron evaluados en 10 dimensiones, nuevamente bajo las mismas condiciones. El análisis de los resultados de dicha experimentación, permitió identificar a los algoritmos con mejor desempeño, los cuales fueron probados nuevamente, ahora con las funciones en 30 dimensiones, esto con el objetivo de seleccionar la versión más robusta. Es decir, el algoritmo con la mejor combinación de valores para los parámetros Tc y cp. Calibración que le brindó una mejor capacidad de exploración y explotación al algoritmo diseñado. Los resultados experimentales se presentan en la Sección 5.2.3 Resultados del rediseño del mejor algoritmo desarrollado.

5.1.5 Análisis de diversidad del mejor algoritmo rediseñado

Una vez rediseñado el algoritmo ε-DE-S ran-Conf1, se llevó a cabo el estudio del comportamiento algorítmico de 6 diferentes versiones de ED para PORs. Algoritmos que fueron fundamentales en diferentes etapas de esta investigación. El objetivo primordial de este experimento fue conocer el efecto de cada uno de los elementos incorporados al algoritmo de ED original. Las variantes analizadas y sus características se dividieron en 3 grupos, los cuales se describen a continuación:

1. Rf-ED ran y ε-ED ran. Variantes de ED/ran/1/bin para PORs que no tienen incorporado un mecanismo para el control de parámetros y difieren en el manejador de restricciones que utilizan.
2. Rf-ED-S ran-Conf1 y ε-ED-S ran-Conf1. Variantes de ED/ran/1/bin para PORs que adaptan sus parámetros con el esquema adaptativo Conf1 (donde F se adapta de forma creciente con base en una función senoidal y CR se mantiene fijo en 0.9) y difieren por el manejador de restricciones que utilizan.
3. ε-ED-S ran-Conf1 (Tc=0.6, cp=1) y ε-ED-S ran-Conf1 (Tc=0.9, cp=1). Variantes de ED/ran/1/bin para PORs que utilizan el manejador de restricciones ε-constrained method y el esquema adaptativo Conf1, las cuales difieren, por la calibración de sus parámetros Tc y cp del manejador de restricciones ε-constrained method.

El estudio consistió en analizar la forma en la que cada algoritmo manejó la diversidad al resolver cada problema en 10 y 30 dimensiones, para lo cual se utilizó la métrica: distancia al vector promedio [32], que básicamente consiste en generar el vector promedio, para posteriormente medir la distancia de cada vector de la población a dicho vector promedio, lo que permite conocer que tan dispersa se encuentra la población en el espacio de búsqueda generación tras generación. Los resultados experimentales se presentan en la Sección 5.2.4 Resultados del análisis de diversidad del mejor algoritmo rediseñado.

5.2 Resultados

5.2.1 Resultados de la experimentación de los algoritmos de ED adaptativos para PORs diseñados

Durante esta etapa experimental, se probaron las 284 variantes diseñadas con las funciones en 10 dimensiones. Las Tablas 5.8 y 5.10 contienen los resultados más sobresalientes de este estudio, donde las columnas 1 y 2 describen las funciones y las medidas de calidad empleadas respectivamente, mientras que las columnas restantes contienen los resultados obtenidos por cada algoritmo en las medidas y funciones estudiadas. Los demás resultados se pueden observar en el Anexo II. Además, se generó un resumen de las Tablas 5.8 y 5.10, que se puede observar en las Tablas 5.9 y 5.11 respectivamente, donde el algoritmo señalado en la primer columna es comparado contra los demás respecto a la medida P, dicha comparación se llevó a acabo utilizaron los siguientes criterios: funciones ganadas (donde el algoritmo de la columna 1 obtuvo mayor porcentaje de convergencia), perdidas (donde obtuvo menos) y empatadas (donde tuvieron igual número de ejecuciones exitosas). Estas tablas fueron generadas con el objetivo de facilitar la identificación de los algoritmos más robustos durante esta experimentación.

Tabla 5.8 Comparativa entre los algoritmos: ε-ED-S best-Conf1, ε-ED-S best-Conf2, ε-ED-S best-Conf3, ε-ED-S best-Conf4, ε-ED-S best-Conf5 y ε-ED-S best-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	0.04	0.8	0.92	0.04	0.8	0.12
	AFES	2.12e+03	8.19e+03	1.32e+04	1.10e+04	9.33e+03	5.92e+03
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0	0.56	0.28	0	0.2	0
	AFES		1.83e+05	1.19e+05		1.47e+05	
C4	**P**	0.44	0.96	1	0.76	1	0.92
	AFES	7.90e+03	3.64e+04	3.43e+04	2.24e+04	3.18e+04	2.78e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0.04	0	0	0
	AFES			5.25e+04			

C7	**P** **AFES**	0	0.64 6.49e+04	.68 5.07e+04	0	0.92 6.03e+04	0
C8	**P** **AFES**	0	0.12 3.84e+04	0.12 4.00e+04	0	0.12 3.34e+04	0.08 7.70e+04
C9	**P** **AFES**	0	0	0.04 1.94e+05	0	0.04 2.30e+04	0
C10	**P** **AFES**	0	0	0	0	0	0
C11	**P** **AFES**	0.2 3.11e+04	0.52 3.91e+04	0.64 4.78e+04	0.16 1.90e+04	0.72 4.84e+04	0.4 5.38e+04
C12	**P** **AFES**	0	0	0	0	0.04 1.36e+04	0
C13	**P** **AFES**	0	0.12 2.11e+04	0.36 2.94e+04	0.04 2.10e+04	0.16 2.10e+04	0.16 2.59e+04
C14	**P** **AFES**	0	0.56 6.51e+04	0.48 7.17e+04	0	0.36 6.67e+04	0.12 1.11e+05
C15	**P** **AFES**	0	0.04 1.42e+05	0.04 4.62e+04	0	0.04 3.04e+04	0
C16	**P** **AFES**	0	0.2 2.10e+04	0.52 2.31e+04	0	0.36 1.95e+04	0.08 5.97e+04
C17	**P** **AFES**	0	0.08 1.00e+05	0.08 5.82e+04	0.12 1.82e+05	0.16 1.23e+05	0
C18	**P** **AFES**	0	0.08 1.16e+05	0.04 1.36e+05	0.121.54e+05	0	0

Tabla 5.9 Resumen de la medida P en los algoritmos: ε-ED-S best-Conf1, ε-ED-S best-Conf4 y ε-ED-S best-Conf6 en 10 dimensiones.

Conf3	**Conf2**	**Conf5**
Ganados(+)	8	7
Perdidos(-)	3	3
Empates(=)	7	8

Tabla 5.10 Comparativa entre los algoritmos: ε-ED-S ran-Conf1, ε-ED-S ran-Conf2, ε-ED-S ran-Conf3, ε-ED-S ran-Conf4, ε-ED-S ran-Conf5 y ε-ED-S ran-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	**Métrica**	**Conf1**	**Conf2**	**Conf3**	**Conf4**	**Conf5**	**Conf6**
C1	**P** **AFES**	1 5.51e+04	1 3.03e+04	1 3.29e+04	.0.56 1.60e+04	1 2.93e+04	1 3.38e+04
C2	**P** **AFES**	0.24 1.07e+04	0.04 8.55e+03	0	0.08 1.00e+05	0.16 4.16e+04	0.08 7.31e+04
C3	**P** **AFES**	0.92 7.04e+04	0.04 1.95e+05	0	0.28 7.31e+04	0.12 1.62e+05	0.04 1.11e+05

C4	**P** **AFES**	1 4.43e+04	1 6.53e+04	1 7.76e+04	0.52 5.30e+04	1 6.83e+04	1 5.54e+04
C5	**P** **AFES**	0	0	0	0	0	0
C6	**P** **AFES**	0.04 4.67e+04	0	0	0.08 3.67e+04	0	0
C7	**P** **AFES**	1 8.07e+04	0.96 1.72e+05	0.92 1.59e+05	0.84 9.56e+04	0.04 1.06e+05	0.04 9.07e+04
C8	**P** **AFES**	0.4 5.50e+04	0.84 1.30e+05	0.92 1.35e+05	0.2 6.98e+04	0.96 6.77e+04	0.68 1.12e+05
C9	**P** **AFES**	0.48 6.69e+04	0	0	0	0.16 1.44e+05	0.24 1.48e+05
C10	**P** **AFES**	0.04 7.01e+04	0	0	0.04 4.23e+04	0	0
C11	**P** **AFES**	1 7.58e+04	1 7.39e+04	1 5.92e+04	0.88 6.97e+04	0.76 6.77e+04	0.96 5.36e+04
C12	**P** **AFES**	0.16 6.55e+04	0.2 1.16e+05	0.36 9.24e+04	0.08 6.61e+04	0.28 9.00e+04	0.36 5.58e+04
C13	**P** **AFES**	1 1.05e+05	1 1.24e+05	1 9.24e+04	0.2 1.02e+05	1 1.22e+05	1 1.22e+05
C14	**P** **AFES**	0.68 9.24e+04	0.6 1.68e+05	0.52 1.79e+05	0.52 1.02e+05	0	0.12 1.16e+05
C15	**P** **AFES**	0.04 5.74e+04	0.16 1.64e+05	0.36 1.63e+05	0.04 8.77e+04	0.08 1.27e+05	0.28 1.15e+05
C16	**P** **AFES**	0	0.44 2.95e+04	0.48 3.14e+04	0.16 2.24e+04	0.24 3.16e+04	0.4 2.93e+04
C17	**P** **AFES**	0.76 2.65e+04	0	0	0.72 2.54e+04	0	0
C18	**P** **AFES**	0.72 3.23e+04	0.04 1.43e+05	0	0.56 5.36e+04	0.04 1.33e+05	0.2 9.95e+04

Tabla 5.11 Resumen de la medida P en los algoritmos: ε-ED-S ran-Conf1, ε-ED-S ran-Conf4 y ε-ED-S ran-Conf6 en 10 dimensiones.

Conf1	**Conf4**	**Conf6**
Ganados(+)	13	10
Perdidos(-)	2	4
Empates(=)	3	4

Los algoritmos plasmados en las Tablas 5.8 y 5.10 (los más sobresalientes en 10 dimensiones), fueron probados con las funciones en 30 dimensiones. Los resultados se detallan en las Tablas 5.12 y 5.14, Además se generó un resumen de las Tablas 5.12 y 5.14, que se puede observar en las Tablas 5.13 y 5.15

respectivamente, las cuales facilitan la identificación de los algoritmos más robustos en esta etapa experimental.

Tabla 5.12 Comparativa entre los algoritmos: ε-ED-S best-Conf1, ε-ED-S best-Conf2, ε-ED-S best-Conf3, ε-ED-S best-Conf4, ε-ED-S best-Conf5 y ε-ED-S best-Conf6 en 30 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	0.84	0.12	0.16	0	0.08	0
	AFES	3..53e+05	5.24e+04	8.00e+04		2.34e+04	
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0	0	0	0	0	0
	AFES						
C4	**P**	0.8	0.04	0.4	0	0.28	0
	AFES	1.96e+05	1.10e+05	2.03e+05		4.10e+04	
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0.16	0	0	0	0
	AFES		2.71e+05				
C7	**P**	0.08	0.92	0.76	0	0.48	0
	AFES	2.75e+05	2.75e+05	2.24e+05		2.72e+05	
C8	**P**	0	0.48	0.24	0	0..32	0
	AFES		2.78e+05	2.79e+05		2.58e+05	
C9	**P**	0.32	0	0	0	0	0
	AFES	3.36e+05					
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	0.44	1	1	0.6	0.96	0.76
	AFES	2.99e+04	6.50e+04	7.88e+04	2.51e+04	6.10e+04	6.10e+04
C12	**P**	0	0.44	0.48	0	0.12	0
	AFES		3.53e+05	2.89e+05		3.23e+05	
C13	**P**	0	0	0.08	0.08	0.04	0
	AFES			3.49e+04	2.21e+04	2.27e+04	
C14	**P**	0	0.16	0.2	0	0	0
	AFES		3.34e+05	2.68e+05			
C15	**P**	0	0	0	0	0	0
	AFES						
C16	**P**	0	0	0.48	0	0	0
	AFES			1.77e+05			
C17	**P**	0	0	0	0	0	0
	AFES						
C18	**P**	0	0	0	0	0	0
	AFES						

Tabla 5.13 Resumen de la medida P en los algoritmos: ε-ED-S best-Conf2, ε-ED-S best-Conf2, ε-ED-S best-Conf3, ε-ED-S best-Conf4, ε-ED-S best-Conf5 y ε-ED-S best-Conf6 en 30 dimensiones.

Conf3	**Conf1**	**Conf2**	**Con4**	**Conf5**	**Conf6**
Ganados (+)	7	5	8	8	9
Perdidos (-)	3	3	0	1	0
Empates (=)	8	10	10	9	9

Tabla 5.14 Comparativa entre los algoritmos: ε-ED-S ran-Conf1, ε-ED-S ran-Conf2, ε-ED-S ran-Conf3, ε-ED-S ran-Conf4, ε-ED-S ran-Conf5 y ε-ED-S ran-Conf6 en 30 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	**Métrica**	**Conf1**	**Conf2**	**Conf3**	**Conf4**	**Conf5**	**Conf6**
C1	**P**	0.84	0.96	1	0.08	0.76	0.8
	AFES	3.53e+05	4.12e+05	1.63e+05	4.08e+04	4.56e+05	4.00e+05
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0	0	0	0	0	0
	AFES						
C4	**P**	0.8	0.2	0.32	0.24	0.48	0.44
	AFES	1.96e+05	7.64e+04	5.49e+04	3.13e+05	3.61e+05	2.78e+05
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	0.08	0	0	0.28	0	0
	AFES	2.75e+05			5.16e+05		
C8	**P**	0	0	0	0.04	0	0
	AFES				5.53e+05		
C9	**P**	0.32	0	0	0	0	0
	AFES	3.36e+05					
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	1	1	1	1	1	1
	AFES	5.65e+05	5.94e+04	5.18e+04	7.90e+04	5.18e+04	4.27e+04
C12	**P**	0.28	0.36	0.04	0.56	0.32	0.36
	AFES	2.51e+05	2.15e+05	2.01e+05	4.29e+05	1.98e+05	2.42e+05
C13	**P**	0	0.32	0.68	0	0	0.12
	AFES		5.59e+05	3.96e+05			1.87e+04
C14	**P**	0.12	0	0	0.04	0	0
	AFES	3.95e+05			4.60e+05		
C15	**P**	0	0	0	0	0	0
	AFES						

C16	**P** **AFES**	0.96 4.86e+04	0.96 4.76e+04	1 5.17e+04	0.44 4	1 4.57e+04	1 4
C17	**P** **AFES**	0	0	0	0	0	0
C18	**P** **AFES**	0.04 5.31e+05	0	0	0	0	0

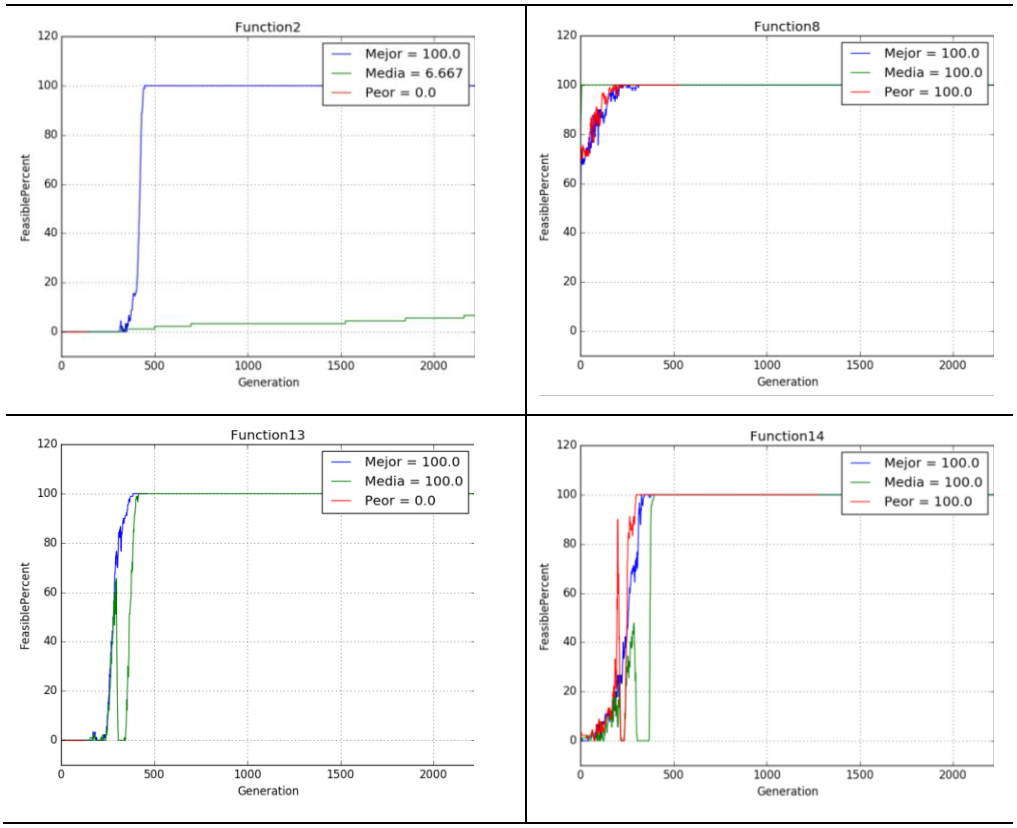
Tabla 5.15 Resumen de la medida P en los algoritmos: ε-ED-S ran-Conf2, ε-ED-S ran-Conf2, ε-ED-S ran-Conf3, ε-ED-S ran-Conf4, ε-ED-S ran-Conf5 y ε-ED-S ran-Conf6 en 30 dimensiones.

Conf1	**Conf2**	**Conf3**	**Con4**	**Conf5**	**Conf6**
ganados (+)	5	6	6	6	6
perdidos (-)	3	3	3	2	3
empates (=)	10	9	9	10	9

5.2.2 Resultados del comportamiento algorítmico del mejor algoritmo diseñado

Derivado de los resultados obtenidos hasta este punto de la investigación, se seleccionó el algoritmo ε-ED-S ran-Conf1, el más robusto en 10 y 30 dimensiones. Posteriormente, se analizó el proceso algorítmico que siguió resolviendo cada una de las funciones en 10 dimensiones, estudio llevado a cabo mediante gráficas de factibilidad (donde se plasmó el porcentaje de individuos factibles generación tras generación). En las gráficas se puede observar el comportamiento que se presentó en la mejor, media y peor ejecución, donde las ejecuciones fueron seleccionadas de acuerdo a la calidad de la solución alcanzada. Las gráficas más representativas fueron plasmadas en la Tabla 5.16, donde el eje *x* indica el porcentaje de individuos factibles en la generación *y*. Las gráficas restantes se pueden observar en el Anexo III.

Tabla 5.16 Comportamiento algorítmico de ε-ED-S ran-Conf1 en las funciones más representativas. Las líneas azul, verde y rojo representan la mejor, la media y la peor ejecución respectivamente.



5.2.3 Resultados del rediseño del mejor algoritmo desarrollado

Se llevó a cabo el rediseño del algoritmo con mejor desempeño en 10 y 30 dimensiones hasta este punto de la investigación, derivado de esto, se generaron 63 nuevas variantes del algoritmo ε-ED-S ran-Conf1, las cuales difieren en los valores utilizados para calibrar los parámetros del manejador de restricciones ε-constrained method. La etapa de rediseño se dividió en dos fases, en la primera sólo se calibró el parámetro Tc, mientras que en la segunda, se calibraron los

parámetros Tc y cp. Los algoritmos diseñados en la fase 1 fueron probados en 10 dimensiones, los resultados se pueden observar en la Tabla 5.17. Los algoritmos diseñados en la fase 2 fueron evaluadas en 10 y 30 dimensiones, los resultados más significativos fueron plasmados en las Tablas 5.19 y 5.21 respectivamente. Cabe resaltar que los algoritmos solo fueron medidos respecto a P, esto debido a que algunas variantes difieren en el tiempo de uso del manejador de restricciones (encargado de promover la exploración), por lo que una comparación utilizando la medida de calidad AFES sería poco justa. Los demás resultados se pueden observar en el Anexo IV. Además se generó un resumen de las Tablas 5.17, 5.19 y 5.21, que se puede observar en las Tablas 5.18, 5.20 y 5.22 respectivamente. Tablas que facilitan la identificación de los algoritmos más robustos.

Tabla 5.17 Comparativa entre los algoritmos: ε-ED-S ran-Conf1 (Tc=0.2), ε-ED-S ran-Conf1 (Tc=0.3), ε-ED-S ran-Conf1 (Tc=0.4), ε-ED-S ran-Conf1 (Tc=0.5), ε-ED-S ran-Conf1 (Tc=0.6), ε-ED-S ran-Conf1 (Tc=0.7), ε-ED-S ran-Conf1 (Tc=0.8) y ε-ED-S ran-Conf1 (Tc=0.9) en 10 dimensiones. P: probabilidad de convergencia.

Función	Métrica	Tc=.2	Tc=.3	Tc=.4	Tc=.5	Tc=.6	Tc=.7	Tc=.8	Tc=.9
C1	**P**	1	1	0.88	0.96	0.92	1	1	1
C2	**P**	0.24	0.44	0.68	0.68	0.88	0.88	0.84	0.92
C3	**P**	0.92	0.96	1	1	0.96	1	0.92	0.96
C4	**P**	1	1	1	1	0.92	1	1	0.96
C5	**P**	0	0	0.04	0	0.04	0.04	0	0
C6	**P**	0.04	0	0.04	0.04	0.08	0.04	0	0
C7	**P**	1	1	1	1	1	1	1	0.96
C8	**P**	0.4	0.2	0.36	0.36	0.2	0.28	0.44	0.4
C9	**P**	0.48	0.92	0.92	0.88	0.88	0.88	0.96	0.96
C10	**P**	0.04	0.68	0.68	0.92	0.92	0.92	0.92	0.92
C11	**P**	1	1	0.96	0.96	0.8	0.8	0.88	0.6
C12	**P**	0.16	0.12	0.2	0.08	0.08	0.12	0.04	0.04
C13	**P**	1	0.96	0.92	0.88	0.76	0.8	0.88	0.76
C14	**P**	0.68	0.92	0.92	0.92	0.84	0.92	0.96	1
C15	**P**	0.04	0.4	0.8	0.92	0.76	0.92	0.88	0.88
C16	**P**	0	0.2	0.48	0.48	0.52	0.76	0.64	0.72
C17	**P**	0.76	0.8	0.96	0.96	0.96	0.92	0.96	1
C18	**P**	0.72	0.68	0.84	0.8	0.76	0.88	0.96	0.92

Tabla 5.18 Resumen de la medida P en los algoritmos: ε-ED-S ran-Conf1 (Tc=0.2), ε-ED-S ran-Conf1 (Tc=0.3), ε-ED-S ran-Conf1 (Tc=0.4), ε-ED-S ran-Conf1 (Tc=0.5), ε-ED-S ran-Conf1 (Tc=0.6), ε-ED-S ran-Conf1 (Tc=0.7), ε-ED-S ran-Conf1 (Tc=0.8) y ε-ED-S ran-Conf1 (Tc=0.9) en 10 dimensiones.

Tc=.7	**Tc =.2**	**Tc=.3**	**Tc =.4**	**Tc =.5**	**Tc =.6**	**Tc =.8**	**Tc =.9**
Ganados(+)	10	10	6	6	10	7	10
Perdidos(-)	4	3	6	4	2	7	6
Empates(=)	4	5	6	8	6	4	2

Tabla 5.19 Comparativa entre las 10 versiones del algoritmo ε-ED-S ran-Conf1 (diferentes en sus parámetros Tc y cp), con mejor desempeño en 10 dimensiones. P: probabilidad de convergencia.

Función	**Métrica**	**Tc=.3 cp=2**	**Tc=.4 cp=5**	**Tc=.6 cp=1**	**Tc=.6 cp=4**	**Tc=.6 cp=5**	**Tc=.7 cp=5**	**Tc=.8 cp=2**	**Tc=.9 cp=1**	**Tc=.9 cp=2**	**Tc=.9 cp=6**
C1	**P**	1	0.88	1	1	0.92	1	0.96	1	0.96	0.96
C2	**P**	0.64	0.68	0.96	0.72	0.88	0.88	0.84	0.96	0.92	0.72
C3	**P**	0.96	1	0.96	0.92	0.96	1	1	0.96	0.92	0.96
C4	**P**	1	1	0.88	0.92	0.92	1	0.64	0.32	0.48	1
C5	**P**	0.08	0.04	0.08	0.04	0.04	0.04	0.08	0.04	0.12	0.04
C6	**P**	0.08	0.04	0.04	0.04	0.08	0.04	0.04	0.04	0.04	0.08
C7	**P**	1	1	0.96	0.96	1	1	1	1	1	1
C8	**P**	0.52	0.36	0.2	0.32	0.2	0.28	0.2	0.16	0.24	0.4
C9	**P**	0.16	0.92	0.96	0.88	0.88	0.88	1	1	0.92	0.96
C10	**P**	0.08	0.68	1	0.96	0.92	0.92	0.92	0.92	0.96	0.96
C11	**P**	0.96	0.96	0.92	0.92	0.8	0.8	0.64	0.76	0.64	0.6
C12	**P**	0.04	0.2	0.04	0.08	0.08	0.12	0.04	0.12	0.04	0.04
C13	**P**	0.92	0.92	0.92	1	0.76	0.8	0.68	0.32	0.16	0.64
C14	**P**	0.94	0.92	0.96	0.84	0.84	0.92	0.92	1	0.88	0.88
C15	**P**	0.72	0.8	0.84	0.88	0.76	0.92	0.88	0.92	0.92	0.92
C16	**P**	0.2	0.48	0.64	0.4	0.52	0.76	0.64	0.52	0.56	0.6
C17	**P**	0.92	0.96	0.96	0.92	0.96	0.92	0.96	1	1	1
C18	**P**	0.92	0.84	0.96	0.88	0.76	0.88	0.92	1	1	0.88

Tabla 5.20 Resumen de la medida P en las mejores versiones del algoritmo: ε-ED-S ran-Conf1 con diferentes valores para los parámetros Tc y cp, en 10 dimensiones.

Tc=.9 cp=1	**Tc=.3 cp=2**	**Tc=.4 cp=5**	**Tc=.6 cp=1**	**Tc=.6 cp=4**	**Tc=.6 cp=5**	**Tc=.7 cp=5**	**Tc=.8 cp=2**	**Tc=.9 cp=2**	**Tc=.9 cp=6**
Ganados(+)	9	9	7	10	8	5	8	8	7
Perdidos(-)	6	6	7	5	5	6	6	5	6
Empates(=)	4	3	4	3	5	7	4	5	5

Tabla 5.21 Comparativa entre las 10 mejores versiones del algoritmo ε-ED-S ran-Conf1 en 30 dimensiones. P: probabilidad de convergencia.

Función	Métrica	Tc=.3 cp=2	Tc=.4 cp=5	Tc=.6 cp=1	Tc=.6 cp=4	Tc=.6 cp=5	Tc=.7 cp=5	Tc=.8 cp=2	Tc=.9 cp=1	Tc=.9 cp=2	Tc=.9 cp=6
C1	**P**	0.88	0.68	1	0.8	0.84	0.68	0.84	0.88	0.84	0.76
C2	**P**	0.44	0.12	0.88	0.52	0.44	0.24	0.84	0.76	0.72	0.4
C3	**P**	0	0.04	0.2	0.16	0.24	0.12	0.28	0.4	0.2	0.12
C4	**P**	0.4	0.6	0.2	0.6	0.72	0.8	0.36	0.4	0.52	0.88
C5	**P**	0	0	0.08	0.04	0	0	0.04	0.12	0	0.04
C6	**P**	0.04	0	0.08	0.04	0.04	0.08	0	0.04	0	0
C7	**P**	0.08	0.12	0.04	0.08	0.12	0.08	0.12	0.04	0.16	0.12
C8	**P**	0	0	0.04	0	0	0	0.04	0.04	0	0
C9	**P**	0.04	0.92	0.12	0.88	0.8	0.84	0.12	0.16	0	0.8
C10	**P**	0	0	0.08	0.08	0.08	0.24	0.28	0.04	0.2	0.08
C11	**P**	1	1	1	1	1	0.96	1	1	1	0.96
C12	**P**	0.2	0	0.12	0.04	0	0	0	0	0.04	0
C13	**P**	0.12	0	0.2	0.04	0	0	0.12	0.52	0.36	0.04
C14	**P**	0.08	0.16	0.12	0.2	0.16	0.2	0.16	0.16	0.08	0.28
C15	**P**	0.08	0	0.04	0.12	0.12	0.12	0.2	0.24	0.12	0.2
C16	**P**	1	0.96	0.96	1	0.96	0.92	1	0.96	0.96	1
C17	**P**	0.08	0.4	0.92	0.6	0.52	0.8	0.72	0.68	0.92	0.8
C18	**P**	0.08	0.2	0.36	0.2	0.36	0.4	0.6	0.28	0.44	0.52

Tabla 5.22 Resumen de la medida P en las 10 mejores versiones del algoritmo: ε-ED-S ran-Conf1 con diferentes valores para los parámetros Tc y cp, en 30 dimensiones.

Tc=.6 cp=1	Tc=.3 cp=2	Tc=.4 cp=5	Tc=.6 cp=4	Tc=.6 cp=5	Tc=.7 cp=5	Tc=.8 cp=2	Tc=.9 cp=1	Tc=.9 cp=2	Tc=.9 cp=6
Ganados(+)	12	12	10	9	9	7	7	8	11
Perdidos(-)	5	4	6	5	7	8	7	6	6
Empates(=)	1	2	2	4	2	3	4	4	1

5.2.4 Resultados del análisis de diversidad del mejor algoritmo rediseñado

Se estudió el comportamiento algorítmico de 6 variantes de ED para PORs (Fr-ED ran, ε-ED ran, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1 ε-ED-S ran-Conf1 (Tc=0.6, cp=1) y ε-ED-S ran-Conf1 (Tc=0.9, cp=1)) que fueron relevantes en diferentes etapas de esta investigación, estudio que se realizó con base en gráficas de diversidad, específicamente con la métrica: distancia al vector promedio [35]. Las Tablas 5.23 y 5.24 contienen las gráficas de la función 4 (en la cual se puede observar claramente el comportamiento

oscilatorio esperado) en 10 y 30 dimensiones respectivamente, donde el eje *x* indica el porcentaje de diversidad en cada generación *y*.

Tabla 5.23 Comparativa entre algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C4 en 10 dimensiones.

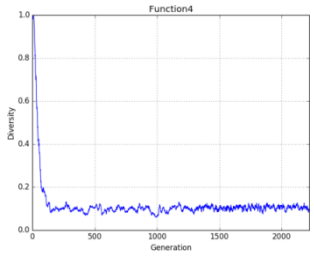
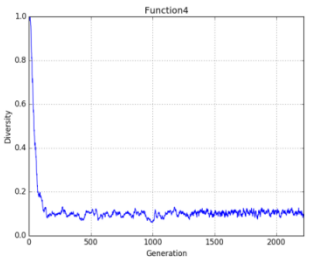
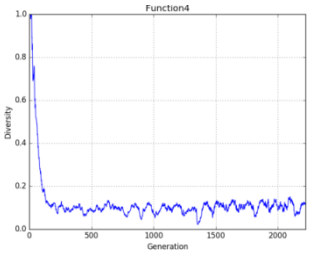
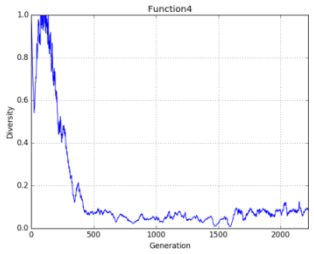
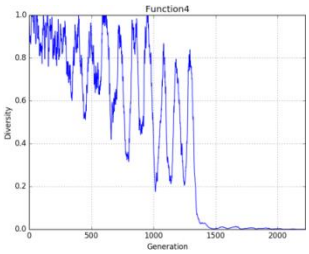
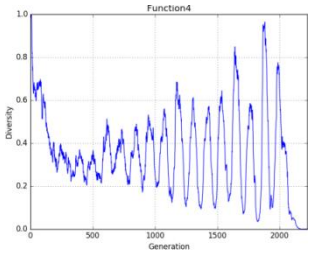
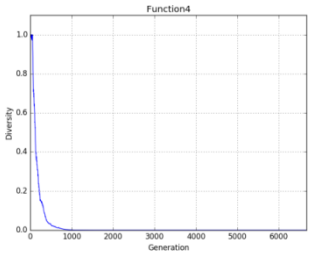
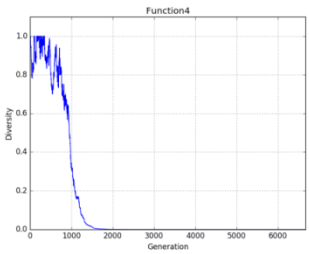
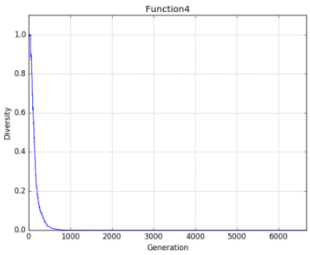
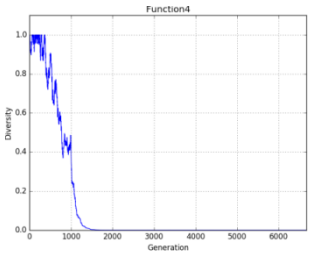
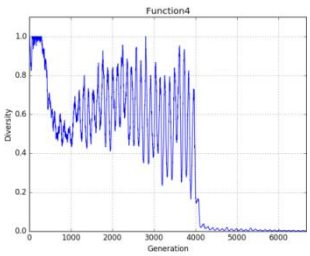
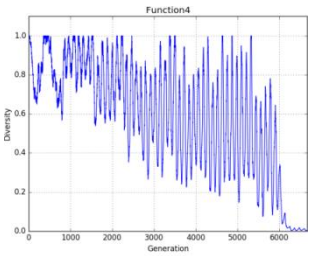
Rf-DE ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.24 Comparativa entre algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C4 en 30 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

5.3 Discusión de resultados

5.3.1 Discusión de resultados de la experimentación de los algoritmos de ED adaptativos para PORs diseñados

Durante el inicio de la presente investigación, se llevó a cabo una experimentación bastante amplia, en la que se diseñaron y probaron un total de 284 algoritmos en 10 dimensiones. Estas versiones pueden diferir en la variante de ED para PORs empleada, el manejador de restricciones y/o el mecanismo para el control de parámetros. Como se puede observar en las Tablas 5.8 y 5.10, las mejores versiones (ε-ED-S ran-Conf1 y ε-ED-S ran-Conf4) lograron resolver a lo mucho 16 de los 18 problemas. En la Tabla 5.1, se puede apreciar que en los dos esquemas adaptativos con mejor desempeño, el parámetro CR se

mantiene fijo, mientras que en Conf1 F se adapta de forma creciente y en Conf2 lo hace de forma decreciente. De tal manera que el comportamiento oscilatorio de los valores que va tomando F, se ve reflejado en vectores mutantes por momentos muy parecidos al vector base y por momentos muy diferentes. Por otro lado, mantener CR fijo en 0.9, permite que los trials generados estén compuestos en su mayoría por genes del vector mutante y no del target. Es decir, cuando F tiende a 0, el algoritmo entra en una etapa de explotación, mientras que cuando tiende a 1, incrementa sus capacidades de exploración. Las Tablas 5.9 y 5.11 permiten identificar a los algoritmos ε-ED-S best-Conf3 y ε-ED-S ran-Conf1 como los más robustos de las variantes ED/best/1/bin y ED/ran/1/bin respectivamente.

Posteriormente, los algoritmos con mejor desempeño en 10 dimensiones (algoritmos plasmados en las Tablas 5.8 y 5.10) fueron probados con las funciones en 30 dimensiones. En las Tablas 5.12 y 5.14 se pueden apreciar los resultados de dicha experimentación, donde se puede observar que las mejores versiones (ε-ED-S ran-Conf1 y ε-ED-S best-Conf3) lograron resolver a lo mucho 9 problemas, desempeño considerado sumamente pobre. Estos algoritmos difieren por los esquemas adaptativos que usan y el vector que están mutando. En ε-ED-S ran-Conf1 se muta un vector elegido aleatoriamente, CR se mantiene fijo en 0.9 y F se adapta con la función senoidal con comportamiento creciente. Por otra parte, en ε-ED-S best-Conf3 CR y F se adaptan con la función senoidal con comportamiento decreciente y se muta el mejor vector de la población actual. Es decir, el esquema Conf3 al igual que Conf1 genera vectores mutantes por momentos cerca del vector base (en este caso, el mejor vector de la población) y por momentos lejos. Sin embargo, difieren en el comportamiento del parámetro CR, de tal manera que los trials que genera Conf3 por momentos se parecen en gran medida a los targets y por momentos a los vectores mutados. En este esquema, ambos parámetros siguen el mismo comportamiento. Cuando tienden a 1, F incrementa el tamaño de paso y se generan vectores mutantes sumamente diferentes al vector base, mientras que CR hace que el vector trial tenga en su mayoría genes del vector mutado y

no del target. Por otro lado, cuando los parámetros tienden a 0, los vectores mutados se parecen al vector base, sin embargo los trials se parecen en gran medida al vector target y no al vector mutado. Las Tablas 5.13 y 5.15 permiten identificar nuevamente a los algoritmos ε-ED-S best-Conf3 y ε-ED-S ran-Conf1 como los más robustos de las variantes ED/best/1/bin y ED/ran/1/bin respectivamente.

Es importante resaltar que en 10 y 30 dimensiones, los algoritmos con mejor desempeño manejaron las restricciones con base en ε-constrained method, manejador que utiliza los parámetros Tc y cp.

Por otro lado, durante el análisis de las tablas comparativas de los resultados alcanzados por los algoritmos con mejor desempeño en 10 y 30 dimensiones (algoritmos plasmados en las Tablas 5.8-5.15), se observó que las funciones fueron resueltas con una baja probabilidad de convergencia y sumamente rápido, utilizando menos de la mitad de las evaluaciones disponibles, lo que permitió inferir que los algoritmos diseñados tienen problemas para escapar de óptimos locales y/o están convergiendo prematuramente.

5.3.2 Discusión de resultados del comportamiento algorítmico del mejor algoritmo diseñado

Derivado de los resultados obtenidos durante la etapa experimental previa, se seleccionó el algoritmo más robusto en 10 y 30 dimensiones. Como se puede observar en las tablas comparativas de dicha experimentación, el algoritmo con mejor desempeño fue ε-ED-S ran-Conf1, el cual resolvió 16 y 9 funciones en 10 y 30 dimensiones respectivamente. Dicho algoritmo tiene las siguientes características: (1) utiliza la variante ED/ran/1/bin, (2) el manejador de restricciones ε-constrained method y (3) el mecanismo para el control de parámetros Conf1, donde F se adapta con la función senoidal de forma creciente y CR se mantiene fijo en 0.9. El análisis del comportamiento algorítmico de esta variante fue llevado a cabo mediante gráficas de

factibilidad, en las que se plasmó el porcentaje de individuos factibles generación tras generación de la mejor, media y peor ejecución, donde las ejecuciones fueron seleccionadas de acuerdo a la calidad de la solución alcanzada con las funciones en 10 dimensiones.

De acuerdo a las gráficas mostradas en la Tabla 5.16, se puede inferir que el algoritmo explora durante el primer 20% del proceso evolutivo, mientras que el resto de la búsqueda, utiliza su capacidad de explotación para encontrar la solución óptima en las zonas identificadas como prometedoras durante la fase de exploración. Cabe resaltar que el manejador de restricciones ε-constrained method es utilizado sólo durante el primer 20% de la búsqueda, para posteriormente dar paso a las reglas de factibilidad. Es decir, se puede concluir que ε-constrained method le brinda al algoritmo la capacidad de explorar, mientras que las reglas de factibilidad le permiten explotar las zonas identificadas como prometedoras durante la etapa de exploración.

Los resultados del análisis del comportamiento algorítmico permitieron validar las conclusiones de la primera experimentación, donde se puede observar que el algoritmo ε-ED-S ran-Conf1, utiliza su capacidad de exploración sólo durante el primer 20% de la búsqueda. De tal forma que si durante dicho lapso no identifica una zona prometedora, la probabilidad de encontrar la solución es muy baja, incrementándose las posibilidades de llegar a una convergencia prematura y/o estancamiento local. Es decir, se puede concluir que el diseño actual de los algoritmos está limitando las capacidades de los esquemas adaptativos empleados, esto debido a que el lapso de exploración proporcionado por el manejador de restricciones es muy corto.

5.3.3 Discusión de resultados del rediseño del mejor algoritmo desarrollado

Después de haber identificado que la calibración del manejador de restricciones estaba limitando las capacidades de exploración y explotación del mejor algoritmo diseñado. Se procedió a rediseñar dicho algoritmo, mejora que

básicamente consistió en la calibración de los parámetros Tc y cp del manejador de restricciones, lo que en principio le permitiría evitar estancarse en óptimos locales y/o converger prematuramente.

El rediseño se dividió en dos fases. En la primera, se consideró el hecho de que el autor del manejador de restricciones ε-constrained method, propuso originalmente que se utilizara sólo durante el primer 20% de la búsqueda, después de haber observado que éste manejador le brinda al algoritmo ε-ED-S ran-Conf1 la capacidad de explorar de mejor forma, éste fue modificado expandiendo el uso de ε-constrained method a 30%, 40%, 50%, 60%, 70%, 80% y 90% del proceso evolutivo. Las variantes generadas fueron probadas con las funciones en 10 dimensiones. En la Tabla 5.17 se puede observar que los resultados obtenidos fueron sumamente prometedores, donde 3 variantes lograron resolver las 18 funciones. La Tabla 5.18, permite identificar a los algoritmos ED-S ran-Conf1 (Tc=.4), ED-S ran-Conf1 (Tc=.7) y ED-S ran-Conf1 (Tc=.8) como los más robustos de esta fase.

Después de haber observado el impacto de la calibración del parámetro Tc, en la segunda fase se exploraron diferentes valores para el parámetro cp, valores enteros entre 1 y 9. De tal forma que en total se diseñaron 63 versiones, 9 variantes de cada algoritmo generado en la primera fase. Como se puede observar en la Tabla 4.19, se identificaron 10 versiones capaces de resolver las 18 funciones en 10 dimensiones. Posteriormente, dichos algoritmos fueron probados con las funciones en 30 dimensiones. Como se puede observar en la Tabla 5.21, si bien no todos los algoritmos mantuvieron el desempeño de la experimentación previa, se identificaron dos versiones que sobresalieron ante las demás (versiones calibradas con los parámetros Tc=0.6, cp=1 y Tc=0.9, cp=1), las cuales resolvieron 18 y 17 funciones respectivamente. Sin embargo, como se pude observar en las Tablas 5.20 y 5.22, al comparar su robustez respecto a P, ambos algoritmos tienen un comportamiento similar en 10 y 30 dimensiones.

5.3.4 Discusión de resultados del análisis de diversidad del mejor algoritmo rediseñado

Finalmente, para concluir con la experimentación de esta investigación y con el objetivo de justificar los ajustes llevados a cabo sobre el algoritmo ED (manejador de restricciones, mecanismo para el control de parámetros y los ajustes efectuados a ε-constrained method), se generó una gráfica de diversidad por cada una de las 18 funciones en 10 y 30 dimensiones, para hacer dicho estudio, se utilizaron 6 algoritmos que fueron fundamentales en diferentes etapas de esta investigación (Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1) y ε-ED-S ran-Conf1 (Tc=0.9, cp=1)).

Las Tablas 5.23 y 5.24 contienen las gráficas de diversidad de la función C4 (una de las más representativas) en 10 y 30 dimensiones respectivamente, donde se puede observar que los algoritmos ε-ED-S (Tc=0.6, cp=1) y ε-ED-S (Tc=0.9, cp=1) manejan la diversidad de mejor forma que las otras 4 variantes. De igual manera, se puede notar como el utilizar el mecanismo para el control de parámetros Conf1 (donde F se adapta de forma creciente, mientras que CR se mantiene fijo en 0.9), mantener la utilización del manejador de restricciones (ε-constrained method) por un porcentaje de búsqueda mayor al que propuso el autor (específicamente por 60% y 90% del proceso evolutivo) y reducir la velocidad con la que se reduce el umbral ε (de 5 a 1), le brinda a estos algoritmos la capacidad de explorar y explotar de forma eficiente el espacio de búsqueda. También se puede percibir cómo estos algoritmos tienen la bondad de dar grandes saltos en el espacio de búsqueda, lo que en principio les permite escapar de óptimos locales y evitar una convergencia prematura. En otras palabras, estas variantes tienen la capacidad de pasar del estado de exploración al de explotación en diversas etapas del proceso evolutivo, esto gracias al comportamiento oscilatorio del mecanismo para el control de parámetros empleado.

Capítulo 6
Conclusiones y trabajo futuro

6.1 Conclusiones

Si bien resolver problemas de optimización sin restricciones de forma eficiente es una tarea sumamente compleja, cuando se habla de problemas restringidos la dificultad aumenta considerablemente. Desde su surgimiento los AEs han sido fuertemente criticados por la fina configuración de parámetros que requieren. A lo largo de la historia se han propuesto diversas configuraciones para ajustar los parámetros del algoritmo ED. Por otro lado, en la literatura especializada se puede observar que cada problema requiere de una calibración de parámetros específica. Uno de los objetivos de la CE es identificar un algoritmo que sea bueno en una amplia gama de problemas de optimización. Para lograr dicho objetivo, en los últimos años la comunidad científica del área ha llevado a cabo grandes esfuerzos para diseñar algoritmos adaptativos, que obtengan buenas soluciones en tiempos de búsqueda considerablemente cortos y que requieran de pequeños ajustes para resolver una amplia gama de problemas.

Bajo la motivación de diseñar un algoritmo basado en ED para PORs, que utilice un mecanismo para el control de sus parámetros principales F y CR a través de funciones matemáticas y una técnica competitiva para el manejo de restricciones, se llevó a cabo esta investigación, donde se obtuvieron las siguientes conclusiones:

- Se presentó el algoritmo ε-ED-S ran-Conf1, variante del algoritmo ED/ran/1/bin que fue mejorada incorporándole un mecanismo para el control de parámetros bajo un enfoque determinista, donde el parámetro F es adaptado con base en una función senoidal, de tal forma que inicia tomando valores cercanos a 0.5 y conforme avanza el proceso evolutivo, dichos valores van oscilando tomando valores por momentos cercanos a

0 y por momentos a 1. Por otro lado CR se mantiene fijo en 0.9. Además, para brindarle a este algoritmo la capacidad de resolver PORs, se le incorporó el manejador de restricciones ε-constrained method, el cual fue mejorado.

- Originalmente el autor de ε-constrained method propuso que se utilizara sólo durante el primer 20% del proceso evolutivo, sin embargo, las conclusiones alcanzadas durante las primeras experimentaciones llevadas a cabo, permitieron inferir que durante dicho primer 20% de la búsqueda, el algoritmo tenía una muy buena capacidad de exploración. Sin embargo, el resto del proceso evolutivo el algoritmo convergía sumamente rápido, estancándose en óptimos locales y/o convergiendo prematuramente.
- Las conclusiones obtenidas del análisis algorítmico de ε-ED-S ran-Conf1 fueron identificadas como un nicho de oportunidad, de tal forma que el manejador de restricciones fue mejorado respecto a la calibración de sus parámetros, con el objetivo de mantener su utilización por un porcentaje mayor del proceso evolutivo y disminuir la velocidad de reducción del umbral ε, lo que esencialmente permitió explotar al máximo la capacidad de exploración que le brinda el esquema adaptativo Conf1 a este algoritmo.
- Al combinar el ε-constrained method mejorado y el comportamiento senoidal del mecanismo para el control de parámetros Conf1, se logró un buen compromiso exploración-explotación.
- El manejador de restricciones le permite al algoritmo mantenerse en un estado de exploración (gracias a la calibración de los parámetros Tc y cp).
- El mecanismo para el control de parámetros por momentos favorece la exploración y por momentos la explotación, gracias a que el tamaño de paso por momentos va creciendo y por momentos decreciendo, lo que se traduce en vectores mutantes por momentos generados muy cerca del vector base y por momentos muy lejos. Aunado a esto, mantener a CR

fijo en 0.9 permite asegurar que el vector trial tenga en su mayoría genes del vector mutante.

- Cuando F tiende a 1, el algoritmo maximiza su capacidad de exploración, mientras que cuando tiende a 0 la minimiza o pasa a un estado de explotación, de tal forma que gracias al comportamiento oscilatorio del mecanismo para el control de parámetros, el algoritmo puede por momentos entrar en un estado de convergencia y por momentos pasar a un estado de divergencia, patrón que se puede repetir una y otra vez de acuerdo a las características específicas del paisaje de aptitud de cada problema, lo que permite llevar a cabo una búsqueda más inteligente.
- Por ejemplo, la función C4 tiene varios óptimos locales, razón por la cual los algoritmos tienden a estancarse, sin embargo las variantes diseñadas con las características antes mencionadas, tienen entre sus bondades la capacidad de dar grandes saltos en el espacio de búsqueda, lo que les permite evitar estancarse en óptimos locales y/o converger prematuramente.
- Ahora bien, los objetivos planteados fueron cumplidos de la siguiente manera: (1) se realizó un análisis profundo de la literatura especializada, lo que permitió conocer más a detalle el funcionamiento del algoritmo ED, (2) se implementaron y probaron los algoritmos ED y EDVC, (3) se diseñaron 284 nuevas variantes, las cuales difieren por el manejador de restricciones y el mecanismo para el control de los parámetros F y CR que les fue incorporado, (4) el análisis del desempeño de los algoritmos fue medido con base en los problemas del benchmarck del CEC2010 y las medidas de calidad P y AFES, (5) se identificó el algoritmo más robusto en 10 y 30 dimensiones ε-ED-S ran-Conf1, (6) se llevó a cabo el análisis del comportamiento algorítmico de ε-ED-S ran-Conf1 y (7) fue rediseñado respecto a los parámetros del manejador de restricciones Tc y cp, mejora que le permitió resolver una mayor cantidad de problemas en 10 y 30 dimensiones. Por otro lado, la hipótesis planteada

fue cumplida de la siguiente forma: el algoritmo diseñado es capaz de resolver todos los problemas del benchmark del CEC210 en 10 y 30 dimensiones en al menos una de las 25 ejecuciones realizadas.

6.2 Trabajo Futuro

Derivado de ésta investigación, se propone el siguiente trabajo futuro:

- Incorporar a otros algoritmos el mecanismo para el control de parámetros utilizado (Conf1).
- Rediseñar el mecanismo para el control de parámetros utilizado, brindándole la capacidad de incrementar y reducir la amplitud de onda de acuerdo a la diversidad de la población.
- Los mecanismos de control utilizados están diseñados para adaptar los parámetros F y CR, se propone adaptar el parámetro NP. Por ejemplo, utilizar una función que lleve a cabo una reducción lineal de la población.
- Adaptar el parámetro cp de ε-constrained method.

Anexos

Anexo I. Detalle de las funciones del CEC2010

C01

$Minimizar$:

$$f(\vec{x}) = \left| \frac{\sum_{i=1}^{D} cos^4 - 2\prod_{i=1}^{D} cos^2(z_i)}{\sqrt{\sum_{i=1}^{D} i z_i^2}} \right| \qquad z = x - 0$$

$Sujeto\ a$:

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^{D} z_i \leq 0$$
$$g_2(\vec{x}) = \sum_{i=1}^{D} -7.5D \leq 0$$
$$x \in [0, 10]^D$$

C02

$Minimizar$:

$$f(\vec{x}) = \max(z) \qquad z = x - 0, y = z - 0.5$$

$Sujeto\ a$:

$$g_1(\vec{x}) = 10 - \frac{1}{D} \sum_{i=1}^{D} [z_i^2 - 10\cos(2\pi z_i) + 10] \leq 0$$
$$g_2(\vec{x}) = \frac{1}{D} \sum_{i=1}^{D} [z_i^2 - 10\cos(2\pi z_i) + 10] - 15 \leq 0$$
$$h(\vec{x}) = \frac{1}{D} \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10] - 20 = 0$$
$$x \in [-5.12, 5.12]^D$$

C03

$Minimizar$:

$$f(\vec{x}) = \sum_{i=1}^{D} (100(z_i^2 - z_i)^2) \qquad z = x - 0$$

$Sujeto\ a$:

$$h(\vec{x}) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2 = 0$$
$$x \in [-1000, 1000]^D$$

C04

$Minimizar$:

$$f(\vec{x}) = \max(z) \qquad z = x - 0$$

Sujeto a:

$$h_1(\vec{x}) = \frac{1}{D}\sum_{i=1}^{D}(z_i cos\sqrt{|z_i|}) = 0$$

$$h_2(\vec{x}) = \sum_{i=1}^{\frac{D}{2-1}}(z_i - z_{i+1})^2 = 0$$

$$h_3(\vec{x}) = \sum_{i=\frac{D}{2+1}}^{D-1}(z_i^2 - z_{i+1})^2 = 0$$

$$x \in [-50,50]^D$$

C05

Minimizar:

$$f(\vec{x}) = \max(z) \qquad z = x - 0$$

Sujeto a:

$$h_1(\vec{x}) = \frac{1}{D}\sum_{i=1}^{D}(-z_i sin\sqrt{|z_i|}) = 0$$

$$h_2(\vec{x}) = \frac{1}{D}\sum_{i=1}^{D}(-z_i \cos\left(0.5\sqrt{|z_i|}\right)) = 0$$

$$x \in [-600,600]^D$$

C06

Minimizar:

$$f(\vec{x}) = \max(z)$$

$$z = x - 0, y = (x + 483.6106156535 - 0)M - 483.6106156535$$

Sujeto a:

$$h_1(\vec{x}) = \frac{1}{D}\sum_{i=1}^{D}(-y_i sin\sqrt{|y_i|}) = 0$$

$$h_2(\vec{x}) = \frac{1}{D}\sum_{i=1}^{D}(-y_i \cos\left(0.5\sqrt{|y_i|}\right)) = 0$$

$$x \in [-600,600]^D$$

C07

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i + 1)^2)$$

$$z = x + 1 - o, y = x - o$$

Sujeto a:

$$g(\vec{x}) = 0.5 - \exp(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(0.1y)\right) + \exp(1) \leq 0$$

$x \in [-140,140]^D$

C08

$Minimizar$:

$$f(\vec{x}) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i + 1)^2)$$
$$z = x + 1 - o, y = (x - o)M$$

$Sujeto\ a$:

$$g(\vec{x}) = 0.5 - \exp(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(0.1y)\right) + \exp(1) \leq 0$$
$$x \in [-140,140]^D$$

C09

$Minimizar$:

$$f(\vec{x}) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i + 1)^2)$$
$$z = x + 1 - o, y = x - o$$

$Sujeto\ a$:

$$h(\vec{x}) = \sum_{i=1}^{D}(ysin(\sqrt{|y_i|})) = 0$$
$$x \in [-500,500]^D$$

C10

$Minimizar$:

$$f(\vec{x}) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})2 + (z_i + 1)^2)$$
$$z = x + 1 - o, y = (x - o)M$$

$Sujeto\ a$:

$$h(\vec{x}) = \sum_{i=1}^{D}(ysin(\sqrt{|y_i|})) = 0$$
$$x \in [-500,500]^D$$

C11

$Minimizar$:

$$f(\vec{x}) = \frac{1}{D}\sum_{i=1}^{D-1}(-z_i\cos(2\sqrt{|z_i|}))$$
$$z = (x - o)M, y = x + 1 - o$$

$Sujeto\ a$:

$$h(\vec{x}) = \sum_{i=1}^{D}(100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2) = 0$$

$$x \in [-100,100]^D$$

C12

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{D}(z_i \sin(\sqrt{|z_i|}))$$
$$z = x - o$$

Sujeto a:

$$g(\vec{x}) = \sum_{i=1}^{D}(z - 100\cos(0.1z) + 10) \leq 0$$
$$h(\vec{x}) = \sum_{i=1}^{D-1}(z^2 - z_{i+1})^2 = 0$$
$$x \in [-1000,1000]^D$$

C13

Minimizar:

$$f(\vec{x}) = \frac{1}{D}\sum_{i=1}^{D}(-z_i \sin(\sqrt{|z_i|}))$$
$$z = x - o$$

Sujeto a:

$$g_1(\vec{x}) = -50 + \frac{1}{100D}\sum_{i=1}^{D} z_i^2 \leq 0$$

$$g_2(\vec{x}) = \frac{50}{D}\sum_{i=1}^{D} \sin(\frac{1}{50}\pi z) \leq 0$$

$$g_3(\vec{x}) = 75 - 50(\sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 \leq 0$$

$$x \in [-500,500]^D$$

C14

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{D-1}(100(z_i^2 - z_i)^2 + (z_i - 1)^2)$$
$$z = x + 1 - o\ , y = x - o$$

Sujeto a:

$$g_1(\vec{x}) = \sum_{i=1}^{D} \left(-y_i \cos\left(\sqrt{|y_i|}\right)\right) - D \leq 0$$

$$g_2(\vec{x}) = \sum_{i=1}^{D} \left(y_i \cos\left(\sqrt{y_i}\right)\right) - \mathrm{D} \leq 0$$

$$g_3(\vec{x}) = \sum_{i=1}^{D} \left(y_i \sin\left(\sqrt{y_i}\right)\right) - 10\mathrm{D} \leq 0$$

$$x \in [-1000,1000]^D$$

C15

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{D-1} (100(z_i^2 - z_i)^2 + (z_i - 1)^2)$$

$$z = x + 1 - o\ , y = (x - o)M$$

Sujeto a:

$$g_1(\vec{x}) = \sum_{i=1}^{D} \left(-y_i \cos\left(\sqrt{|y_i|}\right)\right) - D \leq 0$$

$$g_2(\vec{x}) = \sum_{i=1}^{D} \left(y_i \cos\left(\sqrt{y_i}\right)\right) - \mathrm{D} \leq 0$$

$$g_3(\vec{x}) = \sum_{i=1}^{D} \left(y_i \sin\left(\sqrt{y_i}\right)\right) - 10\mathrm{D} \leq 0$$

$$x \in [-1000,1000]^D$$

C16

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{D} \frac{z_i^2}{40000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$$

$$z = x - o$$

Sujeto a:

$$g_1(\vec{x}) = \sum_{i=1}^{D} [z_i - 100\cos(\pi z) + 10] \leq 0$$

$$g_2(\vec{x}) = \prod_{i=1}^{D} z_i \leq 0$$

$$h_1(\vec{x}) = \sum_{i=1}^{D} \left(z_i \sin\left(\sqrt{z_i}\right)\right) = 0$$

$$h_2(\vec{x}) = \sum_{i=1}^{D} \left(-z_i \sin\left(\sqrt{z_i}\right)\right) = 0$$

$$x \in [-10,10]^D$$

C17

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2$$

$$z = x - o$$

Sujeto a:

$$g_1(\vec{x}) = \prod_{i=1}^{D} z_i \leq 0$$

$$g_2(\vec{x}) = \sum_{i=1}^{D} z_i \ \leq 0$$

$$h_1(\vec{x}) = \sum_{i=1}^{D} \left(z_i \sin\left(4\sqrt{z_i}\right)\right) = 0$$

$$x \in [-10,10]^D$$

C18

Minimizar:

$$f(\vec{x}) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2$$

$$z = x - o$$

Sujeto a:

$$g(\vec{x}) = \sum_{i=1}^{D} \left(-z_i \sin\left(\sqrt{z_i}\right)\right) \ \leq 0$$

$$h(\vec{x}) = \frac{1}{D} \sum_{i=1}^{D} \left(z_i \sin\left(\sqrt{z_i}\right)\right) = 0$$

$$x \in [-50,50]^D$$

Anexo II. Tablas comparativas de la experimentación en 10 dimensiones

Tabla 2.1 Comparativa entre los algoritmos: Rf-ED-S ran-Conf1, Rf-ED-S ran-Conf2, Rf-ED-S ran-Conf3, Rf-ED-S ran-Conf4, Rf-ED-S ran-Conf5 y Rf-ED-S ran-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	0.84	0	0	0.48	0	0
	AFES	1.14e+05			2.85 e+04		
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	1	0	0	0	0	0
	AFES	6.43 e+04					
C4	**P**	1	0.92	0.28	0.44	0.44	0.4
	AFES	2.67 e+04	1.52 e+05	1.63 e+05	3.44 e+04	1.49 e+05	1.06 e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	1	0.96	0.92	0.76	0	0
	AFES	7.39 e+04	1.72 e+05	1.52 e+05	9.60 e+04		
C8	**P**	0.32	0.88	0.88	0.2	0.64	0.6
	AFES	5.64 e+04	1.35 e+05	1.39 e+05	6.08 e+04	1.42 e+05	1.40 e+05
C9	**P**	0	0	0	0	0	0
	AFES						
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	1	1	0.96	0.8	0.56	0.44
	AFES	3.79 e+04	6.89 e+04	6.34 e+04	3.72 e+04	9.36 e+04	5.13 e+04
C12	**P**	0	0	0	0	0	0
	AFES						
C13	**P**	0.96	0	0	0.36	0	0
	AFES	1.10 e+05			4.05 e+04		
C14	**P**	0	0	0	0	0	0
	AFES						
C15	**P**	0	0	0	0	0	0
	AFES						
C16	**P**	0	0	0	0	0	0
	AFES						
C17	**P**	0	0	0	0	0	0
	AFES						
C18	**P**	0	0	0	0	0	0
	AFES						

Tabla 2.2 Comparativa entre los algoritmos: Rf-ED-S best-Conf1, Rf-ED-S best-Conf2, Rf-ED-S best-Conf3, Rf-ED-S best-Conf4, Rf-ED-S best-Conf5 y Rf-ED-S best-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	0	0	0	0	0	0
	AFES						
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0	0.48	0	0	0.16	0
	AFES		1.88e+05			1.68e+05	
C4	**P**	0.16	0.84	0.44	0	0.76	0.2
	AFES	9.88e+03	3.51e+04	5.71e+04		5.14e+04	6.90e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	0	0.8	0.84	0	0.88	0
	AFES		6.77e+04	5.08e+04		6.18e+04	
C8	**P**	0	0.12	0.16	0	0.2	0.04
	AFES		4.49e+04	5.40e+04		3.57e+04	3.16e+04
C9	**P**	0	0	0	0	0	0
	AFES						
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	0.32	0.84	0.84	0.12	0.88	0.92
	AFES	3.62 e+03	2.71 e+04	2.16e+04	4.93 e+03	3.44 e+04	2.75 e+04
C12	**P**	0	0	0	0	0	0
	AFES						
C13	**P**	0	0.04	0.16	0	0.04	0
	AFES		5.77e+04	4.91e+04		3.48e+04	
C14	**P**	0	0.6	0.12	0	0.48	0.52
	AFES		1.37e+05	1.77e+05		1.36e+05	1.53e+05
C15	**P**	0	0	0	0	0	0
	AFES						
C16	**P**	0	0	0	0	0	0
	AFES						
C17	**P**	0	0	0	0	0	0
	AFES						
C18	**P**	0	0	0	0	0	0
	AFES						

Tabla 2.3 Tabla comparativa entre los 6 mecanismos de control (basados en una función senoidal) probados en el algoritmo: Rf-ED-S best-(x, √x, ∜x, x^2, x^4, Sen, Cos,S, 0.5, 0.9)-(x, √x, ∜x, x^2, x^4, Sen, Cos, -S, 0.5, 0.9) en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

CR/F	**x**	**√x**	**∜x**	**x^2**	**x^4**	**Sen**	**Cos**	**S**	**0.5**	**-x**	**-√x**	**-∜x**	**-x^2**	**-x^4**	**-Sen**	**-Cos**	**-S**
x	7	6	5	4	2	1	5	6	8	7	6	5	7	5	3	1	8
√x	0	0	1	0	0	0	0	0	1	5	0	1	5	3	1	0	1
∜x	0	0	1	0	0	1	0	1	1	0	0	1	5	3	0	0	1
x^2	6	5	6	4	1	2	4	6	1	6	5	5	5	4	5	3	7
x^4	4	3	2	4	4	2	4	6	7	5	2	2	6		5	3	7
Sen	2	2	1	2	1	3	4	2	3	2	1	2	3	2	6	1	3
Cos	2	1	1	2	2	2	1	1	2	1	1	1	1	3	2	3	1
S	1	6	6	0	0	1	1	7	8	6	6	6	7	6	0	1	8
0.9	0	3	3	3	0	0	1	2		6	3	2	2	7	0	0	1
-x	2	2	2	0	0	0	0	2	1	5	1	2	6	6	0	0	2
-√x	0	1	1	0	0	0	0	1	1	5	1	1	5	2	1	0	2
-∜x	0	1	1	4	0	0	1	1	1	4	1	1	5	3	0	0	1
-x^2	0	2	2	0	0	0	0	3	1	5	2	2	1	5	0	0	1
-x^4	0	0	2	0	0	0	1	1	2	5	0	2	5	6	0	0	1
-Sen	0	2	2	1	1	0	3	2	2		1	2	2	4	3	1	3
-Cos	1	1	1	1	1	1	2	1	2	1	1	2	1	2	1	1	1
-S	0	4	4	0	0	0	0	7	4	6	4	5	6	7	0	0	7

Tabla 2.4 Comparativa entre los algoritmos: Rf-EDVC-S ran-best-Conf1, Rf-EDVC-S ran-best-Conf2, Rf-EDVC-S ran-best-Conf3, Rf-EDVC-S ran-best-Conf4, Rf-EDVC-S ran-best-Conf5 y Rf-EDVC-S ran-best-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	0	0.48	0.64	0	0.28	0
	AFES		2.39e+04	3.68e+04		1.98e+04	
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	1	0	0	0	0	0
	AFES	5.37e+05					
C4	**P**	1	0.96	0.36	0.28	1	0.84
	AFES	2.93e+04	1.51e+05	1.26e+05	3.01e+04	1.18e+05	6.76e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	0	0.84	0.8	0	1	
	AFES		7.05e+04	5.43e+04		6.09e+04	
C8	**P**	0	0.24	0.08	0	0.2	0
	AFES		3.66e+04	4.13e+04		3.19e+04	
C9	**P**	0	0	0	0	0	0
	AFES						
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	1	1	1	0.88	0.4	0.6
	AFES	3.33e+04	9.36e+04	6.12e+04	4.92e+04	7.22e+04	8.18e+04
C12	**P**	0	0	0	0	0	0
	AFES						
C13	**P**	0.04	0.16	0.56	0	0.12	0.04
	AFES	1.10e+04	5.43e+04	5.64e+04		3.81e+04	2.22e+04
C14	**P**	0	0.12	0	0	0.48	0.08
	AFES		1.96e+05			1.36e+05	1.89e+05
C15	**P**	0	0	0	0	0	0
	AFES						
C16	**P**	0	0	0	0	0	0
	AFES						
C17	**P**	0	0	0	0	0	0
	AFES						
C18	**P**	0	0	0	0	0	0
	AFES						

Tabla 2.5 Comparativa entre los algoritmos: Rf-EDVC-S best-ran-Conf1, Rf-EDVC-S best-ran-Conf2, Rf-EDVC-S best-ran-Conf3, Rf-EDVC-S best-ran-Conf4, Rf-EDVC-S best-ran-Conf5 y Rf-EDVC-S best-ran-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	0	0.44	0.68	0	0.48	0
	AFES		2.38e+04	3.47e+04		2.16e+04	
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0.92	0	0	0	0	0
	AFES	5.29e+04					
C4	**P**	1	0.84	0.4	0.4	1	0.96
	AFES	2.52e+04	1.43e+05	1.45e+05	2.66e+04	1.26e+05	7.40e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	0	0.8	0.72	0	0.8	0
	AFES		6.30e+04	5.17e+04		6.21e+04	
C8	**P**	0	0.12	0.16	0	0.28	0.08
	AFES		4.32e+04	3.97e+04		3.04e+04	2.14e+04
C9	**P**	0	0	0	0	0	0
	AFES						
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	1	1	1	0.8	0.44	0.68
	AFES	3.42e+04	7.75e+04	6.29e+04	4.54e+04	5.75e+04	5.22e+04
C12	**P**	0	0	0	0	0	0
	AFES						
C13	**P**	0.04	0.04	0.08	0	0.12	0
	AFES	7.62e+02	4.19e+04	4.76e+04		4.71e+04	
C14	**P**	0	0.64	0	0	0.04	0.36
	AFES		1.61e+05			1.60e+05	1.62e+05
C15	**P**	0	0	0	0	0	0
	AFES						
C16	**P**	0	0	0	0	0	0
	AFES						
C17	**P**	0	0	0	0	0	0
	AFES						
C18	**P**	0	0	0	0	0	0
	AFES						

Tabla 2.6 Comparativa entre los algoritmos: ε-EDVC-S ran-best-Conf1, ε-EDVC-S ran-best-Conf2, ε-EDVC-S ran-best-Conf3, ε-EDVC-S ran-best-Conf4, ε-EDVC-S ran-best-Conf5 y ε-EDVC-S ran-best-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	1	1	0.64	1	1	0.64
	AFES	2.98e+04	3.11e+04	1.65e+04	3.22e+04	3.26e+04	1.65e+04
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0	0	0.08	0	0.12	0
	AFES			5.34e+04		1.33e+05	
C4	**P**	0.48	0.96	1	0.84	0.96	0.88
	AFES	1.34e+04	3.68e+04	3.42e+04	2.85e+04	3.87e+04	2.50e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	1	1	0.96	0.72	0	0
	AFES	7.31e+04	1.73e+05	1.61e+05	9.06e+04		
C8	**P**	0.12	0.72	0.72	0.2	0.56	0.56
	AFES	4.67e+04	1.30e+05	1.34e+05	7.06e+04	1.23e+05	1.00e+05
C9	**P**	0	0	0	0	0	0
	AFES						
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	0.16	0.68	0.76	0.2	0.72	0.16
	AFES	1.56e+04	3.85e+04	3.88e+04	1.54e+04	5.02e+04	1.56e+04
C12	**P**	0.04	0	0	0	0	0.04
	AFES	2.12e+04					2.12e+04
C13	**P**	0	0.2	0.08	0.08	0.12	0
	AFES		2.87e+04	2.45e+04	3.20e+04	2.58e+04	
C14	**P**	0.24	0.68	0.36	0.08	0	0.24
	AFES	1.63e+05	1.77e+05	1.60e+05	1.12e+05		1.63e+05
C15	**P**	0	0.08	0.12	0	0	0
	AFES		1.82e+05	1.19e+05			
C16	**P**	0	0.64	0.56	0.08	0.44	0
	AFES		1.85e+04	2.27e+04	5.55e+04	3.52e+04	
C17	**P**	0	0.04	0	0	0.04	0
	AFES		2.70e+04			2.55e+04	
C18	**P**	0	0	0	0	0	0
	AFES						

Tabla 2.7 Resumen de la medida P en los algoritmos: ε-EDVC-S ran-best-Conf2, ε-EDVC-S ran-best-Conf3 en 10 dimensiones.

DECV2	DECV3
Ganados(+)	5
Perdidos(-)	4
Empates(=)	9

Tabla 2.8 Comparativa entre los algoritmos: ε-EDVC-S best-ran-Conf1, ε-EDVC-S best-ran-Conf2, ε-EDVC-S best-ran-Conf3, ε-EDVC-S best-ran-Conf4, ε-EDVC-S best-ran-Conf5 y ε-EDVC-S best-ran-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	Métrica	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
C1	**P**	0.96	1	1	0.56	1	1
	AFES	4.39e+04	2.98e+04	3.54e+04	1.64e+04	2.97e+04	3.59e+04
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0	0.04	0.08	0	0.04	0
	AFES		1.69e+05	6.63e+04		1.77e+05	
C4	**P**	0.52	1	1	0.64	1	0.84
	AFES	1.33e+04	2.87e+04	3.81e+04	2.67e+04	1.77e+05	3.53e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	0.36	1	1	0.8	0	0.04
	AFES	5.67e+04	1.71e+05	1.47e+05	9.60e+04		1.66e+05
C8	**P**	0.36	0.92	0.96	0.28	0.84	0.68
	AFES	5.67e+04	1.28e+05	1.37e+05	5.53e+04	1.20e+05	1.00e+05
C9	**P**	0	0	0	0	0	0
	AFES						
C10	**P**	0	0	0	0	0	0
	AFES						
C11	**P**	0.16	0.52	0.88	0.2	0.68	0.6
	AFES	8.51e+03	4.86e+04	3.98e+04	1.57e+04	3.21e+04	5.89e+04
C12	**P**	0	0	0	0	0	0
	AFES						
C13	**P**	0.16	0.44	0.28	0.04	0.48	0.48
	AFES	7.97e+04	6.07e+04	4.59e+04	2.98e+04	6.03e+04	4.99e+04
C14	**P**	0.2	0.8	0.48	0.4	0	0.04
	AFES	1.47e+05	1.74e+05	1.75e+05	1.24e+05		1.88e+05
C15	**P**	0	0.04	0.04	0	0	0
	AFES		8.87e+04	1.18e+05			
C16	**P**	0	0.48	0.6	0.04	0.44	0.04
	AFES		4.62e+04	2.42e+04	5.40e+04	2.09e+04	2.51e+04

C17	**P** **AFES**	0	0	0	0	0.08 9.89e+04	0
C18	**P** **AFES**	0	0	0	0	0	0

Tabla 2.9 Resumen de la medida P en los algoritmos: ε-EDVC-S best-ran-Conf2, ε-EDVC-S best-ran-Conf3 en 10 dimensiones.

DECV3	**DECV2**
Ganados(+)	4
Perdidos(-)	2
Empates(=)	12

Tabla 2.10 Comparativa entre los algoritmos: ε-EDVD-S ran-best-ran-Conf1, ε-EDVD-S ran-best-ran-Conf2, ε-EDVD-S ran-best-ran-Conf3, ε-EDVD-S ran-best-ran-Conf4, ε-EDVD-S ran-best-ran-Conf5 y ε-EDVD-S ran-best-ran-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	**Métrica**	**Conf1**	**Conf2**	**Conf3**	**Conf4**	**Conf5**	**Conf6**
C1	**P** **AFES**	0.96 5.63e+04	1 2.96e+04	1 3.06e+04	0.6 1.68e+04	1 3.21e+04	1 3.10e+04
C2	**P** **AFES**	0.16 8.09e+03	0.16 4.22e+04	0	0.08 1.75e+04	0.12 3.75e+04	0.16 5.77e+04
C3	**P** **AFES**	0.8 7.11e+04	0	0	0.44 1.22e+05	0.12 1.61e+05	0.04 1.48e+05
C4	**P** **AFES**	1 4.61e+04	1 7.40e+04	1 9.02e+04	0.6 4.87e+04	1 7.09e+04	1 5.85e+04
C5	**P** **AFES**	0	0	0	0.04 1.01e+05	0	0
C6	**P** **AFES**	0	0	0	0	0	0.04 1.27e+05
C7	**P** **AFES**	1 7.88e+04	1 1.75e+05	0.96 1.54e+05	0.84 1.07e+05	0	0.04 7.58e+04
C8	**P** **AFES**	0.36 6.25e+04	0.52 1.26e+05	0.76 1.32e+05	0.16 5.69e+04	0.64 1.21e+05	0.6 1.02e+05
C9	**P** **AFES**	0.48 6.80e+04	0.12 1.66e+05	0	0.08 6.27e+04	0.12 1.37e+05	0.16 1.22e+05
C10	**P** **AFES**	0	0	0	0	0	0
C11	**P** **AFES**	0.96 6.99e+04	1 6.30e+04	1 6.53e+04	0.96 6.54e+04	0.8 5.69e+04	0.88 7.03e+04
C12	**P** **AFES**	0.12 6.79e+04	0.2 1.13e+05	0.4 9.86e+04	0	0.44 8.40e+04	0.32 8.16e+04
C13	**P** **AFES**	0.28 7.49e+04	1 9.63e+04	1 5.95e+04	0.08 2.93e+04	1 1.07e+05	1 1.01e+05

C14	**P**	0.6	0.88	0.68	0.48	0.04	0.08
	AFES	8.80e+04	1.70e+04	1.63e+05	1.03e+04	1.49e+05	1.66e+05
C15	**P**	0.08	0.2	0.16	0.04	0.16	0.2
	AFES	4.82e+04	1.27e+04	1.53e+04	4.87e+04	1.38e+05	9.69e+04
C16	**P**	0	0.16	0.48	0.16	0.16	0.32
	AFES		3.05e+04	2.94e+04	1.79e+04	3.19e+04	2.87e+04
C17	**P**	0.64	0	0	0.64	0	0
	AFES	2.65e+04			2.66e+04		
C18	**P**	0.64	0	0	0.6	0	0.12
	AFES	3.08e+04			5.68e+04		8.89e+04

Tabla 2.11 Resumen de la medida P en los algoritmos: ε-EDVC-S ran-best-ran-Conf2, ε-EDVC-S ran-best-ran-Conf3 en 10 dimensiones.

Conf1	**Conf4**	**Conf6**
Ganados(+)	12	7
Perdidos(-)	2	7
Empates(=)	4	4

Tabla 2.12 Comparativa entre los algoritmos: ε-EDVC-S best-ran-best-Conf1, ε-EDVC-S best-ran-best-Conf2, ε-EDVC-S best-ran-best-Conf3, ε-EDVC-S best-ran-best-Conf4, ε-EDVC-S best-ran-best-Conf5 y ε-EDVC-S best-ran-best-Conf6 en 10 dimensiones. P: probabilidad de convergencia y AFES: Promedio de evaluaciones requerida para alcanzar el óptimo.

Función	**Métrica**	**Conf1**	**Conf2**	**Conf3**	**Conf4**	**Conf5**	**Conf6**
C1	**P**	0.08	0.72	0.92	0.12	0.84	0.24
	AFES	3.67e+03	8.52e+03	1.39e+04	6.45e+03	9.06e+03	5.04e+03
C2	**P**	0	0	0	0	0	0
	AFES						
C3	**P**	0	0.6	0.36	0	0.24	0
	AFES		1.68e+05	1.22e+05		1.53e+05	
C4	**P**	.48	1	1	0.64	1	0.88
	AFES	2.06e+05	2.87e+04	3.48e+04	2.53e+04	3.81e+04	3.51e+04
C5	**P**	0	0	0	0	0	0
	AFES						
C6	**P**	0	0	0	0	0	0
	AFES						
C7	**P**	0	0.68	0.88	0	0.8	0
	AFES		6.92e+04	5.21e+04		6.18e+04	
C8	**P**	0	0.2	0.28	0	0.16	0.04
	AFES		3.42e+04	4.33e+04		4.01e+04	6.43e+04
C9	**P**	0	0	0	0	0	0
	AFES						
C10	**P**	0	0.04	0	0	0	0.04
	AFES		1.82e+05				1.33e+05

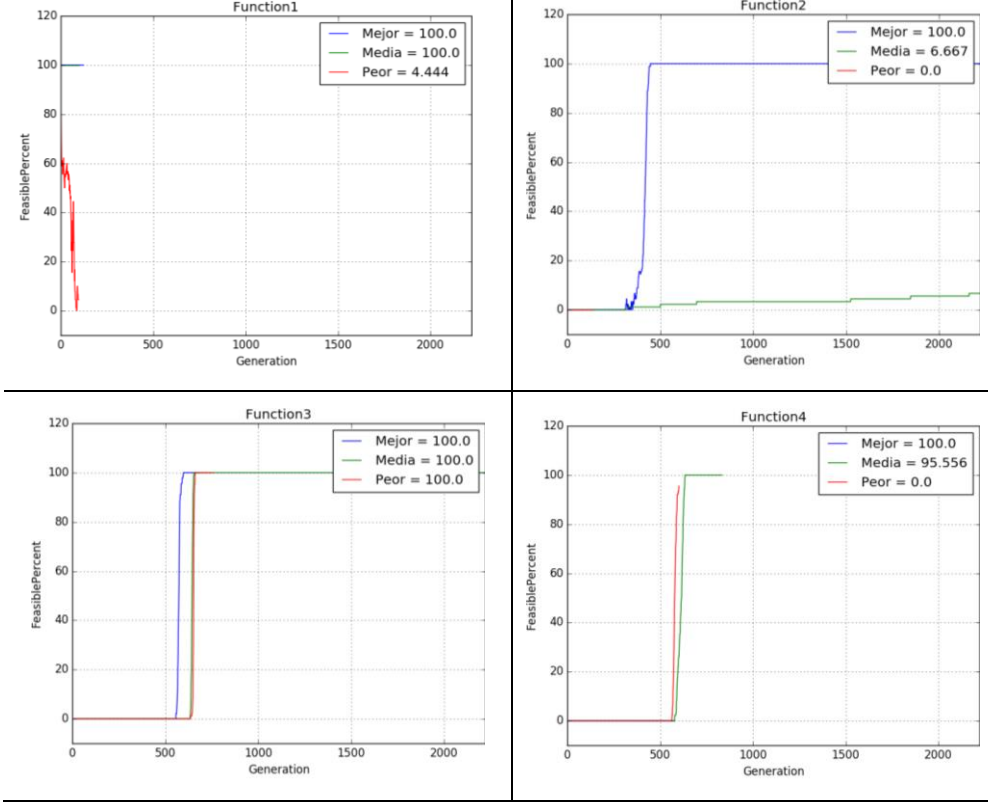
C11	**P** **AFES**	0.12 4.88e+03	0.6 3.79e+04	0.76 5.47e+04	0.2 9.98e+03	0.6 5.08e+04	0.44 4.57e+04
C12	**P** **AFES**	0	0	0	0	0	0
C13	**P** **AFES**	0	0.24 3.07e+04	0.44 3.38e+04	0.08 2.82e+04	0.56 3.29e+04	0.36 3.68e+04
C14	**P** **AFES**	0	0.76 7.08e+04	0.68 7.48e+04	0	0.68 7.65e+04	0.2 1.35e+05
C15	**P** **AFES**	0	0.04 1.17e+05	0.2 6.97	0	0.08 6.52e+04	0
C16	**P** **AFES**	0.08 1.39e+05	0.32 1.65e+04	0.64 2.66e+04	0.04 3.41e+04	0.6 1.75e+04	0.04 2.72e+04
C17	**P** **AFES**	0.04 3.78e+04	0	0.04 8.89e+04	0	0.08 7.36e+04	0
C18	**P** **AFES**	0	0.12 6.08e+04	0	0.04 1.87e+05	0.12 1.89e+05	0

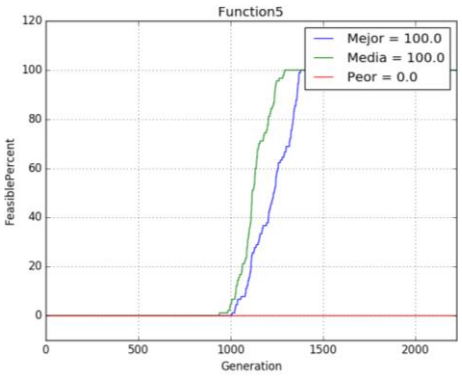
Tabla 2.13 Resumen de la medida P en los algoritmos: ε-EDVC-S best-ran-best-Conf2, ε-EDVC-S best-ran-best-Conf3 en 10 dimensiones.

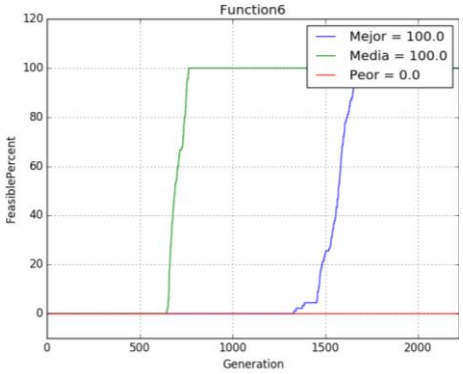
Conf3	**Conf2**	**Conf5**
Ganados(+)	8	7
Perdidos(-)	4	3
Empates(=)	6	8

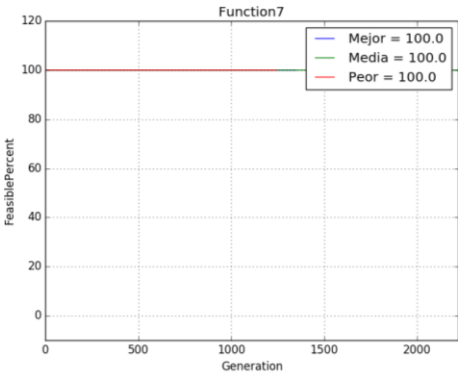
Anexo III. Gráficas de factibilidad

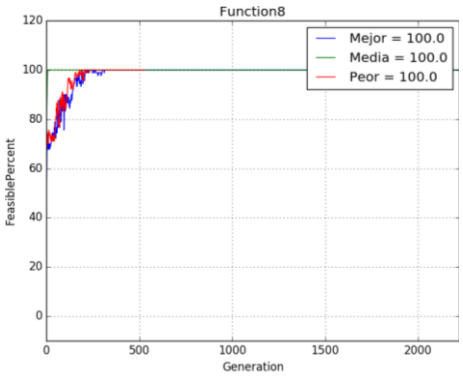
Tabla 3.1 Comportamiento algorítmico de ε-ED-S ran-Conf1 sobre las 18 funciones del benchmarck del CRC 2010 en 10 dimensiones. La línea azul, verde y la roja representan la mejor, la media y la peor ejecución respectivamente.

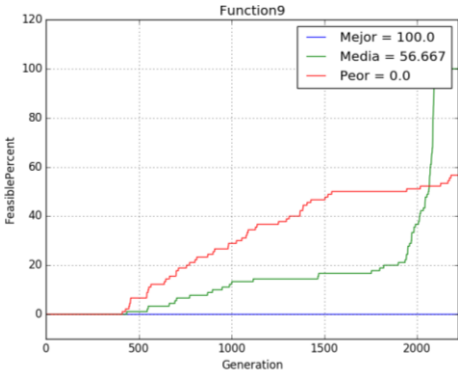


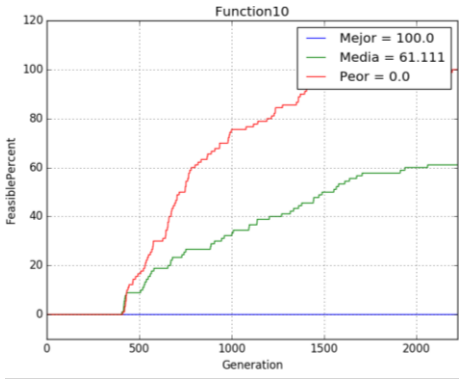

Function5
Mejor = 100.0
Media = 100.0
Peor = 0.0
FeasiblePercent
Generation

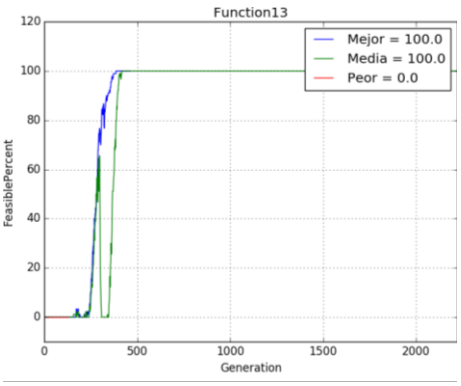

Function6
Mejor = 100.0
Media = 100.0
Peor = 0.0
FeasiblePercent
Generation

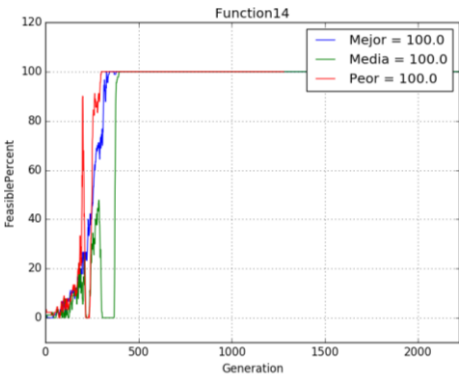

Function7
Mejor = 100.0
Media = 100.0
Peor = 100.0
FeasiblePercent
Generation

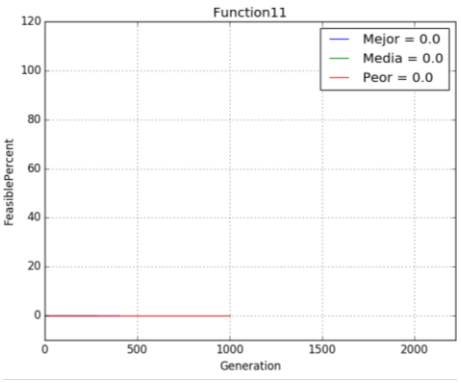

Function8
Mejor = 100.0
Media = 100.0
Peor = 100.0
FeasiblePercent
Generation

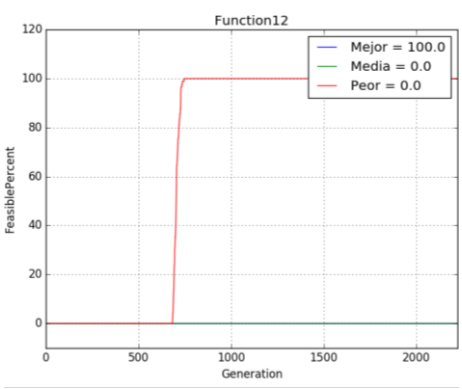

Function9
Mejor = 100.0
Media = 56.667
Peor = 0.0
FeasiblePercent
Generation

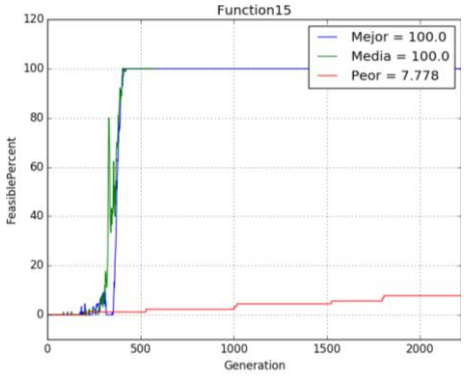

Function10
Mejor = 100.0
Media = 61.111
Peor = 0.0
FeasiblePercent
Generation

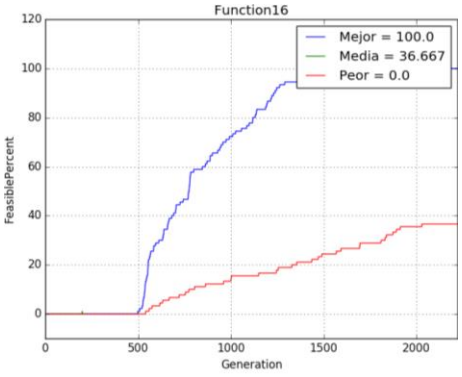

Function13
Mejor = 100.0
Media = 100.0
Peor = 0.0
FeasiblePercent
Generation

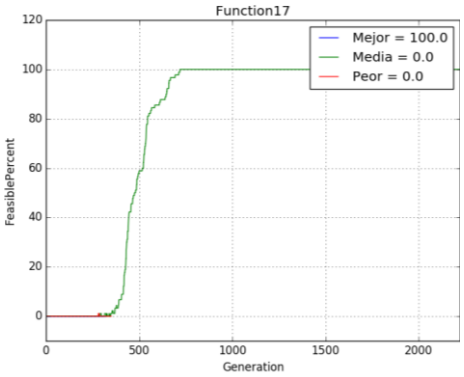

Function14
Mejor = 100.0
Media = 100.0
Peor = 100.0
FeasiblePercent
Generation

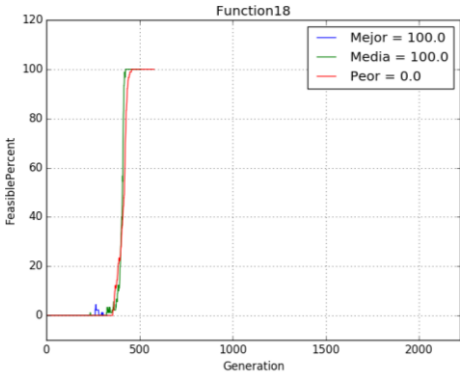

Function11
Mejor = 0.0
Media = 0.0
Peor = 0.0
FeasiblePercent
Generation


Function12
Mejor = 100.0
Media = 0.0
Peor = 0.0
FeasiblePercent
Generation


Function15
Mejor = 100.0
Media = 100.0
Peor = 7.778
FeasiblePercent
Generation


Function16
Mejor = 100.0
Media = 36.667
Peor = 0.0
FeasiblePercent
Generation


Function17
Mejor = 100.0
Media = 0.0
Peor = 0.0
FeasiblePercent
Generation
0
20
40
60
80
100
120
0
500
1000
1500
2000


Function18
Mejor = 100.0
Media = 100.0
Peor = 0.0
FeasiblePercent
Generation
0
20
40
60
80
100
120
0
500
1000
1500
2000

Anexo IV. Tablas comparativas del algoritmo ε-ED-S ran-Conf1 ajustado en 10 dimensiones

Tabla 4.1 Comparativa entre 9 versiones del algoritmo: ε-ED-S ran-Conf1 en 10 dimensiones, variantes respecto al parámetro cp, con valores entre 1 y 9, donde Tc fue fijado en 0.3. P: probabilidad de convergencia.

Función	Métrica	cp=1	cp=2	cp=3	cp=4	cp=5	Tc=6	cp=7	cp=8	cp=9
C1	**P**	1	1	1	1	1	0.96	1	0.96	1
C2	**P**	0.8	0.64	0.76	0.56	0.44	0.4	0.12	0.2	0.04
C3	**P**	0.96	0.96	0.96	1	0.96	0.88	0.92	1	0.96
C4	**P**	0.96	1	1	1	1	1	1	1	1
C5	**P**	0	0.08	0	0	0	0	0	0	0
C6	**P**	0	0.08	0	0.04	0	0.04	0.04	0.08	0
C7	**P**	1	1	1	1	1	1	1	1	1
C8	**P**	0.28	0.52	0.36	0.28	0.2	0.44	0.44	0.4	.32
C9	**P**	0.12	0.16	0.16	0.72	0.92	0.72	0.64	0.64	0.68
C10	**P**	0.24	0.08	0.28	0.16	0.68	0.36	0.16	0.04	0.08
C11	**P**	1	0.96	1	1	1	1	1	1	1
C12	**P**	0.08	0.04	0.04	0.08	0.12	0.16	0.12	0.04	0
C13	**P**	0.96	0.92	0.88	0.84	0.96	0.92	0.92	1	1
C14	**P**	0.88	0.94	0.92	0.84	0.92	0.08	0.76	0.88	0.64
C15	**P**	0.8	0.72	0.56	0.44	0.4	0.08	0.2	0	0.08
C16	**P**	0.28	0.2	0.2	0.2	0.2	0.16	0.24	0.12	0.04
C17	**P**	1	0.92	0.88	0.76	0.8	0.84	0.68	0.8	0.84
C18	**P**	0.88	0.92	0.76	0.8	0.68	0.84	0.72	0.56	0.52

Tabla 4.2 Comparativa entre 9 versiones del algoritmo: ε-ED-S ran-Conf1 en 10 dimensiones, variantes respecto al parámetro cp, con valores entre 1 y 9, donde Tc fue fijado en 0.4. P: probabilidad de convergencia.

Función	Métrica	cp=1	cp=2	cp=3	cp=4	cp=5	Tc=6	cp=7	cp=8	cp=9
C1	**P**	1	0.96	0.96	1	0.88	1	0.96	1	1
C2	**P**	0.96	0.92	0.84	0.76	0.68	0.64	0.6	0.6	0.28
C3	**P**	1	0.96	1	1	1	1	0.96	1	0.96
C4	**P**	0.84	0.8	0.92	0.96	1	1	1	1	1
C5	**P**	0.04	0.04	0	0	0.04	0	0	0	0
C6	**P**	0	0	0.08	0	0.04	0	0.04	0	0
C7	**P**	1	1	1	1	1	1	1	0.92	1
C8	**P**	0.36	0.32	0.28	0.4	0.36	0.48	0.24	0.92	0.36
C9	**P**	0.84	0.96	0.84	0.96	0.92	0.72	0.92	0.2	0.72
C10	**P**	0.68	0.72	0.88	0.76	0.68	0.68	0.68	0.68	0.4
C11	**P**	1	1	0.96	1	0.96	1	1	1	1
C12	**P**	0	0.12	0.04	0.08	0.2	0.08	0.04	0.04	0

C13	**P**	0.96	0.92	0.92	1	0.92	0.88	0.88	0.84	0.96
C14	**P**	0.88	0.8	0.84	0.88	0.92	0.88	0.72	0.68	0.96
C15	**P**	0.84	0.88	0.8	0.84	0.8	0.64	0.64	0.48	0.4
C16	**P**	0.32	0.32	0.52	0.36	0.48	0.36	0.4	0.48	0.24
C17	**P**	0.92	1	1	0.84	0.96	0.88	0.88	0.84	0.84
C18	**P**	1	0.92	0.8	0.96	0.84	0.96	0.88	0.72	0.72

Tabla 4.3 Comparativa entre 9 versiones del algoritmo: ε-ED-S ran-Conf1 en 10 dimensiones, variantes respecto al parámetro cp, con valores entre 1 y 9, donde Tc fue fijado en 0.5. P: probabilidad de convergencia.

Función	**Métrica**	**cp=1**	**cp=2**	**cp=3**	**cp=4**	**cp=5**	**Tc=6**	**cp=7**	**cp=8**	**cp=9**
C1	**P**	1	1	1	0.96	0.96	1	1	0.88	1
C2	**P**	0.88	0.92	0.68	0.84	0.68	0.4	0.52	0.4	0.24
C3	**P**	1	0.92	1	0.96	1	1	1	0.96	1
C4	**P**	0.96	1	0.96	1	1	1	1	1	1
C5	**P**	0.12	0.04	0	0	0	0	0	0	0
C6	**P**	0.04	0	0.08	0.04	0.04	0.04	0.12	0.08	0
C7	**P**	1	0.96	1	1	1	0.96	1	1	0.96
C8	**P**	0.24	0.24	0.2	0.28	0.36	0.24	1	.32	0.56
C9	**P**	0.68	0.72	0.6	0.68	0.88	0.84	0.44	0.68	0.72
C10	**P**	0.68	0.72	0.88	0.76	0.92	0.68	0.76	0.68	0.4
C11	**P**	1	1	0.96	1	0.96	1	1	1	1
C12	**P**	0	0.12	0.04	0.08	0.08	0.08	0.04	0.04	0
C13	**P**	0.96	0.92	0.92	1	0.88	0.88	0.88	0.84	0.96
C14	**P**	0.88	0.8	0.84	0.88	0.92	0.88	0.72	0.68	0.96
C15	**P**	0.84	0.88	0.8	0.84	0.92	0.64	0.64	0.48	0.4
C16	**P**	0.32	0.32	0.52	0.36	0.48	0.36	0.4	0.48	0.24
C17	**P**	0.92	1	1	0.84	0.96	0.88	0.88	0.84	0.84
C18	**P**	1	0.92	0.8	0.96	0.8	0.96	0.88	0.72	0.84

Tabla 4.4 Comparativa entre 9 versiones del algoritmo: ε-ED-S ran-Conf1 en 10 dimensiones, variantes respecto al parámetro cp, con valores entre 1 y 9, donde Tc fue fijado en 0.6. P: probabilidad de convergencia.

Función	**Métrica**	**cp=1**	**cp=2**	**cp=3**	**cp=4**	**cp=5**	**Tc=6**	**cp=7**	**cp=8**	**cp=9**
C1	**P**	1	1	0.92	1	0.92	1	0.92	0.96	0.96
C2	**P**	0.96	0.88	0.88	0.72	0.88	0.56	0.68	0.6	0.56
C3	**P**	0.96	0.92	1	0.92	0.96	1	0.92	1	1
C4	**P**	0.88	0.84	0.92	0.92	0.92	1	1	1	1
C5	**P**	0.08	0.04	0	0.04	0.04	0	0	0	0
C6	**P**	0.04	0	0.04	0.04	0.08	0	0.04	0.16	0.04
C7	**P**	0.96	1	1	0.96	1	1	1	1	1
C8	**P**	0.2	0.12	0.16	0.32	0.2	0.36	0.32	0.32	0.2
C9	**P**	0.96	0.92	0.92	0.88	0.88	0.88	0.88	0.84	0.84
C10	**P**	1	0.92	0.96	0.96	0.92	0.92	0.88	0.72	0.76

C11	**P**	0.92	0.68	0.8	0.92	0.8	0.88	0.76	0.92	0.96
C12	**P**	0.04	0.08	0.12	0.08	0.08	0.08	0.08	0	0
C13	**P**	0.92	0.76	0.8	1	0.76	0.96	1	0.8	1
C14	**P**	0.96	0.96	0.92	0.84	0.84	0.8	0.76	0.8	0.72
C15	**P**	0.84	0.8	0.88	0.88	0.76	0.72	0.64	0.72	0.84
C16	**P**	0.64	0.44	0.48	0.4	0.52	0.6	0.56	0.64	0.48
C17	**P**	0.96	0.92	0.96	0.92	0.96	0.96	0.96	0.92	0.8
C18	**P**	0.96	1	0.96	0.88	0.76	0.92	0.72	0.8	0.76

Tabla 4.5 Comparativa entre 9 versiones del algoritmo: ε-ED-S ran-Conf1 en 10 dimensiones, variantes respecto al parámetro cp, con valores entre 1 y 9, donde Tc fue fijado en 0.7. P: probabilidad de convergencia.

Función	**Métrica**	**cp=2**	**cp=3**	**cp=4**	cp=5	**cp=6**	**cp=7**	**cp=8**
C1	**P**	0.92	1	1	1	1	1	1
C2	**P**	1	0.84	0.68	0.88	0.88	0.6	0.8
C3	**P**	1	0.96	1	1	0.96	1	1
C4	**P**	0.6	0.84	0.92	1	1	1	1
C5	**P**	0.08	0.04	0	0.04	0.04	0	0
C6	**P**	0	0	0.04	0.04	0	0.04	0.04
C7	**P**	1	1	1	1	1	1	0.96
C8	**P**	0.4	0.24	0.28	0.28	0.2	0.32	0.32
C9	**P**	0.92	0.96	0.92	0.88	0.84	0.92	0.92
C10	**P**	0.96	0.84	0.92	0.92	0.88	0.88	0.84
C11	**P**	0.76	0.92	0.8	0.8	0.88	0.88	0.76
C12	**P**	0.08	0	0.04	0.12	0.04	0	0
C13	**P**	0.8	0.8	0.6	0.8	0.8	0.92	0.88
C14	**P**	0.96	0.92	1	0.92	0.8	0.88	0.96
C15	**P**	0.88	0.92	0.88	0.92	0.8	0.8	0.76
C16	**P**	0.48	0.64	0.64	0.76	0.6	0.68	0.4
C17	**P**	0.96	1	1	0.92	0.96	0.84	0.76
C18	**P**	0.92	0.96	0.88	0.88	0.96	0.84	0.88

Tabla 4.6 Comparativa entre 9 versiones del algoritmo: ε-ED-S ran-Conf1 en 10 dimensiones, variantes respecto al parámetro cp, con valores entre 1 y 9, donde Tc fue fijado en 0.8. P: probabilidad de convergencia.

Función	**Métrica**	**cp=1**	**cp=2**	**cp=3**	**cp=4**	**cp=5**	**Tc=6**	**cp=7**	**cp=8**	**cp=9**
C1	**P**	1	0.96	1	1	1	0.96	0.96	1	1
C2	**P**	1	0.84	0.92	0.8	0.84	0.84	0.68	0.68	0.68
C3	**P**	0.92	1	0.96	1	0.92	1	0.96	0.96	0.92
C4	**P**	0.52	0.64	0.84	0.8	1	1	1	1	1
C5	**P**	0.04	0.08	0	0.04	0	0.04	0	0	0
C6	**P**	0	0.04	0.04	0.08	0	0	0	0.04	0
C7	**P**	1	1	1	0.96	1	1	1	1	1
C8	**P**	0.08	0.2	0.4	0.04	0.44	0.28	0.36	0.2	0.28

C9	**P**	1	1	0.96	0.92	0.96	0.96	0.88	0.76	0.88
C10	**P**	0.96	0.92	1	0.84	0.92	0.92	0.76	0.92	0.76
C11	**P**	0.68	0.64	0.88	0.72	0.88	0.8	0.72	0.6	0.8
C12	**P**	0	0.04	0.04	0	0.04	0	0	0.08	0
C13	**P**	0.68	0.68	0.6	0.72	0.88	0.96	0.72	0.96	0.84
C14	**P**	0.92	0.92	0.96	0.88	0.96	0.92	0.92	0.88	0.84
C15	**P**	0.92	0.88	0.96	0.92	0.88	0.88	0.96	0.88	0.8
C16	**P**	0.32	0.64	0.52	0.48	0.64	0.6	0.88	0.76	0.68
C17	**P**	0.96	0.96	0.96	0.92	0.96	0.92	0.96	0.96	0.96
C18	**P**	1	0.92	0.92	0.96	0.96	0.92	0.92	0.8	0.8

Tabla 4.7 Comparativa entre 9 versiones del algoritmo: ε-ED-S ran-Conf1 en 10 dimensiones, variantes respecto al parámetro cp, con valores entre 1 y 9, donde Tc fue fijado en 0.9. P: probabilidad de convergencia.

Función	**Métrica**	**cp=1**	**cp=2**	**cp=3**	**cp=4**	**cp=5**	**Tc=6**	**cp=7**	**cp=8**	**cp=9**
C1	**P**	1	0.96	0.92	0.96	1	0.96	1	0.96	0.96
C2	**P**	0.96	0.92	0.92	0.92	0.84	0.72	0.88	0.72	0.8
C3	**P**	0.96	0.92	1	0.96	0.92	0.96	1	1	0.92
C4	**P**	0.32	0.48	0.64	1	1	1	1	1	1
C5	**P**	0.04	0.12	0.04	0	0	0.04	0	0	0.04
C6	**P**	0.04	0.04	0	0.04	0	0.08	0	0.04	0
C7	**P**	1	1	1	1	1	1	1	1	1
C8	**P**	0.16	0.24	0.12	0.28	0.44	0.4	0.28	0.16	0.2
C9	**P**	1	0.92	0.92	0.92	0.96	0.96	0.84	0.92	0.8
C10	**P**	0.92	0.96	0.88	0.96	0.92	0.96	0.88	0.92	0.92
C11	**P**	0.76	0.64	0.72	0.72	0.88	0.6	0.72	0.64	0.84
C12	**P**	0.12	0.04	0.04	0.04	0.04	0.04	0	0.04	0
C13	**P**	0.32	0.16	0.32	0.8	0.88	0.64	0.88	0.88	0.92
C14	**P**	1	0.88	0.92	0.84	0.96	0.88	1	0.96	0.88
C15	**P**	0.92	0.92	0.92	0.84	0.88	0.92	0.96	0.84	0.72
C16	**P**	0.52	0.56	0.52	0.6	0.64	0.6	0.56	0.36	0.76
C17	**P**	1	1	0.96	0.92	0.96	1	1	0.92	0.96
C18	**P**	1	1	0.96	0.96	0.96	0.88	0.92	1	0.96

Anexo V. Estudio de diversidad en 10 dimensiones

Tabla 5.1 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C1 en 10 dimensiones.

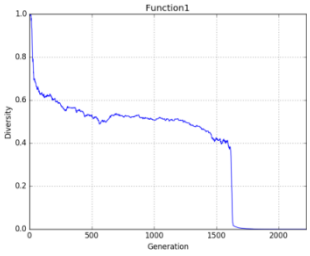
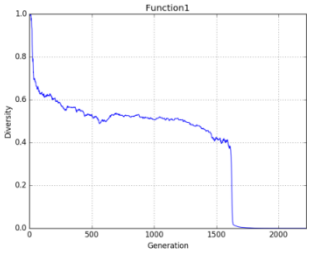
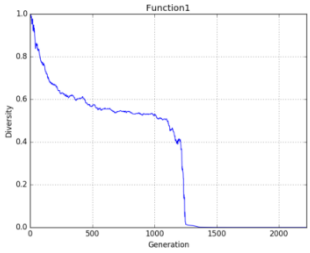
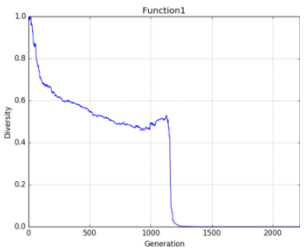
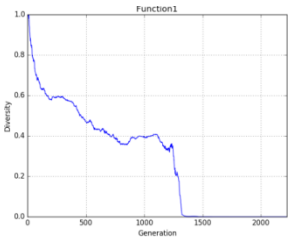
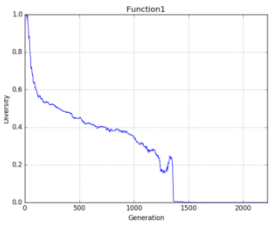
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.2 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C2 en 10 dimensiones.

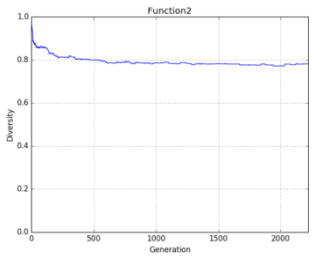
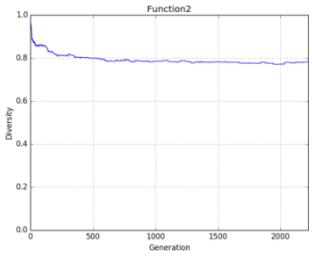
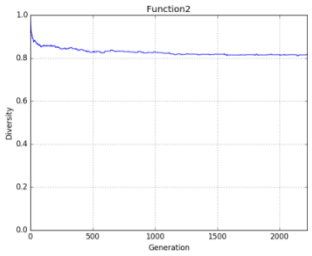
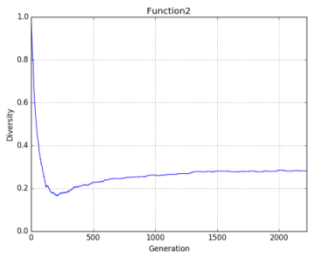
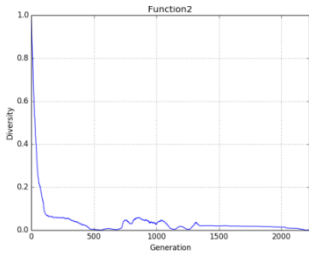
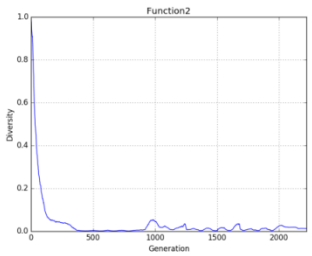
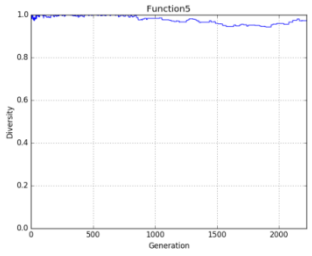
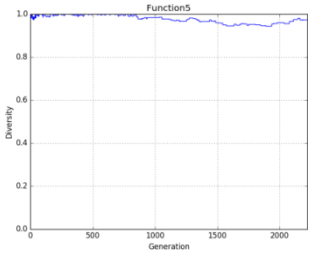
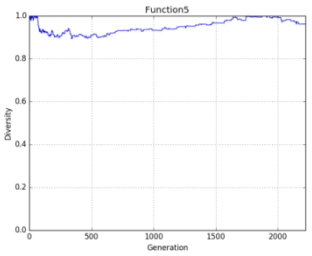
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.3 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C5 en 10 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

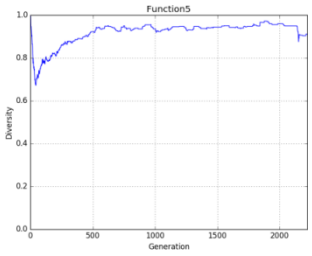
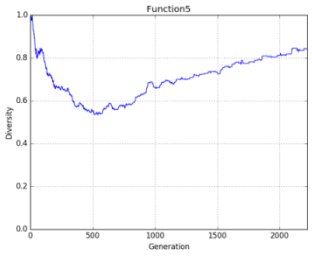
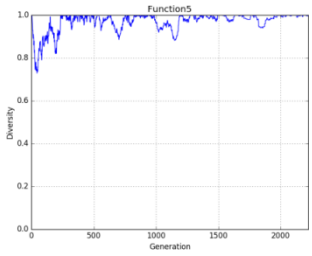
ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.4 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C6 en 10 dimensiones.

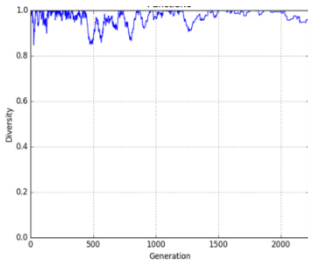
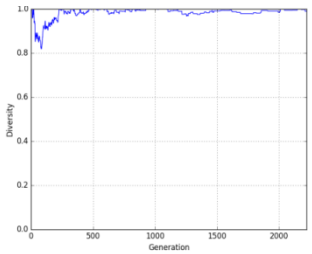
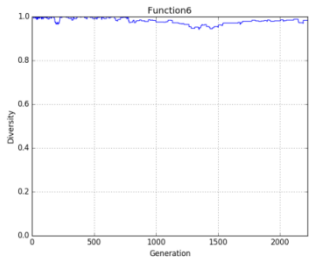
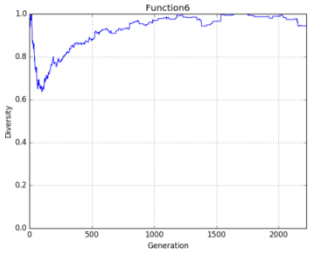
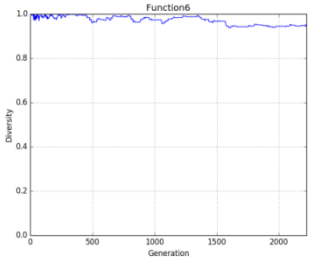
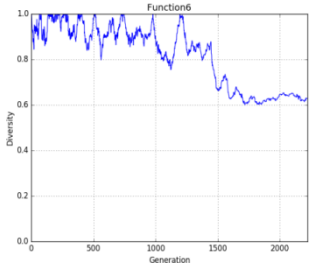
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.5 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C8 en 10 dimensiones.

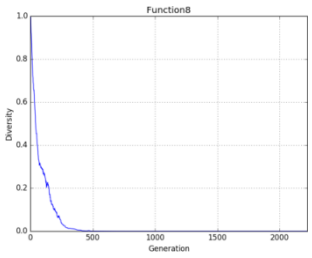
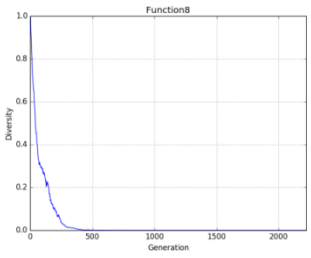
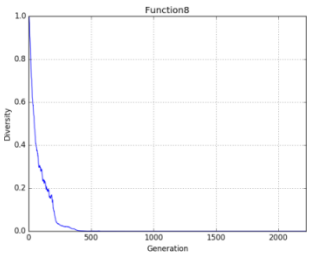
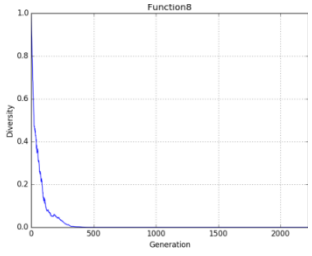
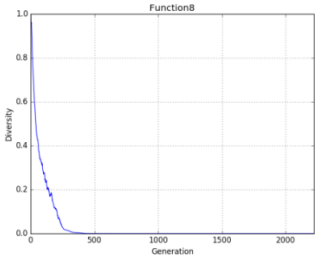
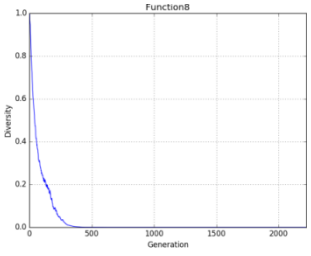
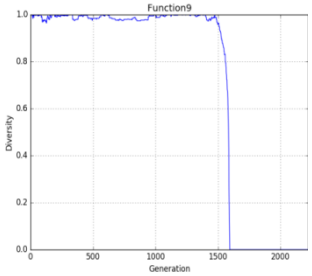
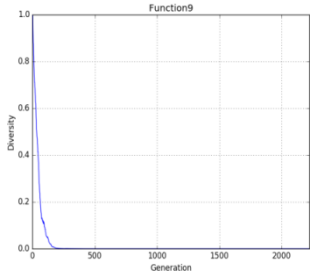
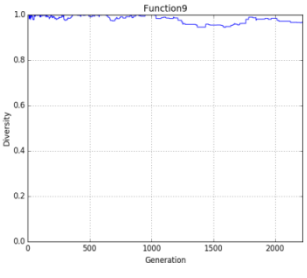
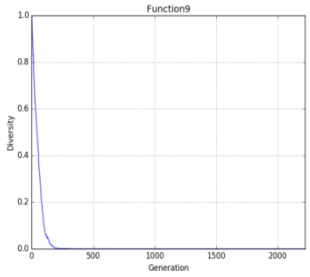
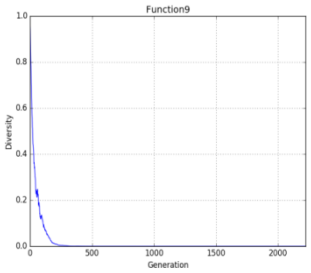
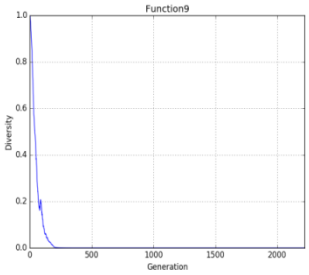
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.6 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C9 en 10 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

5.7 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C11 en 10 dimensiones.

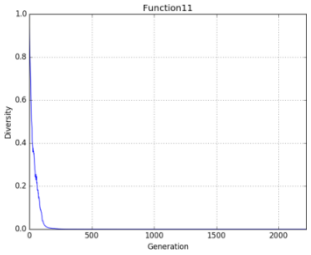
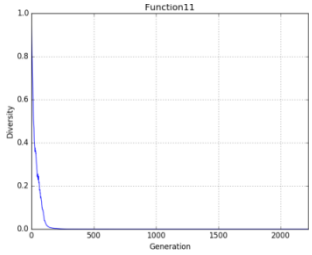
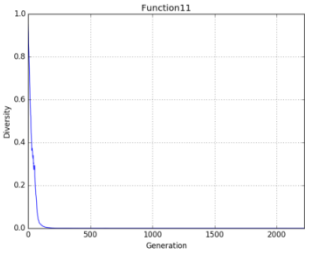
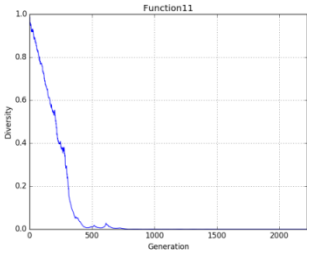
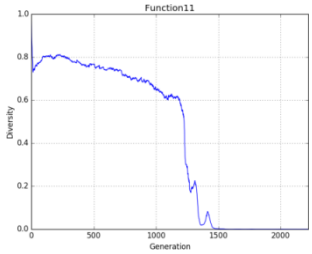
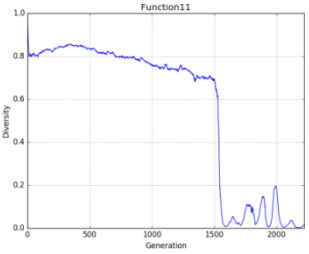
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.8 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C13 en 10 dimensiones.

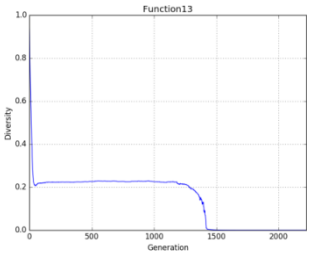
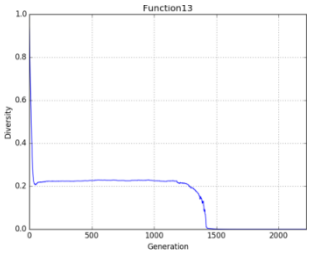
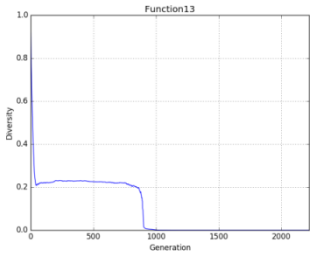
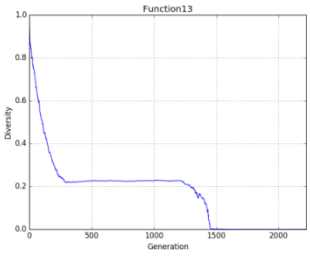
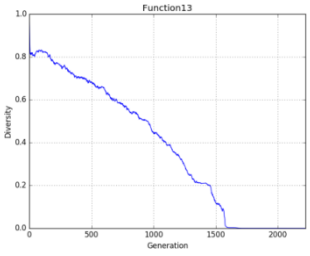
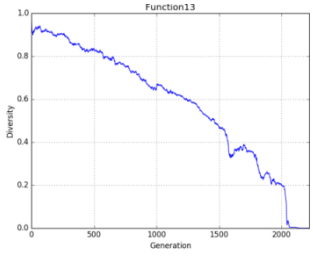
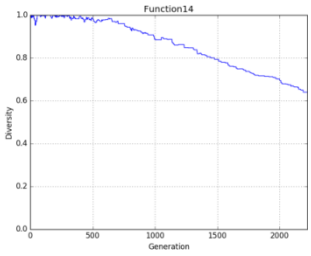
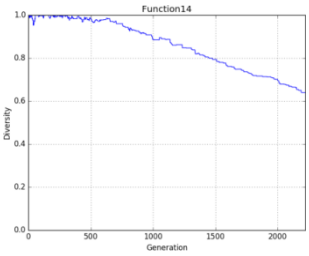
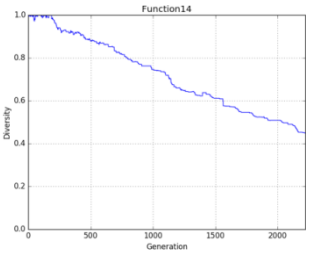
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.9 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C14 en 10 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

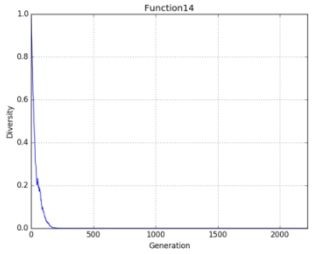
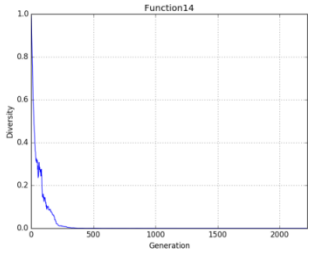
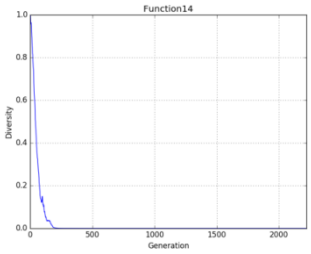
ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.10 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C15 en 10 dimensiones.

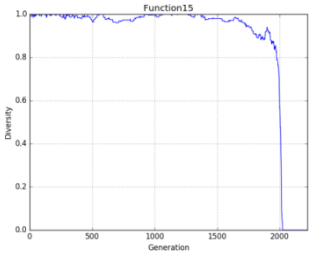
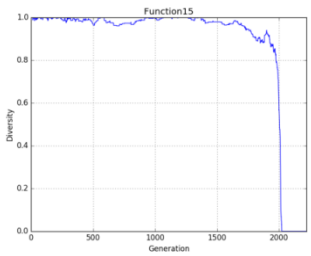
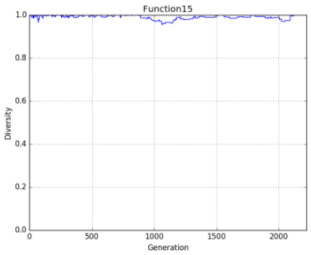
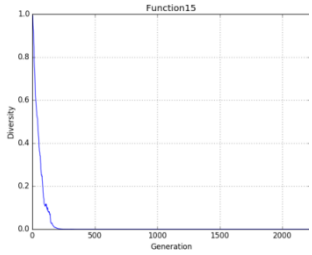
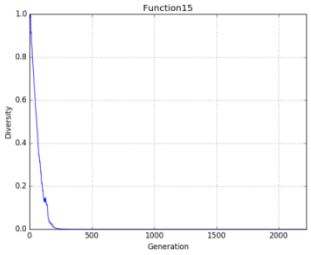
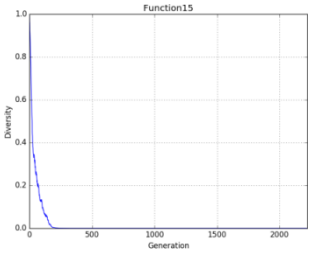
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.11 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C16 en 10 dimensiones.

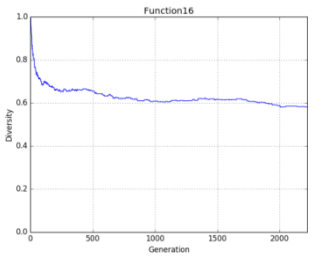
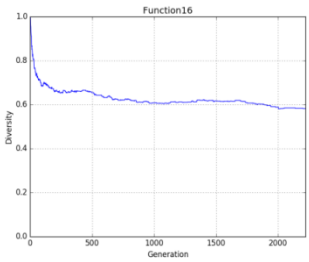
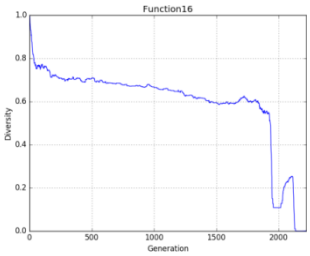
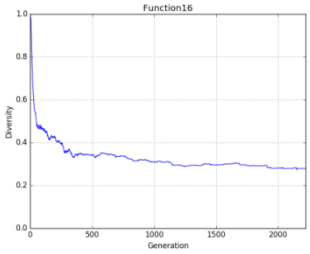
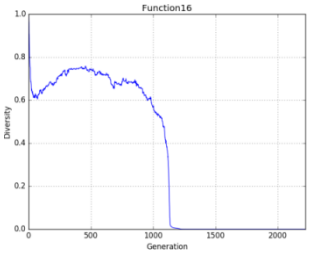
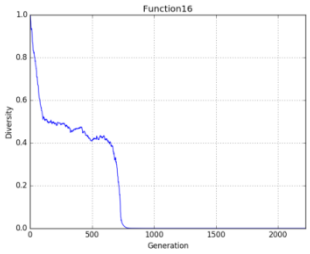
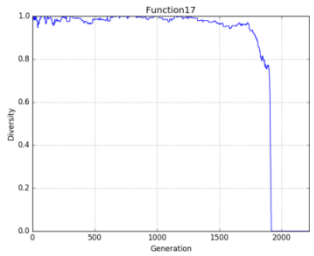
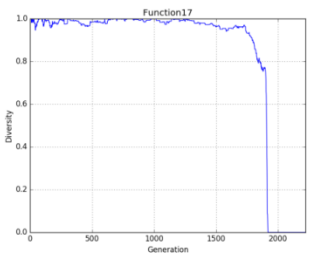
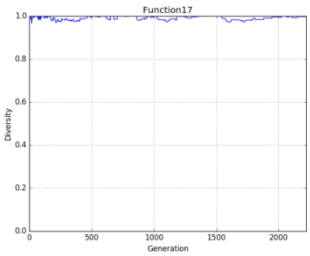
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.12 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C18 en 10 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

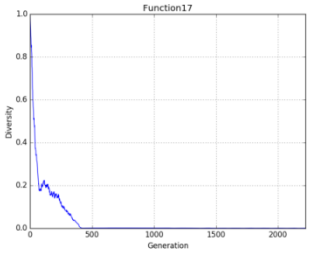
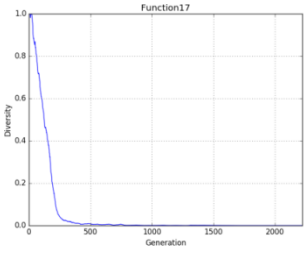
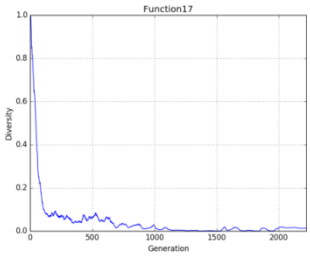
ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.13 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C1 en 30 dimensiones.

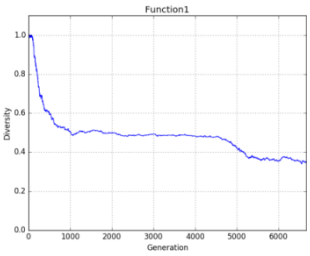
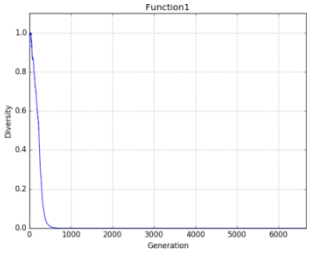
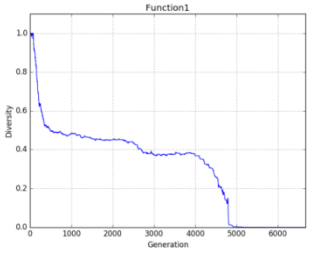
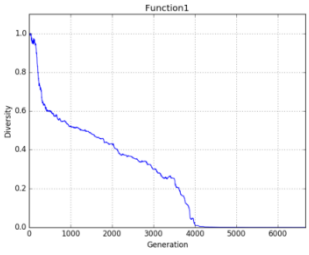
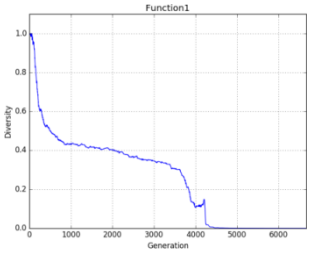
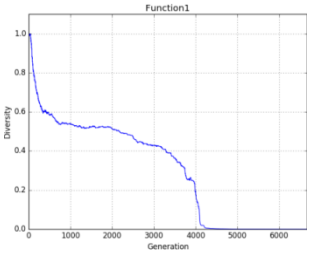
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.14 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C2 en 30 dimensiones.

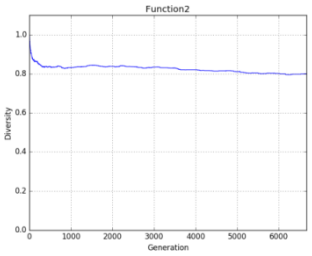
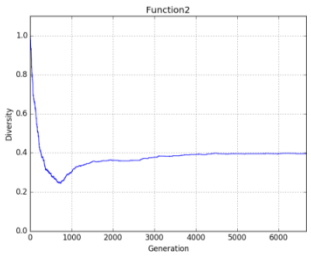
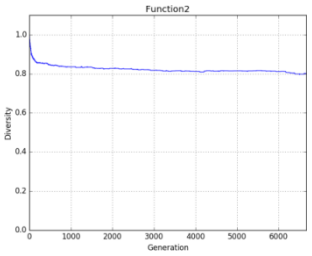
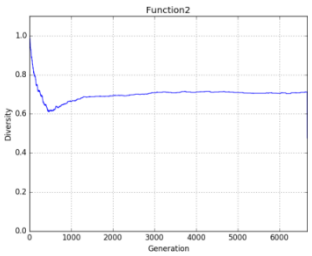
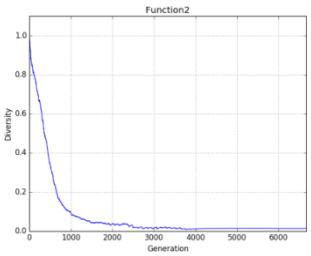
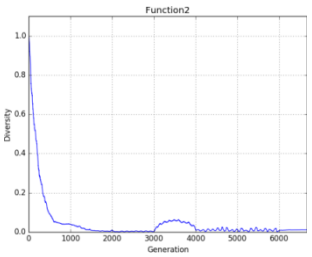
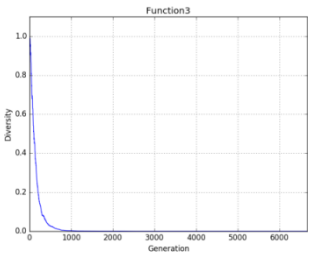
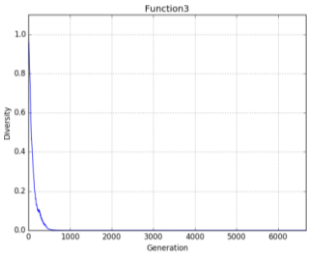
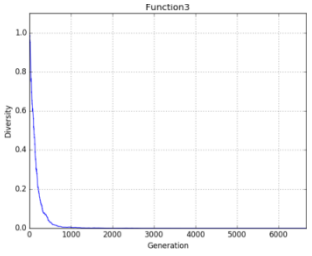
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.15 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C3 en 30 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

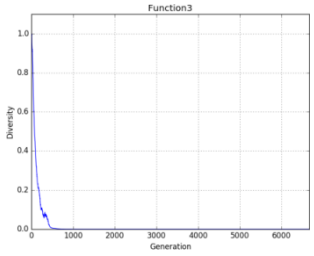
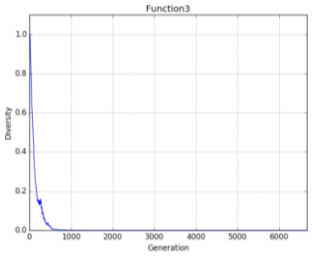
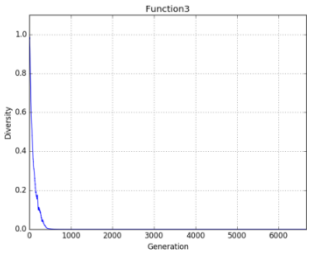
ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.16 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C5 en 30 dimensiones.

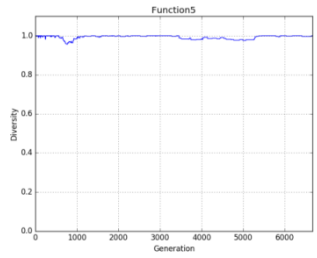
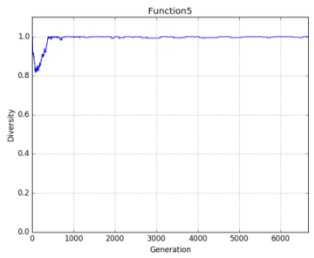
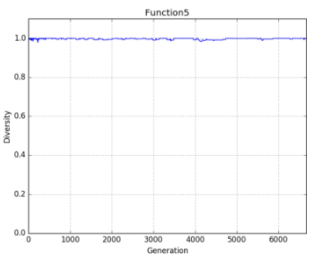
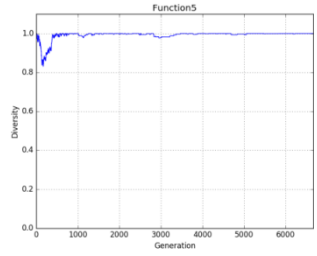
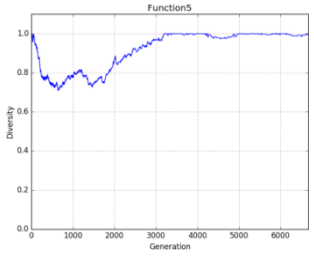
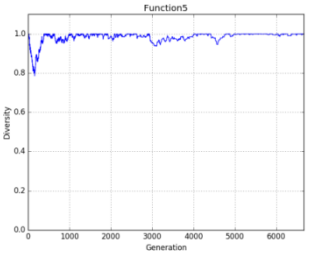
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.17 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C6 en 30 dimensiones.

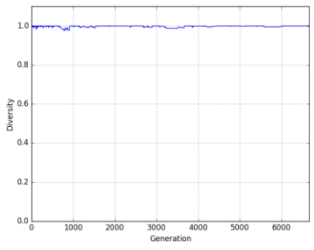
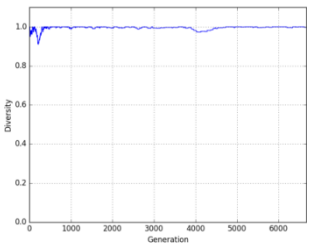
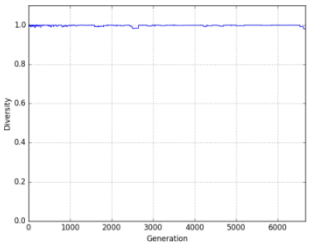
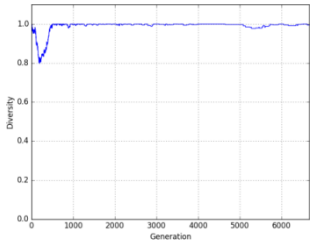
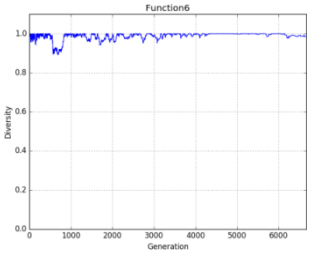
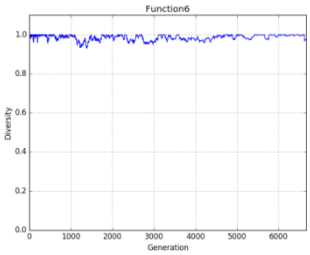
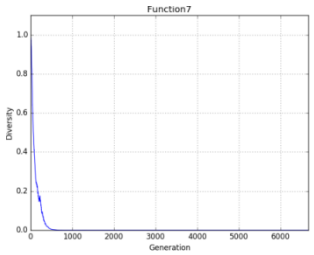
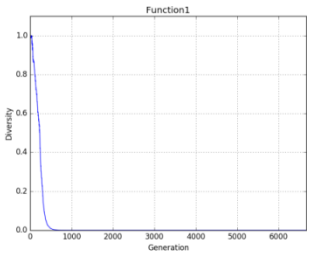
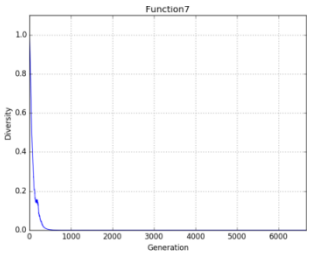
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.18 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C7 en 30 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

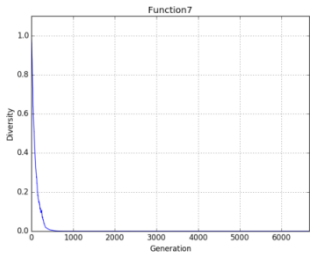
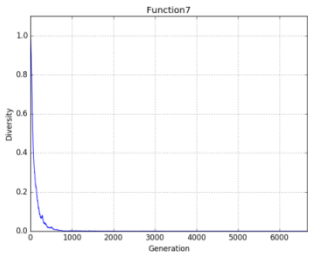
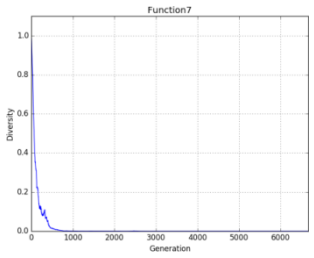
ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.19 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C8 en 30 dimensiones.

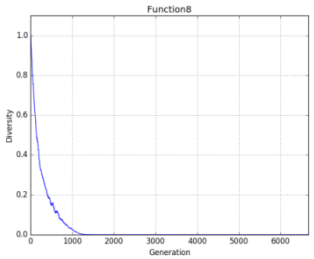
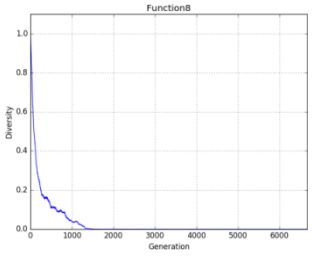
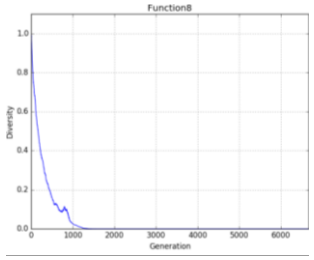
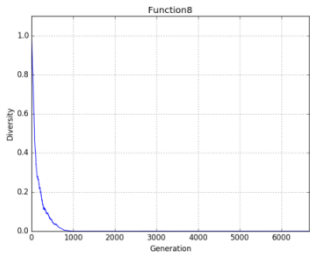
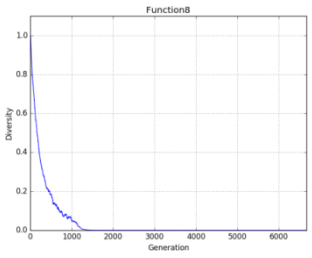
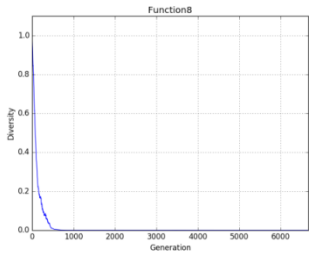
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.20 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C9 en 30 dimensiones.

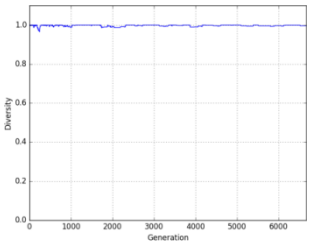
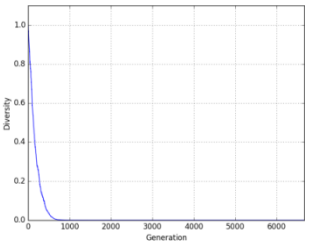
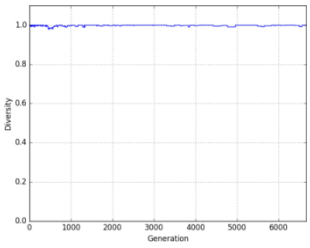
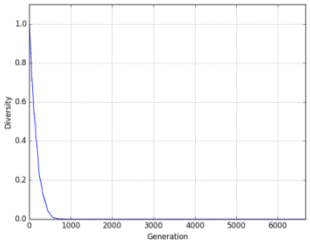
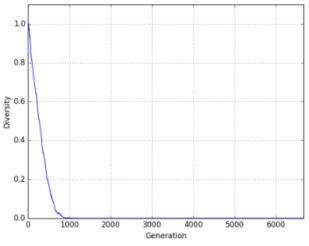
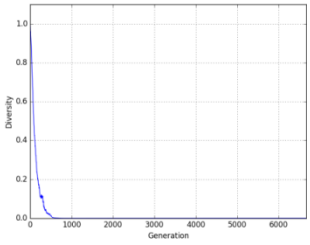
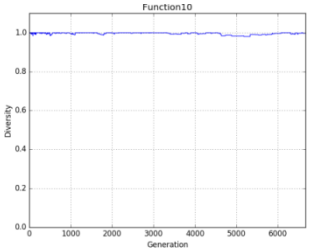
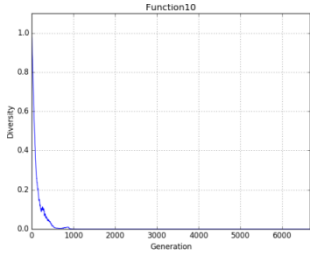
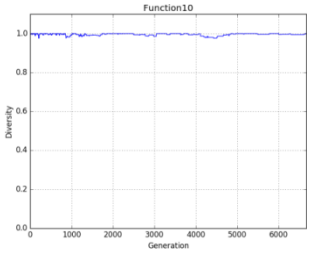
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.21 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C10 en 30 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

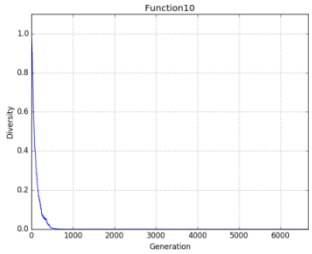
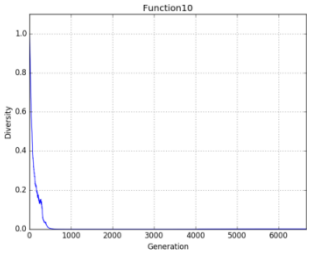
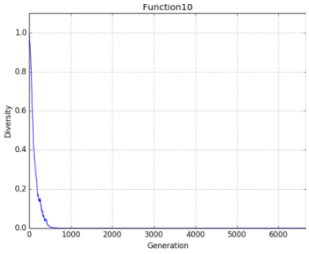
ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.22 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C11 en 30 dimensiones.

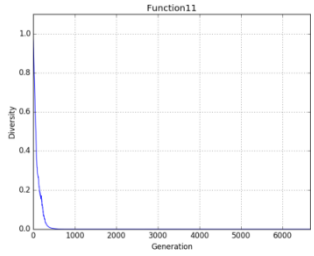
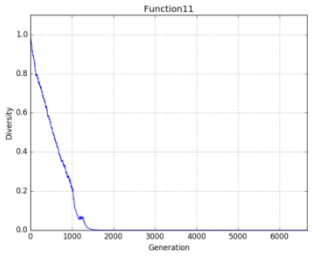
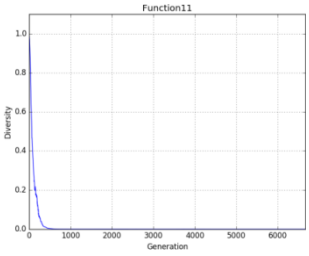
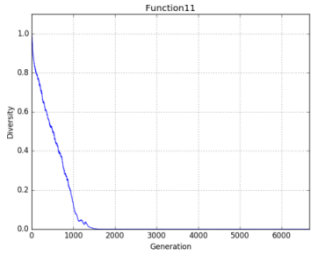
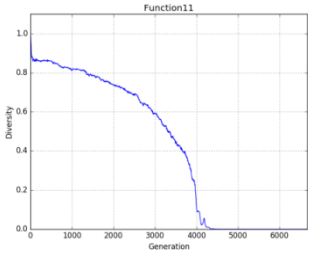
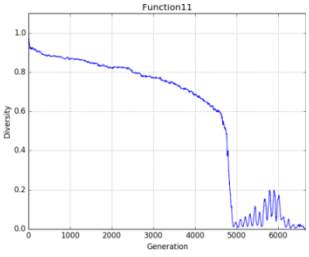
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.23 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C12 en 30 dimensiones.

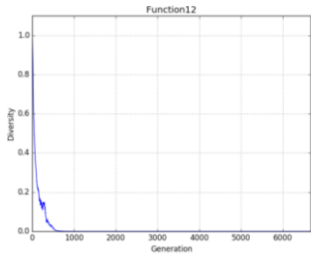
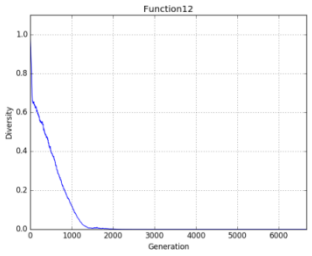
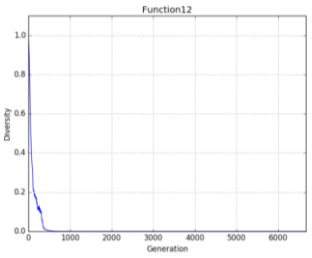
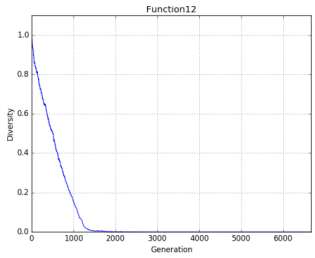
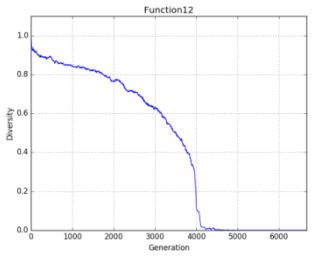
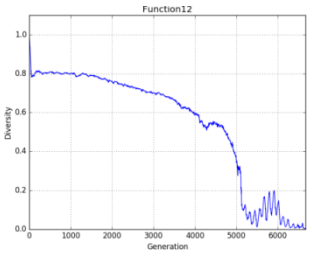
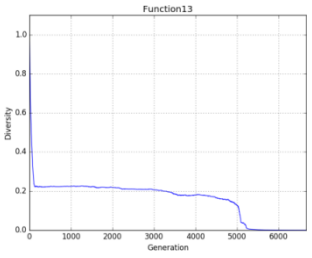
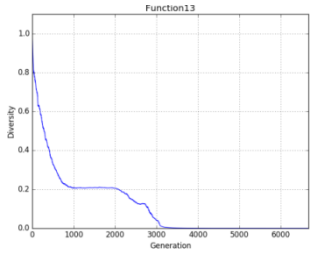
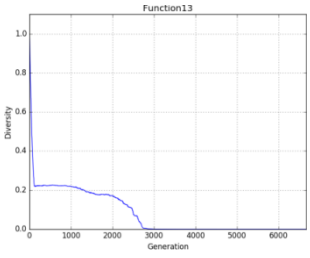
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.24 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C13 en 30 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

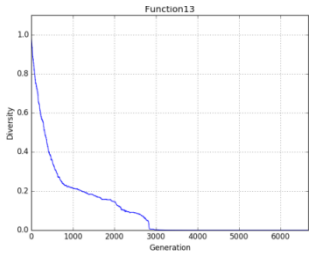
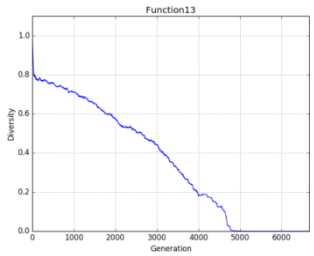
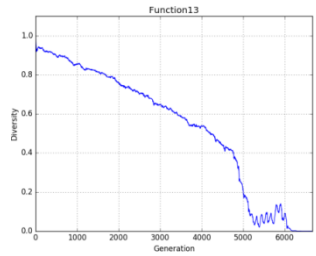
ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.6, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.25 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C15 en 30 dimensiones.

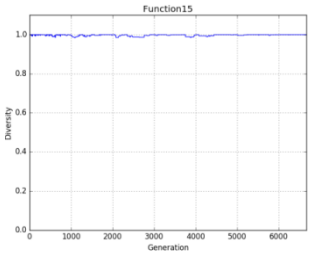
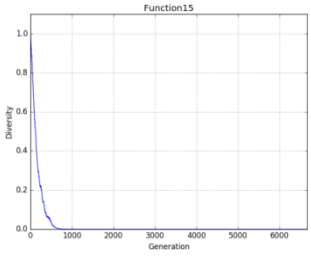
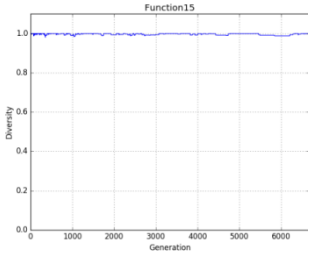
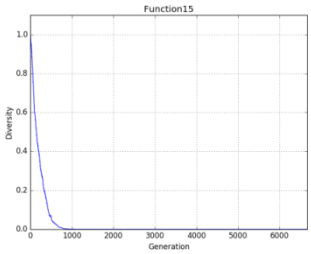
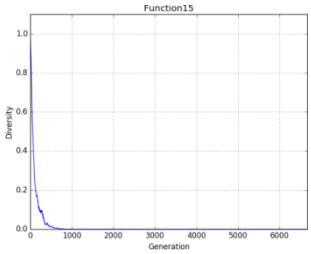
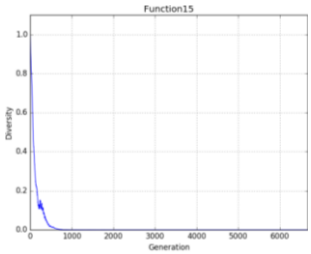
Rf-ED ran	ε-ED ran	Rf-ED-S ran-Conf1
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.26 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C16 en 30 dimensiones.

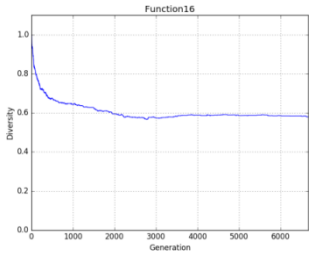
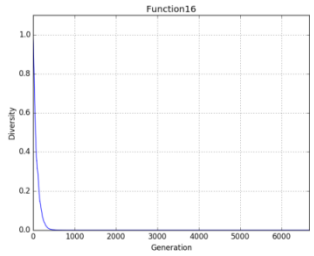
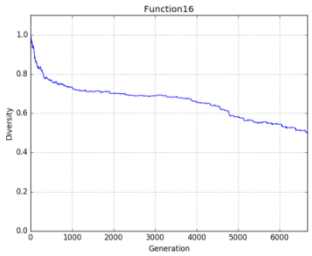
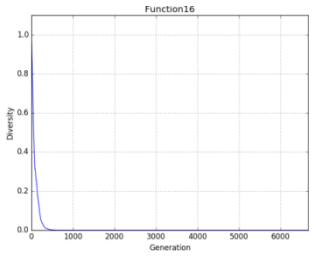
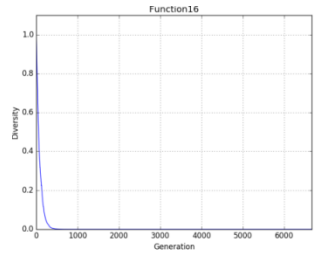
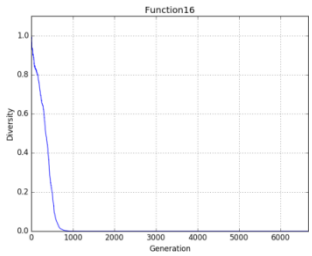
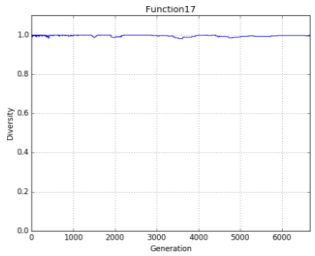
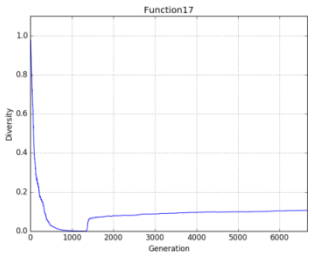
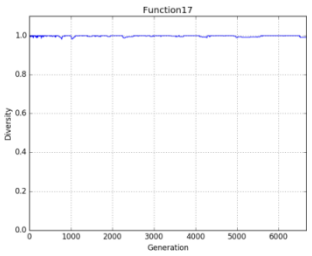
Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		
ε-ED-S ran-Conf1	**ε-ED-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-ED-S ran-Conf1 (Tc=0.9, cp=1)**
		

Tabla 5.27 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C17 en 30 dimensiones.

Rf-ED ran	**ε-ED ran**	**Rf-ED-S ran-Conf1**
		

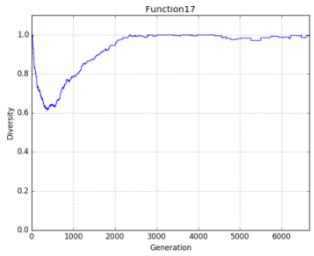
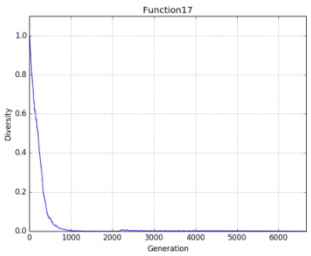
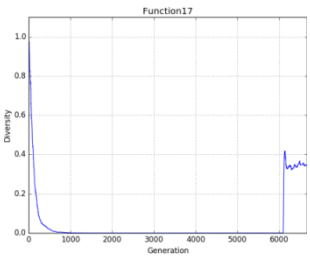
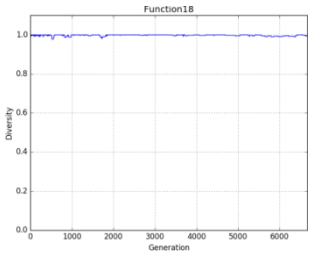
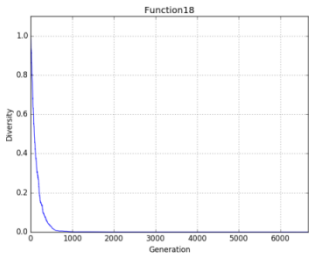
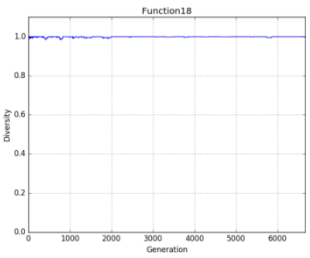
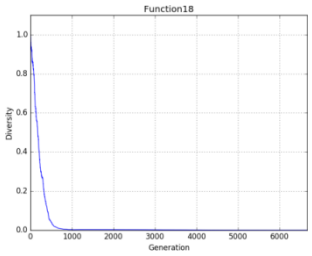
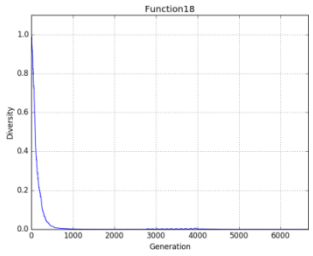
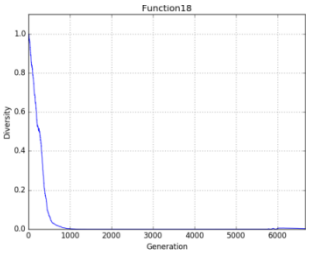
ε-DE-S ran-Conf1	ε-DE-S ran-Conf1 (Tc=0.6, cp=1)	ε-DE-S ran-Conf1 (Tc=0.9, cp=1)
		

Tabla 5.28 Comparativa entre los algoritmos: Rf-ED ran-Conf1, ε-ED ran-Conf1, Rf-ED-S ran-Conf1, ε-ED-S ran-Conf1, ε-ED-S ran-Conf1 (Tc=0.6, cp=1), ε-ED-S ran-Conf1 (Tc=0.9, cp=1), aplicados a la función C18 en 30 dimensiones.

ε-ED-S ran-Conf1	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)	ε-ED-S ran-Conf1 (Tc=0.9, cp=1)
		
ε-DE-S ran-Conf1	**ε-DE-S ran-Conf1 (Tc=0.6, cp=1)**	**ε-DE-S ran-Conf1 (Tc=0.9, cp=1)**
		

Referencias

[1] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-ofthe-Art," IEEE Transactions on Evolutionary Computation, vol. 15, no. 1,pp. 4–31, 2011.

[2] Mezura-Montes, E. Capistran-Gumersindo, E. "Cómo ajustar los parámetros de un algoritmo bio-inspirado", Revista Iztatl (en Impresión), 2015.

[3] A. P. Engelbrecht, "Computational intelligence: an Introduction", 2da edición, John Wiley & Sons, 2007.

[4] A. E. Eiben, Zbigniew Michalewicz, "Parameter Control in Evolutionary Algorithms," VU University Amsterdam, University of Adelaide, 2000.

[5] Tsung-CheChiang, Cheng-NanChen, and Yu-Chieh Lin, "Parameter control mechanisms in differential evolution: A tutorial review and taxonomy," IEEE Symposium Series on Computational Intelligence, 2013.

[6] Saber M. Elsayed, RuhulA. Sarker, Daryl L. Essam, "Adaptive Configuration of evolutionary algorithms for constrained optimization, School of Engineering and Information Technology," University of New South Wales at Canberra, Australia, 2013.

[7] Storn, R., Price, K., "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces", J. of Global Optimization, 1997.

[8] K. Deb, "An efficient constraint handling method for genetic algorithms," Computer Methods in Applied Mechanics and Engineering, vol. 186, no. 2-4, pp. 311 – 338, 2000.

[9] T.Takahama and S. Sakai, "Constrained optimization by the epsilon constrained differential evolution with an archive and gradient-based mutation." WCCI 2010 IEEE World Congress on Computational Intelligence July, 18-23, 2010 - CCIB, Barcelona, Spain, 2010.

[10] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan, "Differential Evolution with Ensemble of Constraint Handling Techniques for solving CEC 2010 Benchmark Problems," School of Electrical and

Electronics Engineering, Nanyang Technological University, Singapore, 2010.
[11] Huang, A. and P. N. Suganthan, "Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization," IEEE Congress on Evolutionary Computation, 2006.
[12] Ana Maria A. and Edite M, "Self-adaptive penalties in the electromagnetism like algorithm for constrained global optimization problems," Department of Production and Systems, School of Engineering, University of Minho, 4710-057 Braga, Portugal, 2009.
[13] Janez Brest, Viljem Zumer and Mirjam Sepesy Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization, IEEE Congress on Evolutionary Computation, 2006.
[14] Sana Ben Hamida and Marc Schoenauer, "An Adaptive Algorithm for Constrained Optimization Problems," Université Paris Ouest Nanterre La Défense and National Institute for Research in Computer Science and Control, 2007.
[15] Farmani R. and Wright J. "Self-adaptative formulation for constrained optimization," IEEE Transactions on Evolutionary Computation, 2005.
[16] Md Asafuddoula, Tapabrata Ray and Ruhul Sarker "An Improved Self-Adaptive Constraint Sequencing approach for constrained optimization problems," School of Engineering and Information Technology, University of New South Wales, Canberra, Australia, 2015.
[17] Saber Mohammed Elsayed and Ruhul Sarker, "Dynamic Configuration of Differential Evolution Control Parameters and Operators," Springer International Publishing Switzerland, 2016.
[18] Saber M. Elsayed, Ruhul A. Sarker and Daryl L. Essam Integrated Strategies Differential Evolution Algorithm with a Local Search for Constrained Optimization, University of New South Wales at Australian, 2011.
[19] Saber M. Elsayed, RuhulA. Sarker,Daryl L. Essam, "Self-adaptive differential evolution incorporating a heuristic mixing of operators, School of Engineering and Information Technology," University of New South Wales, Australia, 2012.
[20] Caponio, A., Cascella, G.L., Neri, F., Salvatore, N. and Sumner, M. "A fast adaptive memetic algorithm for online and offline control design of

PMSM drives," Trans. Syst. Man Cybern., Part B, Cybern.37 (1), 28–41, 2007.
[21] P. N. S. R. Mallipeddi. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore, April 2010.
[22] Kalyanmoy Deb, "Optimization for engineering design: algorithms and examples", Prenrices-Hall of India, 2000.
[23] Zbigniew Michalewicz, "A Survey of Constraint Handling Techniques in Evolutionary Computation Methods", Evolutionary Programming, 1995.
[24] Günter Rudolph: "Evolutionary Strategies", en G. Rozenberg et al. (eds.), Handbook of Natural Computing, 2012.
[25] Gary B. Fogel, "Evolutionary Programming", en G. Rozenberg et al. (eds.), Handbook of Natural Computing, 2012.
[26] David Fogel,"George Friedman - Evolving circuits for robots" [Historic Perspective] IEEE Computational Intelligence Magazine, 2007.
[27] John H. Holland,"Adaptation in Natural and Artificial Systems",The University of Michigan Press, 1975.
[28] J.R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural", Selection, 1992.
[29] Miranda-Varela M. Gomez-Ramon R. Mezura-Montes, E. "Diferential evolution in constrained numerical optimization: an empirical study," Information Sciences, 2010.
[30] Carlos A. 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, 2002.
[31] Amer-Draa, Samira-Bouzoubia, "A sinusoidal differential evolution algorithm for numerical optimization," Applied Soft Computing, 2015.
[32] Margarita Reyez-Sierra and Carlos A. Coello Coello, "Dynamic Fitness Inheritance proportion for Multi-objetive particle swarm optimization", Genetic and Evolutionary Computation Conference, 2006.

[33] Ryoji Tanabe and Alex Fukunaga, ¨Success-History Based Parameter Adaptation for Differential Evolution, IEEE Congress on Evolutionary Computation, 2013.

[34] Efrén Mezura-Montes, Omar Cetina-Dominguez, "Empirical Analysis of a Modified Artificial Bee Colony for Constrained Numerical Optimization", Laboratorio Nacional de Inform´atica Avanzada (LANIA A.C.), Xalapa, Veracruz, México, 2012.

[35] R. K. Ursem, "Diversity-guided evolutionary algorithms," in International Conference on Parallel Problem Solving from Nature. Springer, 2002.