Importancia de la Diversidad en el Diseño de Algoritmos Evolutivos

Carlos Segura[a,*], Joel Chacón Castillo[a]

[a] *Área de Ciencias de Computación, Centro de Investigación de Matemáticas (CIMAT), Callejón Jalisco s/n, Mineral de Valenciana, Guanajuato, Guanajuato 36240, México*

Abstract

La convergencia prematura es una de las mayores problemáticas que afectan al rendimiento de las metaheurísticas poblacionales por lo que a la hora de diseñar algoritmos evolutivos es un aspecto a tener en cuenta. En este capítulo se enumeran diferentes técnicas que se han propuesto a lo largo de las últimas décadas para lidiar con este problema. Una de las alternativas más exitosas consiste en examinar los efectos que los diferentes componentes del algoritmo evolutivo tienen sobre la diversidad mantenida en la población y con base en ello rediseñarlos para modificar su comportamiento de forma dinámica. Con el objetivo de ilustrar de una forma detallada este último grupo de técnicas, se discuten dos mecanismos que pertenecen a este grupo. El primero se aplica en el ámbito de evolución diferencial y consiste en una estrategia de reemplazo que combina una población élite con un mecanismo para mantener la diversidad de forma explícita. El segundo caso está enfocado a los operadores de cruza, donde concretamente se analiza y extiende el Operador de Cruza basado en Simulación Binaria (Simulated Binary Crossover - SBX). En las extensiones se considera el criterio de paro para modificar de forma dinámica el comportamiento del operador con el propósito de inducir un cambio gradual desde exploración hacia intensificación en el proceso de búsqueda. La validación experimental se realizó con algunos de los problemas de prueba más populares del ámbito mono-objetivo y multi-objetivo, alcanzándose mejoras significativas en ambos casos.

Keywords: Diversidad, Convergencia Prematura, Evolución Diferencial, Optimización Multi-objetivo

1. Introducción

Los Algoritmos Evolutivos (Evolutionary Algorithms — EAs) son considerados como uno de los enfoques con mayor eficacia para resolver distintas categorías de problemas de optimización. Se han desarrollado diversas variantes que han sido aplicadas en múltiples campos, como en transporte, economía o ingeniería. Particularmente, se han aplicado tanto en problemas del dominio continuo [32] como del dominio discreto [60]. En general, los EAs han sido especialmente exitosos en la resolución de problemas complejos en los que los enfoques exactos no son actualmente aplicables, como por ejemplo, en problemas NP-completos con espacios de búsqueda grandes [12]. Para el ámbito de este capítulo se hace uso de problemas continuos mono-objetivo y multi-objetivo con restricciones de caja que se pueden definir tal y como se indica en la Ecuación (1).

$$\begin{aligned} Minimizar \quad & \vec{F}(\vec{X}) \\ Sujeto \quad a \quad & \vec{X} \in \Omega \end{aligned} \tag{1}$$

donde $\vec{X}$ es un vector compuesto por D variables continuas de decisión $\vec{X} = [x_1, x_2, ..., x_D]$, cada variable pertenece al conjunto de los reales $x_i \in \Re$, D es la dimensión correspondiente al espacio de las variables de decisión, $\vec{F}$ es un vector compuesto por M funciones objetivo $\vec{F} = [f_1(\vec{X}), f_2(\vec{X}), ..., f_M(\vec{X})]$, Ω es el espacio

*Autor de contacto. Tel.: + 52 473 732 7155

Email addresses: carlos.segura@cimat.mx (Carlos Segura), joel.chacon@cimat.mx (Joel Chacón Castillo)

factible cuyo límite inferior es $x_i^{(L)}$ y límite superior es $x_i^{(U)}$, es decir $\Omega = \prod_{j=1}^{D}[x_j^{(L)}, x_j^{(U)}]$. Cada vector de variables es mapeado con M-funciones objetivo $\vec{F}(\vec{X})(\vec{F} : \Omega \subseteq \Re^D \rightarrow \Re^M)$ al espacio $\Re^M$ que es conocido como el espacio objetivo.

Actualmente, los EAs son una de las metaheurísticas más conocidas [32], pero a pesar de su éxito y de su uso tan extendido, adaptarlas a nuevos problemas implica la toma de varias decisiones de diseño complejas. Particularmente, a la hora de diseñar de forma apropiada un EA, se ha visto que es muy importante conseguir inducir un balanceo adecuado entre la exploración e intensificación del espacio de búsqueda [35]. Nótese en este punto que, de manera informal, la exploración del espacio de búsqueda consiste en evaluar regiones del espacio de búsqueda que no han sido muestreadas con el fin de detectar regiones promisorias, y la explotación consiste en muestrear en zonas ya evaluadas previamente para realizar una búsqueda más profunda con el fin de encontrar soluciones más refinadas y de mayor calidad. Cuando en los algoritmos evolutivos todas o casi todas las soluciones están en regiones distantes — alta diversidad — se produce habitualmente una búsqueda exploratoria, es decir, muchas de las nuevas soluciones evaluadas serán distantes a las ya evaluadas anteriormente. Sin embargo, cuando casi todas las soluciones están en una o en unas pocas regiones, se produce una búsqueda intensificadora. Uno de los problemas a la hora de diseñar los algoritmos evolutivos es que en muchos casos no se comprenden todas las implicaciones que los diferentes componentes tienen sobre el mantenimiento de la diversidad de la población y por tanto, sobre el balanceo entre exploración e intensificación [66]. Por ello, analizar el comportamiento y rediseñar en base a lo que está ocurriendo en este aspecto, es parte del proceso de diseño de los EAs.

Relacionado con lo anterior, aparece el concepto de convergencia prematura [66]. Se dice que un algoritmo converge de forma prematura cuando mucho antes de alcanzar el criterio de paro, todas las soluciones están en una zona muy pequeña del espacio de búsqueda. En este sentido, a partir de ese momento es difícil seguir mejorando las soluciones de forma significativa ya que con alta probabilidad sólo se va a realizar un muestreo de soluciones en dicha región. Por ello, es importante detectar si esto ocurre y en tal caso rediseñar algunos aspectos del EA para preservar una mayor diversidad. Sin embargo, si la población es muy diversa durante todo el proceso de búsqueda, se podría no alcanzar un grado adecuado de intensificación y por lo tanto se tendría una convergencia lenta que posiblemente también resultaría en soluciones de baja calidad. Por esta razón, Mahfoud [16] utilizó el concepto de diversidad útil para referirse a la cantidad de diversidad necesaria para generar soluciones de alta calidad.

En relación al diseño de EAs, se puede observar que en sus inicios la mayoría de enfoques fueron esquemas generacionales [18] en los que las soluciones hijas reemplazaban a la población anterior sin importar su respectiva aptitud o grado de diversidad. En estos esquemas iniciales se usaba la selección de padres para promover que el proceso de muestreo se realizara con mayor probabilidad en las regiones más promisorias encontradas. Por ello, con el propósito de alcanzar un balanceo adecuado entre exploración e intensificación, se desarrollaron muchas estrategias de selección de padres que permitían centrarse con mayor o menor velocidad en las regiones promisorias. Además, se desarrollaron alternativas que modifican otros aspectos como la estrategia de variación [36] y/o el modelo poblacional [2]. En la mayor parte de EAs más recientes, se introduce además una fase de reemplazamiento [27] por lo que la nueva población no tiene que formarse exclusivamente con los hijos, más bien se usan mecanismos para combinar la población anterior con la población hija y determinar así los nuevos sobrevivientes. En este contexto, se suele introducir elitismo en los algoritmos, es decir, el mejor individuo encontrado sobrevivirá a la siguiente generación [29]. De esta forma, ahora también se puede modificar esta última fase para conseguir el balanceo apropiado entre exploración e intensificación.

En los últimos años, algunos de los trabajos más exitosos en el área de evitación de convergencia prematura se han basado en considerar el criterio de parada y evaluaciones realizadas para balancear entre exploración e intensificación [62]. Esto se fundamenta en la premisa de que el grado entre exploración e intensificación debería variar a lo largo de la ejecución, por lo tanto tiene sentido que las decisiones tomadas por las componentes varíen en función del instante de ejecución. Particularmente, en este capítulo se describen dos aportaciones que pertenecen a este grupo de técnicas. La primera, que se aplica en el área de *Evolución Diferencial* (Differential Evolution — DE), recibe el nombre de DE *Mejorado con Mantenimiento de Diversidad* (DE with Enhanced Diversity Maintenance — DE-EDM) e integra una estrategia de reemplazo

que maneja la diversidad de forma explícita con una población élite. En la fase de reemplazo que incorpora el DE-EDM se promueve un balanceo dinámico entre exploración e intensificación, teniendo en cuenta para ello el criterio de paro y las evaluaciones transcurridas para la toma de decisiones. Concretamente, en las primeras etapas se induce un grado de exploración alto ya que los individuos sobrevivientes son diversificados y, posteriormente, conforme transcurren las generaciones se induce un mayor grado de intensificación. La segunda aportación está enfocada a la extensión de operadores de cruza. Particularmente se analizan los componentes que conforman al *Operador de Cruza basado en Simulación Binaria* (Simulated Binary Crossover - SBX), y se propone una variante dinámica (Dynamic Simulated Binary Crossover — DSBX) en la que el comportamiento interno es alterado en base al criterio de paro y a las evaluaciones realizadas hasta el momento.

El resto de este capítulo está organizado de la siguiente forma. En primer lugar, en la sección 2 se revisan algunas de las técnicas más populares que se han propuesto para promover el mantenimiento de diversidad, exponiendo una de las clasificaciones más extendidas. Posteriormente, en la sección 3 se describe y valida experimentalmente la aportación basada en evolución diferencial donde se considera la diversidad de forma explícita en la fase de reemplazamiento. Siguiendo esta misma línea en la sección 4 se lleva a cabo un análisis del operador de cruza SBX mencionando algunas de sus implicaciones en la diversidad, y se expone en base a ello una variante dinámica que se valida experimentalmente con problemas multi-objetivo. Finalmente, en la sección 5 se exponen las conclusiones y trabajos futuros trazados en base a los resultados obtenidos.

2. Preservación de diversidad en algoritmos evolutivos

La convergencia prematura es una problemática muy conocida en el ámbito de los EAs por lo que se han desarrollado gran cantidad de técnicas para lidiar con la misma [55]. Estas técnicas modifican de manera directa o indirecta la cantidad de diversidad mantenida por el algoritmo [67] y varían desde técnicas generales hasta mecanismos dependientes de un problema dado. En este apartado se revisan algunas de las técnicas generales más populares. Inicialmente, se describen algunas maneras de clasificar a este tipo de estrategias, para posteriormente describir mecanismos clásicos específicos, así como algunas estrategias más novedosas que se basan en modificar la estrategia de reemplazamiento. Finalmente, dado que en este capítulo se extiende DE, se revisan también los trabajos de esta área que tienen una relación estrecha con el manejo de diversidad. Se recomienda a los lectores que requieran conocer estos mecanismos con más detalle consultar [67] así como los artículos específicos en que cada método es propuesto.

2.1. Clasificaciones de mecanismos para promover la diversidad

Debido a la gran cantidad de métodos desarrollados en esta área, se han propuesto varias clasificaciones de los mismos. Liu et al. [43] propuso diferenciar entre los enfoques uni-proceso y multi-proceso. En el enfoque uni-proceso se modifica la preservación de la diversidad actuando sobre un único componente del EA. Es importante destacar que en los enfoques uni-proceso no se excluye el uso de otros componentes en el proceso de exploración y/o intensificación, sino que más bien, si no se consigue el balanceo adecuado, sólo se modifica una componente hasta conseguir el comportamiento deseado. Por otra parte, en los enfoques multi-proceso se tiene en cuenta las implicaciones que varios componentes provocan sobre el balanceo, y se actúa modificando o rediseñando varios de ellos hasta conseguir el comportamiento adecuado. Los esquemas uni-proceso son mucho más habituales actualmente [66], y en particular las dos propuestas incluidas en este capítulo son mecanismos uni-proceso.

Extendiendo a lo anterior, se propuso una clasificación más específica [66], en la que se tiene en cuenta cuál es la componente que se cambia para categorizar a cada método. En este sentido, los más populares son los siguientes:

- **Enfoques basados en la selección**: son los más clásicos y se basan en cambiar la presión de selección que se produce hacia las zonas promisorias a la hora de realizar la selección de padres.
- **Enfoques basados en población**: se modifica el modelo poblacional utilizando algunas técnicas como variar el tamaño de la población de forma dinámica, eliminar individuos duplicados, utilizar técnicas de infusión o establecer un estrategia de islas con migraciones.

- **Enfoques basados en ela cruza y/o la mutación**: se basan en rediseñar los operadores de cruza y/o mutación, considerando en algunos casos información específica del problema. También se incluyen en este grupo opciones más generales como aplicar restricciones sobre el emparejamiento y/o incluir operadores disruptivos que podrían ser utilizados sólo en ciertos instantes del proceso de optimización.

2.2. Esquemas clásicos para administrar la diversidad

Los primeros EAs se basaron principalmente en esquemas generacionales que no incluían fase de reemplazo. En estos esquemas, la selección de padres era la principal responsable de que se muestrearan con mayor probabilidad las zonas más promisorias encontradas hasta el momento, y por tanto muchos de los primeros esquemas que trataron de evitar la convergencia prematura se basaron en modificar el proceso de selección de padres. Así, en los 90s se desarrollaron varios esquemas que alteraban la presión de selección [27] de forma estática o dinámica. Sin embargo, con base en varios estudios teóricos y experimentales se observó que, generalmente, actuar exclusivamente sobre el operador de selección no es suficiente, especialmente cuando se quieren realizar ejecuciones a largo plazo, ya que se requerirían poblaciones excesivamente grandes para mantener un grado adecuado de diversidad.

Otra alternativa fue modificar los modelos poblacionales, encontrando en este grupo los esquemas basados en islas [2], los celulares o, más recientemente, los basados en *agrupaciones o clústeres* [31]. La idea de introducir restricciones en el emparejamiento, principalmente con base en la ubicación de los individuos en el espacio de búsqueda también ha sido bastante exitosa aunque controlar los mismos para obtener el balanceo apropiado ante diferentes criterios de parada es bastante complejo. En algunos casos resultó ser más prometedor promover el emparejamiento entre individuos no similares [29], mientras que en otros escenarios se hace exactamente lo opuesto [19]. Otro problema común de muchas de las estrategias anteriores es que suelen introducir parámetros adicionales, por lo que el proceso de ajuste de parámetros, que ya de por sí es un problema importante en los EAs, se vuelve aún más complejo. Es importante resaltar que todas estas estrategias clásicas no evitan por completo la convergencia sino que la idea es disponer de mecanismos para acelerarla o retrasarla.

Otra alternativa diferente ha sido adaptar la fase de variación. En este sentido se han desarrollado diversas técnicas para controlar los parámetros que se consideran en la variación con el propósito de adaptar el balanceo entre exploración e intensificación. En algunos casos esto se consigue usando distintos valores en los parámetros para distintas etapas a lo largo del proceso de optimización [71], mientras que en otros casos se hacen cambios más drásticos y se consideran varios operadores con distintas propiedades [44]. También existen mecanismos adaptativos que usan una memoria para almacenar información histórica sobre los efectos de la variación y con base en ello ir modificándola [72]. Cabe destacar que en la mayor parte de estos esquemas no se considera la diversidad de forma directa, sino que sólo se considera para analizar el comportamiento y con base en ello se procede a un rediseño.

Finalmente, un esquema muy sencillo pero no por ello menos importante es el basado en reinicios. En estos esquemas, en lugar de evitar la convergencia acelerada, se aplica un reinicio total o parcial de la población cada cierto número de generaciones o cuando se detecta que la población ha convergido. Con base en esto se han propuesto diversas estrategias para establecer los puntos de reinicio [40]. Estos esquemas se implementan de forma muy sencilla y en algunos casos han proporcionado mejoras significativas [41] por lo que es un método a tener en cuenta, al menos como alternativa inicial. Es común combinar las estrategias basadas en reinicio con algunas de las técnicas anteriores, ya que dichas técnicas están basadas en mantener la diversidad, mientras que en esta última el objetivo es recuperar la diversidad.

2.3. Esquemas de reemplazamiento basados en diversidad

Recientemente se han propuesto diversos mecanismos que modifican la fase de reemplazo para preservar la diversidad. La idea principal de estos esquemas es inducir un grado de exploración adecuado diversificando a los individuos sobrevivientes, de forma que los operadores de reproducción puedan generar nuevas soluciones en diferentes regiones en las siguientes generaciones. Estos métodos están basados en el principio de que los operadores de cruza tienen un efecto de exploración al considerar individuos distantes y de intensificación al considerar individuos próximos [26].

El esquema de pre-selección propuesto por Cavicchio [33] es uno de los primeros estudios que utilizan la fase de reemplazamiento para controlar la diversidad. El esquema inicial de Cavicchio se extendió para generar el esquema denominado *amontonamiento o crowding* [17], el cual ha sido muy popular en los últimos años [46, 47]. El principio del crowding se basa en que los nuevos individuos que entren en la población sustituyan a individuos similares de generaciones anteriores, y con base en este principio, se han formulado diversas implementaciones.

En esta misma línea, se han propuesto otras estrategias de reemplazo con el propósito de promover la diversidad. Uno de los procedimientos más populares es la *Estrategia de Limpieza* (Clearing Strategy - CLR) [45]. En el procedimiento CLR se agrupan a los individuos en grupos denominados nichos, y los mejores individuos de cada nicho son preservados e incluidos en la población de la siguiente generación. Un inconveniente de este procedimiento es que los casos en que se detectan muchos nichos provocan una fuerte inmovilización de la población. Por ello, Petrowski [56] propuso una variante para únicamente seleccionar a individuos cuya aptitud sea mejor que la media de la población.

Otros métodos de este grupo consideran funciones de aptitud que combinan la función objetivo original con la diversidad. Sin embargo, es complejo construir una función compuesta ya que las dos mediciones podrían no ser directamente compatibles, y por lo tanto, las funciones adecuadas suelen depender de cada problema. Una forma de suavizar este inconveniente fue propuesto en el algoritmo de combinación (COMB) [68] donde los individuos son ordenados y categorizados con base en su aptitud y contribución a la diversidad, y la función compuesta se diseña con base en el orden y no con base en los valores de función objetivo y contribución a diversidad. La principal desventaja del COMB es que requiere dos parámetros de usuario, aunque independientemente de esto, se ha usado con bastante éxito. Otra alternativa es el procedimiento *Reemplazamiento Basado en Contribución a la Diversidad y Sustitución del Peor* (Contribution of Diversity/Replace Worst - CD/RW) [45]. En el método CD/RW un nuevo individuo reemplaza a un miembro de la población cuyo rendimiento sea peor tanto en aptitud como en contribución a la diversidad. En caso de no encontrar un peor individuo bajo estos dos criterios, se procede a reemplazar al peor individuo en la población considerando únicamente a la aptitud. Una última alternativa se basa en considerar a la contribución a la diversidad como un objetivo adicional y aplicar un esquema de optimización multi-objetivo [11] [52]. Estos enfoques son identificados como algoritmos multi-objetivo basados en diversidad. Existen varias estrategias para calcular el objetivo auxiliar [61]. Uno de los enfoques más populares consiste en calcular la contribución a la diversidad de cada individuo con base en la *Distancia al Vecino más Cercano* (Distance to the Closest Neighbor - DCN) [62] de entre los individuos que ya hayan sido seleccionados como supervivientes.

2.4. Diversidad en evolución diferencial

Los algoritmos basados en DE son altamente susceptibles a la pérdida de diversidad debido a que se basan en una estrategia de selección muy elitista. Debido a ello, se han desarrollado varios análisis para lidiar con este problema. Dado que en el área se conoce, al menos de manera general, las implicaciones que tiene cada parámetro sobre la diversidad, algunos autores han trabajado en estimar de forma teórica cuáles deben ser los parámetros adecuados para que se produzca cierto tipo de comportamiento [73]. Otros autores han estudiado el efecto que tiene la norma de los vectores de diferencia sobre la mutación [49] y con base en ello se han propuesto mecanismos que prohíben ciertos movimientos que pueden resultar perjudiciales para la diversidad [51]. En este último estudio, el tipo de movimientos aceptados varía a lo largo de la ejecución, descartando a los movimientos cuyos desplazamientos sean menores a un umbral el cual es decrementado conforme transcurren las generaciones. Además, se han propuesto otras formas para establecer los movimientos aceptados [8].

Una alternativa distinta se basa en alterar el operador de selección [59]. Específicamente, con el propósito de mantener mayor diversidad en la población se altera la presión de selección utilizando una selección probabilística que permite escapar en algunos casos de las bases de atracción de óptimos locales. Sin embargo, este método no es demasiado robusto debido a que considera la aptitud para definir las probabilidades para seleccionar a un individuo por lo que ciertas transformaciones de la función pueden modificar de forma drástica el tipo de búsqueda que se realiza.

Finalmente, la variante DE con *Diversidad de la Población Auto-Mejorado* (*Auto-Enhanced Population Diversity* - AEPD) mide la diversidad de forma explícita y cuando se detecta la existencia de un nivel bajo

de diversidad en la población, se lanza un mecanismo de diversificación [70]. Esta propuesta se ha extendido para considerar diferentes esquemas de perturbación [76].

Es interesante hacer notar que las variantes de DE que alcanzaron los primeros lugares en varias competencias de optimización durante los últimos años no consideran estas modificaciones, y además estas variantes no han sido incorporadas en las herramientas de optimización más populares. Esto puede deberse a que muchos de los concursos están orientados a obtener resultados en un número de evaluaciones bastante limitado en lugar de a largo plazo, que es el ámbito en el que más beneficios suelen dar los mecanismos de control de diversidad.

3. Diseño de evolución diferencial basado en diversidad

3.1. Evolución diferencial: Conceptos básicos

Esta sección está dedicada a revisar la variante clásica de DE y a introducir varios términos importantes que son utilizados en el campo de DE. El esquema clásico de DE es identificado como DE/rand/1/bin y ha sido ampliamente usado como base para el desarrollo de variantes más complejas [15]. De hecho, la propuesta que se presenta en este capítulo como ejemplo extiende al DE/rand/1/bin. DE fue propuesto como un método de búsqueda directa para optimización continua mono-objetivo y es el ámbito en que se usará en este capítulo, es decir, se considera la Ecuación (1), fijando $M = 1$.

DE es un algoritmo estocástico basado en población, por lo que en cada instante maneja un conjunto de soluciones candidatas que van evolucionando de forma iterativa. En DE dichas soluciones candidatas son usualmente conocidas como vectores. En la variante básica de DE, para cada miembro de la población (conocidos como *vectores objetivo*) se genera un nuevo vector que es conocido como el *vector mutado*. A continuación, el vector mutado se combina con el vector objetivo para generar al *vector de prueba* y finalmente se procede con la fase de selección para elegir a los vectores sobrevivientes. De esta forma, las generaciones transcurren de forma iterativa hasta cumplir el criterio de paro. En esta capítulo, el i-ésimo vector de la población en la generación G se denota como$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, ..., x_{D,i,G}]$. A continuación se explica en más detalle cada componente de DE.

3.1.1. Inicialización

Usualmente, DE inicia el proceso de optimización con una población de NP vectores que son creados de forma aleatoria. Habitualmente los vectores de la población inicial son generados con base en una distribución uniforme, ya que usualmente no se posee información sobre cuáles son las zonas más promisorias del espacio de búsqueda. Por lo tanto, el j-ésimo componente del i-ésimo vector es inicializado de la forma $x_{j,i,0} = x_j^{(L)} + rand_{i,j}[0,1](x_j^{(U)} - x_j^{(L)})$, donde $rand_{i,j}[0,1]$ es un número aleatorio uniformemente distribuido entre 0 y 1.

3.1.2. Operador de mutación

Por cada vector objetivo se genera un vector mutado para lo cual se han propuesto múltiples estrategias con la particularidad de que en cierta forma se usen las diferencias entre vectores. La variante clásica de DE aplica la estrategia conocida como rand/1, en la cual se crea un vector mutado $V_{i,G}$ de la siguiente forma:

$$\vec{V}_{i,G} = \vec{X}_{r1,G} + F \times (\vec{X}_{r2,G} - \vec{X}_{r3,G}) \quad r1 \neq r2 \neq r3 \tag{2}$$

En la Ecuación (2) los índices $r1, r2, r3 \in [1, NP]$ deben ser enteros distintos y son generados de forma aleatoria en el rango $[1, NP]$. Además, estos índices son distintos al índice i. Es importante hacer notar que la diferencia entre los vectores es escalada por medio del parámetro F, el cual usualmente se define en el intervalo $[0.4, 1]$. Posteriormente, el vector de diferencia (escalado) es agregado a un tercer vector, lo que significa que los vectores mutados son similares a los vectores objetivo si el grado de diversidad es bajo, ya que los vectores de diferencias tienen norma pequeña. Como consecuencia de esto es claro que es crítico mantener un grado mínimo de diversidad en DE.

3.1.3. Operador de cruza

El operador de cruza se aplica con el objetivo de combinar la información de distintas soluciones candidatas y de incrementar la diversidad de los vectores. Específicamente, cada vector objetivo $\vec{X}_{i,G}$ se mezcla con su correspondiente vector mutado $\vec{V_{i,G}}$ para generar un vector de prueba $\vec{U_{i,G}} = [u_{1,i,G}, u_{2,i,G}, ..., u_{D,i,G}]$. La estrategia de cruza más típica es conocida como *cruza binomial* y actúa de la siguiente forma:

$$\vec{U}_{j,i,G} = \begin{cases} \vec{V}_{j,i,G}, & \text{si} \quad (rand_{i,j}[0,1] \leq CR \quad o \quad j = j_{rand}) \\ \vec{X}_{j,i,G}, & \text{de otra forma} \end{cases} \tag{3}$$

En la Ecuación (3) $rand_{i,j}[0,1]$ es un número uniformemente distribuido, j_{rand} es un índice seleccionado aleatoriamente que asegura que $\vec{U}_{i,G}$ genera al menos un componente de $\vec{V}_{i,G}$ y $CR \in [0,1]$ es la proporción de cruza.

3.1.4. Operador de selección

Finalmente, se aplica una selección elitista para determinar los vectores sobrevivientes que participarán en la siguiente generación. Específicamente, cada vector de prueba se compara con su correspondiente vector objetivo y sobrevive el que tiene la mejor aptitud:

$$\vec{X}_{j,i,G+1} = \begin{cases} \vec{U}_{i,G}, & \text{si} \quad f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G}) \\ \vec{X}_{i,G}, & \text{de otra forma} \end{cases} \tag{4}$$

Debido a la forma de seleccionar a los sobrevivientes en cada generación los miembros de la población permanecen iguales o mejoran. Por ello, se considera que DE hace uso de una selección muy elitista y es una de las razones por la que en ciertos casos puede sufrir de una convergencia demasiado acelerada.

3.2. Propuesta basada en diversidad

El método que se presenta en esta sección está motivado principalmente por dos trabajos. El primero de ellos es un estudio empírico desarrollado por Montgomery et al. [51] que confirma que DE sufre de convergencia prematura en varios problemas. El segundo es un trabajo propuesto por Segura et al. [62] que proporciona mejoras significativas en el campo de optimización combinatoria utilizando para ello una fase de reemplazo conocida como *Reemplazo con Control de Diversidad Dinámica Basada en Varios Objetivos* (Replacement with Multi-objective based Dynamic Diversity Control - RMDDC). Particularmente, el RMDDC controla el grado de diversidad, relacionándolo con el criterio de paro y generaciones transcurridas. Con base en las conclusiones de cada uno de los trabajos, en esta sección se ilustra una propuesta que es una variante novedosa de DE que incluye un mecanismo de reemplazo que sigue los mismos principios que guiaron el diseño de RMDDC. Este nuevo algoritmo recibe el nombre de *Evolución Diferencial con Mantenimiento Mejorado de Diversidad* (Differential Evolution with Enhanced Diversity Maintenance - DE-EDM) y su código fuente está disponible [1].

DE-EDM (ver Algoritmo 1) es bastante similar a la versión clásica de DE; de hecho, la forma en que se crean los vectores de prueba (líneas 5 y 6) se mantiene intacta. La novedad del DE-EDM es que incorpora una población élite (E) y una estrategia de reemplazo basada en diversidad. Específicamente, con el propósito de seleccionar a los miembros de la población élite se considera el operador de selección elitista de la versión clásica de DE (línea 7). Posteriormente, se aplica la estrategia de reemplazo (línea 8) para determinar a los sobrevivientes. Siguiendo la misma filosofía que en el RMDDC, los individuos que contribuyen muy poco a la diversidad no deberían ser aceptados como miembros de la siguiente generación. Para conseguir este propósito se considera tanto el criterio de paro como las generaciones transcurridas en la selección. Así, se establecerá un grado de diversidad mínimo deseado dependiente del tiempo de ejecución.

La estrategia de reemplazo (ver Algoritmo 2) funciona de la siguiente forma. Inicialmente, recibe a la población padre (vectores objetivo), a la población de hijos (vectores de prueba) y a los vectores élite. De

[1] El código en C++ puede ser descargado en la siguiente dirección `https://github.com/joelchaconcastillo/Diversity_DE_Research.git`

Algoritmo 1 Esquema general del DE-EDM

1: Inicializar de forma aleatoria a la población con NP individuos, donde cada uno es distribuido de forma uniforme.
2: $G = 0$
3: **while** El criterio de paro no sea alcanzado **do**
4: **for** $i = 1$ to NP **do**
5: Mutación: Generar al vector mutado ($V_{i,G}$) de acuerdo a la Ecuación (2).
6: Cruza: Utilizar la recombinación para generar al vector de prueba ($U_{i,G}$) de acuerdo a la Ecuación (3).
7: Selección: Actualizar al vector élite ($E_{i,G}$ en lugar de $X_{i,G}$) de acuerdo a la Ecuación (4).
8: Reemplazo: Seleccionar a los vectores objetivo (X_{G+1}) de acuerdo al Algoritmo 2.
9: $G = G + 1$

Algoritmo 2 Fase de Reemplazo

1: Entrada: *Población* (*Vectores Objetivo*), *Hijos* (*Vectores de prueba*), y *Elite*
2: Actualizar $D_t = D_I - D_I * (nfes/(0.95 * max\_nfes))$
3: *Candidatos* = *Población* ∪ *Hijos* ∪ *Elite*.
4: *Sobrevivientes* = *Penalizados* = ∅.
5: **while** $|Sobrevivientes| < NP$ y $|Candidatos| > 0$ **do**
6: *Seleccionados* = Seleccionar al mejor individuo de *Candidatos*.
7: Eliminar *Seleccionado* de *Candidatos*.
8: Copiar *Seleccionado* a *Sobrevivientes*.
9: Encontrar a los individuos de *Candidatos* cuya distancia a *Seleccionados* sea menor que D_t y moverlos a *Penalizados*. En esta parte se considera la distancia normalizada (Ecuación 5).
10: **while** $|Sobrevivientes| < NP$ **do**
11: *Seleccionado* = Seleccionar al individuo de *Penalizados* con la mayor distancia al individuo más cercano a *Sobrevivientes*.
12: Eliminar *Seleccionado* de *Penalizados*.
13: Copiar *Seleccionado* a *Sobrevivientes*
14: **return** *Sobrevivientes*

entre todos ellos, debe seleccionar a NP vectores para formar la siguiente población de padres. En primer lugar, con base en el número de evaluaciones de función transcurridas y el criterio de parada, se calcula una distancia mínima deseada (D_t) para mantener en la población (línea 2). A continuación, se juntan las tres poblaciones en un conjunto de miembros candidatos (línea 3) que contiene en cada momento a los vectores candidatos que podrían ser seleccionados para sobrevivir. Posteriormente, se inicializa tanto el conjunto de individuos sobrevivientes como los penalizados con el conjunto vacío (línea 4). Para seleccionar a los NP sobrevivientes se repite el proceso iterativo (líneas 5 - 13) que se describe a continuación. En cada iteración se comienza seleccionando como sobreviviente al mejor individuo del *Conjunto de Candidatos*, es decir al individuo que tiene la mejor aptitud. Este individuo se mueve al *Conjunto de Sobrevivientes* y los individuos del *Conjunto de Candidatos* cuya distancia al individuo seleccionado sea menor que D_t se transfieren al *Conjunto de Penalizados* (línea 9). La forma de calcular la distancia entre dos individuos es con base en la distancia Euclidiana normalizada descrita en la Ecuación (5), donde D es la dimensión del problema, y $x_d^{(L)}, x_d^{(U)}$ son los límites menores y mayores de la dimensión d. En los casos donde el *conjunto de Candidatos* queda vacío antes de seleccionar a NP individuos, el *Conjunto de Sobrevivientes* se llena seleccionando en cada iteración al individuo *Penalizado* con la mayor distancia al individuo más cercano del *Conjunto de Sobrevivientes* (líneas 10 - 13).

$$distancia(x_i, x_j) = \frac{\sqrt{\sum_{d=1}^{D} \left(\frac{x_i^d - x_j^d}{x_d^{(U)} - x_d^{(L)}} \right)^2}}{\sqrt{D}} \tag{5}$$

Con el propósito de completar la descripción de DE-EDM se especifica la forma en que se calcula D_t y el procedimiento con el cual se actualizan a los individuos élite. El resto del algoritmo se mantiene igual que la variante clásica de DE. El valor de D_t se utiliza para alterar el grado entre exploración e intensificación, por lo tanto el valor adecuado para este parámetro depende del instante de ejecución. Específicamente, este valor debería ser reducido conforme se alcanza el criterio de paro con el objetivo de promover mayor intensificación en las últimas etapas de optimización. En nuestro esquema se requiere asignar un valor inicial para D_t (D_I), y a continuación de manera similar a como se hace en [62], se realiza una reducción lineal de D_t considerando las evaluaciones transcurridas y el criterio de paro. Particularmente, en este trabajo, el criterio de paro se asigna con base en las evaluaciones a función y la reducción se calcula de tal forma que al

95 % del número máximo de evaluaciones el valor de diversidad mínimo requerido es cero, lo que quiere decir que en la última fase la diversidad no es considerada para nada. Entonces, denotando max_nfes al número máximo de evaluaciones y $nfes$ al número de evaluaciones trascurridas, D_t es calculado de la siguiente forma $D_t = D_I - D_I * (nfes/(0.95 * max\_nfes))$.

La distancia inicial (D_I) afecta de forma considerable al rendimiento de DE-EDM. Si este parámetro es elevado, el algoritmo maximiza la diversidad de la población en las primeras etapas de optimización resultando en una exploración adecuada que es muy importante en varios tipos de problemas como los multimodales y deceptivos. Sin embargo, un valor demasiado elevado de D_I podría inducir un grado excesivo de exploración y en consecuencia la intensificación podría fallar. Por otra parte, un valor muy pequeño de D_I provocaría muy poca exploración y por lo tanto sería más difícil evitar óptimos locales. El valor óptimo de D_I podría variar dependiendo del tipo de problema y el criterio de paro. Sin embargo, en esta primera versión no se adapta el valor de D_I para cada problema. En su lugar, se realizó un análisis previo para determinar un valor adecuado, usándose el valor $D_I = 0.3$ en todos los problemas.

Al igual que en la versión estándar de DE, en DE-EDM es necesario asignar una probabilidad de cruza (CR) y un factor de mutación (F). De acuerdo a varios estudios desarrollados por Montgomery y otros [50] la probabilidad de cruza tiene un efecto muy importante. Estos autores mostraron de forma empírica que los valores extremos de CR resultan en comportamientos muy distintos entre sí, pero de gran utilidad. Así, los valores pequeños de CR resultan en una búsqueda alineada a un número reducido de ejes e induce pequeños desplazamientos. Esto provoca una mejora gradual con convergencia lenta que en algunos escenarios podría resultar beneficioso. Por otra parte, los valores elevados de CR en general, proporcionan soluciones de mayor calidad con una menor probabilidad, ya que se tiende a realizar desplazamientos largos. Sin embargo, cuando son exitosos puedan mejorar la aptitud de forma significativa y permiten explorar zonas distantes. Con base en esto, en nuestra propuesta se emplean los dos mecanismos, es decir, valores elevados y pequeños de CR tal y como se muestra en la Ecuación (6).

$$CR = \begin{cases} Normal(0.2, 0.1), & \text{si} \quad rand[0,1] \leq 0.5 \\ Normal(0.9, 0.1), & \text{de otra forma} \end{cases} \tag{6}$$

Por otro lado, siguiendo los principios de distintas variantes del SHADE [3, 9], se consideran las evaluaciones transcurridas para generar el factor de mutación F aplicado. Particularmente, cada valor F se muestrea de una distribución Cauchy (Ecuación 7).

$$Cauchy(0.5, 0.5 * nfes/max\_nfes) \tag{7}$$

El efecto de esta distribución es que en las primeras etapas de optimización se generen valores de F cercanos a 0.5. Posteriormente, conforme la ejecución transcurre, la función de densidad sufre una transformación gradual ya que la varianza se incrementa. Esto implica que se generen valores fuera del intervalo $[0.0, 1.0]$ con una probabilidad más alta. En los casos en que los valores son mayores a 1.0, se utiliza el valor 1.0, mientras que si se genera un valor negativo, se vuelve a muestrear el valor. Uno de los efectos de este enfoque es el de incrementar la probabilidad de generar valores elevados de F conforme transcurren las generaciones, lo que ayuda a evitar la convergencia acelerada en las últimas etapas de optimización.

3.3. Resultados de DE-EDM

En esta sección se presenta la validación experimental de DE-EDM. Específicamente, se muestra que se pueden mejorar los resultados de los algoritmos del estado-del-arte controlando de forma explícita la diversidad. Particularmente, se consideraron los conjuntos de prueba de los concursos de optimización continua organizados en el CEC 2016 y CEC 2017. Cada uno está compuesto de treinta problemas distintos. En la comparativa incluimos a los algoritmos que obtuvieron los primeros lugares en cada año, así como una versión de DE que usa la misma parametrización que DE-EDM pero que no incluye la población élite ni el reemplazamiento basado en diversidad. Los algoritmos considerados del CEC 2016 son el UMOEAs-II [28] y L-SHADE-EpSin [3], que alcanzaron el primero y el segundo lugar respectivamente, mientras que del CEC 2017 se consideran el EBOwithCMAR [42] y el jSO [10]. Todos estos algoritmos fueron probados con los

dos conjuntos de prueba como se sugiere en [48]. Debido a que todos los algoritmos son estocásticos se realizaron 51 ejecuciones con distintas semillas y en cada caso, el criterio de paro fue asignado a $25,000,000$ de evaluaciones de la función objetivo. Además se consideraron diez variables en cada función ($D = 10$). La evaluación de los algoritmos se realizó siguiendo los lineamientos de las competencias del CEC, de forma que se asignó un error igual a 0 si la diferencia entre la mejor solución encontrada y la solución óptima era menor que 10^{-8}. Para cada algoritmo se utilizó la parametrización indicada por sus autores y que se describe a continuación:

- **EBOwithCMAR**: Para la parte EBO, el tamaño máximo de la población de $S_1 = 18D$, el tamaño mínimo de la población de $S_1 = 4$, el tamaño máximo de la población de $S_2 = (146.8)D$, el tamaño mínimo de la población de $S_2 = 10$, el tamaño de la memoria histórica H=6. Para la parte de CMAR el tamaño de la población $S_3 = 4 + 3log(D)$, $\sigma = 0.3$, CS = 50, la probabilidad de búsqueda local $pl = 0.1$ y $cfe_{ls} = 0.4 * FE_{max}$.
- **UMOEAs-II**: Para la parte de MODE, el tamaño máximo de la población de $S_1 = 18D$, el tamaño mínimo de la población de $S_1 = 4$, el tamaño de la memoria histórica H=6. Para la parte del CMA-ES el tamaño de la población $S_2 = 4 + \lfloor 3log(D) \rfloor$, $\mu = \frac{PS}{2}$, $\sigma = 0.3$, CS = 50. Para la búsqueda local, $cfe_{ls} = 0.2 * FE_{max}$.
- **jSO**: El tamaño máximo de la población = $25log(D)\sqrt{D}$, el tamaño de la memoria histórica H= 5, valor de mutación inicial de la memoria $M_F = 0.5$, probabilidad inicial de la memoria $M_{CR} = 0.8$, tamaño mínimo de la población = 4, valor inicial p-best = $0.25 * N$, valor final p-best = 2.
- **L-SHADE-EpSin**: Tamaño máximo de la población = $25log(D)\sqrt{D}$, tamaño de la memoria histórica H= 5, valor de la mutación inicial de la memoria $M_F = 0.5$, probabilidad inicial de la memoria $M_{CR} = 0.5$, frecuencia inicial de la memoria $\mu_F = 0.5$, tamaño mínimo de la población = 4, valor inicial p-best = $0.25 * N$, valor final p-best = 2, generaciones de la búsqueda local $G_{LS} = 250$.
- **DE-EDM**: $D_I = 0.3$, tamaño de la población = 250.
- **Standard-DE**: Tamaño de la población = 250 (mismos operadores que en DE-EDM).

Nuestro análisis experimental se realizó teniendo en cuenta la diferencia entre la solución óptima y la mejor solución obtenida y para comparar los resultados estadísticamente, se siguió un procedimiento similar que el propuesto en [25]. Concretamente, en primer lugar se utilizó la prueba Shapiro-Wilk para comprobar si los resultados se ajustaban a una distribución Gaussiana. En los casos en que sí se ajustaban, se utilizó la prueba de Levene para comprobar la homogeneidad de las varianzas, procediendo con la prueba de ANOVA en caso positivo o con la prueba de Welch en caso negativo. Por otro lado, para los casos que no se ajustaban a distribuciones Guassianas, se utilizó la prueba de Kruskal-Wallis. En todos los casos se fijó el nivel de confianza al 95 %. Se considera que un algoritmo X es superior a un algoritmo Y, si el procedimiento anterior reporta diferencias significativas y si la media y mediana del error obtenido por el método X son inferiores a las obtenidas por el método Y.

En las tablas 1 y 2 se presenta un resumen de los resultados obtenidos para el CEC 2016 y el CEC 2017 respectivamente. La columna etiquetada con "Siempre Resuelto" muestra el número de funciones en que se obtuvo un error de cero en las 51 ejecuciones. La columna etiquetada con "Al menos una vez resuelto" muestra el número de funciones que se resolvieron en al menos una ejecución. Prácticamente todas las funciones del CEC 2017 (28 de ellas) fueron resueltas al menos una vez por nuestra propuesta. Además, se resolvieron 21 funciones del CEC 2016 al menos una vez. Esto representa una diferencia sustancial con los resultados obtenidos por el resto de algoritmos, pues todos ellos resolvieron muchas menos funciones. Con el objetivo de confirmar la superioridad del DE-EDM, se ejecutaron las pruebas estadísticas por pares. La columna etiquetada con el símbolo $\uparrow$ muestra el número de veces en que cada método fue superior, mientras que la columna etiquetada con $\downarrow$ cuenta el número de casos donde el método fue inferior. Finalmente, la columna etiquetada con $\longleftrightarrow$ muestra el número de comparaciones en las que las diferencias no fueron significativas. Las pruebas estadísticas indican que el DE-EDM alcanzó los mejores resultados en los dos años.

Tabla 1: Resumen de los resultados - CEC 2016

Algoritmo	Siempre Resuelto	Resuelto al menos una vez	Pruebas Estadísticas			Puntaje
			↑	↓	⟷	
EBOwithCMAR	8	14	35	56	59	50.28
jSO	9	17	47	51	52	55.43
UMOEAs-II	9	14	51	31	68	62.45
L-SHADE-Epsilon	7	13	20	71	59	50.12
DE-EDM	13	21	77	25	48	100.00
Standard-DE	11	19	50	46	54	56.29

Tabla 2: Resumen de los resultados - CEC 2017

Algoritmo	Siempre Resuelto	Resuelto al menos una vez	Pruebas Estadísticas			Puntaje
			↑	↓	⟷	
EBOwithCMAR	9	18	34	46	70	37.14
jSO	8	15	29	55	66	29.30
UMOEAs-II	11	15	43	40	67	26.89
L-SHADE-Epsilon	8	19	7	81	62	32.78
DE-EDM	21	28	88	6	56	100.00
Standard-DE	12	21	56	29	65	42.91

De hecho el número de comparaciones en que nuestra propuesta ganó en el CEC 2016 y el CEC 2017 fue de 77 y 88 respectivamente y sólo perdió en 25 y 6, respectivamente, que son valores muy superiores a los del resto de algoritmos. La última columna etiquetada con "Puntaje" muestra la puntuación siguiendo los lineamientos propuestos en las competencias del CEC. Particularmente, este método de evaluación combina dos puntajes como se indica en la Ecuación (8), definiendo el puntaje final como $Score = Score_1 + Score_2$.

$$\begin{aligned} Score_1 &= \left(1 - \frac{SE_i - SE_{min}}{SE_i}\right) \times 50, \\ Score_2 &= \left(1 - \frac{SR_i - SR_{min}}{SR_i}\right) \times 50, \end{aligned} \tag{8}$$

Específicamente, SE_i es la sumatoria de errores del i-ésimo algoritmo ($SE_i = \sum_{j=1}^{30} error\_\hat{f}_j$), además $error\_\hat{f}_j$ es el error promedio en la j-ésima función. SE_{min} es la mínima sumatoria de errores entre todos los algoritmos. Por su parte, SR_i es la sumatoria del rango de cada función en el i-ésimo algoritmo ($SR_i = \sum_{j=1}^{30} rango_j$). Además, SR_{min} es la sumatoria mínima de los rangos entre todos los algoritmos. Nuestra propuesta alcanzó el mejor puntaje de 100.00 en los dos años, demostrando su superioridad. Adicionalmente, es destacable que la versión estándar de DE alcanzó resultados buenos, de hecho obtuvo el tercer y el segundo lugar en los años 2016 y 2017, respectivamente. Esto muestra que el rendimiento de los algoritmos del estado-del-arte se deteriora al considerar ejecuciones a largo plazo pues esos algoritmos que fueron muy efectivos a corto plazo, no lo son tanto a largo plazo. La superioridad del DE-EDM es clara con base en todos los tipos de comparativas descritas.

Además, con el fin de proporcionar resultados que puedan usar otros autores para realizar comparativas, en las tablas 3 y 4 se reporta el mejor, peor, mediana, media, desviación estándar y razón de éxito para el CEC 2016 y 2017, respectivamente. En estas tablas se observa que nuestra propuesta resuelve todos los problemas uni-modales y que en la mayor parte de los problemas multi-modales se obtienen resultados muy aceptables con respecto a los reportados por otros métodos. Así, nuestra propuesta resolvió y mejoró significativamente varias funciones complejas que no fueron resueltas por ninguno de los algoritmos restantes.

4. Diseño de operadores de cruza basados en diversidad

La segunda propuesta se aplica en el campo de optimización multi-objetivo y está basada en la extensión del operador de cruza SBX. En esta sección se introducen, en primer lugar, varios conceptos relacionados con los algoritmos evolutivos multi-objetivo, para posteriormente describir algunas de las clasificaciones más populares de operadores de cruza. Finalmente, se realiza un análisis del operador SBX y con base en el mismo

Tabla 3: Resultados del DE-EDM con los problemas del CEC 2016

	Mejor	**Peor**	**Mediana**	**Media**	**Sd**	**Razón de éxito**
f_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_6	0.00E+00	3.60E-02	4.00E-03	7.39E-03	1.15E-02	3.92E-01
f_7	2.00E-02	1.02E-01	5.90E-02	5.77E-02	4.93E-02	0.00E+00
f_8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{10}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{11}	0.00E+00	6.00E-02	0.00E+00	5.88E-03	1.90E-02	9.02E-01
f_{12}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{13}	1.00E-02	8.00E-02	5.00E-02	4.67E-02	2.60E-02	0.00E+00
f_{14}	1.00E-02	5.00E-02	3.00E-02	2.82E-02	2.13E-02	0.00E+00
f_{15}	0.00E+00	4.70E-01	2.20E-01	1.99E-01	1.55E-01	1.96E-02
f_{16}	4.00E-02	1.50E-01	8.00E-02	8.47E-02	4.96E-02	0.00E+00
f_{17}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{18}	0.00E+00	2.00E-02	1.00E-02	7.65E-03	6.32E-03	3.14E-01
f_{19}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{20}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{21}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{22}	0.00E+00	3.00E-02	0.00E+00	3.73E-03	2.76E-02	7.65E-01
f_{23}	0.00E+00	1.00E+02	0.00E+00	2.55E+01	5.10E+01	7.45E-01
f_{24}	0.00E+00	6.90E-01	0.00E+00	2.61E-02	1.33E-01	9.61E-01
f_{25}	1.00E+02	1.00E+02	1.00E+02	1.00E+02	0.00E+00	0.00E+00
f_{26}	8.00E-02	1.00E+02	5.29E+01	5.20E+01	3.19E+01	0.00E+00
f_{27}	2.50E-01	9.10E-01	5.40E-01	5.60E-01	2.92E-01	0.00E+00
f_{28}	0.00E+00	3.57E+02	3.43E+02	2.76E+02	1.60E+02	1.96E-01
f_{29}	1.00E+02	1.00E+02	1.00E+02	1.00E+02	0.00E+00	0.00E+00
f_{30}	1.84E+02	1.84E+02	1.84E+02	1.84E+02	3.25E-02	0.00E+00

Tabla 4: Resultados del DE-EDM con los problemas del CEC 2017

	Mejor	**Peor**	**Mediana**	**Media**	**Sd**	**Razón de éxito**
f_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{10}	0.00E+00	1.20E-01	0.00E+00	1.65E-02	3.39E-02	7.45E-01
f_{11}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{12}	0.00E+00	2.20E-01	0.00E+00	6.37E-02	1.76E-01	6.67E-01
f_{13}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{14}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{15}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{16}	0.00E+00	2.10E-01	0.00E+00	2.47E-02	7.27E-02	8.82E-01
f_{17}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{18}	0.00E+00	1.00E-02	0.00E+00	1.96E-03	4.47E-03	8.04E-01
f_{19}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{20}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{21}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{22}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{23}	0.00E+00	3.00E+02	0.00E+00	3.49E+01	1.03E+02	8.82E-01
f_{24}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{25}	0.00E+00	1.00E+02	0.00E+00	3.92E+00	2.00E+01	9.61E-01
f_{26}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{27}	0.00E+00	3.87E+02	3.87E+02	2.05E+02	2.68E+02	1.96E-02
f_{28}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00
f_{29}	1.45E+02	2.26E+02	2.18E+02	1.99E+02	4.21E+01	0.00E+00
f_{30}	3.95E+02	3.95E+02	3.95E+02	3.95E+02	2.10E-01	0.00E+00

se propone una variante dinámica denominada *Operador de Cruza Dinámico basado en Simulación Binaria* (Dynamic Simulated Binary Crossover - DSBX), el cual es validado experimentalmente.

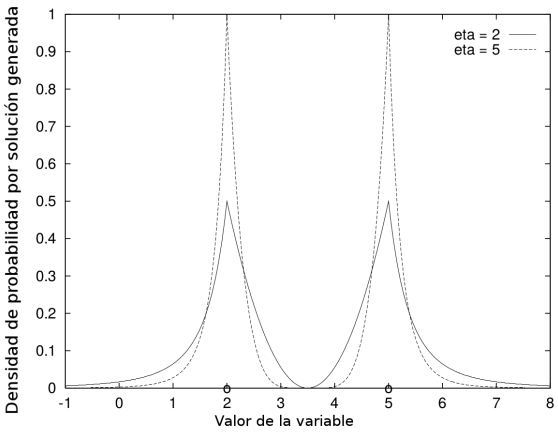

Figura 1: Función de densidad del operador de cruza SBX con índices de distribución 2 y 5. Las soluciones padre están ubicadas en 2 y 5 respectivamente.

4.1. Algoritmos evolutivos multi-objetivo

Los EAs han sido utilizados frecuentemente para lidiar con problemas de optimización multi-objetivo (Multi-objective Optimization Problems - MOPs). Particularmente, un MOP de dominio continuo y que es basado en minimización puede ser definido como se indica en la Ecuación (1) de la sección 1, fijando la M a un valor mayor que uno. Dadas dos soluciones $\vec{x}$, $\vec{y} \in \Omega$, $\vec{x}$ domina a $\vec{y}$, denotado por $\vec{x} \prec \vec{y}$, si y solo si $\forall m \in 1, 2, ..., M : f_m(x_i) \leq f_m(y_i)$ y $\exists m \in 1, 2, ..., M : f_m(x_i) < f_m(y_i)$. Una solución $\vec{x^*} \in \Omega$ es conocida como solución óptima de Pareto si no existe otra solución $\vec{x} \in \Omega$ que domine a $\vec{x^*}$. El conjunto de Pareto es el conjunto de todas las soluciones óptimas de Pareto y el frente de Pareto está formado por las imágenes del conjunto de Pareto. El propósito de un *Algoritmo Evolutivo Multi-Objetivo* (Multi-objective Evolutionary Algorithm - MOEA) es, esencialmente, obtener un conjunto de soluciones bien distribuidas y cercanas a las soluciones del frente de Pareto.

En los últimos años, se han diseñado una gran cantidad de MOEAs siguiendo distintos principios. A raíz de esto se han propuesto varias taxonomías [63] y, por ejemplo, en base a sus principios de diseño se pueden clasificar en basados en la dominancia de Pareto, indicadores y/o descomposición [57]. Hay algoritmos muy competitivos de cada uno de los grupos por lo que en esta sección se consideraron MOEAs de todos los grupos para realizar la validación. Particularmente, la validación experimental se desarrolló incluyendo a los algoritmos *Algoritmo Genético basado en Ordenación de los No-Dominados* (Non-Dominated Sorting Genetic Algorithm - NSGA-II) [23], el *Algoritmo Evolutivo Multi-objetivo basado en descomposición* (MOEA based on Decomposition - MOEA/D) [74] y el *Algoritmo Evolutivo Multi-objetivo basado en la Métrica-S* (the S-Metric Selection Evolutionary Multi-objective Optimization Algorithm - SMS-EMOA) [7]. Estos algoritmos son representativos de los basados en dominancia, basados en descomposición y basados en indicadores respectivamente.

4.2. Operadores de cruza

Los operadores de cruza son utilizados para generar soluciones hijas utilizando la información de las soluciones padre. Así, estos operadores combinan las características de dos o más soluciones padre con el propósito de generar nuevas soluciones candidatas. En base en que en la literatura existen diversos operadores de cruza, se han propuesto varias taxonomías para clasificarlos. Particularmente, las taxonomías se basan en varias características tales como la ubicación relativa entre padres e hijos o el tipo de relaciones existentes entre las variables.

Una clasificación popular hace distinción entre operadores de cruza *basados en las variables* y *basados en los vectores*. En los *basados en las variables*, cada variable de las soluciones padre son combinadas para crear nuevos valores de forma independiente, siendo ideales para lidiar con problemas separables. Algunos

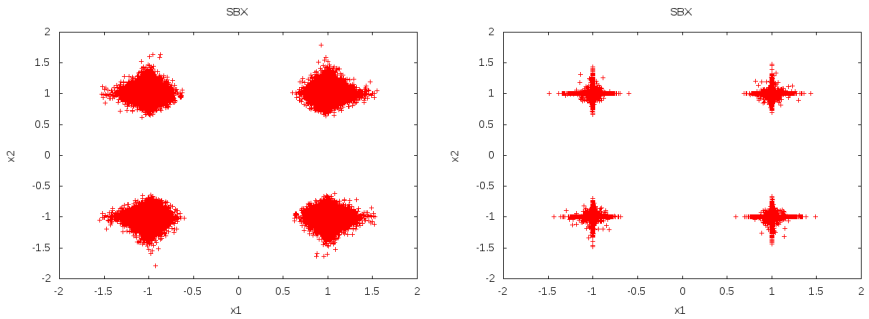

Figura 2: Simulaciones del operador SBX con un índice de distribución igual a 20. Las soluciones padre están ubicadas en $P_1 = (-1.0, -1.0)$ y $P_2 = (1.0, 1.0)$. En la parte izquierda la simulación consiste en alterar cada variable con probabilidad 1.0 y en la parte derecha con probabilidad 0.1 (parámetro δ_1 en el Algoritmo 3).

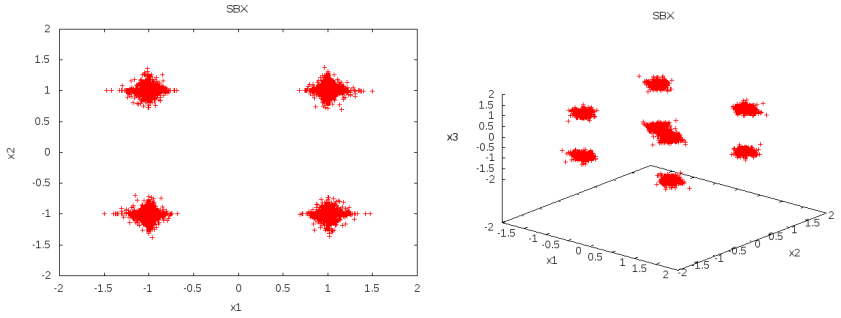

Figura 3: Simulaciones del operador SBX con un índice de distribución de 20. Las soluciones padre están ubicadas en $P_1 = (-1.0, -1.0)$, $P_2 = (1.0, 1.0)$ (parte izquierda) y $P_1 = (-1.0, -1.0, -1.0)$, $P_2 = (1.0, 1.0, 1.0)$ (parte derecha).

operadores que pertenecen a esta categoría son el *Operador de Cruza por Mezcla* (the Blend Crossover - BLX) [30] y el SBX [1]. Por otra parte, los operadores de recombinación *basados en vectores* tienen en cuenta la relación que existe entre las variables, por lo que la combinación no es independiente. Así, este tipo de operadores pueden por ejemplo, realizar combinaciones lineales de las soluciones involucradas. Algunos operadores que pertenecen a esta categoría son el *Operador de Cruza Unimodal Normalmente Distribuido* (Unimodal Normally Distributed Crossover - UNDX) [54], y el *Operador de Cruza Simplex* (Simplex Crossover - SPX) [64].

Adicionalmente, los operadores de cruza pueden ser clasificados como *centrados en los padres* y *centrados en la media* [39]. En los operadores centrados en los padres, las soluciones hijas tienden a ser creadas alrededor de cada solución padre, mientras que en los operadores centrados en la media se tiende a crear a las soluciones hijas alrededor de la media de los valores de las soluciones padres.

4.2.1. El operador de cruza basado en simulación binaria - SBX

Entre los operadores de cruza, el *Operador de Cruza Basado en Simulación Binaria* (Simulated Binary Crossover - SBX) [20] es uno de los más utilizados en dominios continuos y fue el que se decidió extender en nuestra propuesta, por lo que esta sección se centra en este operador de cruza. El operador SBX es clasificado como un operador centrado en los padres, por lo que los valores asociados a los hijos (c_1 y c_2) tienden a ser cercanos a los valores de los padres (p_1 y p_2). Específicamente, el proceso para generar los valores de las soluciones hijas se basa en utilizar una distribución de probabilidad. Esta distribución controla el factor de dispersión $\beta = |c_1 - c_2|/|p_1 - p_2|$, el cual es definido como la razón entre la distancia de los valores de las soluciones hijas y la distancia entre los valores de las soluciones padre. Esta función de densidad se define con base en un índice de distribución η_c (es un parámetro de control especificado por el usuario) el cual altera

Algoritmo 3 Operador de Cruza basado en Simulación Binaria (SBX)

```
1: Entrada: Soluciones padre (P_1, P_2), Índice de distribución (η_c), Probabilidad de cruza (P_c).
2: Salida: Soluciones hijo (C_1, C_2).
3: if U[0,1] ≤ P_c then
4:     for para cada variable d do
5:         if U[0,1] ≤ δ_1 then
6:             Generar C_{1,d} utilizando las distribuciones dadas en [21].
7:             Generar C_{2,d} utilizando las distribuciones dadas en [21].
8:             if U[0,1] ≤ (1 − δ_2) then
9:                 Intercambiar C_{1,d} con C_{2,d}.
10:        else
11:            C_{1,d} = P_{1,d}.
12:            C_{2,d} = P_{2,d}.
13: else
14:    C_1 = P_1.
15:    C_2 = P_2.
```

la capacidad de exploración. Específicamente, un índice pequeño induce una probabilidad elevada de crear valores de las soluciones hijas distantes de los valores de las soluciones padre, mientras que índices elevados tienden a crear soluciones muy similares a las soluciones padre, lo cual se ilustra en la figura 1. La definición matemática de la distribución y la forma de generar los valores de las soluciones hijas pueden ser estudiados en [20]. Las ecuaciones iniciales fueron formuladas con base en un problema de optimización sin límites en las variables. Sin embargo, en muchos problemas, cada variable está limitada dentro de un límite inferior y superior. Para considerar los límites del espacio de decisión se propuso una modificación de la distribución de probabilidad [21], que es la que se usa en este artículo. Para el caso de nuestro método, no es demasiado importante conocer la fórmula exacta, sino el efecto del valor η_c que se ha ilustrado previamente.

Es importante aclarar que la primera versión del SBX fue diseñada con base en una sola variable. Posteriormente, los autores consideraron aplicarlo a problemas con múltiples variables [1], considerando una estrategia simple para escoger las variables que se van a cruzar [54]. Específicamente, con base en los principios del operador de cruza uniforme, cada variable es cruzada con una probabilidad igual a 0.5. A pesar de su sencillez y de no tener en cuenta las posibles dependencias entre variables, hoy en día ésta es la forma más común de aplicar el SBX.

4.2.2. Implementación y análisis del operador SBX

En este apartado se discuten algunas de las principales características de la implementación más utilizada del operador SBX para problemas con múltiples variables. Esencialmente, se consideran tres componentes clave que podrían afectar al rendimiento de los MOEAs. En primer lugar, como ya se mencionó anteriormente, cada variable es alterada con una probabilidad fija igual a 0.5. Si este valor de probabilidad se incrementa, entonces los hijos tienden a crearse a mayor distancia de los padres debido a que se modifican más variables de forma simultánea. En los problemas separables, en general interesa modificar sólo una variable mientras que en los no separables suele ser más conveniente modificar varias variables a la vez. En la figura 2 se pueden observar las implicaciones de modificar esta probabilidad considerando un problema con dos variables. Particularmente, en la parte derecha se muestra que una probabilidad pequeña provoca que algunos valores permanezcan intactos, es decir, hay una tendencia de generar desplazamientos paralelos a los ejes, lo cual suele ser ideal para problemas separables. Por otra parte, en la parte izquierda se muestra que utilizando una probabilidad elevada existe un comportamiento de búsqueda distinto donde la tendencia anterior desaparece, lo cual podría ser oportuno para problemas no separables. Es importante destacar que existe una relación entre esta probabilidad y el índice de distribución, ya que estos dos factores tienen un efecto directo en la similitud que existe entre las soluciones padres e hijas.

El segundo aspecto importante es que después de generar los dos valores de las soluciones hijas, estos valores son intercambiados con una probabilidad fija (usualmente es 0.5), es decir el valor de la solución hija c_1 no siempre es heredado a partir de la solución padre más cercana p_1. Esta es una característica no muy discutida, sin embargo es un aspecto muy relevante que afecta al rendimiento del algoritmo. En algunos contextos esta probabilidad se denomina "Probabilidad de cruza uniforme por variable" (Variable uniform crossover probability) [65] o "Recombinación Discreta" (Discrete Recombination) [53]. Dado que en

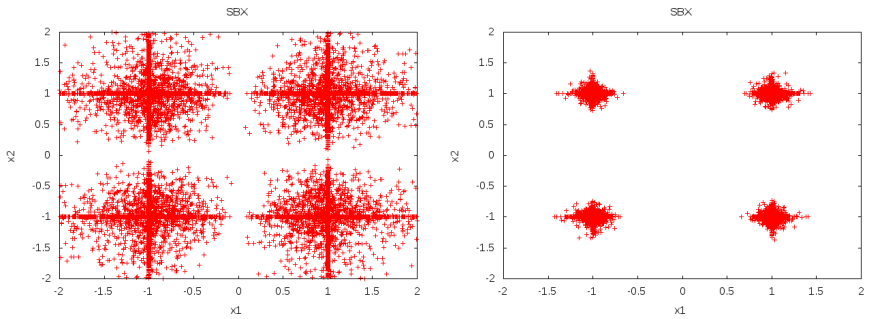

Figura 4: Simulación del operador SBX donde las soluciones padre están ubicadas en $P_1 = (-1.0, -1.0)$ y $P_2 = (1.0, 1.0)$. En la parte izquierda se consideró un índice de distribución de 2 y en la derecha de 20.

el ámbito multi-objetivo se promueve más diversidad en las variables de decisión de forma implícita, estos intercambios podrían ser altamente disruptivos. De hecho, debido a esto, no es totalmente claro que el SBX deba ser categorizado como un operador centrado en los padres. Estos intercambios que existen entre los valores de las soluciones hijas tienen un efecto de realizar múltiples "reflexiones" en el espacio de búsqueda. Así, conforme se incrementa la dimensionalidad en el espacio de las variables, el número de regiones cubiertas crece de forma exponencial como se puede observar en los casos de dos y tres dimensiones en la figura 3. Es importante notar que esta característica tiene un efecto relevante en la distancia entre las soluciones padre y las soluciones hijas.

Finalmente, se discute el índice de distribución que es quizás la característica más conocida del operador SBX. Un índice de distribución pequeño provoca un grado de exploración elevado. De hecho, un índice de distribución igual a uno tiene un efecto similar al *Operador de Recombinación Difusa* (Fuzzy Recombination Operator) [69]. En la figura 4 se puede observar el efecto de aplicar distintos índices de distribución. Particularmente, en la parte izquierda se considera un índice de distribución pequeño, mientras en la parte derecha se considera un índice de distribución grande. Se observa que este último genera soluciones candidatas similares a las soluciones padre.

En el Algoritmo 3 se muestra la implementación del operador SBX. Este pseudocódigo está basado en la implementación que está integrada en el código del NSGA-II propuesto por Deb et al. [23], la cual se considera como la variante más popular. En los parámetros de entrada se requieren dos soluciones padre (P_1, P_2), y éste crea dos soluciones hijas (C_1, C_2). El primero y el segundo componente mencionados previamente corresponden a las líneas 5 y 9 respectivamente. El caso clásico del operador SBX se configura con los parámetros $\delta_1 = \delta_2 = 0.5$ y $\eta_c = 20$. Es importante hacer notar que la implementación clásica no considera la dimensionalidad de las variables o el criterio de paro como parte de sus parámetros internos.

4.3. Propuesta - DSBX

Con base en el análisis anterior y con el propósito de inducir un balanceo entre exploración e intensificación de forma dinámica y que dependa del instante de ejecución, se proponen las siguiente modificaciones. En primer lugar, se modifica la probabilidad de alterar una variable (δ_1) durante la ejecución. La intención de esta modificación es estimular la capacidad de exploración considerando inicialmente una probabilidad elevada para alterar más variables de forma simultánea, mientras que conforme la ejecución avanza se reduce esta probabilidad, con el fin de reducir el número de variables que se modifican y promover así movimientos más pequeños. El valor de δ_1 se cambia con base en un modelo lineal decreciente, donde inicialmente se fija a 1.0 y se decrementa de forma que a la mitad del total de generaciones alcanza el valor 0.5. Este último valor es mantenido hasta el final de la ejecución, es decir, desde la mitad de la ejecución este parámetro se fija al mismo valor que el de la implementación tradicional del SBX. Para asignar el valor δ_1 se utiliza la Ecuación (9), donde $G_{Transcurridas}$ corresponde a la generación actual y G_{Total} corresponde al número total de generaciones.

Tabla 5: Puntos de referencias para el indicador HV

Instancias	**Punto de referencia**
WFG1-WFG9	$[2.1, ..., 2m + 0.1]$
DTLZ 1, 2, 4	$[1.1, ..., 1.1]$
DTLZ 3, 5, 6	$[3, ..., 3]$
DTLZ7	$[1.1, ..., 1.1, 2m]$
UF 1-10	$[2, ..., 2]$

Tabla 6: Información estadística de las métricas considerando dos objetivos

	NSGA-II						MOEA/D						SMS-EMOA					
	1	2	3	4	5	DE	1	2	3	4	5	DE	1	2	3	4	5	DE
Average HV	0.88	0.90	0.90	0.91	0.93	**0.94**	0.87	0.87	0.87	0.90	**0.91**	**0.91**	0.88	0.89	0.87	0.91	0.92	**0.93**
Average IGD+	0.12	0.09	0.11	0.07	0.06	**0.05**	0.14	0.12	0.14	0.09	0.08	**0.07**	0.13	0.11	0.14	0.08	0.07	**0.05**

El segundo cambio está relacionado con la probabilidad de aplicar reflexiones $(1 - \delta_2)$. En este caso, δ_2 es actualizado de acuerdo a la Ecuación (9), por lo tanto la probabilidad de aplicar una reflexión se incrementa de 0.0 a 0.5 durante la ejecución. Esta modificación se realiza con el propósito de evitar el comportamiento disruptivo de intercambiar variables en las primeras generaciones ya que esto provocaría modificaciones muy drásticas. De esta forma, sería más sensato aplicar estas reflexiones una vez que los individuos convergen en cierto grado. Nótese que se incrementa hasta el valor 0.5 el cual es el valor utilizado en la implementación del SBX estándar, ya que no se quiere disponer de un comportamiento excesivamente disruptivo. Sin embargo, en el futuro sería interesante realizar estudios con otros valores finales.

$$\delta_1 = \delta_2 = max\left(0.5, 1.0 - \frac{G_{Transcurridas}}{G_{Total}}\right) \tag{9}$$

Finalmente, el índice de distribución también es modificado durante la ejecución. En las primeras etapas se promueve un índice de distribución pequeño con el propósito de incrementar la capacidad de exploración del SBX. Posteriormente se decrementa de forma lineal, lo cual tiene el efecto de que la curva de distribución se cierre, y por lo tanto se promueve un mayor grado de intensificación en las etapas finales. El incremento lineal es llevado a cabo usando la Ecuación (10), por lo tanto el índice de distribución es alterado de 2 a 22. Es importante aclarar que ya se han considerado modificaciones similares al índice de distribución [79], [34], aunque las otras propiedades que se modifican nunca han sido estudiadas en profundidad.

$$\eta_c = 2 + 20 \times \left(\frac{G_{Transcurridas}}{G_{Total}}\right) \tag{10}$$

4.4. Resultados

En este apartado se analizan los resultados obtenidos con las variantes dinámicas del SBX (DSBX). El nuevo operador de cruza se integró en los algoritmos NSGA-II, MOEA/D y SMS-EMOA. En primer lugar, se analiza cada una de las tres modificaciones propuestas de forma aislada. Posteriormente se construye un caso donde se consideran las dos modificaciones que ofrecieron mejores resultados de forma simultánea. Como parte de la validación experimental se consideran los problemas de prueba WFG [37], DTLZ [24] y UF [75]. Además, con el propósito de comparar nuestra extensión del SBX con otros operadores se incluye la variante de evolución diferencial conocida como DEMO [65].

Para llevar a cabo el análisis experimental, y dado que todos los algoritmos son estocásticos, cada caso fue ejecutado 35 veces con distintas semillas. La configuración global que se aplicó a todos los algoritmos fue la siguiente. Se asignó el criterio de paro a $25,000$ generaciones, el tamaño de la población a 100, se configuraron los problemas de prueba WFG con dos y tres objetivos considerando 24 variables, donde 20 variables son parámetros de distancia y 4 son parámetros de posición. Como es sugerido en [24] se consideró $n = M + r - 1$ variables de decisión en los problemas de prueba DTLZ, donde para los problemas DTLZ1, DTLZ2 - DTLZ6 y DTLZ7 se consideraron $r = \{5, 10, 20\}$ respectivamente. En el caso de los problemas de prueba UF se utilizaron 30 variables de decisión. Finalmente, se hizo uso del operador de mutación polinomial con una

Tabla 7: Información estadística de las métricas considerando tres objetivos

	NSGA-II						MOEA/D						SMS-EMOA					
	1	2	3	4	5	DE	1	2	3	4	5	DE	1	2	3	4	5	DE
Average HV	**0.87**	0.84	**0.87**	**0.87**	**0.87**	0.85	0.84	0.84	0.84	**0.86**	**0.86**	0.85	0.90	0.89	0.88	**0.91**	**0.91**	**0.91**
Average IGD+	0.13	0.16	0.13	**0.12**	**0.12**	0.13	0.15	0.14	0.15	**0.11**	**0.11**	0.13	0.11	0.11	0.13	**0.09**	**0.09**	0.13

Tabla 8: Resumen de las pruebas estadísticas

NSGA-II															
	1			2			3			4			5		
	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷
HV-2obj	16	29	47	6	61	25	28	19	45	31	23	38	**54**	3	35
HV-3obj	15	19	42	12	50	14	17	15	44	**33**	10	33	26	9	41
IGD-2obj	14	30	48	4	60	28	25	17	50	33	19	40	**52**	2	38
IGD-3obj	14	18	44	13	44	19	18	15	43	**33**	15	28	23	9	44

MOEA/D															
	1			2			3			4			5		
	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷
HV-2obj	15	33	44	10	60	22	25	26	41	39	18	35	**57**	9	26
HV-3obj	10	22	44	12	39	25	11	19	46	24	10	42	**38**	5	33
IGD-2obj	16	31	45	9	60	23	23	27	42	37	17	38	**57**	7	28
IGD-3obj	12	22	42	13	43	20	13	24	39	30	9	37	**40**	10	26

SMS-EMOA															
	1			2			3			4			5		
	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷	↑	↓	⟷
HV-2obj	9	35	48	7	43	42	16	31	45	41	9	42	**53**	8	31
HV-3obj	7	21	48	9	35	32	13	21	42	27	6	43	**31**	4	41
IGD-2obj	10	34	48	15	48	29	12	33	47	41	12	39	**55**	6	31
IGD-3obj	8	20	48	13	30	33	9	19	48	22	5	49	**27**	5	44

probabilidad de mutación de $1/n$ y con un índice de distribución igual a 50, mientras que el operador de cruza SBX se utilizó con una probabilidad de cruza igual a 0.9 y un índice de distribución de 20.

A continuación se especifica la parametrización adicional propia de cada algoritmo:

- **DEMO**: CR = 0.3 y F = 0.5.
- **SMS-EMOA**: desplazamiento para calcular el HV = 100.
- **MOEA/D**: tamaño de la vecindad = 10, el número de actualizaciones por subproblema (nr) = 2 y $\delta = 0.9$.

Para comparar los frentes obtenidos por cada método se utiliza el hipervolumen normalizado (HV), que se debe maximizar, y la *Distancia Generacional Invertida Modificada* (Inverted Generational Distance Plus - IGD+), que se debe minimizar. En la tabla 5 se presentan los puntos de referencia utilizados para el indicador del hipervolumen los cuales son similares a los utilizados en [6, 77]. Por otra parte, para comparar los resultados estadísticamente (valores del IGD+ y HV), se siguió un procedimiento similar al que se propuso en [25], siendo el mismo que se aplicó para validar la primera propuesta.

4.5. Análisis de cada modificación en el operador SBX

En este apartado se analiza el efecto que cada modificación propuesta tiene de forma independiente sobre los resultados obtenidos. Para ello basados en el Algoritmo 3 se proponen cuatro casos:

- **Caso 1**: se aplica la versión estándar del operador SBX donde $\delta_1 = \delta_2 = 0.5$ y $\eta_c = 20$.
- **Caso 2**: se actualiza el valor δ_1 como se indica en la Ecuación (9), $\delta_2 = 0.5$ y $\eta_c = 20$.
- **Caso 3**: se actualiza el valor δ_2 como se indica en la Ecuación (9), $\delta_1 = 0.5$ y $\eta_c = 20$.
- **Caso 4**: se actualiza el índice de distribución de acuerdo a la Ecuación (10), $\delta_1 = \delta_2 = 0.5$.

En las tablas 6 y 7 se muestra información del HV normalizado [79] y del IGD+ [38] para cada caso, con dos y tres objetivos respectivamente (el Caso 5 se comenta más adelante). Específicamente, se muestra la media del HV e IGD+ para todos los problemas en dos y tres objetivos. Se observa que considerando dos y tres objetivos el Caso 4 mejora al Caso 1, al Caso 2 y al Caso 3 en todos los algoritmos. Por lo tanto, se aprecian de forma clara los beneficios de incrementar el índice de distribución durante la ejecución. Por otra parte, al considerar tres objetivos, el Caso 2 presentó un rendimiento menor que el Caso 1, posiblemente debido a que al alterar tantas variables de forma simultánea, se produce un comportamiento excesivamente disruptivo. Quizás los resultados podrían mejorar si el parámetro δ_1 es alterado de una forma distinta, sin embargo esto se deja como trabajo futuro. Los análisis anteriores únicamente consideran la media obtenida en todos los problemas de prueba. En la tabla 8 se muestran los resultados de las pruebas estadísticas (el Caso 5 se discute en la siguiente sección). Particularmente, se realizan comparaciones por pares en base a las pruebas estadísticas ya mencionadas entre los cinco casos. Este procedimiento se realizó de forma independiente para el NSGA-II, MOEA/D y el SMS-EMOA. Para cada algoritmo y para cada caso, la columna "↑" reporta el número de comparaciones donde las pruebas estadísticas confirmaron la superioridad del caso correspondiente, mientras que en la columna "↓" se reporta el número de veces donde este caso fue inferior y la columna "⟷" indica el número de comparaciones donde las diferencias no fueron significativas. Con base en estos resultados se puede observar que el caso 3 y caso 4 aportan con respecto al caso 1, mientras que el caso 2 no es robusto, siendo en varios algoritmos inferior al caso 1. Adicionalmente, se anexan resultados detallados para facilitar que otros investigadores puedan realizar comparativas con un mayor nivel de detalle[2].

4.6. Modificación simultánea de varios componentes

Con base en los análisis realizados con anterioridad se propone una variante del operador SBX que incorpora las modificaciones propuestas en el Caso 3 y Caso 4 de forma simultánea, es decir, se incorpora un cambio dinámico tanto en el parámetro δ_2 como en el índice de distribución. Debido a los resultados de baja calidad del Caso 2 el parámetro δ_1 no se modifica de forma dinámica. Particularmente, el Caso 5 se construye con base en el Algoritmo 3 asignando el parámetro δ_1 a 0.5 debido a que es la forma de la versión estándar del SBX, mientras que δ_2 es actualizado de acuerdo a la Ecuación (9) y el parámetro η_c es actualizado como se indica en la Ecuación (10).

De acuerdo a la media del HV y del IGD+ que se obtuvieron en el Caso 5 (ver tablas 6 y 7), es claro que integrar los Casos 3 y 4 proporciona beneficios importantes. En el caso de dos objetivos se observa una ventaja significativa. Sin embargo, en el caso de tres objetivos los Casos 4 y 5 son muy similares con base en la media. Además, al considerar tres objetivos los resultados que se obtuvieron en el Caso 5 son superiores a los obtenidos con DE, mientras que al considerar la versión estándar de SBX se observa un deterioro. Por lo tanto, si el operador SBX es configurado adecuadamente puede generar resultados similares o superiores que el algoritmo DEMO.

Finalmente, en la tabla 8, se muestran los resultados de las pruebas estadísticas. Los beneficios obtenidos por el Caso 5 son muy claros. De hecho, únicamente el Caso 4 obtuvo mejores resultados que el Caso 5 al considerar tres objetivos y al algoritmo NSGA-II. Esto demuestra que se pueden obtener mejores resultados si se integran varias modificaciones dinámicas de forma simultánea. Además, los resultados confirman las ventajas de la versión dinámica frente al caso estándar (Caso 1). El único caso que no presenta ventajas importantes frente a la versión estándar del SBX es el Caso 2, el cual fue discutido con anterioridad.

5. Conclusiones y trabajo futuro

La convergencia prematura es una de las debilidades más importantes de los EAs. Una de las formas de lidiar con este problema consiste en inducir un balanceo adecuado entre exploración e intensificación. Sin embargo, obtener este balanceo no es una tarea trivial debido a que cada problema de optimización tiene características distintas y por ello han surgido numerosos esquemas para tratar esta problemática.

[2] https://github.com/joelchaconcastillo/SBX_CEC2018.

Con base en esto se han desarrollado varias taxonomías con el propósito de clasificar las diferentes formas en que se puede preservar y promover la diversidad. En este trabajo, con el fin de ilustrar como se pueden realizar diseños de EAs que tomen en cuenta el balanceo entre exploración e intensificación se presentan dos ejemplos. De forma general, las propuestas establecen un comportamiento dinámico de algún componente del EA teniendo en cuenta para ello el criterio de paro y el instante en que se encuentra la ejecución. Así, se obtuvo un balanceo adecuado en el que en las primeras fases se promueve más exploración y en las últimas se promueve más la intensificación. De modo más específico, en la primera propuesta se modifica evolución diferencial por medio de una nueva fase de reemplazo con el propósito de conseguir este balanceo. Además, esta contribución incorpora una población élite con el propósito de proporcionar soluciones de calidad tanto a mediano como a largo plazo. Con base en la validación experimental llevada a cabo, se observa que esta propuesta mejora a los mejores algoritmos que se habían propuesto en una competición que consideró las funciones de prueba usadas en este capítulo. Posteriormente, se presenta un análisis del operador de cruza SBX y se desarrolló una variante que modifica varios componentes de forma dinámica considerando también el criterio de paro y las generaciones transcurridas. Con base en los resultados obtenidos con problemas muy populares del ámbito de optimización multi-objetivo, se muestra que usar el operador SBX modificado ofrece ventajas muy importantes en varios MOEAs y para diferentes indicadores. De forma general se observa que obtener un balanceo apropiado entre exploración e intensificación es realmente importante, y que esto se puede conseguir utilizando técnicas radicalmente diferentes aunque siguiendo los mismos principios de diseño.

El campo de diseño de EAs es un campo muy activo y en el que hay muchas ideas aún por explorar. Uno de los aspectos importantes es que las estrategias de control de diversidad ofrecen muchas ventajas a largo plazo, pero se debe hacer el esfuerzo por conseguir que se pueda reducir el número de evaluaciones requeridas para obtener resultados prometedores. Otro aspecto importante es que, en general, la mayor parte de las estrategias de control de diversidad introduce nuevos parámetros, por lo que es importante combinar estos mecanismos con estrategias de control de parámetros adaptativas o auto-adaptativas para facilitar su utilización. Finalmente, para el caso específico del DSBX, sería interesante integrarlo con técnicas de aprendizaje para lidiar con problemas con dependencias y mejorar así aún más el rendimiento de la propuesta.

Para saber más

- Se puede consultar material relacionado con operadores de cruza y algoritmos evolutivos multi-objetivo en [13], [22], [14].
- Si desea consultar información básica sobre algoritmos evolutivos se sugiere consultar [27], [44], [5].
- Para obtener más información relacionada al tema de evolución diferencial se sugiere consultar [58], [12].
- Si desea consultar aplicaciones de algoritmos evolutivos multi-objetivo se puede consultar [78], [4].

Agradeciemientos

Los autores agradecen el apoyo financiero proporcionado por CONACyT a través del proyecto no. 285599.

Referencias

[1] R. B. Agrawal, K. Deb, K. Deb, R. B. Agrawal, Simulated binary crossover for continuous search space, Tech. rep. (1994).
[2] E. Alba, Parallel metaheuristics: a new class of algorithms, vol. 47, John Wiley & Sons, 2005.
[3] N. H. Awad, M. Z. Ali, P. N. Suganthan, R. G. Reynolds, An ensemble sinusoidal parameter adaptation incorporated with l-shade for solving cec2014 benchmark problems, in: Evolutionary Computation (CEC), 2016 IEEE Congress on, IEEE, 2016, pp. 2958–2965.
[4] T. Back, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford university press, 1996.

[5] T. Bäck, D. B. Fogel, Z. Michalewicz, Evolutionary computation 1: Basic algorithms and operators, vol. 1, CRC press, 2000.
[6] J. A. M. Berenguer, C. A. C. Coello, Evolutionary many-objective optimization based on kuhn-munkres' algorithm, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 3–17.
[7] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume , European Journal of Operational Research 181 (3) (2007) 1653 – 1669.
[8] A. Bolufé-Röhler, S. Estévez-Velarde, A. Piad-Morffis, S. Chen, J. Montgomery, Differential evolution with thresheld convergence, in: Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE, 2013, pp. 40–47.
[9] J. Brest, M. S. Maučec, B. Bošković, il-shade: Improved l-shade algorithm for single objective real-parameter optimization, in: Evolutionary Computation (CEC), 2016 IEEE Congress on, IEEE, 2016, pp. 1188–1195.
[10] J. Brest, M. S. Maučec, B. Bošković, Single objective real-parameter optimization: Algorithm jso, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, 2017, pp. 1311–1318.
[11] L. T. Bui, H. A. Abbass, J. Branke, Multiobjective optimization for dynamic environments, in: Evolutionary Computation, 2005. The 2005 IEEE Congress on, vol. 3, IEEE, 2005, pp. 2349–2356.
[12] U. K. Chakraborty, Advances in differential evolution, vol. 143, Springer, 2008.
[13] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, et al., Evolutionary algorithms for solving multi-objective problems, vol. 5, Springer, 2007.
[14] C. A. C. Coello, G. B. Lamont, D. A. V. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[15] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE transactions on evolutionary computation 15 (1) (2011) 4–31.
[16] D. Dasgupta, Z. Michalewicz, Evolutionary algorithms in engineering applications, Springer Science & Business Media, 2013.
[17] K. A. De Jong, Analysis of the behavior of a class of genetic adaptive systems.
[18] K. A. De Jong, Evolutionary computation: a unified approach, MIT press, 2006.
[19] K. Deb, An investigation of niche and species formation in genetic function optimization, ICGA'89 (1989) 42–50.
[20] K. Deb, R. B. Agrawal, Simulated binary crossover for continuous search space, Complex Systems 9 (3) (1994) 1–15.
[21] K. Deb, H.-G. Beyer, Self-adaptive genetic algorithms with simulated binary crossover, Secretary of the SFB 531, 1999.
[22] K. Deb, D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc., New York, NY, USA, 2001.
[23] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[24] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multiobjective optimization, Springer, 2005.
[25] J. J. Durillo, A. J. Nebro, C. A. C. Coello, J. Garcia-Nieto, F. Luna, E. Alba, A Study of Multiobjective Metaheuristics When Solving Parameter Scalable Problems, IEEE Transactions on Evolutionary Computation 14 (4) (2010) 618–635.
[26] A. E. Eiben, C. A. Schippers, On evolutionary exploration and exploitation, Fundamenta Informaticae 35 (1-4) (1998) 35–50.
[27] A. E. Eiben, J. E. Smith, et al., Introduction to evolutionary computing, vol. 53, Springer, 2003.
[28] S. Elsayed, N. Hamza, R. Sarker, Testing united multi-operator evolutionary algorithms-ii on single objective optimization problems, in: Evolutionary Computation (CEC), 2016 IEEE Congress on, IEEE, 2016, pp. 2966–2973.
[29] L. J. Eshelman, The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, in: G. J. Rawlins (ed.), Foundations of Genetic Algorithms, vol. 1, Morgan Kaufmann Publishers, 1991, pp. 265 – 283.
[30] L. J. Eshelman, Real coded genetic algorithms and interval-schemata, Foundations of genetic algorithms 2 (1993) 187–202.
[31] W. Gao, G. G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE transactions on cybernetics 44 (8) (2014) 1314–1327.
[32] F. Glover, G. Kochenberger, Handbook of metaheuristics (international series in operations research and management science), JOURNAL-OPERATIONAL RESEARCH SOCIETY 56 (5) (2005) 614–614.
[33] J. J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Transactions on systems, man, and cybernetics 16 (1) (1986) 122–128.
[34] M. Hamdan, The distribution index in polynomial mutation for evolutionary multiobjective optimisation algorithms: An experimental study, in: International Conference on Electronics Computer Technology, Kanyakumari, India, 2012.
[35] F. Herrera, M. Lozano, Adaptation of genetic algorithm parameters based on fuzzy logic controllers, Genetic Algorithms and Soft Computing 8 (1996) 95–125.
[36] F. Herrera, M. Lozano, Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions, Soft Computing 7 (8) (2003) 545–562.
[37] S. Huband, L. Barone, L. While, P. Hingston, A Scalable Multi-objective Test Problem Toolkit, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 280–295.
[38] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 110–125.
[39] H. Jain, K. Deb, Parent to mean-centric self-adaptation in sbx operator for real-parameter optimization, Swarm, Evolutionary, and Memetic Computing (2011) 299–306.
[40] T. Jansen, On the analysis of dynamic restart strategies for evolutionary algorithms, in: International conference on parallel

problem solving from nature, Springer, 2002, pp. 33–43.
[41] V. K. Koumousis, C. P. Katsaras, A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance, IEEE Transactions on Evolutionary Computation 10 (1) (2006) 19–28.
[42] A. Kumar, R. K. Misra, D. Singh, Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, 2017, pp. 1835–1842.
[43] S.-H. Liu, M. Mernik, B. R. Bryant, To explore or to exploit: An entropy-driven approach for evolutionary algorithms, International Journal of Knowledge-based and Intelligent Engineering Systems 13 (3-4) (2009) 185–206.
[44] F. Lobo, C. F. Lima, Z. Michalewicz, Parameter setting in evolutionary algorithms, vol. 54, Springer Science & Business Media, 2007.
[45] M. Lozano, F. Herrera, J. R. Cano, Replacement strategies to preserve useful diversity in steady-state genetic algorithms, Information Sciences 178 (23) (2008) 4421–4433.
[46] S. W. Mahfoud, Crowding and preselection revisited, Urbana 51 (1992) 61801.
[47] O. J. Mengshoel, S. F. Galán, A. De Dios, Adaptive generalized crowding for genetic algorithms, Information Sciences 258 (2014) 140–159.
[48] D. Molina, F. Moreno-García, F. Herrera, Analysis among winners of different ieee cec competitions on real-parameters optimization: Is there always improvement?, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, 2017, pp. 805–812.
[49] J. Montgomery, Differential evolution: Difference vectors and movement in solution space, in: Evolutionary Computation, 2009. CEC'09. IEEE Congress on, IEEE, 2009, pp. 2833–2840.
[50] J. Montgomery, S. Chen, An analysis of the operation of differential evolution at high and low crossover rates, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE, 2010, pp. 1–8.
[51] J. Montgomery, S. Chen, A simple strategy for maintaining diversity and reducing crowding in differential evolution, in: Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE, 2012, pp. 1–8.
[52] J.-B. Mouret, Novelty-based multiobjectivization, in: New horizons in evolutionary robotics, Springer, 2011, pp. 139–154.
[53] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm i. continuous parameter optimization, Evolutionary computation 1 (1) (1993) 25–49.
[54] I. Ono, H. Kita, S. Kobayashi, Advances in evolutionary computing, chap. A Real-coded Genetic Algorithm Using the Unimodal Normal Distribution Crossover, Springer-Verlag New York, Inc., New York, NY, USA, 2003, pp. 213–237.
URL http://dl.acm.org/citation.cfm?id=903758.903767
[55] H. M. Pandey, A. Chaudhary, D. Mehrotra, A comparative review of approaches to prevent premature convergence in ga, Applied Soft Computing 24 (2014) 1047–1077.
[56] A. Pétrowski, A clearing procedure as a niching method for genetic algorithms, in: Evolutionary Computation, 1996., Proceedings of IEEE International Conference on, IEEE, 1996, pp. 798–803.
[57] M. Pilát, Evolutionary multiobjective optimization: A short survey of the state-of-the-art, Proceedings of the Contributed Papers Part I-Mathematics and Computer Sciences, WDS, Prague, Czech (2010) 1–4.
[58] K. Price, R. M. Storn, J. A. Lampinen, Differential evolution: a practical approach to global optimization, Springer Science & Business Media, 2006.
[59] Â. A. Sá, A. O. Andrade, A. B. Soares, S. J. Nasuto, Exploration vs. exploitation in differential evolution, in: AISB 2008 Convention Communication, Interaction and Social Intelligence, vol. 1, 2008, p. 57.
[60] E. Segredo, C. Segura, C. León, A multiobjectivised memetic algorithm for the frequency assignment problem, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, IEEE, 2011, pp. 1132–1139.
[61] C. Segura, C. A. C. Coello, G. Miranda, C. León, Using multi-objective evolutionary algorithms for single-objective optimization, 4OR 11 (3) (2013) 201–228.
[62] C. Segura, C. A. C. Coello, E. Segredo, A. H. Aguirre, A novel diversity-based replacement strategy for evolutionary algorithms, IEEE transactions on cybernetics 46 (12) (2016) 3233–3246.
[63] A. G. Slim Bechikh, Rituparna Datta, Recent Advances in Evolutionary Multi-objective Optimization, springer.
[64] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. of Global Optimization 11 (4) (1997) 341–359.
URL http://dx.doi.org/10.1023/A:1008202821328
[65] T. Tušar, B. Filipič, Differential evolution versus genetic algorithms in multiobjective optimization, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2007, pp. 257–271.
[66] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, ACM Computing Surveys 45 (3) (2013) 35:1–35:33.
[67] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and Exploitation in Evolutionary Algorithms: A Survey, ACM Computing Surveys 45 (3) (2013) 35:1–35:33.
[68] T. Vidal, T. G. Crainic, M. Gendreau, C. Prins, A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows, Computers & operations research 40 (1) (2013) 475–489.
[69] H.-M. Voigt, H. Mühlenbein, D. Cvetkovic, Fuzzy recombination for the breeder genetic algorithm, in: Proc. Sixth Int. Conf. on Genetic Algorithms, Citeseer, 1995.
[70] M. Yang, C. Li, Z. Cai, J. Guan, Differential evolution with auto-enhanced population diversity, IEEE transactions on cybernetics 45 (2) (2015) 302–315.
[71] W.-J. Yu, M. Shen, W.-N. Chen, Z.-H. Zhan, Y.-J. Gong, Y. Lin, O. Liu, J. Zhang, Differential evolution with two-level parameter adaptation, IEEE Transactions on Cybernetics 44 (7) (2014) 1080–1099.
[72] S. Y. Yuen, C. K. Chow, A genetic algorithm that adaptively mutates and never revisits, IEEE transactions on evolutionary computation 13 (2) (2009) 454–472.

[73] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: Proc. of MENDEL, vol. 9, 2003, pp. 41–46.
[74] Q. Zhang, H. Li, MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.
[75] Q. Zhang, W. Liu, H. Li, The performance of a new version of moea/d on cec09 unconstrained mop test instances, in: Evolutionary Computation, 2009. CEC'09. IEEE Congress on, IEEE, 2009, pp. 203–208.
[76] L. Zhao, C.-j. Sun, X.-c. Huang, B.-x. Zhou, Differential evolution with strategy of improved population diversity, in: Control Conference (CCC), 2016 35th Chinese, IEEE, 2016, pp. 2784–2787.
[77] Q. Zhu, Q. Lin, Z. Du, Z. Liang, W. Wang, Z. Zhu, J. Chen, P. Huang, Z. Ming, A novel adaptive hybrid crossover operator for multiobjective evolutionary algorithm , Information Sciences 345 (2016) 177 – 198.
[78] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications, vol. 63, Citeseer, 1999.
[79] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE transactions on Evolutionary Computation 3 (4) (1999) 257–271.