Un algoritmo inspirado en la naturaleza para resolver problemas de optimización numérica restringida con alta dimensionalidad

---

Adán Enrique Aguilar Justo

Enero 31, 2019

Version:

Universidad Veracruzana

Centro de Investigación en Inteligencia Artificial

Un algoritmo inspirado en la naturaleza para resolver problemas de optimización numérica restringida con alta dimensionalidad

Adán Enrique Aguilar Justo

Director PhD. Efrén Mezura Montes

Enero 31, 2019

Adán Enrique Aguilar Justo
Un algoritmo inspirado en la naturaleza para resolver problemas de optimización numérica restringida con alta dimensionalidad
, Enero 31, 2019
Universidad Veracruzana

Centro de Investigación en Inteligencia Artificial
91000 , Xalapa, Veracruz México

Resumen

Los problemas numéricos actualmente tienden a crecer en la cantidad de características que describen al problema, llevando a un incremento en la dimensión del problema. Cuando la dimensionalidad crece aparecen nuevos problemas que entorpecen la labor de optimizar dichos problemas. El espacio de búsqueda crece exponencialmente, además, las características del espacio de búsqueda cambian y la interacción y dependencia de las variables se ve incrementada. Estos problemas son conocidos en la literatura como la maldición de la dimensionalidad. Los algoritmos evolutivos han mostrado ser capaces de resolver problemas de optimización numérica de manera satisfactoria, sin embargo dicha maldición de la dimensionalidad también afecta su rendimiento, y cuando el problema cuenta con restricciones, la interacción de variables incrementa aún más. Para contender con estos problemas un esquema de Co-evolución cooperativa se ha estudiado a fondo y junto con métodos que descomponen el problema han mostrado eficacia en problemas sin restricciones.

En este trabajo se propone utilizar la descomposición de problemas bajo un enfoque memético que permita guiar la búsqueda local bajo un esquema de cooperación. También se exploran los métodos de descomposición enfocados a problemas con restricciones y se proponen mejora que incrementen su rendimiento.

Los algoritmos propuestos son probados utilizando un conjunto de funciones de prueba con restricciones que trabajan en las dimensiones 100, 500, y 1000. La propuesta se compara contra algoritmos del estado del arte basados en co-evolución cooperativa. Los resultados revelan que la propuesta obtiene un mejor rendimiento en el conjunto de funciones de prueba respecto a los otros algoritmos, encontrando resultados numéricos competitivos. Respecto a su comportamiento de convergencia, muestra una mayor velocidad de

convergencia sin esto significar que el algoritmo quede estancado durante la búsqueda.

Este trabajo abre el área de oportunidad de los algoritmos evolutivos para alta dimensionalidad, al exponer que la descomposición es viable para utilizarse fuere del contexto canónico de la co-evolución cooperativa.

Agradecimientos

> *We are a way for the cosmos to know itself*
>
> **— Carl Sagan**

- Agradezco a mi asesor el Dr. Efrén Mezura Montes, todo su apoyo, conocimientos y compromiso con el proyecto que fueron parte fundamental para que éste trabajo llegara a su término en tiempo y forma.

- Agradezco a todos mis compañeros de la Maestría y del Doctorado en Inteligencia Artificial quienes me aportaron comentarios y criticas valiosas que me ayudaron a mejorar mi trabajo.

- En especial agradezco a mis padres, que siguen siendo un soporte fundamental en mi vida, quienes me han dado las mejores enseñanzas de la vida.

- Agradezco a el CONACYT por el apoyo económico brindado durante la realización de este proyecto.

- Agradezco a la Universidad Veracruzana, esta institución de enorme calidad, que me brindó todo el apoyo durante mi estancia.

Índice general

1 Introducción

> *If knowledge can create problems, it is not through ignorance that we can solve them.*
>
> **— Isaac Asimov**

Hoy en día, diferentes problemas de optimización tienen un número considerablemente alto de variables de decisión, dichos problemas son catalogados como problemas de alta dimensión. Estos problemas suponen un reto importante para los algoritmos de cómputo evolutivo, ya que el espacio de búsqueda crece de manera exponencial, las propiedades de las funciones pueden cambiar conforme la dimensión crece y la evaluación de estos problemas suele ser costosa por la cantidad de variables, además de incrementar la interacción entre las variables. Todas estas características son conocidas en la literatura especializada como "la maldicion de la dimensionalidad" [32, 34, 19]. Por estas razones, cuando los algoritmos evolutivos (EA's por sus siglas en inglés) tratan de resolver este tipo de problemas, su rendimiento se ve afectado.

En la búsqueda por mejorar el rendimiento de los EA's, la estrategia de *divide y vencerás* ha sido adoptada para lidiar con el problema de la separabilidad de variables, y reducir la complejidad del problema. Un primer acercamiento se dio por Potter y De Jong [34], quienes introdujeron el concepto de algoritmos de Co-evolución Cooperativa (CC). Aunque las primeras versiones asumían que los problemas eran completamente separables, rápidamente se desarrollaron estrategias para buscar la interacción de las variables. Estos métodos son conocidos como métodos de descomposición de problemas, y actualmente son la base de los algoritmos CC.

Enfoques de Co-evolución Cooperativa

En la literatura especializada los algoritmos para resolver problemas de alta dimensionalidad se desarrollan bajo el esquema de Co-evolución Cooperativa, y generalmente se enfocan en el proceso de descomponer el problema en

sub-problemas. Por ejemplo, Random grouping propuesto por Yang et al. [45, 18], como su nombre lo indica la descomposición se da de manera, aleatoria y en cada iteración del algoritmo la descomposición se continúa ajustando de manera aleatoria. Yao [35] propone una estrategia de descomposición basada en la correlación de cambio de las variables. Delta Grouping introducido por Omidvar [33], el cual mide el promedio de cambio en un par de variables para determinar si éstas interactúan. Variable Interaction Learning propuesto por Chen et al. [5], asume que todas las variables son independientes, con las iteraciones descubre cuales variables interactúan y las va combinando en grupos. Dependency Identification (DI) fue desarrollado por Sayed [36], en este método se propone una forma de evaluar las configuraciones de sub-grupos. La estrategia que ha mostrado resolver el problema de descomposición con precisión se llama Differential Grouping propuesto Omnidvar [30] y más recientemente propuso la version 2 [31] que reduce el número de evaluaciones requeridas para descomponer el problema.

Otros algoritmos basados en CC

Si bien muchos de los trabajos se han enfocado es la descomposición del problema, otros han buscado mejorar el rendimiento de los algoritmos proponiendo diseños de operadores o hibridación de estrategias. Shi et al. [39] propusieron un algoritmo de Evolución Diferencial en el cual utilizaron múltiples estrategias de mutación, su algoritmo lo probaron en el esquema de CC, con diferentes métodos de descomposición. EL agoritmo MUEDA como optimizador local y que es usando como segunda fase de optimización fue propuesto por Zhang et al. [47]. Cao et al. [4], proponen un algoritmo memético con SaNSDE como búsqueda global y el método de Solis y West como búsqueda local. Ellos probaron el algoritmo en un CC. Una mutación con aprendizaje para la Evolución Diferencial fue propuesto por Ma and Ding [23], probado en el esquema de CC y utilizando differential grouping para descomponer el problema.

Enfoques basados en Algoritmos Meméticos

Existen otras meta-heurísticas que no utilizan un esquema de CC y que resuelven problemas de alta dimensión. Un tipo de estos algoritmos son los

algoritmos meméticos (MA's), que se caracterizan por tener dos fases de búsqueda, la global y la local, algunos de estos algoritmos son:

Hong-qi Li and Li [17] diseña un MA con Optimización por Cúmulo de Partículas (PSO) como búsqueda global y el algoritmo de Harmony Search que se encarga de actualizar las partículas. Existen otros MA's basados en PSO, como el desarrollado por Zhao et al. [48], que mejora a las partículas mediante el método de Quasi-Newton. Muelas et al. proponen [27] un algoritmo de Evolución Diferencial y el método de Múltiple trayectoria (MST) como búsqueda local. Molina et al. [25] presentan un algoritmo memético basado en la search chains, y LaTorre et al. [16] desarrollaron un algoritmo basado en el método de muestreo multiple de descendencia (Multiple Offspring Sampling).

Los resultados de estos algoritmos han sido competitivos en algunos casos han superado el desempeño de aquellos basados en CC.

Algoritmos para resolver problemas restringidos

En la literatura se encuentra que la mayoría de los esfuerzos están enfocados a resolver problemas sin restricciones, o el desarrollo de métodos de descomposición que permitan mejorar el rendimiento de los algoritmos de co-evolución cooperativa. Sin embargo algoritmos que permitan resolver problemas restringidos son muy pocos. Del lado de los algoritmos de descomposición se puede encontrar Variable Interaction Identification for Constrained Problems (VIIC) [37], que es el primer método de descomposición enfocado en problemas con restricciones, y es una extensión del algoritmo Dependency Identification (DI) [36]. Y hablando de algoritmos específicos para este problema, recientemente se han desarrollado propuestas basadas en PSO y basados en el esquema de Cooperación Co-evolutiva y utilizando Differential Grouping y Random Grouping como métodos de descomposición.

Las técnicas de descomposición se han probado mayormente bajo el esquema de CC. Lui et al. [21] concluye en su trabajo que es necesario un estudio de cómo mejorar el diseño eficiente de estrategias de cooperación con las estrategias de agrupación de variables (descomposición). Y recientemente, el uso de la información de la agrupación de variables ha sido utilizado en

otras fases de búsqueda de los algoritmos evolutivos. Como en el trabajo de Zille [49] que resuelve problemas de alta dimensión multi-objetivo, y donde propone diferentes operadores de mutación que utilizan la descomposición para mejorar el rendimiento del algoritmo NSGA-II.

Por lo tanto, basados en la literatura donde los MA's han sido estudiados solos o bajo el esquema de CC, y motivados por el desarrollo de operadores especiales que utilizan la descomposición, en este trabajo se propone combinar las capacidades de búsqueda local de los algoritmos meméticos utilizando la descomposición en esta fase del algoritmo, con la finalidad de resolver problemas de alta dimensión con restricciones. Esto implica que el buscador global trabajará sobre la dimensión completa del problema y sólo cuando la búsqueda local sea requerida, la descomposición del problema se llevara a cabo. A este esquema se le ha llamado Búsqueda Cooperativa Local o LoCoS por sus siglas en inglés (Local Cooperative Search).

1.1 Problema

Un problema de optimización mono-objetivo puede definirse como: encontrar un vector $\mathbf{x}$ de dimensión D que minimice una función $f(\mathbf{x})$ sujeto a restricciones de desigualdad $g(\mathbf{x})$ y/o de igualdad $h(\mathbf{x})$. Ecuación 1.1.

$$\begin{aligned} \text{Minimizar } f(\mathbf{x}), \quad & \mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D \\ & l_i \leq x_i \leq u_i, i = 1, \ldots, D \\ \text{sujeto a:} \quad & \\ & g_j(\mathbf{x}) \leq 0, \text{ for } j = 1, \ldots, q \\ & h_l(\mathbf{x}) = 0, \text{ for } l = 1, \ldots, m \end{aligned} \tag{1.1}$$

donde q y m refieren al número de restricciones de desigualdad e igualdad, respectivamente D es la dimensión del problema. El espacio de búsqueda $\mathbb{S}$ está definido por los límites inferiores $\mathbf{l}$ y superiores $\mathbf{u}$, mientras que la región factible es definida como el subconjunto de soluciones que satisfacen las restricciones del problema $\mathbb{F} \subseteq \mathbb{S}$.

1.2 Hipótesis

La hipótesis principal que se busca responder en este trabajo es la siguiente:

Combinar un algoritmo memético con un método de descomposición durante la fase de búsqueda local, puede mejorar el rendimiento con base en resultados finales de los algoritmos del estado del arte, resolviendo problemas de alta dimensión con restricciones. Este algoritmo tiene las siguientes características:

- Presenta un compromiso entre la exploración por parte del buscador global y explotación mediante la descomposición del problema en el proceso de búsqueda local.
- Es capaz de mejorar los resultados de los algoritmos de estado del arte.
- El rendimiento del algoritmo no se ve afectado por el crecimiento de la dimensión.

1.3 Objetivos

1.3.1 Objetivo general

El objetivo principal de este trabajo es avanzar el *estado del arte* en optimización de problemas de alta dimensión con restricciones, mediante el diseño de un algoritmo memético en el que se combinen las ventajas de los métodos de descomposición usados en algoritmos de cooperación co-evolutiva, dentro de la fase de búsqueda local.

1.3.2 Objetivos específicos

Los objetivos específicos son los siguientes:

- Analizar el estado-del-arte respecto a la optimización de problema de alta dimensión, como se resuelven y cuáles son sus ventajas y desventajas.

- Diseñar un algoritmo memético general como punto de partida.
- Estudiar los diferente métodos de descomposición usados para problemas de alta dimensión con y sin restricciones.
- Seleccionar los métodos de descomposición más relevantes del estado del arte.
- Realizar el diseño experimental para evaluar el rendimiento de la propuesta.
- Llevar a cabo el estudio comparativo contra los algoritmos del estado-del-arte.

1.4 Contribuciones esperadas

- Un nuevo algoritmo evolutivo que resuelva problemas de optimización de alta dimensión.
- Un nuevo esquema de diseño de algoritmos, basado en algoritmos meméticos y métodos de descomposición.
- El análisis empírico del esquema propuesto, evaluado en un conjunto de funciones de prueba de alta dimensión con restricciones.

1.5 Publicaciones

A continuación se presentan los productos obtenidos durante el desarrollo de este trabajo de tesis.

1.5.1 Publicaciones en Revista

- A. E. Aguilar-Justo, E. Mezura-Montes, **A Local Cooperative Approach to Solve Large-Scale Constrained Optimization Problems**, Swarm and Evolutionary Computation. (Sometido en revisión 1)

1.5.2 Publicaciones en Congresos

- A. E. Aguilar-Justo, E. Mezura-Montes and Saber M. Elsayed and Ruhul A. Sarker, **Decomposition of Large-scale Constrained Problems Using a Genetic-based Search** 2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2016). Ixtapa, Mexico

- A. E. Aguilar-Justo and E. Mezura-Montes, **Towards an improvement of variable interaction identification for large-scale constrained problems** 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp. 4167-4174.

1.6 Estructura del documento

Capítulo 2

En este capítulo se introducen los conceptos de optimización, como: (1) la definición de un problema de optimización, (2) concepto de óptimo local y (3) concepto de óptimo global. Además se presenta un conjunto de métodos de optimización multivariable, los cuales se dividen en dos grupos, aquellos utilizan únicamente la información del valor de la función objetivo llamados métodos directos, así como métodos basados en gradiente, que explotan la información de las derivadas del problema.

Capítulo 3

Introduce al cómputo de evolutivo, se exponen las principales características del proceso de la evolución y se describen brevemente los primeros paradigmas dentro del área y cómo es que éstos toman parte de esa inspiración en la evolución de las especies. Dentro de los algoritmos evolutivos existen tres principales enfoques, (1) la programación evolutiva, (2) las estrategias evolutivas, y (3) los algoritmos genéticos.

Capítulo 4

El principal enfoque de cómputo evolutivo que es utilizado para resolver problemas de alta dimensión son los algoritmos de Co-evolución Cooperativa (CC). Estos algoritmos constan de tres partes bien definidas en la literatura, las cuales son: (1) descomponer el problema, (2) Optimizar cada uno de los

sub-grupos o sub-problemas con un algoritmo evolutivo, y (3) realizar un proceso de cooperación entre todos los sub-grupos para crear la siguiente generación de soluciones. Del mismo modo, se introducen tres de los principales métodos para descomponer un problema de acuerdo a la interacción de las variables, estos son: (1) Agrupación aleatoria (Random Grouping), (2) Agrupación por diferencias versión 2 (Differencial Grouping), e (3) Identificación de Interacción de Variables para problemas con restricciones (VIIC).

Capítulo 5

En este capítulo se presenta la propuesta del algoritmos para resolver problemas de alta dimensión con restricciones, basado en un algoritmo memético en el cual se agrega el proceso de descomposición dentro de la fase de búsqueda local. Además, se exponen los cambios propuestos al método de descomposición VIIC, que tienen como finalidad mejorar su rendimiento. Todo ésto llevando a cabo bajo la experimentación de los algoritmos en un conjunto de funciones de prueba de alta dimensión con restricciones, el cual hasta el momento es el primero de su categoría. Se presenta también el análisis de los resultados utilizando pruebas estadísticas y un método de comparación empírica para determinar las capacidades de los algoritmos propuesto con respecto a algoritmos del estado del arte.

Capítulo 6

En este capítulo se proponen mejoras al algoritmo de descomposición VIIC que es el primero en su clase enfocado a problemas de alta dimensionalidad con restricciones. VIIC es un algoritmo de búsqueda de arreglo de variables, sin embargo, entre sus desventajas está que la búsqueda la realiza de una manera voraz y aleatoria. Las mejoras estan enfocadas a realizar una búsqueda guiada por una metaheurística.

Capítulo 7

El capítulo presenta una extension de la experimentación del algoritmo propuesto en el Capítulo 5 junto con las mejoras al algoritmo VIIC expuestas en el Capítulo anterior. Además se compara el rendimiento contra algoritmos de estado del arte.

Capítulo 8

En este capítulo se exponen las conclusiones a las que se llegaron durante el desarrollo de este trabajo.

2

Antecedente de Optimización

> *To every action there is always opposed an equal reaction.*
>
> **— Isaac Newton**

En este capítulo se presenta un panorama general de la optimización desde la perspectiva clásica. Presentando los conceptos generales y teóricos en los que se fundamenta este trabajo. En la siguiente sección se describen los conceptos generales de optimización, seguido de diferentes métodos clásicos que resuelven estos problemas. Si bien existen métodos enfocados a problemas de una sola variable, en este trabajo nos enfocaremos a aquellos que buscan optimizar problemas multivariable.

2.1 Conceptos de Optimización

Hablando de manera general, optimizar es el proceso de encontrar la solución máxima ó mínima de un problema. Los problemas están definidos por una función objetivo también conocida como función de aptitud o costo. Éstos pueden estar sujetos a un conjunto de restricciones; una solución es un conjunto de variables de decisión que determinan el valor de la función objetivo.

Cuando un problema es restringido, las soluciones que se buscan son aquellas que satisfacen todo el conjunto de restricciones y estas forman parte de la región factible.

Antes de entrar al detalle de los métodos de optimización es necesario definir que es un valor óptimo y los diferentes tipos de óptimos que existen.

Definición 1 ***Óptimo local:*** *Se dice que un punto* $\mathbf{x}'$ *es un óptimo local, si no existe ningún otro punto en el* ***vecindario*** *de* $\mathbf{x}'$ *que sea mejor. Hablando de minimización, se puede expresar como sigue:*

$$f(\mathbf{x}') < f(\mathbf{x}) \;|||\; \mathbf{x} - \mathbf{x}' \;||< \epsilon \tag{2.1}$$

donde ϵ *representa el vecindario de* $\mathbf{x}'$.

Definición 2 ***Óptimo global:*** *Un punto* $\mathbf{x}^*$ *se dice que es un óptimo global, si y solo si, no existe otro punto en* ***todo el espacio de búsqueda*** $\mathbb{S}$ *que sea mejor que* $\mathbf{x}^*$.[1] *Esto es:*

$$\forall \mathbf{x} \in \mathbb{S} : f(\mathbf{x}^*) \leq f(\mathbf{x}) \tag{2.2}$$

Estos conceptos pueden verse gráficamente en la figura 2.1.

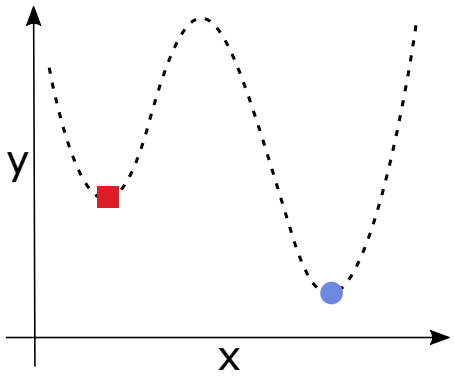


Figura 2.1.: Representación de diferentes puntos y su optimalidad para una función. El cuadrado rojo representa un óptimo local. El círculo azul representa el óptimo global para esta función.

2.2 Métodos directos

En esta sección se presentan algunos métodos de minimización que no requieren la información del gradiente de la función para optimizar. Es decir, únicamente utilizan la información de los valores de la función, por ello se les conoce también como métodos directos [7]. Es importante mencionar que si se cuenta con la información del gradiente, un algoritmo basado en gradiente puede ser más eficiente que un método directo. Sin embargo, en

[1] Un óptimo global no necesariamente es único, esto es, puede existir más de un óptimo global en una función

muchos de los problemas reales es difícil/costoso calcular las derivadas, en este caso los métodos directos son de utilidad. A continuación se presentan dos métodos directos que han mostrado su eficacia en resolver problemas de optimización multi-variable.

2.2.1 Simplex

El método Simplex también conocido como Nelder-Mead, fue propuesto por Jonh A. Nelder y Roger Mead en 1965 [28]. Este método requiere $D + 1$ puntos iniciales para funcionar (D es el número de variables del problema). Dichos puntos forman el Simplex inicial y debe cumplir con la siguiente característica: el Simplex no debe formar un hipercubo de volumen cero. Esto es, hablando de un problema de dos dimensiones, el Simplex no debe ser una línea recta, en tres dimensiones no deberá formar un plano, y en más dimensiones no debe formar un hiperplano.

El proceso del método Simplex en cada iteración es el siguiente:

1. Primero se determina punto del Simplex con el peor valor en la función objetivo y se le llama $\mathbf{x_h}$.

2. Se calcula el centroide de los puntos sin contemplar al peor. El peor punto es *reflejado* respecto al centroide y se obtiene $\mathbf{x_r}$.

3. Si $\mathbf{x_r}$ tiene mejor valor objetivo que el mejor punto en el Simplex, se considera que la región es prometedora y por lo tanto se realiza una *expansión* en dirección del centroide hacia el punto reflejado. La cantidad de expansión es controlada por el parámetro $\gamma > 1$.

4. Si $\mathbf{x_r}$ es peor que el peor punto del Simplex, se considera una región mala, por lo tanto se realiza un movimiento de *contracción*. La cantidad a contraer es determinada por el parámetro $\beta < 0$.

5. Finalmente, si el punto reflejado es mejor que el peor en el simplex, pero peor que el segundo peor, se realiza una contracción pero con valor de $\beta > 0$ positivo.

El caso por defecto es el punto reflejado. El nuevo punto obtenido reemplaza al peor punto del Simplex. En la Figura 2.2 se muestran estos movimientos. El algoritmo a detalle se muestra en Algoritmo 2.1.

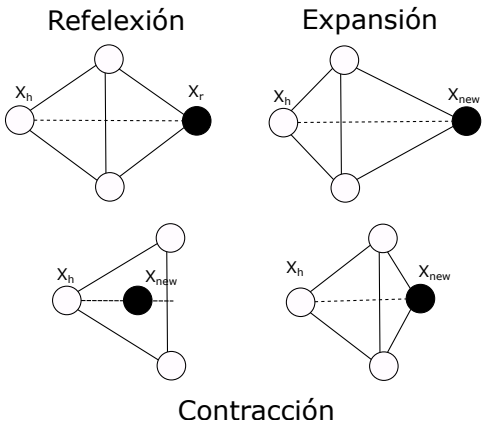


Figura 2.2.: Pasos del método Simplex. Primero un proceso de *Reflexión* del peor punto $\mathbf{x_h}$ respecto al centroide obteniendo $\mathbf{x_r}$. Dependiendo del valor objetivo de $\mathbf{x_r}$ se realiza un paso de *Expansión*, o alguno de los tipos de *Contracción*, la línea punteada indica el eje de movimiento de $\mathbf{x_{new}}$

Una forma de crear el Simplex inicial para cumplir la condición de no cero-hipervolumen, se describe a continuación: Dado un punto $\mathbf{x}_0$ y un factor de escala [2] α, $i, j = 1, 2, ..., N$

$$x_{i,j} = \begin{cases} x_{0,j} + \delta_1 & \text{Si } j = i \\ x_{0,j} + \delta_2 & \text{Si } j \neq i \end{cases} \tag{2.3}$$

donde

$$\delta_1 = \frac{\sqrt{N+1} + N - 1}{N\sqrt{2}}\alpha \tag{2.4}$$

$$\delta_2 = \frac{\sqrt{N+1} - 1}{N\sqrt{2}}\alpha \tag{2.5}$$

2.2.2 Hooke-Jeeves

El método de Hooke-Jeeves también conocido como método de búsqueda de patrón (pattern search method), fue desarrollado por R. Hooke y T. Jeeves in 1961 [13]. Este método crea direcciones de búsqueda de tal manera que se expande por todo el espacio de soluciones. En un problema D-dimensional se requieren por lo menos D direcciones de búsqueda lineales.

[2] Un $\alpha = 1$ crea un Simplex regular de lados de longitud unitaria

ALGORITMO 2.1 Algoritmo Simplex

Entrada: $\gamma > 1$, $\beta \in (0, 1)$, parámetro de paro ϵ, y un Simplex Inicial

$terminar =$ **falso**
mientras No $terminar$ **hacer**
  Encontrar el peor punto x_h, el mejor x_l y el segundo peor x_g
  Calcular centroide sin considerar a x_h, $x_c = \frac{\sum_{i=1, i\neq h}^{N+1} x_i}{N}$
  Calcular el punto reflejado $x_r = 2x_c - x_h$, Asignar $x_{new} = x_r$
  si $f(x_r) < f(x_l)$ **entonces**
    $x_{new} = (1 + \gamma)x_c - \gamma x_h$ (expansión)
  si no, si $f(x_r) \geq f(x_h)$ **entonces**
    $x_{new} = (1 - \beta)x_c + \beta x_h$ (contracción +)
  si no, si $f(x_g) < f(x_r) < f(x_l)$ **entonces**
    $x_{new} = (1 + \beta)x_c - \beta x_h$ (contracción -)
  fin si
  Calcular $f(x_{new})$, reemplazar x_h con x_{new}
  si $\sqrt{\sum_{i=1}^{N+1} \frac{(f(x_i) - f(x_c))^2}{N+1}} \leq \epsilon$ **entonces**
    $terminar =$ **cierto**
  fin si
fin mientras
devolver El mejor de los puntos en el Simplex

El algoritmo de Hooke-Jeeves es una combinación de movimientos de exploración realizados a través de cada una de las dimensiones del problema. Y un movimiento heurístico llamado movimiento de patrón.

El algoritmo requiere de tres parámetros definidos por el usuario: 1) un vector de paso Δ, 2) un factor de reducción de paso α y 3) un criterio de parada ϵ que está dado por la norma del vector de paso $\| \Delta \|$. Los movimientos que realiza el algoritmo se describen a continuación:

1. Movimiento de exploración: Cada una de las variables del punto actual, es perturbada en direcciones positivas y negativas, una a la vez. El mejor punto entre todos los puntos incluyendo la solución actual, es seleccionado como el punto actual.

   Finalmente se dice que el movimiento de exploración fue exitoso si y solo si, el punto actual pertenece a uno de aquellos puntos modificados.(véase el Algoritmo 2.2) , donde $\mathbf{x}_i$ es el punto en la iteración i.

Algoritmo 2.2 Movimiento de exploración

```
1: x = x_i
2: para j = 1 a D hacer
3:    Calcular f(x), f+(x_j + Δ_j), f−(x_j − Δ_j)
4:    x = elMejorde(f, f+, f−)
5: fin para
6: si x ≠ x_i entonces
7:    x_i = x
8:    return Éxito
9: si no
10:   return Fallo
11: fin si
```

2. Movimiento de patrón: Este movimiento es aplicado al punto $\mathbf{x}_i$ solamente si el movimiento de exploración fue **Exitoso**. El movimiento de patrón encuentra un nuevo punto $\mathbf{x}_{i+1}$ utilizando el punto anterior a la solución actual ($\mathbf{x}_{i-1}$) y el punto encontrado por el movimiento de exploración $\mathbf{x}_i$, usando la siguiente ecuación:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{x}_i - \mathbf{x}_{i-1}) \tag{2.6}$$

El algoritmo completo está descrito en Algoritmo 2.3.

Algoritmo 2.3 Algoritmo Hooke-Jeeves

Entrada: $\mathbf{x}_i$ como punto inicial

```
1: mientras ||Δ|| > ε hacer
2:    Realizar un movimiento de exploración basado en x_i (ver Algoritmo 2.2)
3:    si Movimiento de exploración es Exitoso entonces
4:       Realizar movimiento de patrón x_{i+1} = x_i + (x_i − x_{i−1})
5:    si no
6:       Δ = Δ/α
7:    fin si
8: fin mientras
```

Este método es realmente simple, únicamente requiere almacenar la información del punto anterior y actual para funcionar. Sin embargo, puede llevar a una rápida convergencia si el problema es altamente no lineal y con mucha interacción entre sus variables. A su vez, requiere realizar muchas evaluaciones para llegar a soluciones de buena calidad. La velocidad de convergencia recae en el parámetro de reducción de paso α, al cual se le recomienda un valor de

$\alpha = 2$. Mientras que el vector de tamaños de paso se sugiere inicializarlo a $\Delta = 0{,}5$. Y un valor adecuado de condición de término es $\epsilon = 1 \times 10^{-4}$.

2.3 Métodos basado en gradiente

Los siguientes métodos aprovechan la información de la derivada de la función, haciéndolos muy rápidos en la búsqueda. A diferencia de los métodos vistos en la sección anterior, estos métodos gastan menos evaluaciones de la función objetivo. Sin embargo, no siempre son los más adecuados a emplear, ya que muchos de los problemas de optimización no son diferenciables, ya que poseen características como que la función objetivo es discontinua o discreta. Por otro lado, estos métodos basados en gradiente son los más utilizados en el diseño de problemas de ingeniería.

2.3.1 Método de Cauchy (Paso descendiente)

Cauchy utiliza como dirección de búsqueda el gradiente negativo de un punto en particular $\mathbf{x}_t$: $\mathbf{s}_t = -\nabla f(\mathbf{x}_t)$. Esta dirección provee el máximo descenso en los valores de la función. El método de Cauchy también es conocido como paso descendente.

En cada iteración del algoritmo, la derivada del punto actual es calculada y con la ayuda de una búsqueda unidireccional sobre la derivada negativa, se encuentra el valor mínimo de esta dirección. Este punto mínimo se convierte en el punto actual y se continúa generando las derivadas hasta se tiene un vector gradiente lo suficientemente pequeño (criterio de paro).

Este método garantiza mejorar el punto en cada iteración. Una desventaja es que cuando el punto inicial $\mathbf{x}_t$ se encuentra muy cerca del óptimo, el cambio en el vector gradiente es muy pequeño y la búsqueda pierde efectividad. Otra característica, que si bien no necesariamente es una desventaja, es que requiere de un algoritmo que le permita realizar una búsqueda unidireccional[3] para determinar el siguiente punto mínimo de acuerdo al gradiente. El detalle se puede ver en el Algoritmo 2.4.

[3] Algunos métodos unidireccionales son: Búsqueda Dorada, Intervalos por la Mitad, Búsqueda exhaustiva, entre otros

Algoritmo 2.4 Algoritmo Cauchy

Entrada: punto inicial $\mathbf{x}_k$, Número máximo de iteraciones M, dos parámetros de término ϵ_1 y ϵ_2, $k = 0$

1: Calcular $\nabla f(\mathbf{x}_k)$, la primera derivada del punto $\mathbf{x}_k$
2: **mientras** $||\nabla f(\mathbf{x}_k)|| > \epsilon_1$ Y $k < M$ **hacer**
3: Realizar la búsqueda unidireccional para encontrar el valor de α_k utilizando el segundo parámetro de término ϵ_2. Minimizando la siguiente función $f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k))$.
4: Un posible criterio de paro es cuando $|\nabla f(\mathbf{x}_{k+1}) \times \nabla f(\mathbf{x}_k)| \leq \epsilon_2$
5: **si** $\frac{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}{\|\mathbf{x}_k\|} \leq \epsilon_1$ **entonces**
6: **Terminar**
7: **si no**
8: Calcular $\nabla f(\mathbf{x}_k)$ e incrementar $k = k + 1$
9: **fin si**
10: **fin mientras**

2.3.2 Método de Newton

El método de Newton utiliza la derivada de segundo orden para definir la dirección de búsqueda. Esto permite que la velocidad de convergencia hacia el mínimo sea aún más rápida. Por lo tanto la dirección de búsqueda va a estar expresada por la siguiente ecuación: $\mathbf{s}_k = -[\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$

Se debe mencionar que si la matriz $[\nabla^2 f(\mathbf{x}_k)]^{-1}$ es positiva semi-definida, entonces la dirección $\mathbf{s_k}$ debe ser descendente, pero si ésta no es positiva semi-definida podría o no ser una dirección descendente. Por lo tanto la dirección de búsqueda no garantiza que el valor objetivo disminuya. De hecho el método de Newton funciona mejor si el punto de inicio está cerca del óptimo.

Por otro lado, la condición de optimalidad de la derivada de segundo orden dice que en la vecindad del valor mínimo siempre sera una matriz positiva semi-definida, por lo que se garantiza el descenso del valor de la función. Si bien el valor de la función no siempre disminuiría, esto se puede resolver reiniciando el algoritmo usando otros puntos de inicio.

El proceso del algoritmo es similar al visto en el método de Cauchy, solo cambia el cálculo de la dirección, ahora considerando la segunda derivada, y al igual que Cauchy este método requiere de un proceso de búsqueda unidireccional. En el Algoritmo 2.5 se presenta el proceso completo en seudocodigo.

Algoritmo 2.5 Algoritmo de Newton

Entrada: punto inicial $\mathbf{x}_k$, Número máximo de iteraciones M, un parámetro de término ϵ, $k = 0$ y $\lambda_0 = 10e + 4$ un valor muy grande

1: Calcular $\nabla f(\mathbf{x}_k)$, la primera derivada del punto $\mathbf{x}_k$
2: **mientras** $||\nabla f(\mathbf{x}_k)|| > \epsilon_1$ Y $k < M$ **hacer**
3: Realizar la búsqueda unidireccional para encontrar el valor de α_k utilizando el segundo parámetro de término ϵ_2. Minimizando la siguiente función $f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k - \alpha_k[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k))$.
4: Un posible criterio de paro es cuando $|\nabla f(\mathbf{x}_{k+1}) \times \nabla f(\mathbf{x}_k)| \leq \epsilon_2$
5: **si** $\frac{\|\mathbf{x}_{k+1}-\mathbf{x}_k\|}{\|\mathbf{x}_k\|} \leq \epsilon_1$ **entonces**
6: **Terminar**
7: **si no**
8: Calcular $\nabla f(\mathbf{x}_k)$ e incrementar $k = k + 1$
9: **fin si**
10: **fin mientras**

2.3.3 Método de Marquardt

Como se dijo anteriormente el método de Cauchy es eficiente cuando el punto inicial está lejos del mínimo y el método de Newton tiene un mejor rendimiento al acercarse al óptimo. Sin embargo al enfrentarse a un problema de optimización es común no conocer donde se encuentra el valor mínimo y por consiguiente, elegir un punto inicial puede devenir en un nuevo problema.

Con la finalidad de combinar ambas características de los métodos antes definidos, el método de Marquardt inicia el proceso de búsqueda como un método de Cauchy, y posteriormente convirtiendo la búsqueda en un proceso de Newton. Esto lo hace a través de un proceso adaptativo que depende de los valores obtenidos durante la búsqueda.

El método requiere de un parámetro λ que controla el comportamiento del algoritmo, iniciando con un valor muy grande y reduciendo conforme pasan las iteraciones. Además del cálculo de la matriz Hessiana y su inversa. Lo cual lo lleva a tener un costo computacional grande. El algoritmo puede verse en el Algoritmo 2.6.

Algoritmo 2.6 Algoritmo de Marquardt

Entrada: punto inicial $\mathbf{x}_k$, Número máximo de iteraciones M, dos parámetros de término ϵ_1 y ϵ_2, $k = 0$

1: Calcular $\nabla f(\mathbf{x}_k)$, la primera derivada del punto $\mathbf{x}_k$
2: **mientras** $||\nabla f(\mathbf{x}_k)|| > \epsilon_1$ Y $k < M$ **hacer**
3: Calcular $s(\mathbf{x}_k) = -[H_k + \lambda_k I]^{-1} \nabla f(\mathbf{x}_k))$.
4: asignar $\mathbf{x}_{k+1} = \mathbf{x}_k + s(\mathbf{x}_k)$
5: **si** $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ **entonces**
6: $\lambda_{k+1} = 1/2\lambda_k$
7: **si no**
8: $\lambda_{k+1} = 2\lambda_k$
9: **fin si**
10: **fin mientras**

2.4 Métodos para problemas restringidos

Los métodos anteriormente listados trabajan sobre un espacio de búsqueda que no tiene restricciones. A continuación se describirán un par de métodos que ya están diseñados para trabajar con problemas restringidos.

2.4.1 Penalización de funciones

Este método es parte de aquellos basados en transformación, que son simples y populares para el manejo de restricciones [7]. El método de penalización de funciones consiste en transformar el problema restringido en un conjunto de problemas sin restricciones, agregando una penalización a cada una de las restricciones violada. En este método es necesario utilizar alguno de los métodos de optimización mencionados en las secciones anteriores.

El método trabaja de la siguiente forma, en cada iteración se modifican los parámetros de penalización e inicia otro proceso de búsqueda con el punto mínimo encontrado por la configuración anterior. El problema a resolver se encuentra en la siguiente Ecuación 2.7:

$$P(\mathbf{x}, \mathbf{R}) = f(\mathbf{x}) + \Omega(\mathbf{R}, g(\mathbf{x}), h(\mathbf{x})) \tag{2.7}$$

donde $\mathbf{R}$ es un conjunto de parámetros de penalización y Ω es el término de penalización para favorecer la elección de puntos factibles sobre los no-

factibles. Existen diferentes términos de penalización dependiendo del tipo de restricción, algunos son:

- **Penalización parabólica:** es utilizada para restricciones de desigualdad. Cualquier punto no factible es penalizado por la cantidad violada elevada al cuadrado. El parámetro $\mathbf{R}$ es inicialmente pequeño y secuencialmente se va incrementando.

$$\Omega = \mathbf{R}\{h(\mathbf{x})\}^2 \tag{2.8}$$

- **Penalización de barrera infinita:** usada para restricciones de desigualdad. $\mathbf{R}$ es un valor grande y J es el conjunto de restricciones violadas.

$$\Omega = \mathbf{R}\sum_{j\in J}|g_j(\mathbf{x})| \tag{2.9}$$

- **Penalización logarítmica:** también es usada para restricciones de desigualdad. Sin embargo, esta penalización se hace sobre los puntos factibles, dando mayor penalización a aquellos que se encuentran en la frontera de la zona factible. El valor de $\mathbf{R}$ debe iniciar con un valor grande y ser reducido gradualmente.

$$\Omega = -\mathbf{R}\log[g(\mathbf{x})] \tag{2.10}$$

La principal ventaja de este método es que puede aplicarse a cualquier tipo de restricción. Su desventaja es que la función objetivo se puede distorsionar y con esto llevar a un óptimo local artificial en donde el algoritmo puede quedar atrapado. El Algoritmo 2.7 describe el proceso de este método.

2.4.2 Método Complex

El método Complex desarrollado por Box en 1965 [3] está basado en el método Simplex [7], sin embargo a diferencia de éste, Complex toma en cuenta las restricciones al momento de minimizar. A continuación se describe el proceso:

Algoritmo 2.7 Algoritmo de penalización de funciones

Entrada: punto inicial $\mathbf{x}_k$, dos parámetros de termino ϵ_1 y ϵ_2, $k = 0$, Un término de penalización Ω, $\mathbf{R}_0$ inicial, parámetro c para actualizar el valor de $\mathbf{R}$ este valor dependerá directamente del término Ω seleccionado.

1: **repetir**
2: Formar el problema $P(\mathbf{x}_k, \mathbf{R}_k) = f(\mathbf{x}_k) + \Omega(\mathbf{R}_k, g(\mathbf{x}_k), h(\mathbf{x}_k))$
3: evaluar $P(\mathbf{x}_k, \mathbf{R}_k)$
4: realizar una búsqueda con $\mathbf{x}_k$, utiliza ϵ_1 como parámetro de finalización de la búsqueda.
5: actualizar $\mathbf{R}_{k+1} = c\mathbf{R}_k$, $k = k + 1$
6: **hasta que** $|P(\mathbf{x}_{k+1}, \mathbf{R}_k) - P(\mathbf{x}_k, \mathbf{R}_{k-1})| \leq \epsilon_2$
7: **devolver** $\mathbf{x}_k$

1. Crear el Complex inicial: Se requiere un **punto factible** $\mathbf{x}^y$ a partir del cuál se crearan otros K puntos factibles de manera aleatoria dentro de los límites del problema. Cada nuevo punto será evaluado y si no es factible, el punto se retrae hacia el centroide del Complex sin contemplar el punto no factible (Ecuación 2.11). Este proceso se repite hasta que $\mathbf{x}^k$ sea factible, entonces se agrega esta solución al Complex y se continúan generando soluciones hasta que se completen los K requeridos.

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}_k + \tfrac{\overline{\mathbf{x}} - \mathbf{x}_k}{2} \\ \overline{\mathbf{x}} &= \tfrac{\sum_{k=1}^{K} \mathbf{x_k}}{K} \end{aligned} \tag{2.11}$$

2. Ciclo de mejora: En cada iteración el peor punto del Complex es seleccionado $\vec{x}_w$, con este punto se crea un nuevo vector usando la Ecuación 2.12, $alpha$ es un factor de reflexión.

$$\mathbf{x}_n = \overline{\mathbf{x}} + \alpha(\overline{\mathbf{x}} - \mathbf{x}_w) \tag{2.12}$$

Si $\mathbf{x}_n$ es mejor que $\mathbf{x}_w$, entonces lo reemplaza en el Complex, si no es mejorada se continúa creando nuevas soluciones utilizando como punto base $\mathbf{x}_n$. El algoritmo termina cuando la distancia en el Complex es menor o igual a δ o cuando la diferencia en la función de aptitud ha alcanzado un límite φ.

Los valores recomendados de acuerdo a Box [3] son: $K = D + 1$, $\delta = 1e - 5$, $\varphi = 1e - 5$ y para el factor de reflexión $\alpha = 1{,}3$.

Este método asume que se inicia con un punto factible, y que la zona factible es convexa. De no ser así, en el proceso de retracción al centroide los puntos resultantes podrían ser no factibles. Además si el óptimo se encuentra en la frontera de la región factible, el algoritmo se comportara lento y con poca eficiencia. En el Algoritmo 2.8 se muestra el método completo.

Algoritmo 2.8 Complex

1: Inicializar el Complex
2: **mientras** $\sqrt{\sum(||\mathbf{x_k} - \overline{\mathbf{x}}||)^2} > \delta$ y $\sqrt{\sum(f(x_k) - \overline{f})^2} > \varphi$ **hacer**
3: Seleccionar al peor o una solución aleatoria del complex $\mathbf{x}_w$
4: Crear una nueva solución $\mathbf{x}_n$ de acuerdo a la Ecuación 2.12
5: **mientras** $\mathbf{x}_w$ sea mejor que $\mathbf{x}_n$ de acuerdo a factibilidad y valor de función objetivo $||\mathbf{x}_n - \overline{\mathbf{x}}|| > \delta$ **hacer**
6: $\mathbf{x}_n = (\overline{x} + \mathbf{x}_n)/2$
7: **fin mientras**
8: **si** $\mathbf{x}_n$ es mejor que $\mathbf{x}_w$ **entonces**
9: Reemplazar $\mathbf{x}_w$ por $\mathbf{x}_n$
10: **fin si**
11: **fin mientras**
12: Retornar la mejor solución del Complex

2.5 Otros métodos

En esta sección se describirán dos métodos que por sus características no son clasificados dentro de algunas de las otras secciones. Cabe mencionar que los algoritmos evolutivos también son empleados como métodos de optimización, sin embargo a este tipo de algoritmos se les dedicará el capítulo siguiente.

2.5.1 Caminata Aleatoria

El algoritmo de caminata aleatoria pertenece a la categoría de algoritmos aleatorios. Este método se caracteriza por generar una dirección de magnitud unitaria de manera aleatoria $\mathbf{u}$ y con un tamaño de paso λ generar un nuevo punto a partir de $\mathbf{x}_t$. Si el nuevo punto es mejor que el anterior, el algoritmo "camina" hacia este nuevo punto, este proceso se repite durante n veces antes de reducir el tamaño de paso λ, permitiendo explorar el vecindario del punto $\mathbf{x}_t$. El algoritmo termina cuando el tamaño de paso es lo suficientemente pequeño ϵ para considerar que se ha convergido.

Algoritmo 2.9 Caminata Aleatoria

Entrada: punto inicial $\mathbf{x}_t$, un parámetro de término ϵ, un tamaño de paso inicial λ, un número máximo de intentos por tamaño de paso n.

1: **repetir**
2: **para** 1 a n **hacer**
3: Genera un vector unitario aleatorio $\mathbf{u}$
4: Generar $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}$
5: **fin para**
6: Reducir el tamaño de paso λ
7: **hasta que** $|\lambda| < \epsilon$
8: Retornar $\mathbf{x}_t$

2.5.2 Recocido Simulado

El método de recocido simulado, está basado en el proceso de enfriamiento del metal fundido. Cuando el metal está expuesto a altas temperaturas, sus átomos se mueven libremente por todo el metal, pero conforme la temperatura se va reduciendo, el movimiento de los átomos se ve restringido. Finalmente los átomos se ordenan a un punto de mínima energía, y comienzan a formar cristales. Si la temperatura se reduce muy rápido, los cristales formados son débiles. Por lo tanto para llegar al estado mínimo de energía la temperatura debe reducirse lentamente.

Este proceso se simula usando un parámetro que representa la temperatura y es controlada por medio de la distribución de probabilidad de Boltzmann $P(E) = exp(-E/cT)$, donde T es la temperatura, c es una constante y E es la energía del sistema. Esta ecuación siguiere que a mayor temperatura la probabilidad se comportara como una distribución uniforme para permitir cualquier estado de energía, mientras que a bajas temperaturas la probabilidad para permitir altos grados de energía es baja. La energía se puede ver como el valor de aptitud de la función objetivo $E(t) = f(\mathbf{x}_t)$. La energía del punto vecino dependerá de la diferencia entre éste y el anterior, por lo tanto $\Delta E = E(t+1) - E(t)$, dando como resultado la probabilidad de ser aceptado como punto siguiente. En el Algoritmo 2.10 se muestra el algoritmo completo de este método.

Algoritmo 2.10 Recocido Simulado

Entrada: punto inicial $\mathbf{x}_t$, un parámetro de término ϵ, una temperatura suficientemente alta T, un número de iteraciones máximas para una temperatura particular.

1: **repetir**
2: $t = 0$
3: **mientras** t mod $n = 0$ **hacer**
4: Crear un vecino de $\mathbf{x}_t$, $\mathbf{x}_{t+1} = N(\mathbf{x}_t)$, usualmente generado de manera aleatoria.
5: **si** $\Delta E < 0$ Y un número aleatorio entre (0,1] $r \leq exp(-\Delta E/T)$ **entonces**
6: $t = t + 1$, y mantener $\mathbf{x}_{t+1}$
7: **fin si**
8: **fin mientras**
9: Reducir la temperatura T mediante algún proceso de reducción.
10: **hasta que** $|\mathbf{x}_{t+1} - \mathbf{x}_t| < \epsilon$ Y T es suficientemente pequeña
11: Retornar $\mathbf{x}_t$

2.6 Resumen

En este capítulo se introdujeron algunos conceptos básicos de optimización, como lo son: (1) la definición de un problema de optimización, (2) el concepto de óptimo local y (3) el concepto de óptimo global.

También se presentó un conjunto de métodos de optimización multivariable, los cuales se dividen en dos grupos, (1) los métodos directos como lo son: Simplex y Hooke-Jevees y (2) los métodos indirectos tales como el método de Cauchy, el método de Newton y el método de Marquardt. Si bien estos últimos métodos son rápidos, no siempre son viables de aplicar, pues requieren que la función a optimizar sea diferenciable y continua, características que no siempre pueden asegurarse. Mientras que los métodos directos requieren de buenos puntos de inicio y muchas evaluaciones.

Por último se describieron dos algoritmos fuertemente basados en la aleatoriedad, (1) la Caminata aleatoria y (2) el Recocido Simulado, ambos son heurísticas de búsqueda que han mostrado gran eficiencia en problemas donde los métodos tradicionales fallan por sus limitaciones mencionadas en durante el capítulo.

En el siguiente capítulo se introducirán los algoritmo evolutivos, metaheurísticas basadas en el proceso de la evolución que han mostrado ser

eficaces al resolver problemas de optimización con poco o nulo conocimiento del problema.

3 Cómputo Evolutivo

> *Intelligence is based on how efficient a species became at doing the things they need to survive.*
>
> **— Charles Darwin**

En el capítulo anterior se presentaron algunos métodos de optimización clásica, y se expresaron algunas de sus características, como lo son, que requieren de un punto inicial, gastan muchas evaluaciones, y aquellos que son rápidos requieren información de la derivada de la función, la cual puede no estar presente o ser compleja de calcular.

A fin de aliviar las desventajas de estos métodos, otras estrategias como heurísticas y meta-heurísticas se han desarrollado, entre estos se encuentran los algoritmos evolutivos. Entre las características podemos encontrar:

- Tienen elementos estocásticos, i.e., incluso cuando el método inicie en las mismas condiciones no asegura obtener el mismo resultado siempre.

- Generalmente son poblacionales, no requieren un punto de inicio, generan un conjunto de puntos de manera aleatoria y a partir de ellos inicia la búsqueda.

- No requieren de información de la función a optimizar, son tratadas como cajas negras, se da una entrada y se obtiene una salida.

En las siguientes secciones se describirán de manera general, algunos de los paradigmas más importantes.

3.1 Algoritmos Evolutivos

Como se mencionó anteriormente, los algoritmos evolutivos son meta-heurísticas, es decir son procesos de búsqueda que pueden aplicarse a diferentes tipos de problemas.

Estos algoritmos están directamente inspirados en la evolución Darwiniana, de manera general comparten las siguiente características [6]:

- Individuos de una o más poblaciones compiten por recursos.
- Las poblaciones cambian con el paso del tiempo.
- Los individuos son capaces de reproducirse, pasando su carga genética a futuras generaciones.
- Existen procesos de selección de acuerdo a la aptitud de los individuos.

Basados en estos pasos generales, se han desarrollado diferentes algoritmos. En las siguientes sub-secciones se describen algunos de los principales paradigmas. A partir de este punto el concepto de individuos y solución potencial del problema, se consideran sinónimos.

3.1.1 Programación Evolutiva

Fogel et. al. [11] en 1966, proponen un algoritmo donde todos los individuos son capaces de generar un hijo, después de ello, las poblaciones de hijos y padres son combinadas, y solo el 50 % mejor de esta nueva población pasa a la siguiente generación. El algoritmo general se puede ver en el Algoritmo 3.11.

Este algoritmo es elitista, ya que siempre elige a los mejores individuos para pasar a la siguiente generación. También cabe destacar que no existe un proceso de cruza de individuos, pues son considerados como especies diferentes que solo son capaces de mutarse a sí mismos.

Algoritmo 3.11 Algoritmo Programación Evolutiva

1: **para** Cada generación **hacer**
2: **para** cada individuo i en la población **hacer**
3: Generar un hijo i' mediante alguna estrategia de mutación
4: Guardar i' en población de hijos
5: **fin para**
6: Combinar las poblaciones de padres e hijos en una sola
7: Guardar el 50 % mejor de la población combinada, para ser los padres de la siguiente generación.
8: **fin para**

3.1.2 Estrategias Evolutivas

Las estrategias evolutivas (ES) fueron propuestas por Schwefel [38] en 1975, en un inicio este algoritmo no era poblacional, ya que se basa en el proceso de mutación. El proceso de selección es extintivo, ya el padre y el hijo compiten por mantenerse en la población de la siguiente generación.

El proceso de mutación se realiza agregando ruido Gaussiano al padre que será mutado tal como se presenta en la Ecuación 3.1, donde $N(0, \sigma)$ es un vector de números Gausianos con media 0 y desviación estándar σ.

$$\mathbf{x}_{t+1} = \mathbf{x}_t + N(0, \sigma) \tag{3.1}$$

Como se mencionó anteriormente el algoritmo inicialmente no era poblacional, ya que solo existía una única solución y al cual se le aplicaba el método de mutación. A esta estrategia se le llamó (1+1)-ES, donde de manera general $(\mu + \lambda)$ representa el número de padres μ y de hijos λ que generan estos padres. Esto último da cabida a la generación de nuevas versiones como lo son:

- $(\mu + 1)$-ES, μ padres, que generan sólo un descendiente. El proceso de selección es extintivo.
- $(\mu + \lambda)$-ES, los μ padres generan λ hijos. Ambas poblaciones se combinan (de ahí el $+$) y los μ mejores de la nueva población pasan a la siguiente generación.

- (μ, λ)-ES, los padres generan λ hijos. La selección se dá sólo en la población de los λ hijos, seleccionando a los μ mejores.

3.1.3 Algoritmos Genéticos

Los algoritmos genéticos tal vez sean los más conocidos de todas las estrategias que se son parte del cómputo evolutivo. Éstos fueron propuestos por Golberg y Holland [12] en la década de los 60's. Los algoritmos genéticos son poblacionales y a diferencia de los algoritmos anteriormente mencionados, basan su búsqueda en la cruza (reproducción), i.e. consideran a todos los individuos parte de una misma especie y son capaces de generar hijos combinando su información. No por ello dejan de lado el proceso de mutación, pues cada que un hijo es generado es expuesto a dicho proceso, pero no siempre se cumple. El algoritmo general se puede ver en el Algoritmo 3.12. Los algoritmos genéticos en un inicio son extintivos, esto es, la generación nueva reemplaza directamente a la generación anterior, sin embargo una forma de elitismo puede agregarse, siempre manteniendo a la mejor solución dentro de la población actual.

Algoritmo 3.12 Algoritmo Genético

1: Generar y evaluar una población inicial de manera aleatoria
2: **para** cada generación **hacer**
3: Seleccionar de la población aquellos individuos que serán los padres
4: Aplicar la cruza entre padres para generar los descendientes
5: Aplicar el proceso de mutación a los descendientes
6: Evaluar la nueva población
7: Reemplazar la generación anterior con los descendientes
8: **fin para**

3.2 Inteligencia Colectiva

Otro tipo de algoritmos que trabajan de manera similar a los algoritmos evolutivos, son aquellos que su inspiración no parte de la evolución de las especies, pues se centra en emular un comportamiento colectivo. De esta inspiración es que nace la inteligencia colectiva (*Swarm Intelligence*). Formalmente *Swarm* se puede definir como un grupo de agentes (soluciones) que generalmente son móviles y tienen una forma de comunicación directa o indirecta, que les permite actuar sobre su entorno [10]. La interacción

entre los agentes promueve estrategias para resolver problemas de manera distribuida.

El proceso básico de un algoritmo de Inteligencia colectiva es el siguiente:

- Generar un grupo de agentes (swarm).
- Permitir a cada agente aprender de su medio ambiente.
- Cada agente comparte su información a los demás agentes.

Como se puede observar, en este esquema los agentes no mueren, ni son reemplazados por nuevos, cada uno de ellos se mantiene y van aprendiendo de su entorno y de la comunicación entre ellos. Algunos ejemplos que se pueden encontrar en la naturaleza de este comportamiento son: (1) las aves, (2) las hormigas, (3) las abejas, (4) las termitas, entre otros. A continuación se describen brevemente dos algoritmos basados en estos comportamientos.

3.2.1 Optimización por Cúmulo de Partículas

Optimización por Cúmulo de Partículas (PSO por sus siglas en inglés) es un algoritmo desarrollado por Kenedy and Eberhart [15] en 1995. Está basado en el comportamiento de vuelo las aves, observando que estas guardan sincronía al volar, cambian de dirección repentinamente y su forma de distribuirse de manera dispersa o agrupándose ante los cambios en el ambiente.

En este algoritmo las soluciones son llamadas partículas y la población o cúmulo (swarm). Se pueden presentar muchos cúmulos que agruparán a varias partículas, esto con la finalidad de promover la exploración del espacio de búsqueda. Al mismo tiempo, las partículas de diferentes cúmulos pueden interactuar entre ellas. Por otro lado el vector solución al problema representa la posición de la partícula a la cual está asociado.

De esta forma, una partícula conoce su posición actual, mantiene una memoria de su última mejor posición llamada $Pbest$ y conoce su velocidad que le indica la dirección y magnitud de su vuelo. La dirección del vuelo se ve influenciada por el líder del cúmulo (mejor partícula) y por su propia mejor

posición, de esta forma se comparte conocimiento. En el Algoritmo 3.13 se presentan los pasos de este esquema.

Algoritmo 3.13 Algoritmo Optimización por Cúmulo de Partículas (PSO)

1: Generar y evaluar el cúmulo de partículas
2: Actualizar memoria de las partículas $Pbest$
3: **para** cada vuelo **hacer**
4: Seleccionar al líder del cúmulo
5: Aplicar operador de vuelo a cada partícula
6: Actualizar memoria de las partículas $Pbest$
7: **fin para**
8: Reportar la mejor partícula

3.2.2 Colonia Artificial de Abejas

La Colonia Artificial de Abejas (ABC) fue propuesto por Karaboga [14] en 2005. Es un algoritmo basado en el comportamiento de forrajeo de las abejas melíferas. En su comportamiento biológico se logró identificar tres tipos de abejas con trabajos exclusivos durante el forrajeo. Las abejas empleadas son las encargadas de explotar el área en fuentes de alimento conocidas, mientras que las abejas observadoras esperan la información de los mejores lugares para explotar las fuentes de alimento, por otro lado existen las abejas exploradoras que se encargan de buscar nuevas fuentes cuando las existentes ya fueron explotadas.

Basado en ese comportamiento Karaboga et al. diseñaron un algoritmo, donde las soluciones son las fuentes de alimento y los operadores que realizan la búsqueda son las abejas. En el Algoritmo 3.14 se describen los pasos generales de las abejas.

3.3 Evolución Diferencial

Evolución diferencial (DE por sus siglas en inglés) fue propuesto por Storn y Price en 1995 [40], con la finalidad de lidiar con espacios de búsqueda continuos. El algoritmo canónico es DE/rand/1/bin, “rand” significa que utiliza una selección aleatoria del vector base que se usa en la mutación, “1” que el operador está basado en una diferencia, y “bin” que el operador de cruza es binomial. El proceso de Evolución Diferencial se puede describir en cuatro pasos listados a continuación:

Algoritmo 3.14 Algoritmo Colonia Artificial de Abejas (ABC)

1: Generar fuentes de alimento y asignarlas a las abejas empleadas
2: **para** cada iteración **hacer**
3: Las abejas empleadas buscan nuevas fuentes de alimento y solo cambian de fuente si es mejor que la anterior conocida
4: Seleccionar las mejores fuentes de alimento de las conocidas por las abejas empleadas
5: Explotar las fuentes de alimento seleccionadas con las abejas observadoras y actualizar las fuentes si se encuentran mejores
6: Si alguna fuente de alimento no ha mejorado en cierto tiempo, se envía a las abejas exploradoras para generar nuevas de manera aleatoria
7: **fin para**
8: Reportar la mejor fuente de alimento

1. Inicialización: Una población inicial Pop compuesta de NP soluciones $\mathbf{x}_{i,0}$, $i = 1 \ldots, NP$ es generada de manera aleatoria. Por cada variable de decisión $x_{j,i,0}$, se genera un valor aleatorio entre los límites de la función, como se muestra en la Ecuación 3.2.

$$x_{j,i,0} = l_j + rand_j(0,1) \times (u_j - l_j) \tag{3.2}$$

donde $rand_j(0,1)$ es un número con distribución uniforme entre 0 y 1.

2. Operador de mutación: en cada generación g, por cada uno de los vectores en la población (llamado target) $\mathbf{x}_{i,g}$, una solución mutante $\mathbf{v}_{i,g+1}$ es generada de acuerdo a la Ecuación 3.3:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_0,g} + F(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) \tag{3.3}$$

donde $r_0, r_1, r_2 \in \{1, 2 \ldots NP\}$ son índices seleccionados aleatoriamente de la población, diferentes entre ellos y diferentes a i (i.e., $r_0 \neq r_1 \neq r_2 \neq i$). $F > 0$ es un factor de escala que controla la amplitud del vector diferencia, conformado por $\mathbf{x}_{r_1,g}$ y $\mathbf{x}_{r_2,g}$. $\mathbf{x}_{r_0,g}$ es el llamado vector base.

3. Operador de cruza: después de generar el vector mutante, el vector target $\mathbf{x}_{i,g}$ se combina con el vector mutante $\mathbf{v}_{i,g+1}$, para obtener el

vector hijo (trial) $\mathbf{u}_{i,g+1}$. Este operador de cruza se realiza de acuerdo a la Ecuación 3.4.

$$u_{j,i,g+1} = \begin{cases} v_{j,i,g+1}, & \text{if } rand[0,1) \leq CR \text{ or } j = j_r \\ x_{j,i,g} & \text{en otro caso} \end{cases} \tag{3.4}$$

donde $j = 1, 2, \ldots, D$, $rand[0,1)$ es un número aleatorio con distribución uniforme entre 0 y 1, y j_r es un índice de una variable de decisión seleccionado de manera aleatoria, con el objetivo de que al menos un valor del vector mutante pertenezca al vector trial y evitar copias del vector padre. $CR \in [0,1]$ es la tasa de cruza.

4. Selección: de acuerdo al valor de aptitud de la función objetivo, sí $\mathbf{u}_{i,g+1}$ es mejor que el vector target $\mathbf{x}_{i,g}$, entonces $\mathbf{u}_{i,g+1}$ será seleccionado para pasar a la siguiente generación. De otra forma, $\mathbf{x}_{i,g}$ se mantiene en la siguiente generación, este proceso se realiza asumiendo minimización, de acuerdo a la Ecuación 3.5

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1}, & \text{if } f(\mathbf{u}_{i,g+1}) \leq f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{en otro caso} \end{cases} \tag{3.5}$$

En el Algoritmo 3.15 se muestra el algoritmo completo para la versión DE/rand/1/bin. Si bien esta es la versión canónica y básica del algoritmo, se han desarrollado múltiples propuestas y variantes del algoritmo como por ejemplo:

- Operador de cruza exponencial, DE/rand/1/exp. Los valores para el vector trail toman los valores de manera secuencial del vector mutante, mientras se cumpla la taza de cruza CR, los valores restantes se toman del vector target.

- DE/best: dejando de lado el operador de cruza que puede aplicar tanto binomial como exponencial, en esta versión, el vector base para obtener el vector mutante es tomado de la mejor solución de la población actual.

- DE/current-to-rand: en esta versión se utiliza la información del vector target para generar el vector mutante

- DE/current-to-best: el vector mutante es generado con la información del vector target como vector base y la información de la mejor solución en la diferencia de vectores.

Este algoritmo es uno de los más utilizados actualmente, pues es muy simple de entender y de modificar, además de obtener muy buenos resultados en múltiples problemas desde conjuntos de funciones de prueba hasta aplicaciones en problemas reales.

Algoritmo 3.15 DE rand/1/bin

Entrada: el tamaño de población NP, el máximo de evaluaciones permitido $MaxFEs$, un factor de escala F y la tasa de cruza CR

```
Generar la población inicial Pop = (x_1, ..., x_NP)
Evaluar la población
repetir
  para i = 1 a NP hacer
    Seleccionar tres soluciones de manera aleatoria r_0 ≠ r_1 ≠ r_2 ≠ i
    Genera v usando la Ecuación 3.3
    Generar u utilizando la Ecuación 3.4
    si f(u_{i,g+1}) ≤ f(x_i) entonces
      x_{i,g+1} = u_{i,g+1}
    fin si
  fin para
hasta que MaxFEs se alcancen
devolver Mejor solución en la población
```

3.4 Manejadores de restricciones

Al igual que se presentó en el Capítulo 2, donde los métodos originalmente no fueron pensados para atacar problemas con restricciones, los algoritmos evolutivos presentan esta misma característica, de un inicio solo se enfocaron en problemas de optimización sin restricciones.

Sin embargo al ser estos algoritmos meta-heurísticas, es posible adaptarlos para resolver problemas con restricciones. Se han desarrollado diferentes técnicas para lidiar con las restricciones sin necesidad de realizar cambios grandes a la estructura de los algoritmos. Algunas de estas técnicas son mencionadas en este apartado.

3.4.1 Penalización de funciones

Al igual que el método presentado en la sección 2.4.1, esta técnica busca transformar el problema restringido en un problema sin restricciones, haciendo que las meta-heurísticas no tengan que modificar su forma de trabajar.

El problema es transformado agregando a la función objetivo factores de penalización en las restricciones y de esta manera obtener un problema sin restricciones. En la Ecuación 3.6 se muestra una forma de esta transformación:

$$\phi(\mathbf{x}) = f(\mathbf{x}[\sum_{i=1}^{n} r_i \times g_i + \sum_{j=1}^{p} c_j \times h_j]) \tag{3.6}$$

donde, ϕ es la nueva función sin restricciones a optimizar, mientras que g, h representan las restricciones de desigualdad e igualdad respectivamente, r, c son los factores de penalización para cada tipo de restricción, estos factores son parámetros.

El principal problema con esta técnica es la definición de los factores de penalización, ya que no es trivial y dependerá del problema que se esté resolviendo. Un valor grande en estos parámetros permitirá explorar rápidamente el espacio de búsqueda, incrementando la posibilidad de quedar atrapado en óptimos locales. Por otro lado, valores pequeños, permiten explotar la región vecina mejor, pero la búsqueda se vuelve lenta, requiriendo más tiempo para encontrar buenas soluciones.

3.4.2 Reglas de Factibilidad

Las Reglas de Factibilidad fueron propuestas por Deb [8] en el año 2000. Son en esencia un método de selección binaria, i.e., comparan dos soluciones y determinan cuál es mejor que otra. Esto lo realiza utilizando tanto la información del valor de la función objetivo, como la factibilidad de dichas soluciones.

Esta técnica siempre considera que una solución factible es mejor que una que no lo es, generando una gran presión de selección. De esta manera se puede ver que las reglas de factibilidad se pueden agregar a cualquiera de los algoritmos mencionados sin hacer ningún cambio, más que al momento

de comparar la calidad de dos soluciones. Las reglas de factibilidad son las siguientes:

1. Dadas dos soluciones, una factible y otra no factible, la solución factible es seleccionada.

2. Entre dos soluciones factibles, aquella con mejor valor en la función objetivo es seleccionada.

3. Entre dos soluciones no factibles, aquélla con menor grado de violación de restricción es seleccionada.

Esta técnica es intuitiva y de fácil implementación, sin embargo una de sus desventajas, es que la presión de selección es muy fuerte cuando de una solución factible se compara con una no factible, llevando al algoritmo a una convergencia rápida y posibles óptimos locales en la región factible.

3.4.3 Método ε-Constrained

El método de ε-Constrained puede verse como una extensión de las reglas de factibilidad y fue desarrollado por Takahama y Sakai [42]. Este método busca relajar la influencia de las restricciones, basado en una tolerancia ε que permite considerar factibles soluciones que no lo son. Básicamente ataca el problema de presión de selección que presentan las reglas de factibilidad.

El valor ε, se reduce conforme pasan las generaciones hasta que esta tolerancia se vuelve 0 y a partir de aquí el método se comporta exactamente igual que las reglas de factibilidad.

La comparación entre dos soluciones $\mathbf{x}_1$ y $\mathbf{x}_2$ se realiza de acuerdo a la siguiente Ecuación 3.7:

$$f(\mathbf{x_1}), \phi(\mathbf{x_1}) <_\varepsilon f(\mathbf{x_2}), \phi(\mathbf{x_2}) \iff \begin{cases} f(\mathbf{x_1}) < f(\mathbf{x_2}), & \text{si } \phi(\mathbf{x_1}), \phi(\mathbf{x_2}) \leq \varepsilon; \\ f(\mathbf{x_1}) < f(\mathbf{x_2}), & \text{si } \phi(\mathbf{x_1}) = \phi(\mathbf{x_2}); \\ \phi(\mathbf{x_1}) < \phi(\mathbf{x_2}) & \text{en otro caso} \end{cases} \tag{3.7}$$

El valor de la sumatoria de violación de restricciones se puede calcular de dos formas, (1) como el valor máximo de las todas las restricciones (Ecuación 3.8), o (2) como la sumatoria de todas las restricciones tal como se muestra en la Ecuación 3.9 donde P es un número entero positivo.

$$\phi(\mathbf{x}) = max\{max\{0, g_j(\mathbf{x})\}, max\{|h_j(\mathbf{x})|\}\} \tag{3.8}$$

$$\phi(\mathbf{x}) = \sum ||max\{0, g_j(\mathbf{x})\}||^P + \sum ||h_j(\mathbf{x})||^P \tag{3.9}$$

El valor inicial de ε está definido por la sumatoria de violación de restricciones de la θ-ésima solución de la población inicial. Este valor es actualizado durante cada iteración del algoritmo de acuerdo a la Ecuación 3.10, hasta que el número de iteraciones t llegue al número de control TC, y es cuando el valor de ε se volverá 0.

$$\begin{aligned} \varepsilon(0) &= \vartheta(\mathbf{x}_\theta) \\ \varepsilon(t) &= \begin{cases} \varepsilon(0)(1-(t/TC))^{cp}, & 0 < t < TC \\ 0, & t \geq TC \end{cases} \end{aligned} \tag{3.10}$$

donde $\mathbf{x}_\theta$ es la θ-ésima solución, y $\theta = 0{,}2 \times NP$, donde NP es número de soluciones, y cp es un número entero mayor o igual a 1 y se recomienda que tome un valor de 3.

3.5 Resumen

En este capítulo se introdujo al cómputo de evolutivo, exponiendo su principal característica de estar basado en el proceso de la evolución. Se describieron brevemente los primeros paradigmas dentro del área y cómo es que estos toman parte de esa inspiración en la evolución de las especies.

Dentro de los algoritmos evolutivos se expusieron tres principales enfoques. La programación evolutiva, que se basa en la mutación y selección elitista de los individuos generados. Las estrategias evolutivas, que usan la mutación

como principal generador de individuos, siendo extintivos y que inicialmente no era una estrategia poblacional, sin embargo, propuestas posteriores dieron paso a adquirir esta característica, permitiendo explorar nuevas formas de reemplazo generacional. Y los algoritmos genéticos, estos son poblacionales desde su inicio y contrariamente a las otras propuestas apuesta por la cruza ente individuos y poca mutación de los hijos.

Por otra parte, se expuso el enfoque de inteligencia colectiva, que busca generar algoritmos inspirados en el comportamiento de los individuos con su medio y entre ellos mismos. La principal diferencia entre inteligencia colectiva y los algoritmos evolutivos, es que los agentes en la inteligencia colectiva se mantienen durante todo el proceso de búsqueda, mientras que en el cómputo evolutivo los individuos son reemplazados por aquellos que se van creando en cada generación.

Los algoritmo de inteligencia colectiva aquí expuestos fueron: 1) Optimización de Cúmulo de Partículas, que está inspirado en el vuelo de las aves, su característica principal, es que realizan un vuelo con información de su posición anterior, información del líder, y una velocidad que actualizan siempre que se mueven. Por otro lado, 2) el algoritmo de Colonia Artificial de Abejas, donde los operadores son las abejas y las soluciones son fuentes de alimento, su característica principal es que cada operador (abeja) tiene un objetivo particular, las empleadas exploran en busca de más fuentes, las observadoras explotan las fuentes conocidas y las exploradoras buscan nuevas fuentes de alimento cuando estas dejan de mejorar.

Finalmente se expuso el algoritmo de Evolución Diferencial, que está basado en diferencias de vectores para generar nuevas soluciones, usando el proceso iterativo básico del proceso evolutivo ha logrado posicionarse como uno de los algoritmos favoritos en el área por su sencillez y rendimiento al resolver problemas.

Todos estos enfoques expuestos, fueron creados con la finalidad de atender problemas numéricos sin considerar restricciones, y es por ello que en la última sección se habló de tres diferentes estrategias que le permiten a los algoritmos lidiar con las restricciones. Por un lado se tiene una estrategia clásica que es la penalización, en la cual, el problema restringido se convierte en un problema sin restricciones utilizando factores de penalización en las restricciones. Después se introdujeron las reglas de factibilidad, que como su

nombre lo dice, es un conjunto de reglas que permiten seleccionar en una comparación a pares cual solución es mejor basados en la factibilidad de éstas. Y finalmente, el método de ε-constrained, que puede verse como una extensión de las reglas de factibilidad, el cual busca relajar las restricciones en un inicio del proceso de búsqueda hasta llegar a convertirse en las reglas de factibilidad hacia el final de la búsqueda.

En el siguiente capítulo, se presentarán las estrategias seguidas para resolver problemas de alta-dimensión desde el contexto de cómputo evolutivo.

4 Cómputo Evolutivo en problemas de alta dimensión

> *Never memorize something that you can look up.*
>
> **— Albert Einstein**

4.1 Introducción

Los problemas de optimización que se enfrentan actualmente, involucran cada vez más variables de diseño para resolver el problema en cuestión. Si bien, los algoritmos evolutivos descritos en el Capítulo 3, han mostrado ser una opción para resolver problemas de optimización numérica, su rendimiento se ve afectado de manera negativa cuando se agregan más variables de diseño al problema, sufriendo la "maldición de la dimensionalidad" [32, 34, 19]. Esto es, el número de soluciones requeridas para explorar el espacio de búsqueda crece exponencialmente con relación al incremento del número de dimensiones del problema, el costo computacional para la evaluación de una solución se ve incrementado y las características del problema pueden llegar a cambiar (forma del espacio de búsqueda, región factible, modalidad).

Para contender con estos problemas, una estrategia de *divide y vencerás* se ha adoptado y agregado a los algoritmos evolutivos, mejorando su eficiencia al resolver problemas de alta dimensionalidad, dando lugar a un nuevo enfoque de algoritmos que se encuentran basados en la cooperación de diferentes especies [34] que evolucionan al *mismo tiempo* sobre subespacios de búsqueda del problema original.

En este capítulo se describirá el enfoque general de los algoritmos de Coevolución Cooperativa, también se expondrán tres de los principales enfoques de descomposición de problemas como son: el agrupamiento aleatorio

(Random Grouping), agrupamiento diferencial (Differential Grouping), e Identificación de la interacción de variables para problemas con restricciones (VIIC).

4.2 Co-evolución Cooperativa

Uno de los primeros enfoques de Co-evolución Cooperativa (CC) fue propuesto por Potter y De Jong [34]. Ellos desarrollaron un algoritmo genético (GA) al que llamaron CCGA, tomando como inspiración las técnicas clásicas que optimizan cada una de las variables del problema por separado dejando las restantes con un valor constante. De esta forma se crean tantas *especies* como variables tenga el problema, aplicando el algoritmo genético a cada una de las variables. Tal como ellos mencionan en su trabajo, este algoritmo sirvió para comprobar que el esquema CC fue significativamente mejor que el algoritmo genético tradicional en las funciones Rastrigin, Schwefel, y Ackley. Por otro lado, en la función Griewangk CCGA fue ligeramente mejor que el GA. Concluyendo que al igual que los métodos clásicos, su desempeño se degrada cuando las variables interactúan unas con otras.

Algoritmo 4.16 Algoritmo CCGA

1: $gen = 0$
2: **para** cada especie s **hacer**
3: Inicializar las $Pop_s(gen)$ especies de manera aleatoria
4: Evaluar cada individuo de $Pop_s(gen)$
5: **fin para**
6: **mientras** condición de término no se alcance **hacer**
7: $gen = gen + 1$
8: **para** cada especies s **hacer**
9: Seleccionar $Pop_s(gen)$ de $Pop_s(gen-1)$ de acuerdo a la aptitud
10: Aplicar los operadores genéticos sobre $Pop_s(gen)$
11: Evaluar $Pop_s(gen)$
12: **fin para**
13: **fin mientras**

El algoritmo CCGA es descrito en el Algoritmo 4.16, en la primera fase del algoritmo se crean especies por cada variable de diseño, y estas son evaluadas utilizando la información de otras soluciones seleccionadas aleatoriamente de las otras especies. Cada especie es optimizada una a una, por separado, y para la evaluación se utiliza la mejor solución de las otras especies, que de manera temporal son constantes. Con esto, Potter y De Jong, introducen el concepto de CC (Cooperative Coevolutionary Evolutive Algorithm), dejando abierta

la exploración para integrar diferentes algoritmos evolutivos al enfoque de CC.

4.2.1 Esquema general de la Co-evolución Cooperativa

El esquema propuesto por Potter y De Jong, puede generalizarse de la siguiente forma:

1. **Descomponer el problema** en múltiples grupos de variables de decisión de acuerdo a la interacción de las variables.

2. **Optimización** de cada sub-grupo de variables con algún algoritmo de cómputo evolutivo.

3. **Cooperación** de los sub-grupos para crear una nueva población que formará parte de la nueva generación de individuos.

Estos tres pasos generales componen el esquema de Co-evolución Cooperativa. En la Figura 4.1 se muestra de manera gráfica y general este proceso. En ella se observa que el proceso es iterativo por lo que el problema se descompone múltiples veces; sin embargo, si el proceso de descomposición es determinista o especificado por el experto en el problema, no es necesario iterar esta primera fase; concentrándose directamente en la optimización y la cooperación de los sub-grupos. En la sección 4.3 se profundizará en los algoritmos de descomposición. Durante la evolución de cada sub-grupo de variables la evaluación se realiza considerando la información los otros sub-grupos, este proceso se ejemplificará más adelante. Finalmente la cooperación es la unión de todos los sub-grupos para crear la nueva población, la cual deberá ser evaluada nuevamente; de esta forma cada sub-grupo comparte su *conocimiento* al proceso general de CC.

Evaluación de las soluciones

Durante el proceso de optimización (a través de un algoritmo evolutivo) de cada sub-grupo de variables, las soluciones deben ser evaluadas. Al considerar

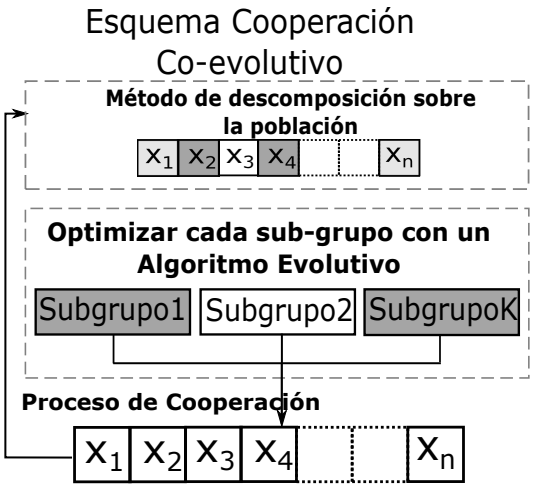
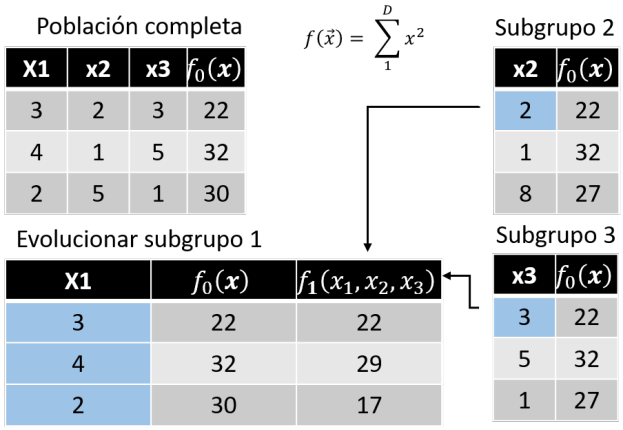


Figura 4.1.: Co-evolución Cooperativa

el problema como una caja negra que tiene como entrada todas las variables y salida la evaluación, es necesario una estrategia para complementar las soluciones de cada sub-grupo con las variables que le faltan. Durante este proceso, se da la cooperación entre los diferentes sub-grupos de variables. En la Figura 4.2, se presenta un ejemplo del proceso de evaluación durante la optimización de un sub-grupo.



X1	x2	x3	$f_0(x)$
3	2	3	22
4	1	5	32
2	5	1	30

x2	$f_0(x)$
2	22
1	32
8	27

X1	$f_0(x)$	$f_1(x_1, x_2, x_3)$
3	22	22
4	32	29
2	30	17

x3	$f_0(x)$
3	22
5	32
1	27

Figura 4.2.: Evaluación de soluciones, utilizando la información de los demás sub-grupos

Asumamos que queremos optimizar la función $f(\mathbf{x}) = \sum_i^D x_i^2$ y que consta de 3 variables de decisión. (1) El primer paso es crear una población aleatoria y evaluarla. (2) El problema en este ejemplo es descompuesto en tres sub-grupos cada uno formado por cada una de las variables del problema, por lo tanto se tienen 3 sub-poblaciones en la generación cero del CC. (3) Durante el proceso de evolución de cada sub-población, los operadores del algoritmo evolutivo seleccionado sólo afectarán a las variables correspondientes del

sub-grupo de variables asignadas a esa sub-población. (4) La evaluación se realiza completando la solución con valores de las otras sub-poblaciones, algunas estrategias ocupadas en la literatura son: seleccionar aleatoriamente soluciones de las otras sub-poblaciones o después de evaluar la población completa, seleccionar una solución como vector de contexto, el cual se mantendrá estático durante la evolución de cada sub-población, y de este se tomaran las variables necesarias para completar cada solución.

4.3 Métodos de descomposición

Como se mencionó en la sección anterior, los algoritmos de co-evolución cooperativa requieren dividir el problema en sub-problemas de menor dimensión, este proceso puede verse como un problema de optimización que busca la mejor manera de agrupar las variables del problema de acuerdo a su interacción. En la introducción de este trabajo se mencionaron diferentes algoritmos que se enfocan en resolver este problema de descomposición.

De acuerdo a Omidvar et al. [30], los algoritmos de descomposición se pueden categorizar en cuatro clases. **(1) Métodos aleatorios**, los cuales no tienen un procedimiento "inteligente", las variables se permutan para buscar incrementar la probabilidad de agrupar aquellas que interactúan. **(2) Los métodos de perturbación** monitorean los cambios en la función objetivo tratando de encontrar las variables que tienen interacción. En la mayoría de los casos la descomposición se realiza fuera de linea, es decir, se aplica antes de empezar la optimización, ejemplos de estos son CC con aprendizaje de interaccion de variables (CCVIL) [5] y Differential Grouping [30][31]. **(3) Adaptación de la interacción**: estos métodos incorporan mecanismos para la detección de la interacción, los cuales a diferencia de los métodos de perturbación, evolucionan la descomposición a través del proceso evolutivo que optimiza el problema. **(4) Métodos de construcción de modelo** que construyen un modelo probabilista basado en las soluciones prometedoras de la población, el modelo se actualiza de manera iterativa durante el proceso evolutivo, por ejemplo Ray y Yao [35] presentan un algoritmo basado en correlación.

En este trabajo nos enfocaremos en tres métodos del *estado del arte* que son representativos y que han sido probados en problemas restringidos, estos son; (1) Random Grouping, (2) Differential Grouping en su version 2, ya

que es una mejora a su antecesor; y (3) Variable Interaction Identification for Constrained Problems (VIIC) que, de acuerdo a la literatura, es el primer método de descomposición pensado en problemas con restricciones.

4.3.1 Random Grouping

Random Grouping es un método aleatorio presentado por Yang et. al. [45], inicialmente propusieron utilizar un vector de pesos que evolucionaría junto con la población para actualizar la interacción en las variables, sin embargo en una investigación posterior realizada por Omidvar et al. [32], concluyeron que este vector de pesos no aportaba la suficiente calidad a las soluciones respecto al costo que tenia optimizar el vector de pesos. Por esta razón, en este trabajo se decide utilizar la versión más simple de Random Grouping.

El algoritmo de Random Grouping requiere de dos parámetros iniciales, que son el número de subproblemas m y la cantidad de variables para cada subproblema definido por D/m donde D es la dimensión del problema original. La agrupación de variables se realiza a través de permutar las variables, asignando a cada grupo las variables que son contiguas de acuerdo al tamaño y numero de subproblemas requerido.

Consideremos el siguiente ejemplo, se requiere descomponer un problema de 10 variables en 2 sub-grupos, el proceso es el siguiente:

1. $m = 2$
2. Determinar el número de variables por sub-grupo $D/m = 10/2 = 5$
3. Permutar las variables $\{9, 7, 8, 5, 3, 2, 6, 1, 4, 10\}$
4. La descomposición del problema sería el siguiente $grupo1 = \{9, 7, 8, 5, 3\}$ y el $grupo2 = \{2, 6, 1, 4, 10\}$

El proceso de descomposición de Random Grouping se realiza en cada iteración del algoritmo de Co-evolución Cooperativa, tal como se muestra en la Figura 4.1. Las desventajas que presenta este método es que requiere se ingrese el número de sub-grupos que se quiere como salida, y también el tamaño de cada uno de estos sub-grupos aunque este punto puede automatizarse.

4.3.2 Differential Grouping versión 2

El método de Differential Grouping (DG) fue propuesto por Omidvar et al. [30] para la identificación de interacción en variables y aplicarlo en la optimización de problemas de alta dimensión. De acuerdo a sus autores este algoritmo es un método basado en la perturbación de variables, identificando la interacción de las variables por medio del cambio del valor de la función objetivo al perturbar dos variables con un mismo valor. DG fue comparado contra otros métodos de descomposición utilizando el conjunto de funciones de prueba del CEC'2010 [43]. Los resultados mostraron que DG fue superior a los otros métodos.

Sin embargo, el rendimiento del algoritmo no fue el mismo cuando éste fue probado en el conjunto de funciones de pruebas del CEC'2013 [19], de cuyo trabajo concluyeron que DG tenia las siguiente deficiencias, (1) un alto costo computacional en funciones altamente separables, (2) incapacidad para detectar la interacción de variables en problemas con superposición de variables, (3) sensibilidad a errores de redondeo computacional y (4) que necesita un parámetro de umbral ϵ. Dado este análisis los autores en el 2017 proponen mejoras al algoritmo y lo llaman Differential Grouping version 2 (DG2) [31]. En DG2 el número de evaluaciones requeridas por el algoritmo se redujo a la mitad y el parámetro ϵ se calcula automáticamente, haciendo de DG2 un algoritmo libre de parámetros a ajustar. Este algoritmo toma como base la siguiente definición:

Definición 3 *Sea* $f(\mathbf{x})$ *una función separable en sumatorias.*$\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$*, las variables* x_p *y* x_q *interactúan si:*

$\Delta_{\delta,x_p}[f](\mathbf{x}) \mid_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathbf{x}) \mid_{x_p=a,x_q=b_2}$*, donde:*

$\Delta_{\delta,x_p}[f](\mathbf{x}) = f(..., x_p + \delta, ...) - f(..., x_p, ...)$,

refiriéndose a una diferencia positiva de f *respecto a la variable* x_p *con el intervalo* δ.

Dada esta definición, es claro que una interacción entre las variables x_p y x_q, se da si evaluados usando cualquiera de los valores para x_q existe una diferencia.

Esta diferencia será calculada para cada par de variables, creando una matriz de estructura de interacción Λ (ISM). Dada esta matriz, todas aquellas diferencias que sean mayores al umbral ϵ se consideran variables que interactúan, formando la matriz de diseño(DSM) Θ, que contendrá 0 para variables sin interacción y 1 para las que interactúan. Mediante un algoritmo de componentes conexas se puede encontrar la estructura final de cada uno de los grupos de variables que interaccionan en el problema. En el Algoritmo 4.17, se muestra el algoritmo general descrito anteriormente. El detalle del algoritmo esta fuera del ámbito de este trabajo y puede consultarse en DG2 [31], donde también se explica a detalle el cálculo del umbral ϵ.

Algoritmo 4.17 Algoritmo DG2 general

1: {f es la función a descomponer,$límite_{sup}$, $límite_{inf}$ límites de las variables del problema.}
Entrada: f,$límite_{sup}$,$límite_{inf}$
2: $\Lambda = ISM(f, límite_{sup}, límite_{inf})$
3: $\Theta = DSM(\Lambda, f)$
4: $(k, y1, ..., y2) = ConnComp(\Theta)$
5: $x_sep = \{\}, g = 0$
6: **para** $i = 1$ a k **hacer**
7: **si** $|y_i| = 1$ **entonces**
8: $x_sep = x_sep \cup y_i$
9: **si no**
10: $g = g + 1, x_g = y_i$
11: **fin si**
12: **fin para**

4.3.3 Variable Interaction Identification for Constrained Problems (VIIC)

VIIC es un algoritmo de descomposición de problemas de alta dimensión con restricciones. VIIC fue desarrollado por Sayed et. al. [37], proponiendo una forma de evaluar el rendimiento de un esquema de descomposición para un problema establecido de alta dimensión.

El algoritmo de VIIC deriva de la definición de "separabilidad de problemas" [26], la cual dice que, un problema puede descomponerse en m sub-grupos de V variables dependientes y ninguna de las variables pertenece a más de un sub-grupo. La suma de evaluar cada sub-grupo es igual a la evaluación de la solución completa $F(\mathbf{x}) = \sum_{k=1}^{m} F(x_k)$, si m es igual a la dimensión del problema, entonces se dice que el problema es *completamente separable*,

de otra forma, el problema es *parcialmente separable*. Sin embargo, si alguna variable, se encuentra en más de un sub-grupo, entonces los sub-grupos son interdependientes y por lo tanto $F(\mathbf{x}) \neq \sum_{k=1}^{m} F(x_k)$. A continuación se describe VIIC.

VIIC inicia con un arreglo de índices de variables generado de manera aleatoria S_g, tanto para la función objetivo como para cada una de las restricciones. VIIC regresa solo un arreglo de índices de variables S_N para el problema completo. Este arreglo indica cómo las variables son agrupadas en sub-problemas de mínima interdependencia, y máxima interdependencia entre las variables, lo cual es dado por $gprs_{diff}$ en la Ecuación 4.4, el valor buscado por VIIC para $gprs_{diff}$ es cero, significando que la descomposición se realizó de manera correcta, por lo tanto éste es un problema de minimización. El cálculo de $gprs_{diff}$ se realizar de la siguiente manera:

1. Definir el numero de sub-grupos m y la cantidad de variables para cada sub-grupo V.

2. Para calcular $gprs_{diff}$ en la Ecuación 4.4, se requiere de dos valores aleatorios $C_1 > 0$ y $C_2 > 0$ definidos entres los límites del problema.

3. Con los valores $C_1 > 0$ y $C_2 > 0$ crear y evaluar dos vectores usando las Ecuaciones 4.2 y 4.3, donde f es la función objetivo o la restricción correspondiente a descomponer.

4. Calcular la aptitud total ($fit_{all_{C_1C_2}}$) usando la Ecuación 4.1.

5. El siguiente paso es calcular la aptitud de la sumatoria de cada sub-grupo $fit_{groups_{C_1C_2}}$ (Ecuación 4.5), usando la evaluación de cada sub-grupo $k \in m$ $fit_{grp_{C_1C_2=k}}$ (Ecuación 4.6)

6. El cálculo de $fit_{grp_{C_1=k}}$ en la Ecuación 4.7 para cada sub-grupo k, se realiza asignado el valor de $C1$ a las V variables que integran el sub-grupo y el resto $(N - V)$ se asigna el valor $C2$. De esta misma forma se calcula la aptitud de $fit_{grp_{C_2=k}}$ en las Ecuaciones 4.9 y 4.10.

El proceso antes mencionado para el cálculo de $gprs_{diff}$ es realizado tanto para la función objetivo como para cada una de las restricciones del problema. En la Figura 4.3 se muestra de manera gráfica y simplificado el cálculo de

$gprs_{diff}$. En este punto sólo se ha hecho una evaluación de un arreglo de variables.

VIIC es un algoritmo voraz (greedy) que itera generando arreglos aleatorios de variables y evaluándolos por un número máximo de iteraciones $maxgpr_{iter} = m \times 10^4$. Este número de iteraciones se aplica tanto para la función objetivo como para cada restricción, con el objetivo de obtener el mejor arreglo de variables que minimice el valor de $grps_{diff}$.

Los arreglos de variables obtenidos para la función objetivo y para cada una de las restricciones, son almacenados en una matriz de frecuencias ($FrequencyMatrix\ FM$). Esta matriz es utilizada para obtener el arreglo de variables final S_n que representará la descomposición del problema con restricciones completo. Para obtener S_n se contabiliza la frecuencia con que aparecen las variables por cada sub-grupo, aquellas con mayor frecuencia serán las que conformen el sub-grupo $k \leq m$. En la Figura 4.4 se muestra un ejemplo del uso de la matriz de frecuencias. El algoritmo de VIIC se detalla en el Algoritmo 4.18.

Al igual que Random Grouping la desventaja de este método frente a Differential Grouping es que requiere que el usuario ingrese el número de sub-grupos que desea obtener. Además de que no incluye ninguna estrategia para la búsqueda de soluciones potenciales, siendo un algoritmo aleatorio y voraz.

$$fit_{all_{C_1C_2}} = m \times [fit_{all_{C_1}} + fit_{all_{C_2}}] \tag{4.1}$$

$$fit_{all_{C_1}} = f(\mathbf{x} \mid x_i = C_1, \forall i = [1, N]) \tag{4.2}$$

$$fit_{all_{C_2}} = f(\mathbf{x} \mid x_i = C_2, \forall i = [1, N]) \tag{4.3}$$

$$gprs_{diff} = |fit_{all_{C_1C_2}} - fit_{groups_{C_1C_2}}| \tag{4.4}$$

$$fit_{groups_{C_1C_2}} = \sum_{k=1}^{m} fit_{grp_{C_1C_2=k}} \tag{4.5}$$

$$fit_{grp_{C_1C_2=k}} = fit_{C_1grp=k} + fit_{C_2grp=k} \forall k \leq m \tag{4.6}$$

$$fit_{grp_{C_1=k}} = f(\mathbf{x}_{C_1}) \tag{4.7}$$

$$x_{C_1} = \begin{cases} C_1 & \forall x \in V \\ C_2 & \text{de otra manera} \end{cases} \tag{4.8}$$

$$fit_{grp_{C_2=k}} = f(\mathbf{x}_{C_2}) \tag{4.9}$$

$$x_{C_2} = \begin{cases} C_2 & \forall x \in V \\ C_1 & \text{de otra manera} \end{cases} \tag{4.10}$$

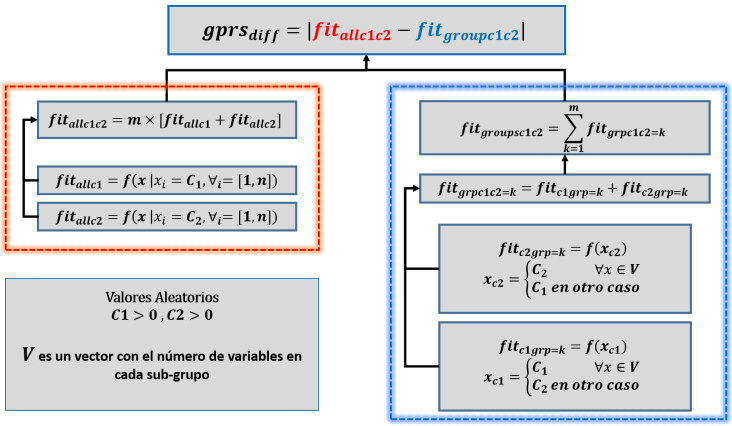


Figura 4.3.: Evaluación de un arreglo de descomposición

4.4 Resumen

En este capítulo se explicó cuál es el principal enfoque de cómputo evolutivo que es utilizado para resolver problemas de alta dimensión, siendo los Algoritmos de Co-evolución Cooperativa (CC). Estos algoritmos constan de tres partes bien definidas en la literatura, las cuales son: (1) descomponer el problema, (2) optimizar cada uno de los sub-grupos o subproblemas con un algoritmo evolutivo, y (3) realizar un proceso de cooperación entre todos los sub-grupos para crear la siguiente generación de soluciones.

Algoritmo 4.18 Algoritmo VIIC

Entrada: m número de sub-grupos

$max\_constr = nc$, si existe una solución factible entonces $max\_constr = nc + 1$
$maxgpr_{iter} = m \times 10^4$, V tamaño de cada sub-grupo m, $c\_num = 0$, $S_N = \emptyset$
mientras $c\_num < max\_constr$ **hacer**
    Inicializar el arreglo S_g, generar dos variables aleatorias $C_1 > 0$ y $C_2 > 0$
    Calcular $fit_{all_{C_1C_2}}$ Ecuación 4.1 y $fit_{grp_{C_1C_2=k}}$ Ecuación 4.6
    Calcular $grps_{diff}$ para cada arreglo S_g con la ecuación 4.4
    mientras $grps_{diff} \neq 0$ y $gpr_{iter} < maxgpr_{iter}$ **hacer**
        Generar un nuevo arreglo de variables de manera aleatoria S'_g
        Calcular $newgrps_{diff}$ para S'_g
        si $newgrps_{diff} < grps_{diff}$ **entonces**
            Actualizar $S_g = S'_g$ y $grps_{diff} = newgrps_{diff}$
        fin si
    fin mientras
    Añadir S_g en la c_num fila de la matriz de frecuencias FM, actualizar $c\_num + 1$
fin mientras
Contar la frecuencia de todas las variables en cada sub-grupo usando FM
Añadir las variables con mayor frecuencia al grupo correspondiente S_N
retornar S_N

Los CC, requieren que el problema se descomponga, y este proceso es por sí mismo un problema de optimización que busca las mejor manera de agrupar las variables. En este sentido, múltiples métodos se han desarrollado para atacar esta problemática, como lo son Random Grouping un método simple pero que ha mostrado ser competitivo, Differential Grouping que es más complejo y requiere de un número elevado de evaluaciones para determinar la interacción de variables, pero con la ventaja de ser un método que requiere ejecutarse sólo una vez, y VIIC una estrategia enfocada en problemas con restricciones que combina una búsqueda aleatoria con una estrategia de evaluación de la interacción de las variables. Estos tres métodos se escogieron para este trabajo por ser los más representativos de la literatura.

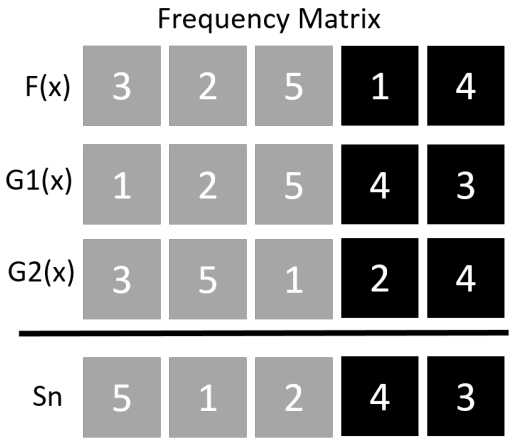


Figura 4.4.: Ejemplo de $FrequencyMatrix$ (FM) para una función hipotética de cinco variables y dos restricciones. El número de sub-grupos buscado es $m = 2$. En el primer sub-grupo de variables (gris) aquellas variables con mayor frecuencia son 5,1 y 2. El segundo sub-grupo (negro) las variables más repetidas son 4 y 3. Las variables 1 y 2 en el segundo sub-grupo con son consideras, ya que estas se encuentran asignadas al primer sub-grupo

5 Búsqueda Local Cooperativa (LoCoS)

> *I have no special talent. I am only passionately curious.*
>
> — **Albert Einstein**

5.1 Introducción

En este capítulo se presentara las propuestas de algoritmos para resolver problemas de alta dimensión con restricciones, y se expondrán las mejoras realizadas al método de descomposición VIIC para mejorar su rendimiento.

Con la finalidad de mantener claro el desarrollo experimental, las herramientas utilizadas para la evaluación de las propuestas y comparación con los algoritmos del estado del arte se presentarán en esta sección introductoria al capítulo.

5.1.1 Conjunto de prueba de alta-dimensión con restricciones

Para evaluar los algoritmos evolutivos, generalmente se utilizan conjuntos de funciones de prueba que son diseñados considerando diferentes características que cambian la dificultad de resolver cada problema.

En el ámbito de problemas con restricciones, múltiples conjuntos de prueba han sido propuestos, especialmente para las competencias del Congreso en Cómputo Evolutivo (CEC) de la IEEE. En el 2006 se propone un conjunto de prueba [20] de 24 problemas con restricciones, y respecto a la dimensionalidad el problema No. 20 presenta la mayor cantidad de variables con 24. Otros conjuntos de prueba destacados son el CEC2010 [24], que cuenta con 18 problemas con restricciones que pueden ser evaluados en las dimensiones

10 y 30. Finalmente el ultimo conjunto de problemas fue publicado para la competencia del CEC2017 [44], que consta de 28 funciones de prueba y que puede ser resuelto en las dimensiones 10, 30, 50 y 100.

En el ámbito de problemas de alta dimensión sin restricciones, existe el conjunto de problemas del CEC2013 [19], que consta de 18 problemas y que se resuelve con una dimensión de 1000 variables. De acuerdo a la revisión de la literatura el único conjunto de pruebas que considera restricciones y fue diseñado como problemas de alta dimensión es el propuesto por Emman Sayed en 2015 [37]. Este conjunto consta de 18 problemas que pueden resolverse en dimensión 100, 500 y 1000. Los 18 problemas son el resultado de combinar 6 funciones objetivo con una, dos o tres funciones de restricción. En la Tabla 5.1, se resumen las características de separabilidad de las funciones objetivo y restricciones. Del mismo modo, también se muestra cómo se generan los 18 problemas a resolver. Este conjunto de problemas será el que se usará para probar el rendimiento de los algoritmos que este trabajo propone. El detalle de las funciones puede encontrarse en el Apéndice A.

Tabla. 5.1.: Resumen de las funciones objetivo y restricciones para el conjunto de prueba Sayed [37]. Los 18 problemas restringidos son generados a partir de la combinación de una de las 6 funciones objetivo, con una, dos o tres restricciones.

Función	Descripción		Objetivo	g1	g2	g3		Objetivo	g1	g2	g3
Obj1	Completamente separable	**F1**	**Obj1**	✓			**F10**	**Obj4**	✓		
Obj2	Parcialmente no separable	**F2**		✓	✓		**F11**		✓	✓	
Obj3	Parcialmente no separable	**F3**		✓	✓	✓	**F12**		✓	✓	✓
Obj4	Parcialmente no separable, sobrelapado	**F4**	**Obj2**	✓			**F13**	**Obj5**	✓		
Obj5	Empalmada no separable, sobrelapado	**F5**		✓	✓		**F14**		✓	✓	
Obj6	Empalmada no separable, sobrelapado	**F6**		✓	✓	✓	**F15**		✓	✓	✓
g1	Separable en grupos de 6 variables	**F7**	**Obj3**	✓			**F16**	**Obj6**	✓		
g2	No separable en grupos de 3 variables	**F8**		✓	✓		**F17**		✓	✓	
g3	Empalmada en pares no separables	**F9**		✓	✓	✓	**F18**		✓	✓	✓

5.1.2 Complejidad en algoritmos evolutivos

Se han propuestos metodologías empíricas que permiten una comparación de la complejidad entre los algoritmos. Una de estas metodologías fue propuesta por Suganthan [41] para la competencia del CEC2005.

Este análisis empírico se realiza de la siguiente forma:

1. Calcular el valor $T0$. Este valor es el tiempo usado para realizar los siguientes cálculos matemáticos $1,000,000$ de veces: $x = x + x$, $x = x/2$,

$x = x \times x$, $x = sqrt(x)$, $x = ln(x)$, $x = exp(x)$, y $y = x/x$. el valor de $x = (double)5{,}55$.

2. Calcular el valor $T1$. Es el tiempo que se toma para evaluar una solución $200,000$ veces.

3. Calcular $T2$. Es el promedio de tiempo de cinco ejecuciones del algoritmo completando $200,000$ evaluaciones de la función.

4. La complejidad del algoritmo es calculada mediante la siguiente expresión $comp = (T2 - T1)/T0$.

5.1.3 Método de clasificación de algoritmos

Con la finalidad de comparar dos o más algoritmos y poder categorizarlos de acuerdo a su rendimiento en el conjunto de problemas de prueba, Elsayed [9] propone una técnica para clasificarlos. Esta técnica utiliza la información del valor de la función objetivo y la factibilidad. Su funcionamiento es descrito a continuación:

Dado un conjunto de algoritmos Z y un conjunto de problemas Y, se obtienen los mejores valores S_{zy}^{best} de cada problema, de acuerdo a la Ecuación 5.1

$$S_{zy}^{best} = \left(1 - \frac{|F_{zy} - BF_y|}{a|BF_y - WF_y|}\right)^{\zeta} \tag{5.1}$$

donde F_{zy} es el mejor valor de aptitud del algoritmo z para la función y. BF_y y WF_y son todos los mejores y peores valores de la función y para todos los algoritmos Z. Los parámetros de control, $a \geq 1$ hace diferencia entre la peor solución factible y cualquier otra no-factible. Por otro lado, $\zeta > 1$ hace énfasis en las buenas soluciones. $S_{zy}^{average}$ refiere a los valores promedios y es calculado de la misma forma que S_{zy}^{best} La puntuación final FS_z se calcula como lo indica la Ecuación 5.2:

$$FS_z = \frac{\sum_{y=1}^{Y} S_{zy}^{best} + \sum_{y=1}^{Y} S_{zy}^{average}}{2} FR_z \tag{5.2}$$

Las puntuaciones finales permiten categorizar los algoritmos de acuerdo a su desempeño general en todo el conjunto de problemas de prueba. Cabe aclarar que esta estrategia no es una prueba estadística.

5.2 Local Cooperative Search (LoCoS)

Después de revisar la literatura, se encontró que la manera de atacar los problemas de alta dimensión es en su mayoría por medio de la estrategia "divide y vencerás", derivando en los algoritmos de Cooperación Co-evolutiva. Estos algoritmos proponen utilizar métodos que permitan dividir el problema en problemas de menor tamaño y optimizar cada uno de ellos mediante algún algoritmo de optimización. Siendo la problemática de la descomposición la que mayor investigación tiene, pocos esfuerzos se han centrado en mejorar los operadores de los algoritmos o en proponer nuevas opciones. Por otra parte, si bien muchos de los métodos de descomposición han sido probados en problemas restringidos, sólo se encontró evidencia de un método enfocado específicamente a problemas con restricciones, siendo este VIIC [37], descrito en detalle en el capítulo anterior.

Observando el éxito de la cooperación co-evolutiva en conjunto con los métodos de descomposición y el rendimiento competitivo de otros enfoques que no utilizan dicha estrategia, como lo son los algoritmos meméticos, en este trabajo se busca diseñar un algoritmo memético que utilice los métodos de descomposición dentro del ámbito de búsqueda local.

Por lo tanto, el trabajo se centrará en proponer un esquema diferente a cooperación co-evolutiva, el cual se le ha llamado "Búsqueda Local Cooperativa" o en inglés "Local Cooperative Search" (LoCoS) como será referido a partir de este momento.

5.2.1 Algoritmo Memético

Antes de entrar a la propuesta y comparación de LoCoS contra el esquema de Cooperación Co-evolutiva, se describirá qué es un algoritmo memético y cuáles son las partes que lo conforman.

Aunque en la literatura hay una delgada línea entre lo que es un algoritmo memético y un algoritmo híbrido, la definición propuesta por Nery y Cota [29] de un algoritmo memético es una de las más aceptadas en la literatura, ya que describe las características que hacen diferentes a los algoritmos meméticos de los híbridos. Esta definición es la siguiente:

"*Los algoritmos meméticos son meta-heurísticas basadas en población, que se componen de un algoritmo evolutivo y un conjunto de algoritmos de búsqueda local , los cuales son activados dentro del ciclo generacional del algoritmo externo en este caso los algoritmo evolutivos.*"

Esta definición es clara, ya que determina bien la manera en que los algoritmos meméticos trabajan. Por un lado tenemos un algoritmo evolutivo que resuelve el problema de manera global, del otro uno o más buscadores locales que tienen por objetivo refinar las soluciones de la población del algoritmo evolutivo, y que se activan dentro del proceso de evolución. Esto último es la característica principal que los diferencia de un algoritmo híbrido, ya que el buscador local no afecta o se combina con los operadores del algoritmo evolutivo, como sí sucede en un algoritmo híbrido.

A partir de la definición anterior, podemos deducir tres fases principales que componen a un algoritmo memético, éstas son:

- Ciclo evolutivo o generación: Este es realizado por el algoritmo evolutivo seleccionado.

- Fase de búsqueda local: el objetivo de esta fase es mejorar una o más soluciones de la población mediante la aplicación de uno o más buscadores locales. Además, esta fase se realiza dentro del ciclo evolutivo.

- Reemplazo del resultado de la fase local: después de aplicar la fase de búsqueda local, el resultado debe afectar a la solución que se buscó mejorar, o ser agregado a la población.

5.2.2 Propuesta Local Cooperative Search (LoCoS)

La propuesta principal es un esquema de diseño de algoritmos basado en la combinación de la descomposición y la fase de búsqueda local en los algoritmos meméticos para resolver problemas de optimización de alta dimensión con restricciones. Ésta es la principal diferencia con el esquema de Cooperación Co-evolutiva (CC), En la Figura 5.1, se muestra un comparativo del funcionamiento general entre LoCoS y CC.

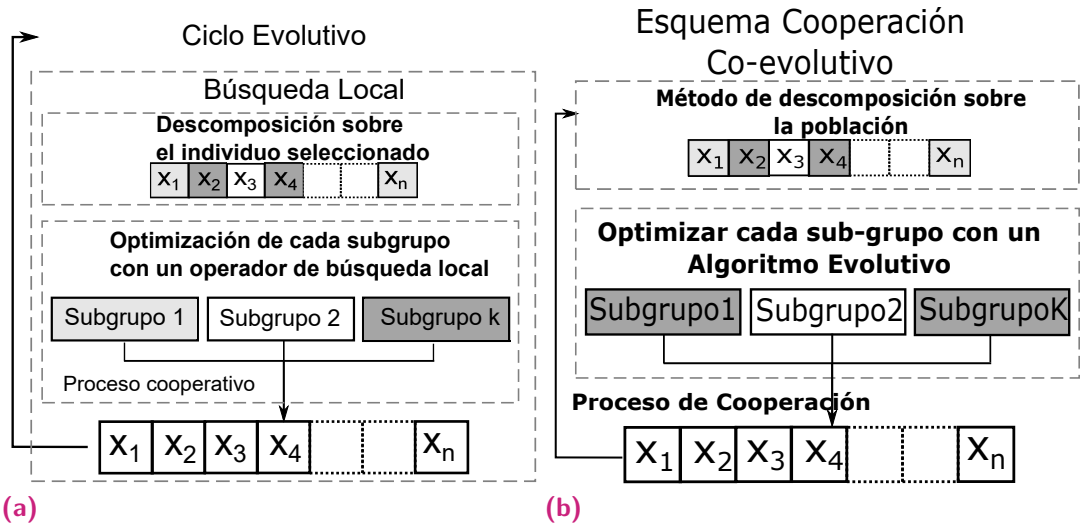


Figura 5.1.: Comparación de los enfoques de Búsqueda Local Cooperativa (LoCoS) y Co-evolución Cooperativa. En 5.1a Enfoque de Búsqueda Local Cooperativa (LoCoS) y en 5.1b el enfoque Co-evolución Cooperativa

Básicamente, como se explicó en el capítulo anterior, CC descompone el problema antes de realizar la optimización, y posteriormente cada uno de los subproblemas es optimizado por un algoritmo evolutivo. En la propuesta LoCoS, la descomposición se realiza cuando se llega a la fase de búsqueda local, optimizando la solución candidata de acuerdo a los sub-grupos obtenidos por el método de descomposición. El intercambio de información entre los sub-grupos puede seguir las mismas estrategias que en CC, por ejemplo, un intercambio secuencial, donde después de optimizar un sub-grupo, la información del sub-grupo optimizado es utilizada para continuar la optimización de los demás grupos de variables, o bien, una estrategia paralela, donde todos los sub-grupos son optimizados al mismo tiempo y al final, la información de todos es unida para generar una única solución.

Una de las características de los algoritmos mémeticos es que la fase de búsqueda local no necesariamente se activa en cada generación del algoritmo evolutivo. Esto implica, que si se toma una estrategia de aplicar la descomposición siempre que esta sea requerida como lo hacen algunos algoritmos CC, el número de evaluaciones utilizadas por el método de descomposición es menor a las realizadas por algoritmos CC tradicionales.

Basados en el esquema LoCoS propuesto, se desarrollaron dos algoritmos. Ambos algoritmos tienen el mismo diseño mémetico, sin embargo la diferencia es la forma en que la fase de búsqueda local será asistida por la información de la descomposición del problema.

Los métodos de descomposición por lo general entregan como resultado un arreglo de variables S_n que indica el orden de las variables, y un vector V utilizado para auxiliar a S_n, proveyendo la información de cuántas variables componen a cada sub-grupo. Por ejemplo, $S_n = [1, 3, 4, 2, 5]$ y $V = [3, 2]$, V indica que existen dos sub-grupos el primero consta de 3 variables y el segundo de 2, y S_n nos dice que al primer sub-grupo S_1 pertenecen las variables $1, 3, 4$ y al segundo $2, 5$.

Partiendo de la información anterior, el primer algoritmo diseñado, únicamente utiliza la información de S_n y V como el orden de las variables que deberá utilizar el buscador local. A este algoritmo se le ha denominado Local Decomposition Search (LoDeS), ya que no existe cooperación entre los diferentes subgrupos. El segundo algoritmo está totalmente basado en LoCoS, la búsqueda local se realiza por sub-grupo, la estrategia de cooperación es secuencial, es decir, los subgrupos de variables se optimizan uno a continuación del otro, y no todos al mismo tiempo.

Algoritmo memético

A continuación se describe el diseño del algoritmo mémetico que utilizará las dos propuestas descritas anteriormente.

- Como buscador global, se utilizó el algoritmo DE/rand/1/bin, descrito en el Capítulo 4.

- El buscador local seleccionado fue Hooke-Jeeves (Capítulo 3) con una pequeña variación, el parámetro $\Delta = stdx(Pop_g)$ es proporcionado por la desviación estándar de cada variable de decisión de la población actual.

- Siempre se selecciona la mejor solución $\vec{x}_{best}$ de la población actual para ser mejorada por el buscador local.

- Con la finalidad de utilizar un máximo del 30 % del total de evaluaciones durante la búsqueda local, la frecuencia de aplicación del buscador local se calcula de la siguiente forma. $Cyclefrecuency = (4 \times D)/(0{,}3 \times NP)$. Donde D es la dimensión del problema y NP es el tamaño de población del algoritmo evolutivo.

- El resultado de la búsqueda local $\vec{x}_{local}$ sustituye a la mejor solución de la población actual si y solo si $\vec{x}_{local}$ es mejor que $\vec{x}_{best}$.

- Para manejar las restricciones, se utilizaron las reglas de factibilidad de Deb [8], descritas en el Capítulo 3.

Ya que el algoritmo seleccionado como buscador global es Evolución Diferencial, los algoritmos propuestos fueron nombrados DE-LoDeS y DE-LoCoS. El algoritmo memético para ambas versiones está descrito en el Algoritmo 5.19. En éste se pueden ver la fase local para cada versión en las líneas 18 y 20.

5.2.3 Experimentación y resultados

Para probar el desempeño de los dos algoritmos propuestos, ambos fueron probados en el conjunto de 18 problemas de prueba [37] descritos en la introducción del capítulo.

Continuando con la metodología de Sayed [37], se diseñaron dos experimentos. (1) El primero tiene como finalidad conocer el número de sub-problemas adecuado para que los algoritmos resuelvan el problema de alta dimensión restringido, y (2) el segundo experimento busca comparar el comportamiento de los algoritmos propuestos contra los algoritmos DEVIIC [37] y DERG [32]. Dado que no se pudieron replicar los resultados reportados por DEVIIC y DERG, se realizó una comparación indirecta, utilizando como medio esta-

Algoritmo 5.19 Algoritmo Memético, DE-LoCoS y DE-LoDeS

1: Asignar el número máximo de evaluaciones de la función $MaxFEs$
2: Calcular la frecuencia de activación de la búsqueda local $Cyclefrecuency = (4 \times D)/(0{,}3 \times NP)$
3: Calcular el máximo número de evaluaciones permitidas para el buscador local $MaxLocalFes = MaxFEs \times 0{,}3$
4: Generar aleatoriamente la población inicial $Pop = (\mathbf{x}_1, \ldots, \mathbf{x}_{NP})$
5: Evaluar la población inicial
6: **repetir**
7: **para** $i = 1$ a NP **hacer**
8: Seleccionar tres vectores aleatorios $r_0 \neq r_1 \neq r_2 \neq i$
9: Calcular $\mathbf{u}_{i,g+1}$ utilizando el operador $rand/1/bin$ (ver Ecuaciones 3.3 & 3.4)
10: **si** $f(\mathbf{u}_{i,g+1})$ es mejor de acuerdo a las reglas de factibilidad $f(\mathbf{x}_i)$ **entonces**
11: $\mathbf{x}_{i,g+1} = \mathbf{u}_{i,g+1}$
12: **fin si**
13: **fin para**
14: **si** $currentCycle \ mod \ Cyclefrecuency == 0$ y $MaxLocalFes$ aún no se alcanzan **entonces**
15: Calcular $\Delta = stdx(Pop_g)$
16: Obtener el vector de descomposición $S_n =$ decomposition(K) (Algoritmo 4.18)
17: **si** DE-LODES **entonces**
18: asignar $\mathbf{x}_{new} =$ HookeJeevesMethod$(\mathbf{x}_{best}, \Delta, S_n)$
19: **si no**, **si** DE-LOCOS **entonces**
20: **para** $k = 1$ a K **hacer**
21: asignar $\mathbf{x}_{new} =$ HookeJeevesMethod$(\mathbf{x}_{new}, \Delta, S_n^k)$
22: **fin para**
23: **fin si**
24: **si** $\mathbf{x}_{new}$ es mejor de acuerdo alas reglas de factibilidad $\mathbf{x}_{best}$ **entonces**
25: $\mathbf{x}_{best} = \vec{x}_{new}$
26: **fin si**
27: **fin si**
28: **hasta que** $MaxFEs$ sean alcanzadas
29: Regresar la mejor solución encontrada

dístico la prueba de Kruskal-Wallis con prueba post-hoc de Bonferroni, con una confianza del 95 %. Esta prueba estadística se realizó con los mejores resultados de cada uno de los problemas.

Experimento 1: análisis del parámetro de sub-grupos de VIIC

Para definir el número de subgrupos k requeridos para VIIC, se realizó un experimento probando diferentes valores $K = \{2, 3, 4, 6, 8\}$, a su vez las pruebas se realizaron en las tres dimensiones que permite el conjunto de problemas (100, 500, y 1000). Ya que este experimento es de calibración para el parámetro k de VIIC, se seleccionó un subconjunto de problemas que son los problemas 3, 6, 9, 12, 15, y 18, ya que estos presentan las 3 restricciones propuestas en el conjunto de pruebas y por lo tanto son los más complicados de resolver. Se realizaron 10 ejecuciones de cada uno de los problemas y el número máximo de evaluaciones se fijó en $MaxFEs = 7{,}5$e+6. Este diseño experimental está basado en [37]. De la misma forma, los parámetros para VIIC y DE fueron los mismos que se describieron por Sayed [37], estos parámetros son descritos a continuación:

- DE/rand/1/bin: Factor de escala $F = 0{,}5$, probabilidad de cruza $CR = 0{,}9$ y tamaño de población $NP = 50$.
- Hooke-Jeeves : Desviación estándar de la población actual como vector de tamaño de paso $\Delta = stdx(Pop)$, criterio de paro $\epsilon = 1{,}0$e-4, y factor de reducción $\alpha = 2$.
- VIIC: Número de diferentes arreglos de variables generados para el objetivo y cada una de las restricciones $maxgpr_{iter} = K \times 10^4$, y $c_1, c_2 \in [l_d, u_d]$.

Resultados

En la Figura 5.2, se muestran los resultados de la prueba post-hoc de Bonferroni. Los resultados de esta prueba sugieren que el valor adecuado de numero de sub-grupos es $k = 2$ ya que este mostró diferencia significativa contra los valores de 6 y 8 sub-grupos. Este resultado es el mismo al que llegaron los autores de VIIC [37]. Por lo tanto, este valor será el utilizado en el siguiente experimento. Es importante destacar que DE-LoDeS no fue afectado por el número de sub-grupos. Esto es una característica a esperar, ya que el esquema LoDeS no optimiza por sub-grupos si no por el orden de las variables que le indica el arreglo de variables S_n.

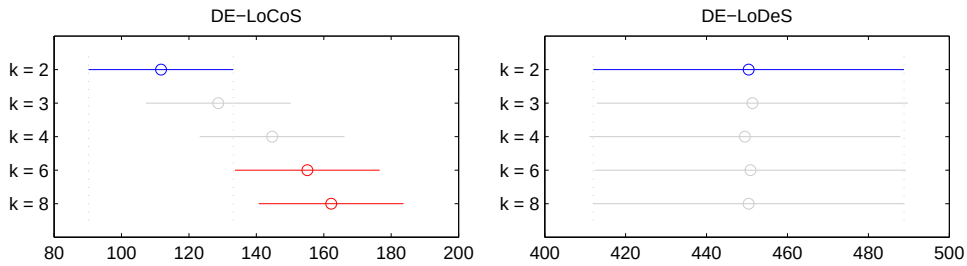


Figura 5.2.: Prueba post-hoc de Bonferroni, para el experimento 1, determinar el número de subgrupos. DE-LoCos y DE-LoDeS

Experimento 2

La configuración del experimento 2, se realizó de acuerdo a la competencia de optimización con restricciones del CEC2010 [24]. Por lo tanto el número máximo de evaluaciones se fijó en $MaxFEs = 20,000 \times D$. Se realizaron veinticinco ejecuciones independientes por cada uno de los problemas del conjunto de pruebas sobre las tres dimensiones.

Los parámetros para los algoritmos de Hooke-Jeeves y VIIC ($K = 2$) son los mismos que los del experimento 1. Sin embargo, los parámetros para el algoritmo de Evolución Diferencial fueron calibrados utilizando la herramienta IRACE [22], los parámetros son los siguientes:

- DE/rand/1/bin: Factor de escala $F = 0{,}51$, probabilidad de cruza $CR = 0{,}73$ y tamaño de población $NP = 56$.

Por otra parte, el análisis empírico de complejidad y la clasificación de los algoritmos descritos en la introducción del capítulo, se realizaron para comparar los algoritmos propuestos contra DEVIIC y DERG. Finalmente, se presenta un análisis de la convergencia de los algoritmos DE-LoDeS y DE-LoCoS.

Resultados

Las Tablas 5.4, 5.5, y 5.6 muestran los resultados estadísticos y el porcentaje de factibilidad (FR) para las dimensiones 100, 500 y 1000 respectivamente. Los resultados muestran que ambos algoritmos propuestos logran una factibilidad del 100 % en todos los problemas y en cualquiera de las dimensiones

probadas. En contraste, DEVIIC logra el 100 % de factibilidad en todas las funciones de dimensión 500, y DERG solo en las funciones F8 y F11 en dimensión 100, en las funciones F1, F4, F5, F7, F10, F13, F16 y F17 en dimensión 500, y de dimensión 1000 en las funciones F1, F4, F7, F10, F13 y F16.

En dimensión 100 (Tabla 5.4), todos los algoritmos logran alcanzar el mismo valor mínimo en las funciones F2 y F3, sin embargo, DE-LoDeS y DE-LoCoS muestran una desviación estándar menor que DEVIIC y DERG. En las funciones F1, F5, F6, F9, F11, F12, F14, F15, F17, y F18, los algoritmos propuestos mejoran los resultados obtenidos por DVIIC y DERG Finalmente, DE-LoCoS logra encontrar mejores resultados que los demás algoritmos en las funciones F4, F8, F10, F13, F16.

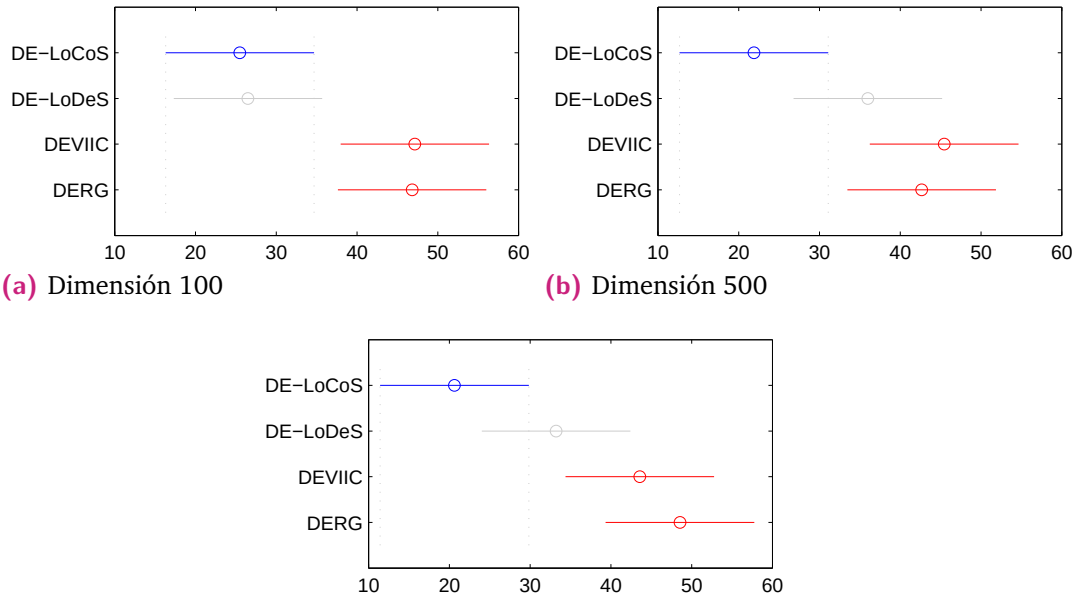
Para la dimensión 500 (Tabla 5.5), en las funciones F1, F4 a F10, F12, F16, F17, y F18, DE-LoCoS supera a los otros algoritmos de acuerdo al mejor valor encontrado y a la desviación estándar. Sin embargo, en la función F3 DVIIC logra un mejor rendimiento que cualquiera de los otros algoritmos.

Respecto a la dimensión 1000 (Tabla 5.6), el rendimiento de DE-LoCoS fue mejor en la mayoría de las funciones contra los demás algoritmos. Sólamente en las funciones F10, F11, y F12, su homólogo DE-LoDeS logra alcanzar los mismos valores.

En la Figura 5.3 se muestran los resultados de la prueba post-hoc de Bonferroni aplicada a este experimento. Para la dimensión 100 (Fig. 5.3a), las propuestas DE-LoCoS y DE-LoDeS no muestran diferencia significativa en su comportamiento, pero sí la muestran contra DEVIIC y DERG. Para las dimensiones 500 (Fig. 5.3b) y 1000 (Fig. 5.3c), el comportamiento entre los algoritmos propuestos continúa siendo similar, sin embargo, en estas dimensiones sólo DE-LoCoS supera a las versiones de CC.

Complejidad Empírica y Categorización de los algoritmos

La complejidad de los algoritmos DEVIIC y DERG fueron tomadas de la literatura [37] en donde sólo se reportan los valores para las funciones F1 a F6 en dimensión 500. Por lo cual, la complejidad de DE-LoCoS y DE-



(a) Dimensión 100 **(b)** Dimensión 500

(c) Dimensión 1000

Figura 5.3.: Comparación múltiple de rangos de medias utilizando la prueba Post-Hoc de Bonferroni.

LoDeS fue calculada para la misma dimensión y funciones. Los resultados se muestran en la Tabla 5.2.

Los resultados revelan que el esquema propuesto es aproximadamente 2.5 veces más complejo que los algoritmos basados en CC. Si bien se mencionó que la cantidad de evaluaciones utilizadas por VIIC se reduciría, el hecho es que la inclusión de una búsqueda local incrementa la complejidad de un algoritmo evolutivo. Sin embargo, este crecimiento de complejidad puede ser justificado por los resultados finales obtenidos, ya que la factibilidad fue de un 100 % en las tres dimensiones probadas. Además, la calidad del resultado final fue mejor en la mayoría de las funciones comparado con las obtenidas por las versiones CC.

Conforme a la categorización de los algoritmos, considerando el hecho de que la comparación entre los algoritmos CC de la literatura y los propuestos en este trabajo es un estudio indirecto, los mismos parámetros para el método de categorización $a = 1{,}001$ y $\zeta = 2$ fueron tomados de [37].

Los puntajes finales (FS) y la probabilidad de factibilidad de los cuatro algoritmos para cada una de las dimensiones probadas, se presentan en la Tabla 5.3.

Tabla. 5.2.: Comparación de Complejidad Empírica [41], los valores representan el tiempo (milisegundos) promedio para ejecutar el algoritmo con un límite de 200,000 evaluaciones. En negrita se resaltan los menores tiempos.

	F1			F2			F3		
	D100	D500	D1000	D100	D500	D1000	D100	D500	D1000
DE-LoCoS	33.56	135.7675	230.01	32.6975	110.44	190.55	32.2725	97.3375	196.3075
DE-LoDeS	29.7475	128.7075	241.6875	30.3	109.1325	182.1925	31.2325	92.6075	192.315
DEVIIC	**14.30423**	44.89417	81.13926	**11.26537**	**44.07581**	78.24339	**11.44087**	**43.61048**	**77.30665**
DERG	17.55115	**44.23293**	**80.42092**	11.33477	44.39214	**77.54337**	11.91235	48.0554	85.10668
	F4			F5			F6		
	D100	D500	D1000	D100	D500	D1000	D100	D500	D1000
DE-LoCoS	30.8175	98.0525	144.085	30.7775	97.675	148.3075	30.885	98.0975	154.3475
DE-LoDeS	30.4125	94.9975	149.04	30.53	93.0075	144.5625	30.2575	96.4525	151.3175
DEVIIC	14.37971	42.48397	76.1842	14.53068	**42.62068**	**78.2964**	**14.70415**	41.18799	**78.39024**
DERG	**14.26132**	**42.44519**	**76.08006**	**14.41032**	42.67988	81.18622	15.50621	**40.93086**	80.87395

Se puede observar que la factibilidad de DE-LoCoS y DE-LoDeS no decae cuando la dimensión se incrementó a diferencia de DVIIC y DERG. Sin embargo, en dimensión 500, DE-LoDeS quedó en tercer lugar superado por DERG y DVIIC. Esto se debe a que la calidad de las soluciones de DE-LoDeS en dimensión 500 no son equiparables a los de los otros algoritmos. Estos resultados muestran que, en efecto, la cooperación entre subgrupos utilizada por DE-LoCoS ayuda a mejorar el rendimiento del esquema que se propone.

Tabla. 5.3.: Categorías de los modelos utilizando el método propuesto por Elsayed [9]. FS representa el puntaje final del algoritmo. FR el promedio de factibilidad del algoritmo

	100D		500D		1000D	
	FS	FR	FS	FR	FS	FR
DE-LoCoS	**18.00**	100 %	**16.35**	100 %	**18.00**	100 %
DE-LoDeS	17.00	100 %	4.88	100 %	11.10	100 %
DEVIIC	9.06	100 %	7.02	100 %	7.36	98 %
DERG	1.84	89 %	6.81	95 %	0.50	81 %

Convergencia

Se comparó la convergencia de DE-LoCoS y DE-LoDeS con la finalidad de observar el comportamiento de ambas propuestas. En la Figura 5.4 se muestran gráficas de convergencia representativas, estas pertenecen a las funciones F6 y F18, ambas con tres restricciones, y con variables parcialmente no separables para la función F6 y variables no separables con superposición para la

función F18. Dichas gráficas de convergencia fueron tomadas de la ejecución correspondiente al valor de la mediana de las 25 ejecuciones.

En dimensión 500 y 1000 para la función F6, DE-LoDeS presenta un comportamiento de convergencia prematura, desde generaciones tempranas el algoritmo alcanza un mínimo. En la función F18 en cualquier dimensión, ambos algoritmos alcanzan valores objetivos similares, pero la velocidad de convergencia de DE-LoCoS es mayor en ambas funciones y en las tres dimensiones, sin sufrir de convergencia prematura como lo hace DE-LoDeS.

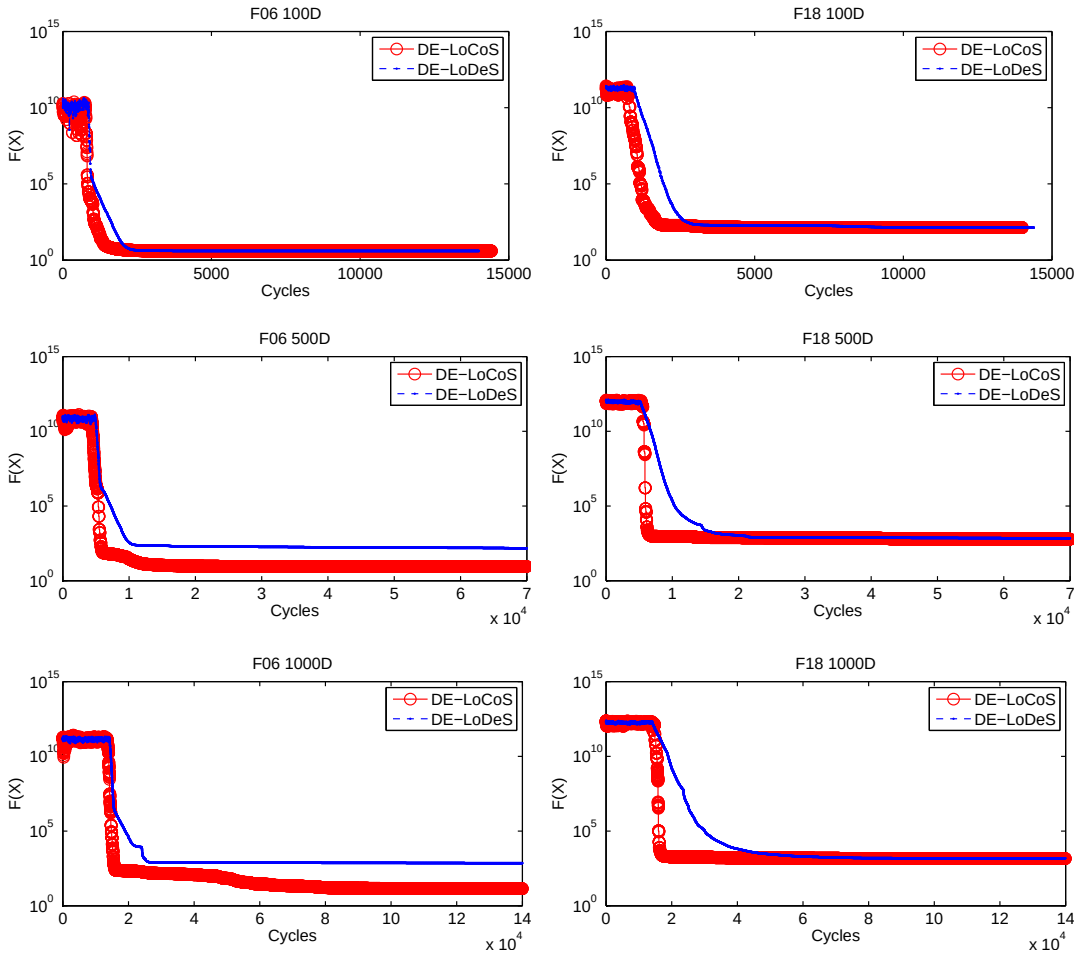


Figura 5.4.: Gráficas de convergencia de las funciones F06 y F18 para las dimensiones 100, 500 y 1000

5.2.4 Conclusiones

En esta sección se presentó un algoritmo memético donde su fase de búsqueda local es guiada por la descomposición del problema en sub-problemas. Este algoritmo fue diseñado para resolver problemas de alta dimensión con

Tabla. 5.4.: DELoCoS y DELoDeS con VIIC: Resultados Estadísticos dimensión 100. Se muestra el Mejor, Mediana, std (desviación estandar) y el porcentaje de factibilidad (FR). En Negrita se resaltan los mejores valores para cada estadística

	F1				F2				F3			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	**0.00E+00**	**0.00E+00**	1.90E-98	5.98E-89	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**	**5.00E+00**	**5.00E+00**	**5.00E+00**	**5.00E+00**
Mediana	**0.00E+00**	**0.00E+00**	8.86E-86	6.00E-87	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**	**5.00E+00**	**5.00E+00**	**5.00E+00**	**5.00E+00**
std	**0.00E+00**	**0.00E+00**	1.78E-83	6.89E-85	1.36E-15	**6.54E-16**	3.98E-08	4.32E+03	2.72E-15	**9.06E-16**	4.27E-08	5.42E-08
FR	**100 %**	**100 %**	**100 %**	92 %	**100 %**	**100 %**	**100 %**	84 %	**100 %**	**100 %**	**100 %**	84 %
	F4				F5				F6			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	4.19E-30	**0.00E+00**	4.14E-03	3.02E-01	**3.00E+00**	**3.00E+00**	3.02E+00	3.17E+00	**4.00E+00**	**4.00E+00**	4.11E+00	4.13E+00
Mediana	1.48E-28	**0.00E+00**	3.95E+00	7.41E+00	**3.00E+00**	**3.00E+00**	7.01E+00	7.31E+00	**4.00E+00**	**4.00E+00**	7.24E+01	1.21E+01
std	2.43E-26	**1.17E-30**	7.84E+00	2.50E+07	3.69E-13	**1.36E-15**	9.74E+00	3.43E+01	4.35E-15	**1.68E-15**	3.96E+01	4.14E+01
FR	**100 %**	**100 %**	**100 %**	74 %	**100 %**	**100 %**	**100 %**	84 %	**100 %**	**100 %**	**100 %**	92 %
	F7				F8				F9			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	**3.87E-36**	1.99E-30	4.43E-03	3.38E-03	1.51E-42	**9.58E-220**	1.07E-05	4.00E+00	**2.00E+00**	**2.00E+00**	5.99E+00	2.00E+00
Mediana	**4.54E-30**	2.23E-26	2.90E+01	5.22E+00	3.95E-30	**1.23E-32**	2.84E+01	8.08E+00	**2.00E+00**	**2.00E+00**	1.43E+01	1.00E+01
std	**3.71E-27**	2.03E-19	3.74E+01	1.78E+01	8.85E-30	**5.25E-31**	1.99E+01	3.25E+03	**4.53E-16**	**4.53E-16**	1.88E+01	1.26E+01
FR	**100 %**	**100 %**	**100 %**	84 %	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	96 %
	F10				F11				F12			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	5.93E-20	**0.00E+00**	9.22E-02	1.36E+00	**3.00E+00**	**3.00E+00**	3.24E+00	3.82E+00	**4.00E+00**	**4.00E+00**	4.30E+00	4.70E+00
Mediana	4.74E-17	**2.68E-268**	4.06E+00	5.78E+00	**3.00E+00**	**3.00E+00**	8.63E+00	1.16E+01	**4.00E+00**	**4.00E+00**	8.09E+01	8.20E+01
std	4.85E-15	**1.99E-30**	1.43E+01	3.03E+01	1.06E-09	**1.28E-15**	5.19E+01	4.59E+01	3.42E-07	**2.11E-15**	4.24E+01	1.48E+09
FR	**100 %**	**100 %**	**100 %**	88 %	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	96 %
	F13				F14				F15			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	3.51E-11	**1.91E-23**	4.59E-02	8.08E-01	**3.00E+00**	**3.00E+00**	3.16E+00	4.04E+00	**4.00E+00**	**4.00E+00**	4.01E+00	4.70E+00
Mediana	3.00E-08	**9.72E-19**	8.03E+00	1.34E+01	**3.00E+00**	**3.00E+00**	1.42E+01	1.71E+01	**4.00E+00**	**4.00E+00**	2.01E+01	2.12E+01
std	1.70E-06	**1.14E-14**	4.48E+01	1.86E+02	8.34E-04	**1.39E-04**	1.25E+02	7.24E+04	2.12E-03	**1.79E-05**	7.49E+01	8.96E+01
FR	**100 %**	**100 %**	**100 %**	88 %	**100 %**	**100 %**	**100 %**	92 %	**100 %**	**100 %**	**100 %**	96 %
	F16				F17				F18			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	1.37E+02	**8.17E+01**	**8.17E+01**	**8.17E+01**	**1.37E+02**	**1.37E+02**	1.46E+02	1.49E+02	**1.37E+02**	**1.37E+02**	1.47E+02	1.46E+02
Mediana	1.37E+02	**1.37E+02**	1.71E+02	2.47E+02	**1.37E+02**	**1.37E+02**	2.41E+02	2.25E+02	**1.37E+02**	**1.37E+02**	3.06E+02	3.18E+02
std	2.50E-01	**2.31E+01**	1.03E+02	2.76E+05	**3.87E-01**	4.75E-01	1.84E+02	1.74E+02	7.66E-01	**7.81E-04**	1.47E+02	5.92E+07
FR	**100 %**	**100 %**	**100 %**	84 %	**100 %**	**100 %**	**100 %**	76 %	**100 %**	**100 %**	96 %	92 %

restricciones. Bajo este esquema, se desarrollaron dos algoritmos DE-LoDeS y DE-LoCoS. Ambos utilizan DE/rand/1/bin como buscador local, y el método de Hooke-Jeeves como buscador local. El método de descomposición seleccionado fue VIIC. Estos algoritmos fueron probados en un conjunto de problemas de prueba diseñado para ser evaluado en dimensiones 100, 500 y 1000. Finalmente, los resultados fueron comparados contra dos algoritmos del estado del arte, DEVIIC y DERG.

De los resultados se puede concluir que el incremento de la dimensión no afecta la factibilidad de DE-LoCoS y DE-LoDeS. Sin embargo DE-LoDeS disminuye la calidad de sus soluciones al incrementar la dimensión. La diferencia entre estos dos algoritmos es la forma de usar la información de la descomposición. Y los resultados remarcan que la cooperación de los sub-grupos de variables mejora el rendimiento.

Partiendo de lo anterior y conociendo que el buscador local Hooke-Jeeves es un buscador de trayectoria y que se ha probado que su desempeño mejora cuando el problema es separable, se puede concluir que la descomposición del

Tabla. 5.5.: DELoCoS y DELoDeS con VIIC: Resultados Estadísticos dimensión 500. Se muestra el Mejor, Mediana, std (desviación estandar) y el porcentaje de factibilidad (FR). En Negrita se resaltan los mejores valores para cada estadística

	F1				F2				F3			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	2.69E-302	**0.00E+00**	3.34E-27	1.81E-27	**6.00E+00**	**6.00E+00**	**6.00E+00**	**6.00E+00**	1.00E+01	1.00E+01	**1.81E-27**	1.00E+01
Mediana	1.13E-276	**0.00E+00**	4.29E-26	8.53E-25	**6.00E+00**	**6.00E+00**	**6.00E+00**	**6.00E+00**	1.00E+01	1.00E+01	1.83E+02	1.00E+01
std	**0.00E+00**	**0.00E+00**	1.61E-23	1.03E+00	1.61E-11	**3.51E-13**	5.44E-08	1.80E+03	1.42E-11	**2.80E-12**	6.15E+03	1.60E-07
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	88 %	**100 %**	**100 %**	**100 %**	84 %
	F4				F5				F6			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	2.23E+01	**3.02E-16**	7.70E+00	9.80E+00	5.44E+01	**6.00E+00**	2.74E+01	2.64E+01	8.04E+01	**9.00E+00**	2.33E+01	2.36E+01
Mediana	5.23E+01	**6.08E-13**	2.00E+02	4.06E+02	1.24E+02	**6.00E+00**	2.95E+02	2.27E+02	1.46E+02	**9.00E+00**	2.99E+02	2.99E+02
std	2.57E+01	**5.45E-07**	4.35E+02	4.54E+02	3.91E+01	**1.06E-02**	2.96E+02	4.01E+02	5.28E+01	**6.41E-02**	4.34E+02	3.95E+02
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	84 %
	F7				F8				F9			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	4.25E+01	**2.05E-01**	3.30E+02	1.83E+02	8.38E+01	**2.32E-02**	1.71E+02	1.31E+02	1.43E+02	**4.02E+00**	1.83E+02	1.15E+02
Mediana	2.16E+02	**3.04E+01**	5.42E+02	3.92E+02	1.71E+02	**1.15E+00**	3.54E+02	2.55E+02	2.68E+02	**4.54E+00**	3.05E+02	3.03E+02
std	**8.31E+01**	1.15E+02	1.48E+02	1.09E+02	6.95E+01	**1.99E+01**	1.08E+02	7.30E+01	8.87E+01	**4.04E+01**	7.99E+01	7.89E+02
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	92 %	**100 %**	**100 %**	**100 %**	96 %
	F10				F11				F12			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	1.68E+02	**6.80E+01**	8.04E+01	8.21E+01	1.74E+02	**7.40E+01**	**7.40E+01**	7.80E+01	1.77E+02	**7.60E+01**	8.00E+01	7.61E+01
Mediana	2.22E+02	**1.18E+02**	2.20E+02	2.05E+02	2.08E+02	**7.40E+01**	2.56E+02	2.09E+02	2.33E+02	**1.26E+02**	2.24E+02	2.35E+02
std	3.49E+01	**3.39E+01**	1.57E+02	4.68E+03	3.86E+01	**3.28E+01**	1.98E+02	3.20E+02	4.52E+01	**3.24E+01**	1.99E+02	3.21E+08
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	96 %	**100 %**	**100 %**	**100 %**	96 %
	F13				F14				F15			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	2.16E+02	**5.06E+01**	1.15E+02	5.06E+01	2.39E+02	**5.66E+01**	5.66E+01	7.98E+01	2.58E+02	**6.02E+01**	6.02E+01	8.06E+01
Mediana	2.80E+02	**5.06E+01**	3.24E+02	2.56E+02	3.09E+02	**5.66E+01**	4.03E+02	2.73E+02	3.21E+02	**6.02E+01**	4.34E+02	5.03E+02
std	3.16E+01	**2.58E-13**	3.19E+02	4.18E+02	4.04E+01	**7.01E-12**	1.93E+02	1.01E+07	3.36E+01	**1.23E-11**	4.05E+02	4.23E+02
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	96 %	**100 %**	**100 %**	**100 %**	92 %
	F16				F17				F18			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	6.83E+02	**5.72E+02**	1.02E+03	9.04E+02	6.89E+02	**5.99E+02**	1.04E+03	1.02E+03	6.84E+02	**6.01E+02**	9.44E+02	1.18E+03
Mediana	7.26E+02	**5.73E+02**	1.35E+03	1.28E+03	7.42E+02	**6.07E+02**	1.35E+03	1.34E+03	7.01E+02	**6.05E+02**	1.48E+03	1.46E+03
std	2.93E+01	**2.37E+01**	2.64E+02	8.74E+04	**3.16E+01**	3.68E+01	4.40E+02	3.06E+02	**1.47E+01**	4.32E+01	2.03E+02	2.46E+06
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	92 %

problema como se aplica en LoCoS permite a Hooke-Jeeves una exploración adecuada de la vecindad de cada uno de los sub-grupos. Esto también es fundamentado por el comportamiento mostrado por DE-LoCoS en las gráficas de convergencia, donde se observó una convergencia rápida pero sin llegar a ser prematura o mostrar algún efecto de estancamiento.

Los resultados estadísticos, en específico la baja desviación estándar de DE-LoCoS y DE-LoDeS indican que ambos son robustos y estables al realizar su búsqueda. Comparado con los algoritmos DEVIIC y DERG, los algoritmos propuestos tienen un costo computacional más elevado de acuerdo al método empírico para medir la complejidad. Ello derivado de que un algoritmo memético por lo regular siempre es más costoso que un algoritmo evolutivo tradicional. Por otra parte, el método de Hooke-Jeeves explora toda la vecindad de la solución a mejorar e itera sobre ésto hasta que ya no exista mejora o converger, lo cual supone un gasto mayor de evaluaciones. Por lo tanto el parámetro que controla la frecuencia de aplicación puede ayudar a reducir esta desventaja.

Tabla. 5.6.: DELoCoS y DELoDeS con VIIC: Resultados Estadísticos dimensión 1000. Se muestra el Mejor, Mediana, std (desviación estandar) y el porcentaje de factibilidad (FR). En Negrita se resaltan los mejores valores para cada estadística

	F1				F2				F3			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	6.39E-283	**0.00E+00**	1.90E-11	1.50E-10	**9.00E+00**	**9.00E+00**	**9.00E+00**	**9.00E+00**	**1.50E+01**	**1.50E+01**	**1.50E+01**	1.52E+01
Mediana	4.55E-218	**0.00E+00**	3.66E-09	5.12E-09	**9.00E+00**	**9.00E+00**	**9.00E+00**	9.09E+00	**1.50E+01**	**1.50E+01**	**1.50E+01**	7.50E+02
std	**0.00E+00**	**0.00E+00**	5.92E-08	3.64E-08	3.60E-11	**1.72E-12**	1.03E-01	1.34E+04	5.03E-11	**7.65E-12**	1.70E+00	3.12E+04
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	92 %	**100 %**	**100 %**	**100 %**	68 %
	F4				F5				F6			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	4.05E+02	**1.12E-09**	7.02E+02	3.74E+02	5.11E+02	**9.00E+00**	1.86E+02	5.16E+02	5.11E+02	**1.40E+01**	2.08E+02	1.84E+03
Mediana	5.37E+02	**4.89E-05**	1.63E+03	1.15E+03	6.46E+02	**9.00E+00**	1.33E+03	2.00E+03	7.13E+02	**1.42E+01**	1.56E+03	8.69E+03
std	6.39E+01	**1.32E-01**	7.70E+02	6.38E+02	7.61E+01	**5.92E-02**	8.07E+02	7.81E+07	8.67E+01	**8.35E-01**	7.11E+03	3.57E+05
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	68 %	**100 %**	**100 %**	92 %	64 %
	F7				F8				F9			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	1.20E+03	**2.69E-03**	1.08E+03	8.04E+02	9.31E+02	**1.48E-05**	5.53E+02	6.15E+02	1.12E+03	**5.00E+00**	6.33E+02	1.68E+03
Mediana	1.52E+03	**2.51E+00**	1.34E+03	1.19E+03	1.22E+03	**7.19E-04**	8.73E+02	8.52E+02	1.30E+03	**5.62E+00**	9.60E+02	7.81E+04
std	1.30E+02	**9.07E+01**	2.42E+02	3.32E+02	1.53E+02	**1.56E+00**	1.56E+02	2.79E+09	1.14E+02	**3.09E+01**	1.99E+04	2.36E+10
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	72 %	**100 %**	**100 %**	**100 %**	68 %
	F10				F11				F12			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	**1.41E+02**	**1.41E+02**	3.10E+02	2.84E+02	**1.47E+02**	**1.47E+02**	2.21E+02	5.75E+02	**1.54E+02**	**1.54E+02**	2.53E+02	8.04E+02
Mediana	**1.42E+02**	1.64E+02	6.54E+02	5.07E+02	**1.53E+02**	1.70E+02	6.45E+02	1.24E+03	**1.73E+02**	1.77E+02	1.43E+08	1.16E+04
std	**7.34E+00**	3.47E+01	2.40E+02	1.96E+02	**1.04E+01**	1.90E+01	3.57E+02	1.55E+10	1.69E+01	**1.65E+01**	6.71E+08	2.70E+09
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	60 %	**100 %**	**100 %**	88 %	76 %
	F13				F14				F15			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	2.84E+02	**1.54E+02**	3.07E+02	3.73E+02	2.91E+02	**2.20E+02**	4.97E+02	5.78E+02	3.44E+02	**2.53E+02**	4.08E+02	7.07E+02
Mediana	3.14E+02	**2.45E+02**	7.73E+02	6.31E+02	3.45E+02	**2.51E+02**	7.09E+02	1.17E+03	4.13E+02	**2.83E+02**	9.42E+02	5.97E+03
std	4.68E+01	3.65E+01	5.74E+02	**3.16E+02**	4.28E+01	**2.90E+01**	3.75E+02	2.79E+03	3.31E+01	**2.56E+01**	1.41E+03	1.83E+08
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	80 %	**100 %**	**100 %**	96 %	56 %
	F16				F17				F18			
	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG	DE-LoDeS	DE-LoCoS	DEVIIC	DERG
Mejor	1.38E+03	**1.12E+03**	3.60E+03	2.63E+03	1.38E+03	**1.30E+03**	2.78E+03	3.50E+03	1.40E+03	**1.32E+03**	2.46E+03	1.23E+04
Mediana	1.46E+03	**1.20E+03**	4.53E+03	3.13E+03	1.47E+03	**1.44E+03**	4.72E+03	4.51E+04	1.51E+03	**1.51E+03**	3.59E+03	1.61E+07
std	**5.82E+01**	8.59E+01	3.27E+03	6.13E+07	**4.56E+01**	9.23E+01	3.47E+04	5.84E+09	**6.31E+01**	1.21E+02	4.27E+07	4.09E+09
FR	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	68 %	**100 %**	**100 %**	92 %	80 %

Mejorando VIIC 6

6.1 Introducción

Como se explicó en la sección 4.3.3, VIIC es un método de descomposición de problemas con restricciones, tratando de encontrar el mejor arreglo de variables para este propósito. Analizando el proceso del algoritmo VIIC, se encontró que el algoritmo posee características que pueden ser mejoradas, y se listan a continuación:

(a) VIIC optimiza más de un vector de arreglo de variables, es decir, uno por la función objetivo y uno por cada una de las restricciones del problema.

(b) La creación de nuevos vectores de arreglo de variables es realizado de manera aleatoria, es decir, no se usa la información del vector actual.

(c) La búsqueda es realizada de manera voraz (greedy). El algoritmo no utiliza ninguna estrategia para guiar la búsqueda.

Basados en estos tres puntos, algunos cambios que buscan mejorar el rendimiento son propuestos a continuación.

6.2 Optimizando un solo arreglo de variables

El primer punto a considerar es la creación de múltiples vectores de arreglo de variables. El método dice que en cada iteración se generen tantos arreglos de variables como funciones objetivo y restricciones tenga el problema, y de esta forma, utilizar la información de cada uno de los arreglos para generar una matriz de frecuencias donde se combinan y generar un único arreglo de variables. Si el problema crece en número de restricciones, la matriz

de frecuencias será de mayor tamaño y el tiempo del proceso de unir la información es mayor.

Para evitar este problema, se propuso evaluar un solo arreglo de variables. Permitiendo de este modo, prescindir de la matriz de frecuencias. Con la finalidad de modificar la manera de evaluar un arreglo de variables, es necesario revisitar la definición de un problema con restricciones, y el concepto de separabilidad en sumas de variables.

La definición de un problema con restricciones fue dada en la sección 3.1, por claridad se definirá una vez más:

Definición 4 *Un problema de optimización mono-objetivo con restricciones puede definirse como: encontrar un vector* $\mathbf{x}$ *de dimensión* D *que minimice una función* $f(\mathbf{x})$ *sujeto a restricciones de desigualdad* $g(\mathbf{x})$ *y/o de igualdad* $h(\mathbf{x})$. *Ecuación 6.1.*

$$\begin{aligned} &\textit{minimizar } f(\mathbf{x}), \quad && \mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D \\ & && l_i \leq x_i \leq u_i, i = 1, \ldots, D \\ &\textit{sujeto a:} && \\ & && g_j(\mathbf{x}) \leq 0, \textit{ for } j = 1, \ldots, q \\ & && h_l(\mathbf{x}) = 0, \textit{ for } l = 1, \ldots, m \end{aligned} \tag{6.1}$$

donde q *y* m *refieren al número de restricciones de desigualdad e igualdad respectivamente.* D *es la dimensión del problema. El espacio de búsqueda* $\mathbb{S}$ *está definido por los límites inferiores* $\mathbf{l}$ *y superiores* $\mathbf{u}$*, mientras que la región factible es definida como el subconjunto de soluciones que satisfacen las restricciones del problema* $\mathbb{F} \subseteq \mathbb{S}$.

Partiendo de esta definición, comúnmente en el área de cómputo evolutivo, las restricciones se manejan mediante una suma de éstas. Por lo tanto una solución es factible cuando la suma de violación de restricciones cvs es cero. Para calcular este valor se utiliza la Ecuación 6.2:

$$cvs(\mathbf{x}) = \sum_{j}^{q} max(0, g_j(\mathbf{x})) + \sum_{l}^{m} max(0, |h_l(\mathbf{x})| - \epsilon) \tag{6.2}$$

donde las restricciones de igualdad son transformadas a desigualdad de la siguiente forma $(|h_l(\mathbf{x})| - \epsilon \leq 0)$, y $\epsilon = 1e^{-4}$ es una pequeña tolerancia para relajar la restricción.

Las siguientes definiciones refieren a la separabilidad de variables en sumas, de forma completa y parcial:

Definición 5 ***Completamente separable:*** *Se dice que una función F es completamente separable en forma de sumas, si existen funciones f_1, f_2, f_3, ..., f_d para cada una de las variables, de forma que:*

$$F(\mathbf{x}) = f_1(x_1) + f_2(x_2) + f_3(x_3) + ... + f_d(x_d); \tag{6.3}$$

Definición 6 ***Parcialmente separable:*** *Se dice que una función F es parcialmente separable en forma de sumas, cuando las variables pueden ser agrupadas en conjuntos disjuntos. Suponiendo que se tienen dos conjuntos de variables uno A y otro B, un problema parcialmente separable puede expresarse de la siguiente manera:*

$$F(\mathbf{x}) = f_a(x_i, i \in A) + f_b(x_i, i \in B); \tag{6.4}$$

A partir de estas definiciones podemos concluir que, si agregamos en forma de suma el valor de la sumatoria de violación de restricciones a la función objetivo, la separabilidad de las variables de ambas funciones serán combinadas en una sola función. De este modo es posible evaluar un solo arreglo de variables para todo el problema, sin perder las características de separabilidad propias de la función objetivo y restricciones. De esta manera también se evita el uso de la matriz de frecuencias y todo el proceso de cálculo del arreglo de variables final.

Por lo tanto, el único cambio necesario en el método VIIC es en las Ecuaciones 4.2, 4.3, 4.7 y 4.9 que son usadas por VIIC para evaluar la separabilidad son sustituidas por las siguientes Ecuaciones 6.5, 6.6, 6.7 y 6.8, respectivamente.

$$fit_{all_{C_1}} = Obj(\mathbf{x}) + cvs(\mathbf{x}) \mid x_i = C_1, \forall i = [1, N] \tag{6.5}$$

$$fit_{all_{C_2}} = Obj(\mathbf{x}) + cvs(\mathbf{x}) \mid x_i = C_2, \forall i = [1, N] \tag{6.6}$$

$$fit_{grp_{C_1=k}} = Obj(\mathbf{x}_{C_1}) + (\mathbf{x}_{C_1}) \tag{6.7}$$

$$fit_{grp_{C_2=k}} = Obj(\mathbf{x}_{C_2}) + cvs(\mathbf{x}_{C_1}) \tag{6.8}$$

Otra característica del método original VIIC, es que cuando se encuentran soluciones factibles durante la optimización, VIIC empieza a considerar la función objetivo para determinar el arreglo de variables. Con esta nueva propuesta, el enfoque cambia, dado que la cvs de una solución factible es cero, ésta no aporta información de las restricciones para descomponer el problema. Por lo tanto, una vez encontradas soluciones factibles, el algoritmo solo trabajará con la función objetivo.

6.2.1 Estrategias de vecindario

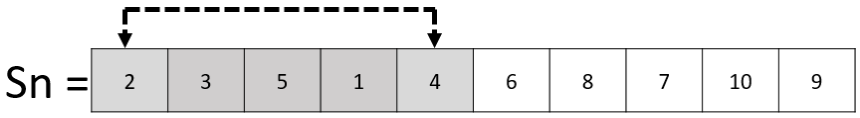
La siguiente característica a considerar es la creación de un nuevo arreglo de variables. VIIC crea un nuevo arreglo de manera totalmente aleatoria, sin considerar la información del arreglo de variables actual.

Con la finalidad de utilizar las características del arreglo de variables actual, se desarrollaron dos estrategias de generación de vecindario (Figura 6.1).

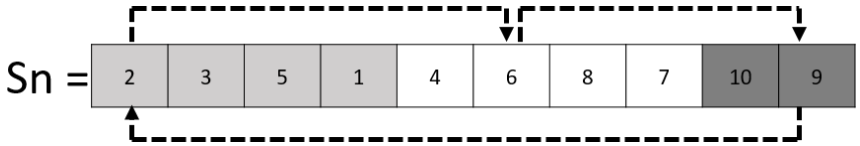
Dado un arreglo de variables, la primera estrategia intercambia dos o más variables con otras, sin considerar el sub-grupo al que pertenecen, esta estrategia es representada en la Figura 6.1a.

La segunda estrategia, tiene por objetivo, considerar los sub-grupos de variables, para no llevar a cabo intercambios de variables dentro del mismo sub-grupo. Se selecciona un número de variables a intercambiar por sub-grupo, estas variables serán intercambiadas de grupo hacia el siguiente adyacente, esta estrategia se muestra en la Figura 6.1b.

Usando estas estrategias de generación de vecindario y la forma de evaluar considerando la sumatoria de violación de restricciones, la nueva propuesta se detalla en el Algoritmo 6.20.



(a) Estrategia 1: estrategia basada en el vector completo, una o más variables intercambian posición



(b) Estrategia 2: estrategia basada en los sub-grupos, las variables cambian posición con el sub-grupo consecutivo.

Figura 6.1.: Estrategias de vecindario para generar nuevos arreglos de variables a partir de un arreglo especifico. Los valores representan el número de variable y el color el sub-grupo.

6.2.2 Nuevas versiones de VIIC

Basado en todo lo anterior, se desarrollaron varios algoritmos para probar la eficacia de las mejoras propuestas. Para contender con el último punto mencionado al inicio del capítulo, se eligió la heurística de recocido simulado (SA por sus siglas en inglés), este método fue expuesto en la sección 2.5. Básicamente el cambio en el método nuevo de VIIC (Algoritmo 6.20) es en la forma de selección, la cual se realiza a través de la temperatura como lo hace SA.

Los detalles de los cinco algoritmos de prueba se presentan en la Tabla 6.1. Cada uno utiliza una estrategia de vecindario y una forma de búsqueda.

Tabla. 6.1.: Las cinco diferentes versiones de algoritmo propuesta para realizar las pruebas.

Nombre del Algoritmo	Algoritmo de búsqueda	Estrategia de vecindario
SA1_VIIC_F+CVS	Recocido simulado	Estrategia 1
SA2_VIIC_F+CVS	Recocido simulado	Estrategia 2
VIIC_F+CVS	Voraz	Aleatorio
VIIC_F+CVS_N1	Voraz	Estrategia 1
VIIC_F+CVS_N2	Voraz	Estrategia 2

Algoritmo 6.20 Nuevo algoritmo VIIC

1: Asignar $maxgpr_{iter} = m * 10^4$, V tamaño de cada sub-grupo, m número de subgrupos, $c\_num = 0$, $S_N = \emptyset$
2: Inicializar in arreglo de variables aleatorio S_n
3: Generar dos valores aleatorios, $C_1 > 0$ y $C_2 > 0$
4: **Calcular $grps_{diff}$ para el arreglo S_n , usando las ecuaciones 4.4 pero ahora considerando las Ecuaciones 6.5 a 6.8**
5: **mientras** $grps_{diff} \neq 0$ y $gpr_{iter} < maxgpr_{iter}$ **hacer**
6: **Generar un nuevo arreglo de variables S'_n de acuerdo a alguna estrategia de vecindario (ver Figura 6.1**
7: Calcular $newgrps_{diff}$ para S'_n
8: **si** $newgrps_{diff} < grps_{diff}$ **entonces**
9: Actualizar $S_n = S'_n$ y $grps_{diff} = newgrps_{diff}$
10: **fin si**
11: **fin mientras**
12: Retornar el ultimo arreglo de variables S_N

6.2.3 Resultados

Para evaluar el desempeño de las cinco versiones de VIIC propuestas, seleccionando un sub-conjunto representativo de los 18 problemas de prueba descritos en la introducción de este capítulo. Los problemas seleccionados corresponden a las funciones $F3$, $F6$, $F9$, $F12$, $F15$, y $F18$. Estos problemas tienen 3 restricciones, lo que los convierte en los problemas más complicados de descomponer en sub-grupos.

El experimento se diseñó de la siguiente forma: se realizaron veinticinco ejecuciones independientes por cada problema en cada una de las tres dimensiones (100, 500, y 1000). El arreglo de variables S_n final fue evaluado para la función objetivo y cada una de las restricciones del problema, con la finalidad de comparar el comportamiento de la descomposición. Por lo cual, el experimento completo constó de 72 funciones (6 objetivos, 3 restricciones por objetivo y todas estas en 3 diferentes dimensiones). A los resultados se les aplicó la prueba estadística de Wilcoxon con un 95 % de confianza. Los resultados de las cinco versiones fueron comparadas contra los resultados obtenidos por el algoritmo VIIC original.

Los parámetros para todos los algoritmos que se compararon fueron: (1) El número de sub-grupos $m = 2$, (2) para ambas estrategias de vecindario el intercambio de variables fue del $40\,\%$ y (3) para las versiones que utilizan el

algoritmo de recocido simulado, la probabilidad de aceptación inicial se fijó en $0{,}3$ y la probabilidad final fue $1E-3$.

Resultados Numéricos

La comparación de desempeño en este experimento se basó en el arreglo de variables S_n final obtenido por las diferentes versiones del algoritmo VIIC contra los arreglos de variables del algoritmo original VIIC (uno por la función objetivo y uno por cada restricción del problema). Es necesario hacer énfasis en que los diferentes vectores de arreglo de variables obtenidos por VIIC, son unidos en uno solo de acuerdo a una matriz de frecuencias, y que este último arreglo de variables no es evaluado en el problema.

Para lidiar con estas diferencias y realizar una comparación justa, el arreglo de variables obtenido por los algoritmos propuestos se evalúa en la función objetivo y cada una de las restricciones. De esta forma los resultados se comparan con los arreglos de variables de VIIC. El objetivo de esta forma la comparación es probar que un solo arreglo de variables obtenido por los algoritmos propuestos es competitivo contra aquellos obtenidos por VIIC.

Los resultados estadísticos se encuentran resumidos en las Tablas 6.2, 6.3, y 6.4 para las dimensiones 100, 500, y 1000, respectivamente. Estas tablas muestran el mejor, mediana, y desviación estándar, para cada uno de los algoritmos en los respectivos problemas.

De manera general, se puede observar que el rendimiento de todos los algoritmos decae conforme la dimensión del problema se ve incrementado. Sin embargo en el problema $F3$ donde su función objetivo es separable, todo los algoritmos logran encontrar arreglos de variables competitivos en las tres dimensiones. Por otro lado, los problemas $F12$, $F15$, y $F16$ los cuales son más complejos de resolver dada las características de separabilidad de sus funciones objetivo, ningún algoritmo logro una descomposición adecuada. Sin embargo, sí lo hicieron para las restricciones asociadas a estos problemas.

En la Figura 6.2 se resume el resultado de aplicar la prueba de Wilcoxon. La figura muestra el número de diferencias significativas de cada uno de los algoritmos que se proponen en contra del algoritmo original de VIIC. Solo se muestran las diferencias significativas a favor de los algoritmos propuestos

por que en las funciones restantes no hubo diferencias significativas en contra de estos.

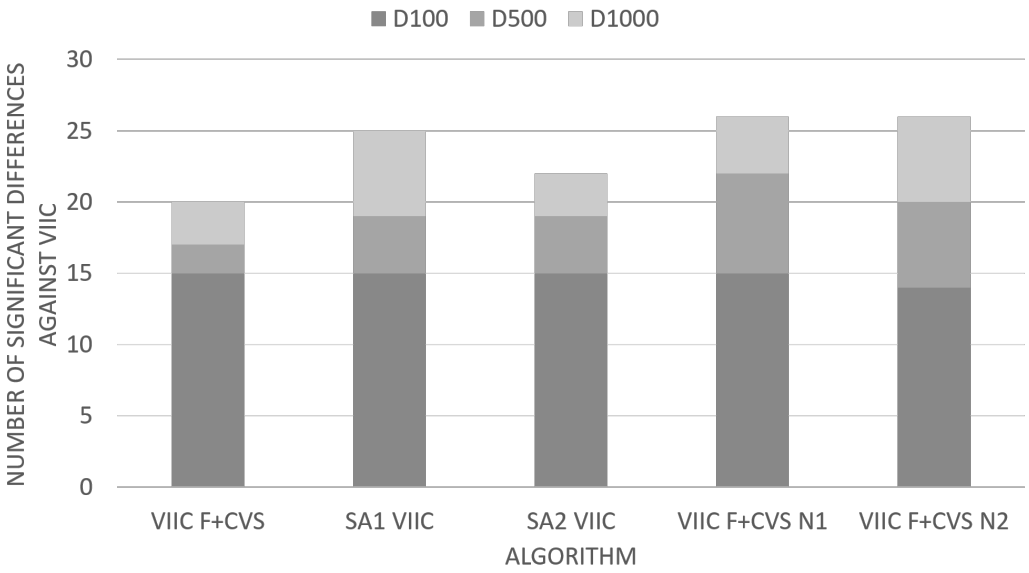


Figura 6.2.: Resumen de la prueba de Wilcoxon. La gráfica muestra el conteo acumulado de diferencias significativas en favor de cada algoritmo contra VIIC original.

VIIC_F+CVS muestra mejor rendimiento que VIIC, lo cual comprueba que optimizar un solo arreglo de variables es una alternativa viable para obtener resultados competitivos sin necesitar de la matriz de frecuencias que utiliza originalmente VIIC.

Ambos algoritmos basados en recocido simulado (SA1_VIIC y SA2_VIIC) logran mejores resultados que VIIC en la mayoría de las funciones. Por otro lado, el algoritmo SA1_VIIC que utiliza la estrategia de vecindario 1, es mejor que aquel que utiliza el vecindario 2.

Finalmente, los mejores resultados los obtuvieron los algoritmos VIIC_F+CVS_N1 y VIIC_F+CVS_N2, estos algoritmos usan una selección totalmente voraz, y combinado con cualquiera de las estrategias de vecindario parece ser lo más adecuado para descomponer un problema de alta dimensión con restricciones en este conjunto de funciones de prueba.

Tabla. 6.2.: Mejora 1 a VIIC: Resultados estadísticos para la dimensión 100. Se presenta el Mejor, Mediana y desviación estándar (std).

			F3					F12			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		6.17E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		6.98E-10	**0.00E+00**	**0.00E+00**	6.28E+05		1.35E+08	**0.00E+00**	2.06E+05	**0.00E+00**
	std		1.30E-09	5.46E-12	8.65E+07	6.87E+07		5.30E+08	4.18E-12	1.01E+08	5.67E+07
SA1_VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.14E+05	7.02E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	4.66E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.82E+08	5.13E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	2.11E-06	1.30E-09	4.80E-12	0.00E+00	0.00E+00	6.72E+08	1.14E+08	5.49E-12	5.47E+07	3.87E+07
SA2_VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.76E+05	3.52E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	6.98E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.01E+08	4.56E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	1.68E-06	1.09E-09	5.32E-12	**0.00E+00**	**0.00E+00**	9.20E+08	1.62E+08	8.48E-12	7.68E+07	5.53E+07
VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.68E+03	3.05E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	1.16E-09	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.87E+07	5.76E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	1.53E-06	1.40E-09	5.07E-12	**0.00E+00**	**0.00E+00**	1.41E+08	1.22E+08	4.44E-12	5.04E+07	1.65E+07
VIIC_F+CVS_N1	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.03E+04	1.86E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	4.66E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.08E+08	6.92E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	1.53E-06	1.23E-09	5.14E-12	**0.00E+00**	**0.00E+00**	1.63E+08	1.66E+08	5.09E-12	3.94E+07	2.74E+07
VIIC_F+CVS_N2	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.16E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	4.66E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.41E+07	3.92E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	7.80E-07	8.93E-10	5.49E-12	**0.00E+00**	**0.00E+00**	2.12E+08	1.51E+08	4.54E-12	4.52E+07	7.75E+07
			F6					F15			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		5.03E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**		1.33E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		6.46E+07	**0.00E+00**	6.70E+05	0.00E+00		2.36E+08	**0.00E+00**	8.72E+06	1.43E+07
	std		3.20E+08	8.36E-12	7.44E+07	7.45E+07		6.73E+08	5.01E-12	8.44E+07	5.71E+07
SA1_VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.96E+06	4.88E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.62E+08	6.10E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	5.72E+06	8.75E-06	5.94E-12	**0.00E+00**	**0.00E+00**	6.95E+08	1.06E+08	5.51E-12	9.46E+07	**0.00E+00**
SA2_VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.88E+04	3.05E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.93E+08	9.01E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	1.57E+06	4.66E-06	7.48E-12	**0.00E+00**	**0.00E+00**	8.11E+08	9.93E+07	4.92E-12	3.90E+07	3.73E+07
VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.43E+05	1.43E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.21E+07	4.21E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	8.79E-06	7.22E-06	8.80E-12	0.00E+00	0.00E+00	1.17E+08	9.67E+07	6.46E-12	3.33E+07	7.86E+06
VIIC_F+CVS_N1	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.10E-05	6.10E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.21E+07	5.21E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	**0.00E+00**	8.73E-06	7.19E-12	**0.00E+00**	**0.00E+00**	1.98E+08	1.98E+08	3.24E-12	2.54E+06	8.67E+05
VIIC_F+CVS_N2	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.02E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.77E+07	2.25E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	8.45E-06	3.38E-06	3.50E-12	**0.00E+00**	**0.00E+00**	1.43E+08	9.02E+07	5.81E-12	9.37E+07	1.13E+07
			F9					F18			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		4.20E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		5.86E+07	**0.00E+00**	3.82E+05	3.09E+06		2.64E+09	**0.00E+00**	1.51E+07	4.73E+06
	std		1.91E+08	4.33E-12	1.31E+08	7.61E+07		4.56E+09	5.94E-12	1.42E+08	8.69E+07
SA1_VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.88E+07	7.86E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.17E+09	1.39E+09	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	4.02E+06	3.43E-06	7.52E-12	0.00E+00	0.00E+00	3.54E+09	2.51E+09	4.44E-12	7.03E+07	4.11E+07
SA2_VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.18E+06	7.77E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.16E+09	1.33E+09	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	1.02E+08	3.40E-06	3.47E-12	**0.00E+00**	**0.00E+00**	3.87E+09	2.51E+09	4.33E-12	3.65E+07	5.30E+07
VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.13E+07	1.13E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.09E+09	1.03E+09	**0.00E+00**	**0.00E+00**	2.89E+07
	std	**0.00E+00**	4.48E-06	4.96E-12	0.00E+00	0.00E+00	2.59E+09	2.54E+09	6.97E-12	6.65E+07	5.86E+07
VIIC_F+CVS_N1	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.12E+04	6.12E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.78E+08	5.78E+08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	**0.00E+00**	3.19E-06	3.79E-12	**0.00E+00**	**0.00E+00**	3.53E+09	3.50E+09	5.92E-12	1.67E+06	6.48E+07
VIIC_F+CVS_N2	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.74E+06	2.74E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.51E+09	2.36E+09	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	1.68E-06	3.79E-06	5.51E-12	0.00E+00	0.00E+00	3.05E+09	3.04E+09	9.16E-12	3.57E+07	4.03E+07

6.2.4 Conclusiones

En esta sección se presentaron tres modificaciones al algoritmo de VIIC, con la finalidad de mejorar el rendimiento para encontrar arreglos de variables adecuados para problemas de alta dimensionalidad con restricciones.

La inclusión de la sumatoria de violación de restricciones durante la evaluación del arreglo de variables fue la primera modificación. Este cambio permite optimizar un solo arreglo de variables contemplando la información de las restricciones, tal como lo realiza VIIC pero sin requerir de la matriz de frecuencias. La siguiente modificación se realizó en el proceso de creación de los nuevos arreglos de variables. VIIC lo realiza de manera aleatoria en cada

Tabla. 6.3.: Mejora 1 a VIIC: Resultados estadísticos para la dimensión 500. Se presenta el Mejor, Mediana y desviación estándar (std).

			F3					F12			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		2.77E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		1.49E-08	2.91E-11	2.15E+07	6.48E+06		4.39E+08	5.82E-11	9.34E+06	2.80E+06
	std		2.54E-08	7.39E-11	1.58E+08	5.01E+07		3.36E+09	7.71E-11	1.37E+08	1.07E+08
SA1_VIIC_F+CVS	Mejor	**0.00E+00**	2.91E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.02E+06	6.71E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	3.26E-08	5.82E-11	**0.00E+00**	**0.00E+00**	8.93E+08	5.18E+08	2.18E-11	1.08E+07	5.35E+06
	std	**0.00E+00**	3.84E-08	6.67E-11	1.68E-06	**0.00E+00**	5.12E+09	3.09E+09	8.48E-11	2.07E+08	1.27E+08
SA2_VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.08E+05	4.27E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	1.35E-08	5.82E-11	**0.00E+00**	**0.00E+00**	1.45E+09	8.91E+08	2.91E-11	3.94E+06	2.17E+06
	std	**0.00E+00**	2.61E-08	7.94E-11	3.11E-06	**0.00E+00**	3.60E+09	2.38E+09	8.59E-11	1.80E+08	5.96E+07
VIIC_F+CVS	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.11E+05	8.11E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	1.86E-08	5.82E-11	**0.00E+00**	**0.00E+00**	1.27E+09	1.20E+09	2.91E-11	3.52E+07	0.00E+00
	std	3.11E-06	2.76E-08	6.71E-11	3.35E-06	**0.00E+00**	3.15E+09	2.96E+09	7.02E-11	2.00E+08	2.85E+07
VIIC_F+CVS_N1	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.63E+07	1.58E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	8.38E-09	2.91E-11	0.00E+00	**0.00E+00**	2.97E+09	2.80E+09	5.82E-11	1.68E+07	1.79E+06
	std	3.11E-06	2.71E-08	9.31E-11	1.53E-06	**0.00E+00**	3.96E+09	3.78E+09	5.30E-11	1.91E+08	1.12E+08
VIIC_F+CVS_N2	Mejor	**0.00E+00**	2.33E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.26E+07	1.15E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	2.42E-08	4.37E-11	**0.00E+00**	**0.00E+00**	7.29E+08	7.09E+08	5.82E-11	9.35E+06	**0.00E+00**
	std	**0.00E+00**	2.16E-08	6.22E-11	7.65E-07	**0.00E+00**	2.09E+09	2.00E+09	6.97E-11	8.48E+07	8.62E+07
			F6					F15			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		7.08E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**		9.36E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		6.90E+08	4.37E-11	2.41E+07	2.41E+07		3.19E+08	7.28E-12	8.38E+06	5.73E+06
	std		1.79E+09	7.68E-11	2.22E+08	1.46E+08		2.70E+09	4.96E-11	8.46E+07	7.17E+07
SA1_VIIC_F+CVS	Mejor	7.88E+06	3.64E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.75E+06	1.42E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	5.27E+08	2.41E+08	5.82E-11	6.58E+06	**0.00E+00**	1.81E+09	1.00E+09	5.82E-11	1.68E+07	9.84E+06
	std	1.49E+09	6.71E+08	9.46E-11	5.02E+07	4.57E+07	4.02E+09	2.67E+09	7.90E-11	1.37E+08	8.76E+07
SA2_VIIC_F+CVS	Mejor	2.73E+03	1.50E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.98E+06	5.08E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	7.87E+08	3.68E+08	5.82E-11	1.02E+06	1.11E+05	1.97E+09	1.21E+09	2.91E-11	1.16E+07	1.03E+07
	std	1.46E+09	7.69E+08	1.12E-10	1.15E+08	9.14E+07	3.77E+09	2.32E+09	6.79E-11	9.30E+07	6.70E+07
VIIC_F+CVS	Mejor	3.92E+04	3.52E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.71E+07	2.64E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	2.24E+08	1.79E+08	2.91E-11	9.54E-07	3.26E+05	7.25E+08	6.80E+08	5.82E-11	1.40E+07	7.44E+06
	std	9.68E+08	7.75E+08	8.92E-11	2.26E+08	7.01E+07	2.96E+09	2.90E+09	6.37E-11	7.03E+07	9.22E+07
VIIC_F+CVS_N1	Mejor	1.37E+05	1.37E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.29E+07	2.23E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	1.99E+08	1.76E+08	1.46E-11	1.83E+06	0.00E+00	1.66E+09	1.48E+09	5.82E-11	1.88E+07	1.37E+07
	std	7.01E+08	6.37E+08	3.54E-11	7.56E+07	3.22E+07	3.44E+09	3.17E+09	8.92E-11	1.91E+08	1.11E+08
VIIC_F+CVS_N2	Mejor	8.10E+04	8.10E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.70E+05	7.49E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	1.38E+08	1.14E+08	2.91E-11	3.91E+06	**0.00E+00**	2.84E+09	2.69E+09	5.82E-11	7.68E+07	**0.00E+00**
	std	5.16E+08	4.79E+08	6.33E-11	5.41E+07	4.84E+07	3.91E+09	3.67E+09	7.97E-11	1.85E+08	1.49E+08
			F9					F18			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		7.86E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**		1.18E+08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		3.54E+08	4.37E-11	4.17E+07	3.93E+07		4.65E+09	5.82E-11	7.68E+06	6.64E+06
	std		1.03E+09	5.67E-11	1.65E+08	1.51E+08		2.66E+10	1.18E-10	1.07E+08	1.65E+08
SA1_VIIC_F+CVS	Mejor	1.05E+07	2.41E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.38E+07	2.95E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	3.30E+08	8.87E+07	5.82E-11	1.38E+07	0.00E+00	1.17E+10	9.46E+09	2.91E-11	5.97E+07	5.04E+06
	std	1.36E+09	4.33E+08	7.66E-11	9.58E+07	1.52E+08	2.07E+10	1.79E+10	1.08E-10	1.77E+08	1.01E+08
SA2_VIIC_F+CVS	Mejor	1.75E+05	3.88E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.43E+06	8.35E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	8.00E+08	2.86E+08	5.82E-11	1.91E-06	0.00E+00	2.03E+09	1.72E+09	2.91E-11	4.96E+06	1.98E+06
	std	1.09E+09	3.12E+08	7.48E-11	1.14E+08	6.37E+07	2.56E+10	2.21E+10	6.49E-11	1.09E+08	1.55E+08
VIIC_F+CVS	Mejor	2.39E+05	2.39E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.69E+07	5.66E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	4.37E+08	3.28E+08	5.82E-11	**0.00E+00**	**0.00E+00**	1.23E+10	1.22E+10	2.91E-11	2.80E+07	2.44E+05
	std	3.87E+08	3.33E+08	9.10E-11	6.51E+07	8.65E+07	2.15E+10	2.14E+10	1.11E-10	1.63E+08	8.48E+07
VIIC_F+CVS_N1	Mejor	8.05E+04	6.44E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.35E+06	3.29E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	1.45E+08	1.28E+08	2.91E-11	3.24E+06	**0.00E+00**	1.97E+10	1.96E+10	5.82E-11	1.13E+07	**0.00E+00**
	std	3.51E+08	3.49E+08	6.69E-11	3.98E+07	1.74E+07	2.42E+10	2.40E+10	1.19E-10	1.73E+08	1.37E+08
VIIC_F+CVS_N2	Mejor	8.20E+05	4.62E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.26E+06	1.25E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	8.58E+07	6.87E+07	2.91E-11	3.51E+06	**0.00E+00**	7.95E+09	7.89E+09	1.46E-11	1.19E+07	1.92E+06
	std	3.42E+08	1.65E+08	6.19E-11	9.26E+07	1.27E+08	1.86E+10	1.84E+10	6.88E-11	1.70E+08	8.93E+07

iteración; para ello se propusieron dos estrategias de vecindario para crear los nuevos vectores de variables. Finalmente, la manera voraz de búsqueda de VIIC, fue cambiada por una estrategia de recocido simulado. Basado en estas modificaciones se desarrollaron cinco algoritmos y se compararon contra la versión VIIC original.

A pesar de que todos los algoritmos se ven afectados de manera negativa en su rendimiento cuando la dimensión del problema crece, se encontraron cosas interesantes, como lo son: (1) el evaluar sólo un arreglo de variables no afecta e incluso mejora los resultados finales para encontrar un arreglo de variables adecuado, (2) las dos estrategias de vecindario para generar arreglos de variables a partir del ya existente mejoran el rendimiento de VIIC,

Tabla. 6.4.: Mejora 1 a VIIC: Resultados estadísticos para la dimensión 1000. Se presenta el Mejor, Mediana y desviación estándar (std).

			F3					F12			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		2.99E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		4.84E-08	**5.82E-11**	6.49E+07	3.77E+07		3.03E+09	1.75E-10	4.57E+07	3.48E+07
	std		9.54E-08	2.89E-10	1.72E+08	1.07E+08		8.40E+09	**3.78E-10**	3.01E+08	1.76E+08
SA1_VIIC_F+CVS	Mejor	**0.00E+00**	4.66E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.04E+05	2.28E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	5.96E-08	1.16E-10	**0.00E+00**	**0.00E+00**	**1.25E+09**	**1.04E+09**	1.16E-10	5.08E+07	**8.25E+06**
	std	3.95E+06	1.05E-07	2.92E-10	3.05E-06	**8.35E-07**	8.45E+09	6.31E+09	3.61E-10	2.05E+08	9.35E+07
SA2_VIIC_F+CVS	Mejor	**0.00E+00**	1.40E-09	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.04E+06	1.99E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	6.33E-08	3.49E-10	**0.00E+00**	**0.00E+00**	5.02E+09	3.80E+09	1.16E-10	6.11E+07	2.49E+07
	std	1.66E+08	1.22E-07	3.23E-10	4.53E-06	3.82E-06	1.11E+10	8.73E+09	3.17E-10	2.33E+08	1.66E+08
VIIC_F+CVS	Mejor	**0.00E+00**	6.98E-10	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.99E+03	5.65E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	1.12E-07	1.75E-10	**0.00E+00**	**0.00E+00**	3.63E+09	3.59E+09	1.75E-10	4.33E+07	2.90E+07
	std	6.71E-06	1.38E-07	3.61E-10	4.40E-06	2.20E-06	9.42E+09	9.11E+09	3.24E-10	2.17E+08	1.34E+08
VIIC_F+CVS_N1	Mejor	**0.00E+00**	5.59E-09	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.50E+07	1.46E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	4.47E-08	1.75E-10	**0.00E+00**	**0.00E+00**	1.37E+09	1.35E+09	2.04E-10	1.78E+07	1.37E+07
	std	3.06E-06	8.71E-08	3.44E-10	4.41E-06	1.72E-06	7.48E+09	7.11E+09	2.16E-10	3.08E+08	1.30E+08
VIIC_F+CVS_N2	Mejor	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.73E+05	3.60E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**0.00E+00**	7.64E-08	2.33E-10	**0.00E+00**	**0.00E+00**	3.11E+09	3.00E+09	2.33E-10	5.34E+07	1.07E+07
	std	1.53E-06	**1.29E-07**	**4.48E-10**	5.15E-06	1.58E-06	**6.21E+09**	**6.04E+09**	2.21E-10	**1.60E+08**	**8.15E+07**
			F6					F15			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		6.70E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**		4.23E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		2.00E+09	1.16E-10	1.26E+08	3.99E+07		1.97E+09	1.16E-10	5.21E+07	4.44E+07
	std		3.55E+09	3.49E-10	3.42E+08	1.98E+08		9.57E+09	2.61E-10	2.15E+08	1.95E+08
SA1_VIIC_F+CVS	Mejor	**4.12E+04**	**2.32E+04**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.01E+07	2.28E+07	**0.00E+00**	8.45E+05	**0.00E+00**
	Mediana	**3.67E+08**	**1.93E+08**	1.16E-10	**1.11E+07**	4.65E+06	2.68E+09	1.90E+09	2.33E-10	8.71E+07	4.14E+07
	std	2.58E+09	1.72E+09	2.20E-10	**1.49E+08**	9.87E+07	1.15E+10	8.78E+09	**3.64E-10**	2.48E+08	1.75E+08
SA2_VIIC_F+CVS	Mejor	1.21E+06	7.44E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**9.87E+05**	**7.80E+05**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	2.01E+09	1.38E+09	**5.82E-11**	8.45E+07	6.65E+07	3.02E+09	2.52E+09	2.33E-10	5.54E+07	2.77E+07
	std	3.62E+09	1.95E+09	**3.83E-10**	2.51E+08	1.11E+08	9.01E+09	7.07E+09	3.46E-10	1.89E+08	1.32E+08
VIIC_F+CVS	Mejor	4.19E+06	4.02E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.39E+07	**2.34E+07**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	1.63E+09	1.30E+09	2.33E-10	1.01E+08	1.31E+07	2.40E+09	2.24E+09	1.16E-10	4.37E+07	1.12E+07
	std	2.52E+09	2.37E+09	3.21E-10	1.67E+08	1.11E+08	6.36E+09	6.11E+09	2.03E-10	1.97E+08	9.19E+07
VIIC_F+CVS_N1	Mejor	5.14E+06	4.28E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.13E+07	1.06E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	9.54E+08	8.62E+08	1.16E-10	2.80E+07	**0.00E+00**	9.32E+08	**9.12E+08**	**8.73E-11**	**1.83E+07**	1.83E+07
	std	**1.92E+09**	**1.60E+09**	2.52E-10	2.76E+08	9.16E+07	7.07E+09	6.83E+09	2.83E-10	1.72E+08	9.39E+07
VIIC_F+CVS_N2	Mejor	4.97E+05	4.97E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.11E+08	1.04E+08	**0.00E+00**	4.98E+06	**0.00E+00**
	Mediana	7.58E+08	6.32E+08	1.75E-10	4.00E+07	4.30E+04	**2.14E+09**	2.02E+09	1.16E-10	5.92E+07	**1.10E+07**
	std	2.19E+09	2.02E+09	2.65E-10	1.60E+08	**7.94E+07**	**5.45E+09**	**5.32E+09**	3.50E-10	**1.04E+08**	**7.32E+07**
			F9					F18			
		Sn	Obj	G1	G2	G3	Sn	Obj	G1	G2	G3
VIIC	Mejor		3.65E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**		3.97E+08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana		7.48E+08	2.33E-10	5.30E+07	7.10E+07		1.63E+10	**2.33E-10**	4.62E+07	3.41E+07
	std		1.76E+09	3.88E-10	2.67E+08	1.82E+08		4.37E+10	**3.92E-10**	2.21E+08	1.13E+08
SA1_VIIC_F+CVS	Mejor	1.36E+06	5.71E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.59E+08	1.38E+08	**0.00E+00**	5.76E+05	**0.00E+00**
	Mediana	1.51E+09	7.54E+08	2.33E-10	4.84E+07	**7.63E-06**	7.49E+09	**6.63E+09**	1.16E-10	**3.67E+07**	2.69E+07
	std	2.23E+09	1.16E+09	3.55E-10	2.00E+08	1.30E+08	3.39E+10	3.05E+10	2.32E-10	**1.61E+08**	1.28E+08
SA2_VIIC_F+CVS	Mejor	1.26E+05	5.27E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**2.98E+05**	**2.66E+05**	**0.00E+00**	4.66E-10	**0.00E+00**
	Mediana	6.72E+08	3.31E+08	1.16E-10	1.17E+07	1.05E+04	2.23E+10	2.03E+10	2.33E-10	1.06E+08	3.74E+07
	std	2.40E+09	1.32E+09	**4.89E-10**	2.28E+08	3.92E+07	3.98E+10	3.66E+10	3.75E-10	1.65E+08	1.03E+08
VIIC_F+CVS	Mejor	1.61E+06	1.61E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.38E+07	1.37E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	7.92E+08	6.93E+08	2.04E-10	4.94E+07	1.87E+07	1.18E+10	1.17E+10	2.33E-10	4.32E+07	2.40E+07
	std	1.30E+09	1.01E+09	3.30E-10	2.34E+08	1.70E+08	4.26E+10	4.23E+10	2.88E-10	2.38E+08	1.61E+08
VIIC_F+CVS_N1	Mejor	3.43E+05	2.74E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.55E+06	8.52E+06	**0.00E+00**	1.74E+04	**0.00E+00**
	Mediana	3.73E+08	2.80E+08	2.33E-10	2.48E+07	3.81E-06	1.38E+10	1.37E+10	1.16E-10	5.67E+07	1.68E+07
	std	1.23E+09	1.08E+09	2.10E-10	1.63E+08	3.21E+07	3.84E+10	3.80E+10	3.54E-10	1.74E+08	2.08E+08
VIIC_F+CVS_N2	Mejor	**1.86E+03**	**1.72E+03**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.55E+06	5.50E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Mediana	**3.02E+08**	**2.82E+08**	1.16E-10	**1.01E+07**	1.63E+06	**7.08E+09**	7.01E+09	1.75E-10	4.15E+07	**8.18E+06**
	std	**8.76E+08**	**7.61E+08**	1.82E-10	**8.49E+07**	8.93E+07	**2.56E+10**	**2.54E+10**	1.42E-10	8.58E+07	**1.01E+08**

y (3) es interesante que al combinar estas estrategias con la versión voraz de VIIC proveen mejores resultados que cuando se aplica una metaheurística como recocido simulado.

Los resultados de este trabajo fueron publicados en el congreso de cómputo evolutivo (CEC2016):

A. E. Aguilar-Justo and E. Mezura-Montes, **Towards an improvement of variable interaction identification for large-scale constrained problems** 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp. 4167-4174. [1].

6.3 VIIC guiado por un algoritmo genético

Muchos de los primeros métodos de descomposición necesitaban el número de sub-grupos que se querían obtener, como el caso de las versiones de VIIC ([36], [37]). Por otro lado algunos no requieren este parámetro pero los sub-grupos contenían un tamaño fijo de variables, un ejemplo es Random Grouping [32]. Actualmente existen métodos de descomposición que no requieren alguno de estos parámetros como lo es Diferential Grouping [30], sin embargo este método está enfocado en problemas sin restricciones.

Como se mencionó anteriormente, el algoritmo VIIC original y las mejoras que se le realizaron a éste en la sección 6, requieren el número de sub-grupos, además de generar los grupos con un tamaño fijo de variables.

Una vez probado que la inclusión de generación de vecindarios y uso de heurísticas mejora el rendimiento de VIIC, el siguiente paso es buscar la forma de no requerir el número de sub-grupos a crear y que estos no tengan fijo el número de variables. En esta sección se propone el uso de un algoritmo genético que realice la búsqueda del arreglo de variables, utilizando el cálculo de $gprs_{diff}$ de VIIC mejorado como función de aptitud de los individuos. El algoritmo descrito en esta sección ha sido llamado Dynamic Variable Interaction Identification for Constrained problems (DVIIC).

Los algoritmos genéticos ya fueron descritos en el capítulo 3, sin embargo en la Figura 6.3 se presenta un diagrama de flujo del algoritmo con sus partes importantes, los cuales serán descritas a detalle continuación.

6.3.1 Detalle del Algoritmo Genético

Evaluación

Para evaluar un individuo se utiliza el procedimiento explicado en la sección 6.2, que fue incluido en el proceso de VIIC, donde el valor $gprs_{diff}$ es el objetivo a minimizar. Sin embargo, con la finalidad de maximizar el número de sub-problemas el valor de $gprs_{diff}$ es modificado de la siguiente forma (Ecuación 6.9):

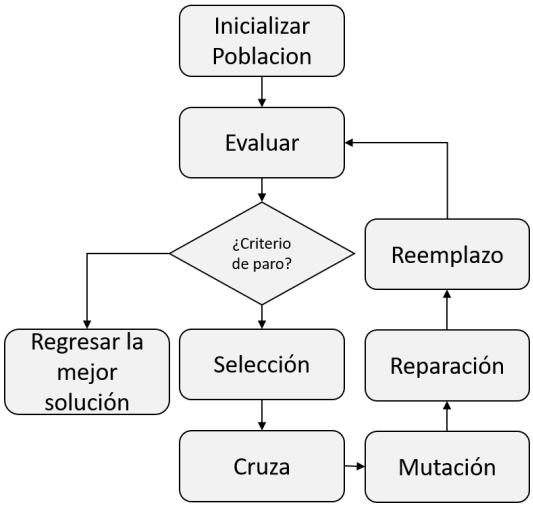


Figura 6.3.: Diagrama de flujo del algoritmo genético propuesto

- Si el número de sub-grupos ($nSubgroups$) es uno, entonces al valor de $gprs_{diff}$ se le asigna un valor grande.

- Si $gprs_{diff} = 0$, significa que la descomposición es exacta, y es gratificada, substrayendo el número de sub-grupos obtenidos.

- En cualquier otro caso el valor de $gprs_{diff}$ se mantiene.

Estas reglas buscan beneficiar a aquellos individuos que descompongan el problema y además tengan el mayor número de subgrupos.

$$gprs_{diff} = \begin{cases} infinito & , nSubgroups = 1 \\ gprs_{diff} - nSubgroups & , gprs_{diff} = 0 \\ gprs_{diff} & , \text{de otra manera} \end{cases} \tag{6.9}$$

Representación

La representación de un individuo es un vector de números enteros. La longitud de este vector es igual a la dimensión del problema. El valor de cada posición del vector representa la etiqueta del grupo al que pertenece la variable. Esta es la representación codificada sobre la cual trabajarán los operadores del algoritmo genético (ver Figura 6.4a).

Por otro lado, para evaluar el individuo es necesario transformar la representación. Esta representación es conocida como decodificada. Dicha representación está compuesta por el arreglo de variables S_n y un vector que indica la cantidad de variables en cada uno de los subgrupos V. La longitud de V indica el número de sub-grupos a probar.

$ind = [1, 1, 1, 2, 1, 2, 2, 2, 3, 3]$ $S_n = [1, 2, 3, 5, 4, 6, 7, 8, 9, 10]$ $V = [4, 4, 2]$

(a) Representación codificada **(b)** Representación decodificada

Figura 6.4.: Representaciones de un individuo

Selección

Un proceso importante en un algoritmo genético es la selección de padres. Existen diversas técnicas como: selección proporcional y torneo. Sin embargo, para esta propuesta todo individuo será seleccionado para la cruza.

Cruza

Como se mencionó anteriormente, todos los individuos podrán cruzarse para crear una nueva generación de individuos. Dada la probabilidad de cruza, el método de recombinación utilizado será la cruza de dos puntos. En la Figura 6.5 se muestra un ejemplo de este operador.

$P1 = [1, 1, 1, 2, 1, 2, 2, 2, 3, 3]$ $C1 = [1, 1, 1, 2, 2, 2, 4, 2, 3, 3]$

$P2 = [1, 2, 1, 1, 2, 2, 4, 2, 3, 4]$ $C2 = [1, 2, 1, 1, 1, 2, 2, 2, 3, 4]$

Figura 6.5.: Cruza de dos puntos. P1: padre 1, P2: padre 2, C1: hijo 1, C2: hijo 2

Mutación

Después de aplicar el operador de cruza, se puede aplicar un operador de mutación dada una probabilidad de mutación. Para este trabajo se utilizó la mutación uniforme, cambiando cada una de las etiquetas por otra seleccionada de manera aleatoria entre 1 y la dimensión del problema.

$$ind = [1, 1, \mathbf{1}, 2, 1, \mathbf{2}, \mathbf{2}, 2, 3, 3]$$
$$mut = [1, 1, \mathbf{5}, 2, 1, \mathbf{3}, \mathbf{5}, 2, 3, 3]$$

Figura 6.6.: Mutación de un individuo. Los valores mutados se ven remarcados en negrita

Reparador

Una vez aplicados los operadores de cruza y mutación, los individuos pueden terminar con múltiples etiquetas de grupo diferentes. Sin embargo, para mantener la representación clara y simple, es necesario reparar la solución.

El proceso de reparación consiste en el re-etiquetado del individuo para mantener las etiquetas en orden ascendente. Tomando como ejemplo la Figura 6.6, se puede apreciar que el individuo tiene las siguientes etiquetas diferentes $\{1, 2, 3, 5\}$. Esto indica que la solución representa 4 subgrupos pero la etiqueta mayor es 5. Por lo tanto después del proceso de reparación la solución es la siguiente $mut = [1, 1, 4, 2, 1, 3, 4, 2, 3, 3]$.

Otro proceso de reparación consiste en unir los grupos de variables que son separables, por ejemplo, dada la siguiente solución de 5 variables y la descomposición $ind = [1, 2, 3, 4, 4]$, se puede apreciar que son 4 grupos de variables donde 3 de ellos sólo tienen una variable asignada. De este modo, la solución reparada seria la siguiente $ind_{repair} = [1, 1, 1, 2, 2]$.

Reemplazo

El reemplazo es generacional con elitismo, es decir, el mejor individuo siempre es seleccionado para pasar a la siguiente generación, reemplazando de manera a aleatoria a un individuo de la población de la nueva generación.

6.3.2 Resultados

Con la finalidad de evaluar el rendimiento de DVIIC, se utilizó el conjunto de 18 funciones de prueba [37] descritos en la introducción de este capítulo. DVIIC fue comparado contra las dos mejores versiones de los algoritmos expuestos en la sección anterior (6.2.2). Estos algoritmos fueron las ver-

siones voraces con las estrategias de vecindario, para esta experimentación se han nombrado VIICN1, VIICN2 para la estrategia de vecindario 1 y 2, respectivamente.

Para cada algoritmo se realizaron veinticinco ejecuciones independientes por cada una de las 18 funciones de prueba. Cada función se ejecutó en las tres dimensiones soportadas por el conjunto de prueba: 100, 500 y 1000. El soporte estadístico se obtuvo con la prueba de Wilcoxon con un 95 % de confianza. También se calculó la complejidad de los algoritmos de manera empírica con la metodología presentada en el CEC2005 [41] y descrita en la sección 5.1.2.

Los parámetros utilizados para DVIIC son los siguientes: (1) el tamaño de población $popSize = 6$, (2) probabilidad de cruza $pc = 0{,}9$, (3) probabilidad de mutación $pm = 0{,}1$, y (4) el número máximo de evaluaciones permitidas $maxFes = 100000$. Para VIICN1 y VIICN2, (1) el porcentaje de intercambio de variables se asignó a $40\,\%$, (2) el número de sub-grupos $m = 2$, y (3) el máximo de iteraciones $maxIter = m \times 10000$.

El análisis de estos algoritmos está centrado en qué tan bien se hace la descomposición del problema tomando como medida el $gprs_{diff}$ propuesto para VIIC. Por lo tanto, los resultados numéricos presentados aquí no corresponden al valor de la función objetivo del problema a optimizar.

Resultados Estadísticos

En las Tablas 6.7, 6.8, y 6.9 se resumen los resultados estadísticos de los resultados numéricos para las dimensiones 100, 500, y 1000, respectivamente. El resultado de la prueba de Wilcoxon se muestra en estas tablas, en la columna "W". El conteo resumen de esta prueba es presentado en la Tabla 6.5.

De acuerdo a los resultados de la prueba estadística, de manera general, DVIIC tuvo diferencias significativas a favor en la mayoría de los problemas respecto a los otros dos algoritmos (VIICN1, VIICN2). En dimensión 100, tanto VIICN1 como VIICN2, tienen un desempeño a favor en los problemas F4 a F9. Estos problemas tienen como caracteristica que su función objetivo son separables y parcialmente separables, lo que incrementa la probabilidad de encontrar un arreglo de variables para dos sub-grupos. En los problemas

restantes, DVIIC es mejor en cada una de las funciones que los algoritmos VIICN, mostrando alcanzar arreglos dinámicos de variables adecuados para problemas con alta complejidad en la separabilidad de sus variables.

En las dimensiones 500 y 1000, DVIIC encuentra arreglos de variables de alta calidad en todas las funciones. Sin embargo, se nota un decremento en el rendimiento del algoritmo cuando la dimensión aumenta. Este efecto se observa de manera más clara en los problemas F9 a F18. Además, los algoritmos VIICN1 y VIICN2, logran obtener mejores resultados en las funciones F2 y F3. En estos problemas, la función objetivo es totalmente separable.

Tabla. 6.5.: Resumen de la prueba de Wilcoxon. SdS sin diferencia significativa

	DVIIC - VIICN1			DVIIC - VIICN2		
	D100	**D500**	**D1000**	**D100**	**D500**	**D1000**
Favor	**11**	**16**	**16**	**11**	**16**	**16**
SdS	1	1	1	1	0	0
En contra	6	1	1	6	2	2

Complejidad

La Tabla 5.2 muestra los resultados de complejidad de tiempo para aquellos problemas que tienen 3 restricciones (F3, F6, F9, F12, F15, y F18). Por lo tanto, su costo computacional es el más alto de todo el conjunto de prueba.

Se puede observar que en los problemas de dimensión 100, los tiempos entre los algoritmos son muy similares. DVIIC tiene los mejores tiempos en los problemas F3, F9, y F18 y sólo la versión VIICN1 logra mejorar el tiempo en los problemas F6, F12, y F15. Por otro lado, al incrementar la dimensión a 500 y 1000, DVIIC logra los mejores tiempos en todas las funciones y los algoritmos VIICN1 y VIICN2 reducen su comportamiento de manera notable.

Este comportamiento en los resultados de tiempo da a entender que el proceso de creación de los vecinos en los algoritmos VIICN1 y VIICN2 es más costoso conforme la dimensión se incrementa.

Tabla. 6.6.: Comparación de Complejidad Empírica [41], los valores representan el tiempo (milisegundos) promedio para ejecutar el algoritmo con un límite de 200,000 evaluaciones. En negrita se resaltan los menores tiempos.

D100	**DVIIC**	**VIICN1**	**VIICN2**
F3	**8.67E-01**	1.40E+00	1.73E+00
F6	1.20E+00	**6.00E-01**	2.60E+00
F9	**8.00E-01**	3.60E+00	5.60E+00
F12	2.40E+00	**2.20E+00**	7.53E+00
F15	4.60E+00	**3.87E+00**	4.27E+00
F18	**2.07E+00**	5.53E+00	8.93E+00
D500	**DVIIC**	**VIICN1**	**VIICN2**
F3	**5.60E+00**	1.89E+01	3.01E+01
F6	**5.93E+00**	1.37E+01	2.75E+01
F9	**3.73E+00**	1.73E+01	2.41E+01
F12	**5.27E+00**	1.42E+01	3.61E+01
F15	**4.67E+00**	3.47E+01	3.47E+01
F18	**6.27E+00**	1.95E+01	6.13E+01
D1000	**DVIIC**	**VIICN1**	**VIICN2**
F3	**2.33E+00**	3.14E+01	5.83E+01
F6	**8.87E+00**	5.36E+01	6.39E+01
F9	**2.43E+01**	6.01E+01	4.02E+01
F12	**1.37E+01**	4.73E+01	6.39E+01
F15	**2.89E+01**	8.19E+01	4.17E+01
F18	**9.87E+00**	4.33E+01	2.91E+01

6.3.3 Conclusiones

En esta sección se presentó al algoritmo Dynamic Variable Interaction Identification for Constrained problems (DVIIC). Este algoritmo busca contender con el problema de definir el número de sub-grupos tal como lo hace VIIC en su forma canónica. Para realizar ésto, DVIIC utiliza como motor de búsqueda un algoritmo genético tradicional con representación entera. El algoritmo propuesto fue probado en un conjunto de funciones de prueba que tienen diferentes características de separabilidad. Y se compararon sus resultados contra dos algoritmos basados en VIIC con estrategias de vecindario (VIICN1 y VIICN2).

Los resultados muestran que DVIIC supera a VIICN en 11 de los 18 problemas en dimensión 100. Incluso encuentra arreglos de variables de alta calidad (i.e. $gprs_{diff} = 0$) en seis de los problemas más complejos. Además, cuando la dimensión se incrementa a 500 y 1000 variables, DVIIC continúa mostrando

un alto desempeño comparado contra los algoritmos VIICN1 y VIICN2. Sin embargo, DVIIC no logra alcanzar soluciones de tan alta calidad para las funciones F10 a F17 como lo hace en la dimensión 100. Finalmente, la desviación estándar y la media de los resultados finales, suguieren que DVIIC requiere de más evaluaciones conforme la dimensión incrementa.

De acuerdo a la complejidad de los algoritmos, DVIIC mostró ser más rápido que VIICN1 y VIICN2, a pesar del hecho de que DVIIC esta basado en un algoritmo genético y realiza más cálculos para su funcionamiento, comparado con VIICN que sólo realiza la exploración del vecindario. Por lo tanto, se concluye que los procesos de vecindario son computacionalmente costosos y deben ser debidamente calibrados.

Como se mencionó anteriormente, estos trabajos están enfocados en mejorar el rendimiento del algoritmo de descomposición VIIC, el cual proporcionó una manera de evaluar la calidad de un arreglo de variables. Por lo tanto, en las siguientes secciones, se probará el rendimiento de esta nueva propuesta en la optimización del problema, utilizando la estrategia de LoCoS.

Cabe remarcar que es importante profundizar en el estudio del impacto de sub-grupos dinámicos en la optimización del problema.

Esta investigación fue desarrollada durante la estancia en la Universidad de Nueva Gales del Sur (UNSW) Australia, bajo la asesoría del Dr. Ruhul A. Sarker y el Dr. Saber M. Elsayed. El trabajo derivó en una publicación de congreso:

A. E. Aguilar-Justo, E. Mezura-Montes and Saber M. Elsayed and Ruhul A. Sarker, **Decomposition of Large-scale Constrained Problems Using a Genetic-based Search** 2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2016). Ixtapa, Mexico. [2]

Tabla. 6.7.: Estadísticas dimensión 100. "W" prueba de Wilcoxon, "✓" diferencia significativa a favor de DVIIC, "="no existe diferencia significativa y "✗" diferencia significativa en contra de DVIIC

D100	DVIIC			VIICN1				VIICN2			
	Mejor	Mediana	Desv. Est.	Mejor	Mediana	Desv. Est.	W	Mejor	Mediana	Desv. Est.	W
F1	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	4.66E-10	6.67E-10	✓	**0.00E+00**	2.33E-10	1.05E-09	✓
F2	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**	1.58E-06	✓	**0.00E+00**	**0.00E+00**	1.91E-06	✓
F3	**0.00E+00**	**0.00E+00**	1.69E+03	**0.00E+00**	**0.00E+00**	1.53E-06	=	**0.00E+00**	**0.00E+00**	7.79E-07	=
F4	**0.00E+00**	7.63E-06	1.72E+03	**0.00E+00**	**0.00E+00**	0.00E+00	✗	**0.00E+00**	**0.00E+00**	0.00E+00	✗
F5	**0.00E+00**	5.70E+02	2.63E+04	**0.00E+00**	**0.00E+00**	3.06E-06	✗	**0.00E+00**	**0.00E+00**	3.06E-06	✗
F6	**0.00E+00**	4.81E+02	2.04E+04	**0.00E+00**	**0.00E+00**	0.00E+00	✗	**0.00E+00**	**0.00E+00**	8.45E-06	✗
F7	**0.00E+00**	1.19E-07	6.31E+02	**0.00E+00**	**0.00E+00**	0.00E+00	✗	**0.00E+00**	**0.00E+00**	7.62E-07	✗
F8	**0.00E+00**	1.53E-05	6.42E+03	**0.00E+00**	**0.00E+00**	0.00E+00	✗	**0.00E+00**	**0.00E+00**	3.06E-06	✗
F9	**0.00E+00**	7.63E-06	1.71E+04	**0.00E+00**	**0.00E+00**	0.00E+00	✗	**0.00E+00**	**0.00E+00**	1.68E-06	✗
F10	**0.00E+00**	**4.18E+02**	2.27E+04	0.00E+00	2.57E+07	5.19E+07	✓	0.00E+00	9.50E+06	6.34E+07	✓
F11	**0.00E+00**	**2.76E+03**	2.13E+04	5.34E-05	5.69E+07	1.58E+08	✓	1.13E+01	1.65E+07	1.64E+08	✓
F12	**0.00E+00**	**5.42E+03**	7.97E+04	1.03E+04	1.08E+08	1.63E+08	✓	5.16E+05	6.41E+07	2.12E+08	✓
F13	**0.00E+00**	**1.24E+03**	1.16E+04	0.00E+00	9.96E+06	9.70E+07	✓	0.00E+00	1.55E+06	9.60E+07	✓
F14	**0.00E+00**	**1.47E+04**	7.54E+04	3.81E-06	4.03E+07	1.19E+08	✓	2.38E+05	5.17E+07	9.53E+07	✓
F15	**0.00E+00**	**5.14E+03**	7.99E+04	6.10E-05	5.21E+07	1.98E+08	✓	2.02E+05	4.77E+07	1.43E+08	✓
F16	**2.87E+01**	**1.66E+04**	7.96E+04	1.12E+06	7.30E+08	2.99E+09	✓	1.85E+06	1.08E+09	2.60E+09	✓
F17	**3.21E+02**	**4.31E+04**	2.79E+05	5.39E+05	4.80E+08	3.31E+09	✓	1.85E+02	6.64E+08	1.70E+09	✓
F18	**5.49E+02**	**3.31E+04**	3.82E+05	6.12E+04	5.78E+08	3.53E+09	✓	2.74E+06	2.51E+09	3.05E+09	✓

Tabla. 6.8.: Estadísticas dimensión 500. "W" prueba de Wilcoxon, "✓" diferencia significativa a favor de DVIIC, "="no existe diferencia significativa y "✗" diferencia significativa en contra de DVIIC

D500	DVIIC			VIICN1				VIICN2			
	Mejor	Mediana	Desv. Est.	Mejor	Mediana	Desv. Est.	W	Mejor	Mediana	Desv. Est.	W
F1	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	3.73E-09	2.99E-08	✓	9.31E-10	2.33E-08	2.50E-08	✓
F2	**0.00E+00**	**0.00E+00**	5.20E+02	**0.00E+00**	**0.00E+00**	3.12E-06	=	**0.00E+00**	**0.00E+00**	0.00E+00	✗
F3	**0.00E+00**	**0.00E+00**	2.41E+04	**0.00E+00**	**0.00E+00**	3.12E-06	✗	**0.00E+00**	**0.00E+00**	0.00E+00	✗
F4	**0.00E+00**	**7.17E+03**	1.75E+05	4.18E+06	1.69E+08	6.68E+08	✓	1.07E+05	1.32E+08	4.79E+08	✓
F5	**0.00E+00**	**1.13E+04**	1.41E+05	4.22E+05	3.60E+08	8.41E+08	✓	2.60E+05	2.44E+08	5.17E+08	✓
F6	**0.00E+00**	**1.08E+04**	1.31E+05	1.37E+05	1.99E+08	7.01E+08	✓	8.10E+04	1.38E+08	5.16E+08	✓
F7	**0.00E+00**	**9.30E+03**	3.59E+04	7.62E+04	1.24E+08	1.76E+08	✓	1.22E+02	1.58E+08	2.49E+08	✓
F8	**0.00E+00**	**9.66E+03**	4.75E+04	2.21E+05	8.09E+07	3.28E+08	✓	3.50E+06	1.96E+08	3.45E+08	✓
F9	**1.22E-04**	**1.50E+04**	2.09E+05	8.05E+04	1.45E+08	3.51E+08	✓	8.20E+05	8.58E+07	3.42E+08	✓
F10	**0.00E+00**	**3.69E+04**	9.88E+05	2.67E+06	7.54E+08	2.72E+09	✓	6.09E+05	1.66E+09	3.23E+09	✓
F11	**0.00E+00**	**5.21E+04**	8.70E+05	8.88E+05	4.27E+08	2.56E+09	✓	3.03E+04	4.77E+08	1.73E+09	✓
F12	**2.44E-04**	**6.43E+04**	3.68E+05	1.63E+07	2.97E+09	3.97E+09	✓	1.26E+07	7.29E+08	2.09E+09	✓
F13	**0.00E+00**	**6.03E+04**	2.74E+05	4.19E+06	1.89E+09	3.10E+09	✓	1.88E+06	1.12E+09	1.56E+09	✓
F14	**1.22E-04**	**1.73E+05**	4.44E+05	2.45E+05	2.78E+09	3.27E+09	✓	5.49E+05	9.50E+08	3.36E+09	✓
F15	**1.45E+02**	**3.31E+05**	1.27E+06	2.29E+07	1.66E+09	3.44E+09	✓	7.70E+05	2.84E+09	3.91E+09	✓
F16	**9.44E+02**	**5.36E+04**	1.92E+06	3.72E+07	3.76E+09	1.35E+10	✓	3.34E+07	4.09E+09	2.16E+10	✓
F17	**5.76E+03**	**3.11E+05**	2.74E+06	7.68E+06	1.40E+10	2.19E+10	✓	1.81E+07	8.14E+09	2.16E+10	✓
F18	**3.53E+04**	**4.94E+05**	3.43E+06	3.35E+06	1.97E+10	2.42E+10	✓	1.26E+06	7.95E+09	1.86E+10	✓

Tabla. 6.9.: Estadísticas dimensión 1000. "W" prueba de Wilcoxon, "✓" diferencia significativa a favor de DVIIC, "="no existe diferencia significativa and "✗" diferencia significativa en contra de DVIIC

D1000	DVIIC			VIICN1				VIICN2			
	Mejor	Mediana	Desv. Est.	Mejor	Mediana	Desv. Est.	W	Mejor	Mediana	Desv. Est.	W
F1	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	7.26E-08	1.09E-07	✓	**0.00E+00**	4.10E-08	1.40E-07	✓
F2	**0.00E+00**	**0.00E+00**	3.80E+04	**0.00E+00**	**0.00E+00**	7.62E-07	=	**0.00E+00**	**0.00E+00**	0.00E+00	✗
F3	**0.00E+00**	1.12E+03	3.28E+04	**0.00E+00**	**0.00E+00**	3.06E-06	✗	**0.00E+00**	**0.00E+00**	1.53E-06	✗
F4	**0.00E+00**	**4.08E+04**	6.88E+05	1.43E+06	6.67E+08	1.57E+09	✓	3.89E+07	1.10E+09	1.38E+09	✓
F5	**0.00E+00**	**5.50E+04**	1.20E+06	1.23E+06	1.72E+09	2.22E+09	✓	3.77E+06	6.11E+08	2.01E+09	✓
F6	**0.00E+00**	**1.09E+05**	1.21E+06	5.14E+06	9.54E+08	1.92E+09	✓	4.97E+05	7.58E+08	2.19E+09	✓
F7	**0.00E+00**	**1.21E+04**	2.00E+05	1.42E+04	2.88E+08	7.31E+08	✓	5.53E+05	2.12E+08	6.74E+08	✓
F8	**0.00E+00**	**4.22E+04**	7.25E+05	2.01E+03	4.18E+08	5.54E+08	✓	2.38E+06	6.21E+08	1.08E+09	✓
F9	**2.64E+01**	**6.21E+04**	4.48E+05	3.43E+05	3.73E+08	1.23E+09	✓	1.86E+03	3.02E+08	8.76E+08	✓
F10	**0.00E+00**	**1.62E+05**	7.70E+05	6.32E+06	3.90E+09	9.33E+09	✓	6.69E+06	1.01E+09	3.51E+09	✓
F11	**4.48E+02**	**2.57E+05**	2.81E+06	4.82E+06	1.58E+09	6.50E+09	✓	1.01E+06	1.27E+09	5.55E+09	✓
F12	**2.80E+01**	**3.92E+05**	2.61E+06	1.50E+07	1.37E+09	7.48E+09	✓	3.73E+05	3.11E+09	6.21E+09	✓
F13	**8.24E+01**	**6.10E+04**	9.64E+05	7.93E+05	3.10E+09	5.58E+09	✓	3.50E+07	1.71E+09	4.62E+09	✓
F14	**0.00E+00**	**6.35E+04**	1.48E+06	7.54E+06	3.17E+09	9.16E+09	✓	4.08E+05	3.61E+09	8.19E+09	✓
F15	**0.00E+00**	**1.21E+05**	1.96E+06	1.13E+07	9.32E+08	7.06E+09	✓	1.11E+08	2.14E+09	5.46E+09	✓
F16	**3.84E+03**	**6.76E+05**	6.92E+06	1.00E+05	3.92E+10	4.63E+10	✓	2.54E+07	2.05E+10	4.39E+10	✓
F17	**1.02E+02**	**1.43E+06**	1.14E+07	7.86E+06	1.35E+10	3.59E+10	✓	2.84E+08	1.58E+10	4.67E+10	✓
F18	**2.72E+04**	**7.83E+05**	1.04E+07	8.55E+06	1.38E+10	3.84E+10	✓	5.55E+06	7.08E+09	2.56E+10	✓

7 Estudio de DE-LoCoS con DG2 y DVIIC

7.1 Comparativa de DG2 y DVIIC bajo el esquema LoCoS

En la Sección 5.2, se introdujo la propuesta de este trabajo, los algoritmos DE-LoDeS y DE-LoCoS; los cuales fueron probados utilizando el método de descomposición VIIC y comparados contra el esquema de Co-evolución Cooperativa, mostrando que el esquema de Cooperación Local es efectivo y altamente competitivo para resolver problemas de alta dimensión con restricciones. De acuerdo a los resultados, se concluyó que DE-LoCoS es mejor que DE-LoDeS.

Posteriormente en las Secciones 6 y 6.3, se propusieron mejoras al algoritmo de descomposición VIIC, agregando estrategias heurísticas (recocido simulado, estrategias de vecindario) y meta-heurísticas (algoritmo genético), dando como resultado una mejora en la capacidad para descomponer los problemas.

A partir de estos resultados, en esta sección se presenta la experimentación para comparar el rendimiento de DE-LoCoS utilizando la versión propuesta de VIIC que utiliza el algoritmo genético (DVIIC), y el método de DG2 [31] descrito en la sección 4.3.2. Ambos algoritmos son comparados con CCSaNSDE-DG2, un algoritmo de Co-evolución Cooperativa del *estado del arte* que utiliza como algoritmo de optimización SaNSDE [46] y DG2 como método de descomposición. También se presenta la experimentación del efecto de agregar la descomposición dentro de la fase de búsqueda local. A continuación en el Algoritmo 7.21 se describe el algoritmo de DE-LoCoS utilizado para esta experimentación.

Algoritmo 7.21 Algoritmo Memético, DE-LoCoS usando DVIIC o DG2

Asignar el número máximo de evaluaciones de la función $MaxFEs$
Calcular la frecuencia de activación de la búsqueda local $Cyclefrecuency = (4 \times D)/(0{,}3 \times NP)$
Calcular el máximo número de evaluaciones permitidas para el buscador local $MaxLocalFes = MaxFEs \times 0{,}3$
Obtener $[S, K]$ de DVIIC or DG2
Generar aleatoriamente la población inicial $Pop = (\mathbf{x}_1, \ldots, \mathbf{x}_{NP})$
Evaluar la población inicial
repetir
  para $i = 1$ a NP **hacer**
    Seleccionar tres vectores aleatorios $r_0 \neq r_1 \neq r_2 \neq i$
    Calcular $\mathbf{u}_{i,g+1}$ utilizando el operador $rand/1/bin$ (ver Ecuaciones 3.3 & 3.4)
    si $f(\mathbf{u}_{i,g+1})$ es mejor de acuerdo a las reglas de factibilidad $f(\mathbf{x}_i)$ **entonces**
      $\mathbf{x}_{i,g+1} = \mathbf{u}_{i,g+1}$
    fin si
  fin para
  si $currentCycle \ mod \ Cyclefrecuency == 0$ y $MaxLocalFes$ aun no se alcanzan **entonces**
    para $k = 1$ a K **hacer**
      asignar $\mathbf{x}_{new} =$ HookeJeevesMethod$(\mathbf{x}_{new}, \Delta, S_n^k)$
    fin para
    si $\mathbf{x}_{new}$ es mejor de acuerdo alas reglas de factibilidad $\mathbf{x}_{best}$ **entonces**
      $\mathbf{x}_{best} = \vec{x}_{new}$
    fin si
  fin si
hasta que $MaxFEs$ sean alcanzadas
Regresar la mejor solución encontrada

7.1.1 Diseño experimental

Se realizaron tres experimentos: (1) El primero con el objetivo de analizar el efecto de agregar la descomposición del problema durante la fase local del algoritmo memético, por lo que, DE-LoCoS con DVIIC fue comparado contra el mismo DE-LoCoS pero sin incorporar DVIIC, al cual se le ha nombrado como MDE (Memetic Differential Evolution). (2) El segundo experimento busca comparar el rendimiento entre DVIIC y DG2 bajo el esquema de DE-LoCoS. Tal como se mencionó en la sección 4.3.2, DG2 es un algoritmo nuevo, que se ha utilizado en problemas sin restricciones. Sin embargo, utilizando la misma estrategia que la utilizada en DVIIC para incorporar la información de las restricciones (ver Ecuación 7.1 o la sección 6.2), DG2 es capaz de atacar

este tipo de problemas y (3) por último, el tercer experimento, compara las mejor versión de DE-LoCoS del experimento 2, contra CCSaNSDE-DG2.

Para evaluar los algoritmos se utilizó el conjunto de funciones de prueba propuestos por Sayed [37]. En todos los experimentos se utilizó la prueba estadística de suma de rangos de Wilcoxon con una confianza del 95 %, para comparar función a función si existe diferencia significativa en el rendimiento de los algoritmos. Del mismo modo, se realizó la clasificación de algoritmos propuesta por Elsayed [9] como evidencia empírica para determinar la eficacia de los algoritmos. Estas herramientas fueron descritas a detalle en la Sección 5.1.

$$f'(\mathbf{x}) = f(\mathbf{x}) + cvs(\mathbf{x}) \tag{7.1}$$

Configuración de Parámetros

A continuación se presenta la configuración de los parámetros para los algoritmos de esta experimentación:

- Los parámetros para DE fueron calibrados con la herramienta IRACE [22], los valores sugeridos por IRACE son: factor de escala $F = 0{,}51$, probabilidad de cruza $CR = 0{,}73$ y tamaño de la población $NP = 56$.

- Para el buscador local Hooke-Jeeves, la desviación estándar de las variables de la población actual fue utilizada como tamaño de paso $\Delta = stdx(Pop)$, el criterio de parada es $\epsilon = 1{,}0$e-4, y el factor de reducción del tamaño de paso es $\alpha = 2$.

- SaNSDE [46] es una variante de DE que auto-adapta los parámetros de factor de escala F y probabilidad de CR; para tamaño de población se utilizó el sugerido por la herramienta IRACE $NP = 56$.

Respecto a los métodos de descomposición, DG2 tiene la ventaja de ser un algoritmo que no requiere parámetros, y es un algoritmo determinista, esto quiere decir que provee la misma descomposición para un problema en cada ejecución del método. Por esta razón, para realizar una comparación justa

entre DG2 y DVIIC, ambos métodos son ejecutados únicamente al inicio del algoritmo, como se muestra en la línea 4 del Algoritmo 7.21.

Por otro lado, DVIIC está basado en un algoritmo genético, y requiere parámetros los cuales fueron tomados de la literatura [2], estos son: (1) tamaño de población $popSize = 6$, (2) probabilidad de cruza $pc = 0{,}9$, (3) probabilidad de mutación $pm = 0{,}1$, y (4) finalmente, el máximo número de evaluaciones se ajustó a las mismas evaluaciones requeridas por DG2 (ver Eq. 7.2)[31].

$$maxDecompositionFes = \frac{D(D+1)}{2} + 1 \tag{7.2}$$

7.1.2 Resultados

En esta sección se presentan y discuten los resultados obtenidos en cada uno de los tres experimentos. Las Tablas 7.8, 7.9, y 7.10 muestran los resultados numéricos para las dimensiones 100, 500 y 1000, respectivamente. Los valores resaltados en negrita indican los mejores valores obtenidos por aquellos algoritmos que alcanzaron el 100 % de factibilidad (FR), es decir, alcanzaron una solución factible en las 25 ejecuciones independientes. Las gráficas de convergencia representativas se muestran en las Figuras 7.1, 7.2 y 7.3 para las tres dimensiones 100, 500 y 1000, respectivamente. Las gráficas de convergencia fueron tomadas de la ejecución con el valor correspondiente a la mediana de los problemas que tiene tres restricciones. Las gráficas de convergencia inician una vez que la región factible fue alcanzada. Por último, los modelos de clasificación de algoritmos para cada una de las dimensiones se muestran en las Tablas 7.5, 7.6 y 7.7.

Experimento 1: comparando DE-LoCoS con y sin descomposición

DE-LoCoS fue comparado contra su propia versión sin el método de descomposición en la fase de búsqueda local. Dicha versión fue nombrada Memetic Differential Evolution (MDE). En este experimento, DVIIC fue elegido como método de descomposición porque fue diseñado para descomponer problemas con restricciones.

Tabla. 7.1.: Resumen de la prueba de suma de rangos de Wilcoxon. Número de diferencias a favor (F) , sin diferencia significativa (SdS) y diferencias en contra (C) de DELoCoS-DVIIC contra MDE en las tres dimensiones.

DELoCoS-DVIIC contra MDE									
	100D			500D			1000D		
	F	**SdS**	**C**	**F**	**SdS**	**C**	**F**	**SdS**	**C**
F1-F3	1	2	0	2	1	0	**3**	0	0
F4-F6	**3**	0	0	**3**	0	0	**3**	0	0
F7-F9	2	1	0	0	1	2	**3**	0	0
F10-F12	**3**	0	0	**3**	0	0	0	3	0
F13-F15	**3**	0	0	**3**	0	0	**3**	0	0
F16-F18	**3**	0	0	**3**	0	0	1	2	0
Total	**15**	3	0	**14**	2	2	**13**	5	0

Ambos algoritmos alcanzaron la región factible de forma consistente, llegando al 100 % de factibilidad $FR = 100\,\%$ en las dimensiones 100 y 500 (Tablas 7.8, y 7.9). Por otro lado, DELoCoS-DVIIC encontró soluciones factibles en cada una de las ejecuciones en dimensión 1000 (Tabla 7.10). En contraste, MDE en la dimensión 1000 no pudo encontrar soluciones factibles en los problemas que están compuestos por dos y tres restricciones, alcanzando la zona factible solo en las funciones F01, F04, F07, F10, F13, y F16 que están compuestas de una restricción solamente.

La Tabla 7.1 resume los resultados de aplicar la prueba de suma de rangos de Wilcoxon que compara a DELoCoS y MDE.

Respecto a la dimensión 100, DELoCoS-DVIIC supera a MDE en 15 de las 18 funciones de acuerdo a la prueba de Wilcoxon, mientras que para las funciones F02, F03, y F09 ambos algoritmos muestran un desempeño similar. En la dimensión 500, DELoCoS-DVIIC supera a MDE en 14 de las 18 funciones. En las funciones F03 y F09 no se observan diferencias significativas. Estas dos funciones están compuestas de tres restricciones, y sus funciones objetivo son completa y parcialmente separables respectivamente. Por otro lado, DELoCoS-DVIIC fue superado por MDE en las funciones F07 y F08, las cuales son funciones parcialmente no-separables, y constan de una y dos restricciones. DELoCoS-DVIIC tiene diferencias significativas a favor respecto a MDE en 13 de las 18 funciones en dimensión 1000, y en las funciones F10 a F12, y F17-F18 ambos algoritmos muestran un rendimiento similar.

Las gráficas de convergencia (Figuras. 7.1, 7.2, 7.3) revelan que agregar la descomposición a la fase de búsqueda local, mejora la velocidad para alcanzar la zona factible, permitiendo más exploración en funciones que son parcialmente no-separables y también en aquellas que contienen variables en superposición.

A partir de estos resultados, se puede concluir que integrar la descomposición a un esquema memético tiene un impacto positivo al momento de optimizar problemas de alta dimensión con restricciones. Los resultados muestran que se mejora la habilidad para alcanzar la zona factible, se obtienen mejores resultados finales y la velocidad de convergencia es mayor cuando la descomposición está activa.

Experimento 2: Comparando DELoCoS con DVIIC y DG2

Respecto al porcentaje de factibilidad, ambos algoritmos muestran ser constantes en las tres dimensiones; con excepción de la función F06 en dimensión 1000, en la cual DELoCoS-DG2 no logro alcanzar el 100 % de factibilidad. (Tablas 7.8, 7.9, and 7.10)

Los resultados finales analizados con la prueba estadística de Wilcoxon se muestran en la Tabla 7.2. En dimensión 100, para la mitad de las funciones de prueba DELoCoS-DVIIC fue superado por DELoCoS-DG2, particularmente en aquellas funciones que tienen variables con superposición y grupos de variables empalmadas no-separables (F7 to F15).

Respecto a dimensión 500, DELoCoS-DVIIC supera en rendimiento a DELoCoS-DG2 en las funciones F7, F12, F14, F18. En las funciones F01, F08, F10, F11, F15 y F17 no se observa diferencia significativa entre los dos algoritmos. Finalmente, DELoCoS-DG2 supera a DeLoCoS-DVIIC en las 8 restantes funciones de prueba.

Analizando la dimensión 1000, cuando las funciones son totalmente separables (F1 a F3), DG2 obtiene mejores resultados que DVIIC en dos de estas funciones. El mismo comportamiento ocurre cuando los problemas tienen variables empalmadas o con superposición (Funciones F16-F18). Sin embargo, en las funciones F7 a F12, con variables parcialmente separables, DVIIC muestra un mejor desempeño.

Tabla. 7.2.: Resumen de la prueba de suma de rangos de Wilcoxon. Número de diferencias a favor (F) , sin diferencia significativa (SdS) y diferencias en contra (C) de DELoCoS-DVIIC contra DELoCoS-DG2 en las tres dimensiones.

DELoCoS-DVIIC contra DELoCoS-DG2									
	100D			500D			1000D		
	F	SdS	C	F	SdS	C	F	SdS	C
F1-F3	0	**3**	0	0	1	**2**	0	1	**2**
F4-F6	**2**	1	0	0	0	**3**	1	1	1
F7-F9	0	1	**2**	1	1	1	**2**	0	1
F10-F12	0	0	**3**	1	**2**	0	**2**	1	0
F13-F15	0	0	**3**	1	1	1	1	**2**	0
F16-F18	0	**2**	1	1	1	1	0	1	**2**
Total	2	7	**9**	4	6	**8**	**6**	**6**	**6**

Continuando con el análisis de los resultados de estos dos algoritmos, la Tabla 7.3 muestra los resultados de la prueba de Wilcoxon agrupando las funciones de prueba por el número de restricciones. Vale la pena notar que DG2 obtuvo mejores resultados cuando los problemas tienen tres restricciones en dimensión 1000. Sin embargo, considerando todo el conjunto de funciones de prueba, los resultados de la prueba de Wilcoxon indican que el rendimiento general de ambos algoritmos es bastante similar.

El comportamiento de convergencia en dimensión 100 (Figura 7.1) para las funciones con tres restricciones indica que DELoCoS-DG2 tiene una convergencia más rápida que DeLoCoS-DVIIC. Sin embargo, la versión DG2 alcanza la región factible después de DeLoCoS-DVIIC. En las dimensiones 500 y 1000, la región factible fue alcanzada primero por DeLoCoS-DG2.

A partir de los resultados de este segundo experimento, se puede concluir, que DG2 provee un mejor rendimiento en las dimensiones 100 y 500. Por otro lado, en la dimensión 1000 el número de restricciones del problema es una característica que marca diferencia y debe ser considerada para seleccionar el método de descomposición más adecuado. Es decir, si el problema tiene una gran cantidad de restricciones, DG2 parece ser la mejor opción, por el contrario, DVIIC tiene un buen desempeño en problemas con una o dos restricciones, sobre todo en aquellas que son parcialmente separables.

Tabla. 7.3.: Número de diferencias a favor (F) , sin diferencia significativa (SdS) y diferencias en contra (C) de DELoCoS-DVIIC contra DELoCoS-DG2 agrupando las funciones de acuerdo al número de restricciones

DELoCoS-DVIIC contra DELoCoS-DG2									
	100D			500D			1000D		
	F	SdS	C	F	SdS	C	F	SdS	C
1C	1	1	**4**	1	2	**3**	**3**	1	2
2C	0	3	**3**	1	**3**	2	**3**	2	1
3C	1	**3**	2	2	1	**3**	0	2	**4**
Total	2	7	**9**	4	6	**8**	**6**	**6**	**6**

Experimento 3: DELoCoS-DG2 contra CCSaNSDE-DG2

Considerando los resultados del experimento 2, DELoCoS-DG2 fue seleccionado para ser comparado contra el algoritmo de Co-evolución Cooperativa que utiliza SaNSDE como algoritmo de búsqueda y DG2 como método de descomposición.

De las Tablas 7.8, 7.9, y 7.10 es interesante observar que CCSaNSDE-DG2 logra el 100 % de factibilidad en todo el conjunto de problemas de prueba en las dimensiones 100 y 1000. Sin embargo, en dimensión 500, CCSaNSDE-DG2 no logra ser consistente para alcanzar la región factible en las funciones F13, F16, F17, y F18.

Los resultados de la prueba de Wilcoxon en la Tabla 7.4 revelan que el rendimiento de CCSaNSDE-DG2 contra DELoCoS-DG2, disminuye cuando la dimensión de los problemas incrementa.

De acuerdo a las gráficas de convergencia en dimensión 100 (Figura 7.1), DELoCoS-DG2 alcanza la zona factible antes que CCSaNSDE-DG2. De hecho, en la función F18 (Fig. 7.1c) CCSaNSDE-DG2 quedo atrapado en un óptimo local. En dimensión 500 (Figura 7.2) la misma dificultad con los óptimos locales de CCSaNSDE-DG2 aparece en las funciones F12 y F18 (Figuras 7.2a, 7.2c).

En la dimensión 1000, incluso cuando CCSaNSDE alcanzó la región factible más rápido que DELoCoS-DG2, tal como se muestra en la gráfica de convergencia en la Figura 7.3, se puede observar un comportamiento de convergencia prematura de nuevo en las funciones F12, F15 y F18. Tales

Tabla. 7.4.: Resumen de la prueba de suma de rangos de Wilcoxon. Número de diferencias a favor (F) , sin diferencia significativa (SdS) y diferencias en contra (C) de DELoCoS-DG2 contra CCSaNSDE-DG2 en las tres dimensiones

DELoCoS-DG2 contra CCSaNSDE-DG2									
	100D			500D			1000D		
	F	SdS	C	F	SdS	C	F	SdS	C
F1-F3	1	0	**2**	1	0	**2**	**3**	0	0
F4-F6	1	0	**2**	1	**2**	0	**3**	0	0
F7-F9	**2**	0	1	**2**	0	1	**3**	0	0
F10-F12	1	0	**2**	**3**	0	0	**3**	0	0
F13-F15	1	0	**2**	0	0	**3**	**3**	0	0
F16-F18	**3**	0	0	**3**	0	0	**3**	0	0
Total	**9**	0	**9**	**10**	2	6	**18**	0	0

Tabla. 7.5.: Puntaje de los cuatro algoritmos en dimensión 100, ordenados de acuerdo a su puntaje final (FS_z), S_z^{best} es el puntaje de acuerdo a los mejores valores obtenidos, $S_z^{avarage}$ el puntaje de acuerdo a la media de los resultados y FR_{mean} la media de factibilidad del algoritmo en las funciones de prueba.

100D	CCSaNSDE-DG2	DELoCoS-DG2	DELoCoS-DVIIC	MDE
FS_z	**10.75**	9	9	5.7
S_z^{best}	**11.99**	9	9	4.49
$S_z^{avarage}$	**10.99**	9	9	6.93
FR_{mean}	93.55 %	**100 %**	**100 %**	99.77 %

problemas de prueba están compuestos por variables empalmadas y en superposición.

La discusión anterior lleva a concluir que, la combinación de un algoritmo de evolución diferencial (como buscador global) y un algoritmo de búsqueda directa como lo es el método de Hooke-Jeeves, pero agregando un método de descomposición, provee mejores resultados que SaNSDE (una variante de DE claramente más compleja) bajo el esquema de Co-evolución Cooperativa.

Método de clasificación de algoritmos

Los resultados de la clasificación de algoritmos, para las dimensiones 100, 500 and 1000 son mostrados en las Tablas 7.5, 7.6 y 7.7, respectivamente. Los mejores resultados se muestran marcados en negrita. Esta técnica permite

Tabla. 7.6.: Puntaje de los cuatro algoritmos en dimensión 500, ordenados de acuerdo a su puntaje final (FS_z), S_z^{best} es el puntaje de acuerdo a los mejores valores obtenidos, $S_z^{avarage}$ el puntaje de acuerdo a la media de los resultados y FR_{mean} la media de factibilidad del algoritmo en las funciones de prueba.

500D	DELoCoS-DVIIC	DELoCoS-DG2	MDE	CCSaNSDE-DG2
FS_z	**13.66**	12.01	9.48	6.77
S_z^{best}	**13.36**	10.79	8.57	8.17
$S_z^{avarage}$	**13.96**	13.23	10.4	6.92
FR_{mean}	**100 %**	**100 %**	**100 %**	89.77 %

Tabla. 7.7.: Puntaje de los cuatro algoritmos en dimensión 1000, ordenados de acuerdo a su puntaje final (FS_z), S_z^{best} es el puntaje de acuerdo a los mejores valores obtenidos, $S_z^{avarage}$ el puntaje de acuerdo a la media de los resultados y FR_{mean} la media de factibilidad del algoritmo en las funciones de prueba.

1000D	DELoCoS-DVIIC	DELoCoS-DG2	MDE	CCSaNSDE-DG2
FS_z	**17.99**	17.1	10.04	0.49
S_z^{best}	**17.99**	17.99	17.18	0
$S_z^{avarage}$	**17.99**	16.29	17.97	0.99
FR_{mean}	**100 %**	99.77 %	57.11 %	**100 %**

clasificar los algoritmos de acuerdo a los resultados finales y el porcentaje de factibilidad.

El puntaje en dimensión 100 (Table 7.5), sugiere que el mejor rendimiento lo obtuvo CCSaNSDE-DG2. Ello porque el mejor valor final (S_z^{best}) y el valor medio (S_z^{best}) de los resultados numéricos fueron mejores que los resultados de los otros algoritmos. Cuando CCSaNSDE-DG2 no logra el 100 % de factibilidad, las versiones DELoCoS están posicionadas en el segundo y tercer lugar, pero su puntaje final no muestra una diferencia clara entre ellos.

La clasificación en dimensión 500 (Tabla 7.6) muestra que las versiones de DELoCoS tienen el mejor rendimiento, seguidos por MDE y CCSaNSDE-DG2.

En dimensión 1000, el mejor puntaje final (FS_z) lo obtuvieron las versiones de DELoCoS, seguidos por los algoritmos MDE y CCSaNSDE-DG2. Es importantes remarcar que MDE presenta un puntaje más alto respecto a CCSaNSDE-DG2. Esto se debe a que MDE logra mejores resultados en el conjunto de funciones de prueba completo incluso cuando CCSaNSDE-DG2 logra el 100 % de factibilidad.

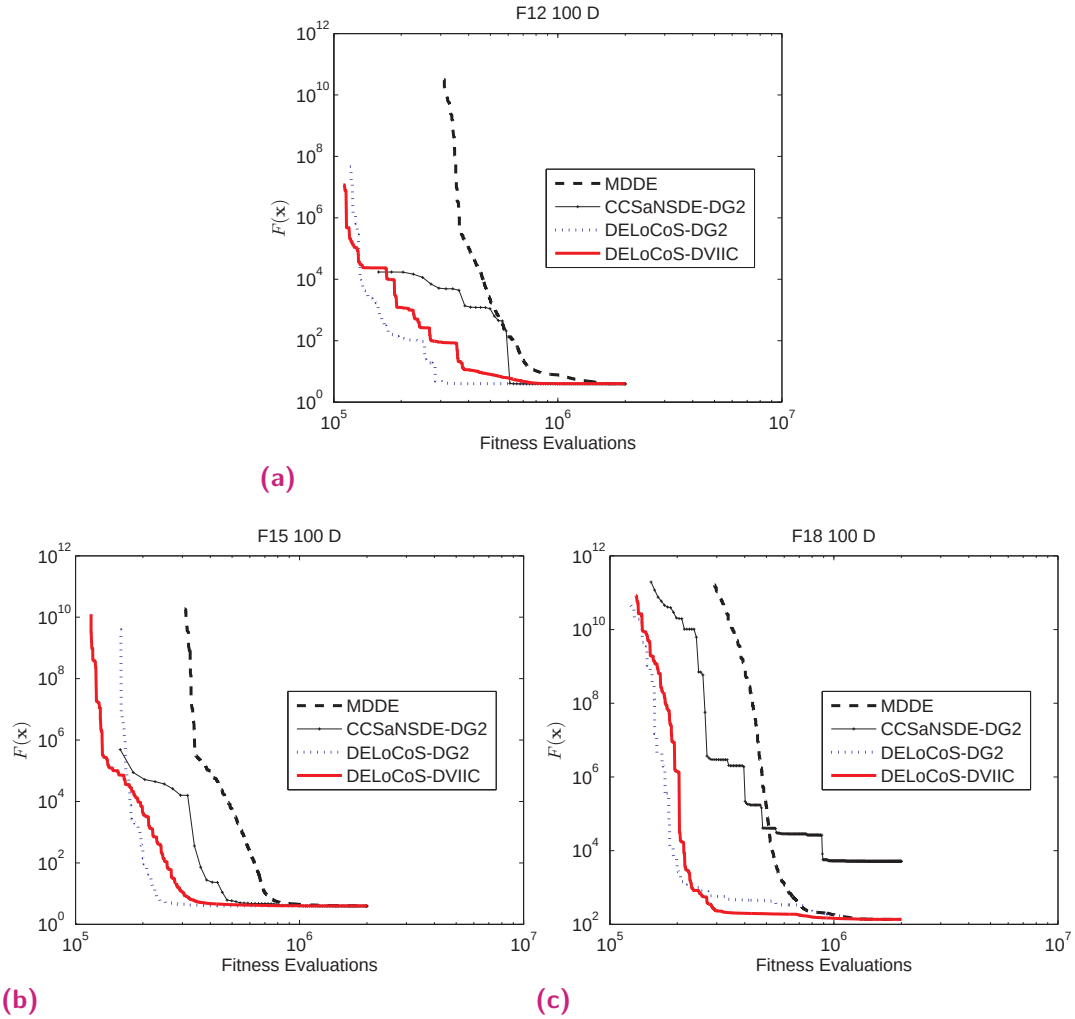


Figura 7.1.: Gráfica de convergencia para la dimensión 100 correspondientes a la ejecución mediana de las funciones F12, F15 and F18. Ambos ejes se encuentran en escala logarítmica.

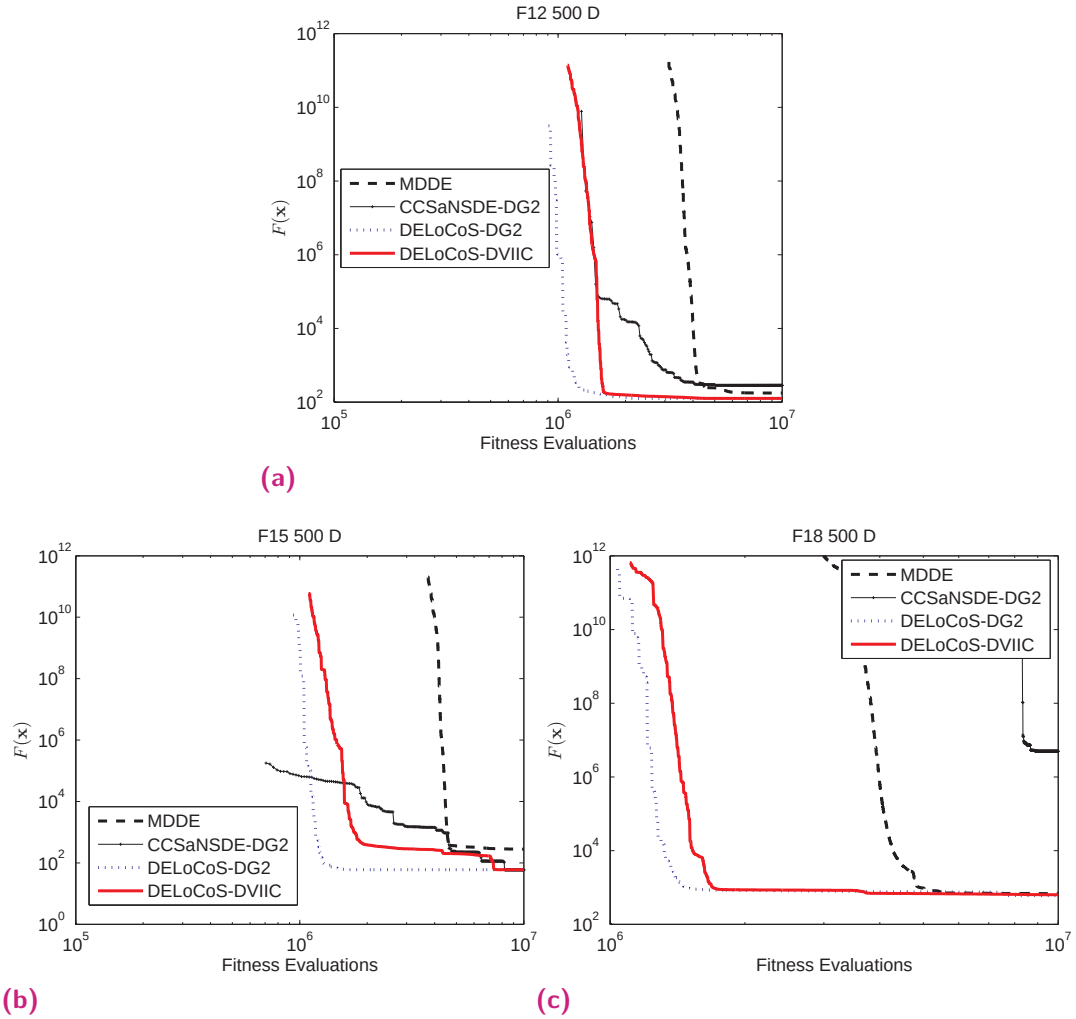


Figura 7.2.: Gráfica de convergencia para la dimensión 500 correspondientes a la ejecución mediana de las funciones F12, F15 and F18. Ambos ejes se encuentran en escala logarítmica.

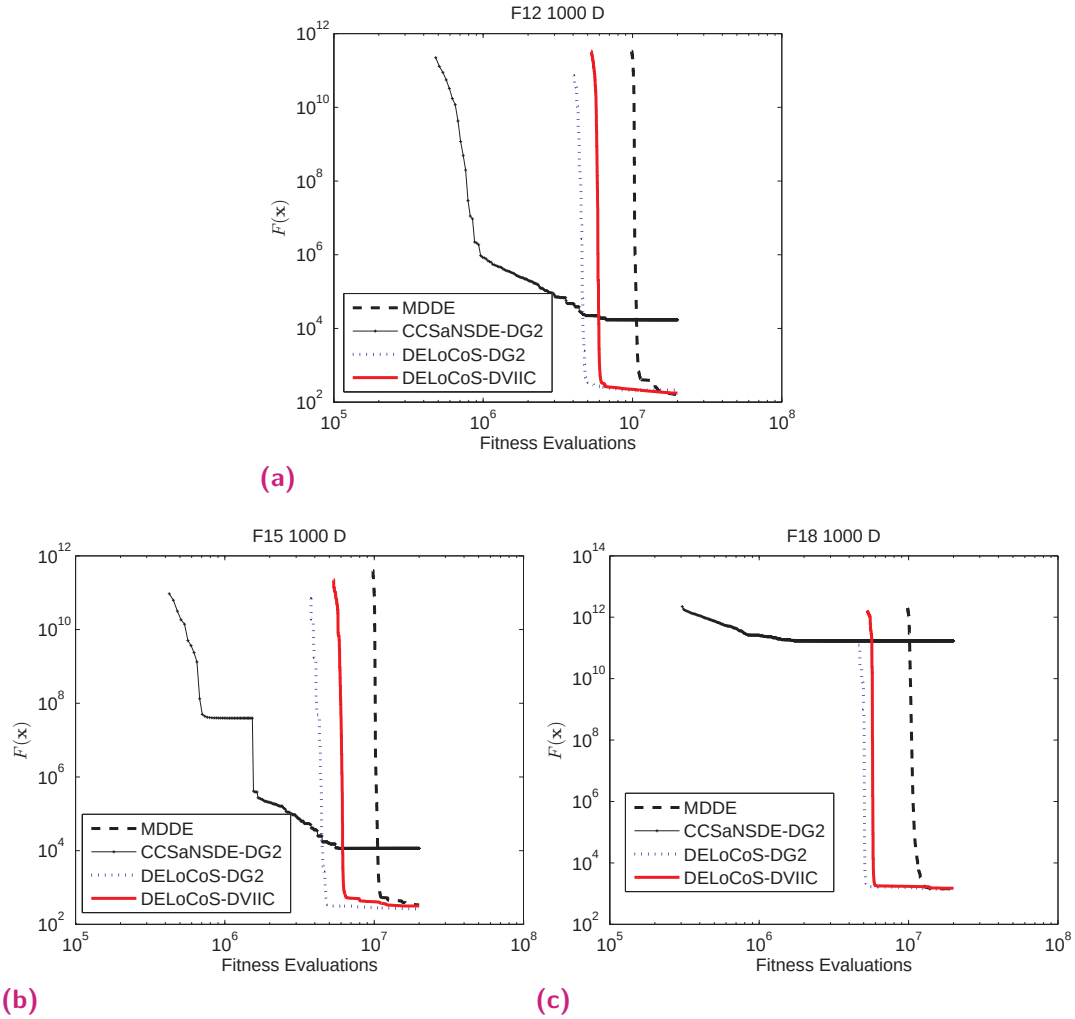


Figura 7.3.: Gráfica de convergencia para la dimensión 1000 correspondientes a la ejecución mediana de las funciones F12, F15 and F18. Ambos ejes se encuentran en escala logarítmica.

Tabla. 7.8.: Resultados estadísticos para la dimensión 100. FR porcentaje de factibilidad. En negrita se presentan los mejores resultados para los algoritmos cuando alcanzan el 100 % de factibilidad.

	F01				F02				F03			
	FR	Mejor	Mediana	std	FR	Mejor	Mediana	std	FR	Mejor	Mediana	std
DELOCOS-DG2	**100 %**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**100 %**	**3.00E+00**	**3.00E+00**	**6.21E-16**	**100 %**	**5.00E+00**	**5.00E+00**	**9.06E-16**
DELOCOS-DVIIC	**100 %**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**100 %**	**3.00E+00**	**3.00E+00**	5.66E-16	**100 %**	**5.00E+00**	**5.00E+00**	**9.06E-16**
MDDE	**100 %**	2.39E-169	4.47E-166	0.00E+00	**100 %**	**3.00E+00**	**3.00E+00**	6.08E-16	**100 %**	**5.00E+00**	**5.00E+00**	**9.06E-16**
CCSaNSDE-DG2	**100 %**	7.92E-122	6.04E-10	3.02E-09	**100 %**	2.94E+00	3.10E+00	8.00E-01	**100 %**	4.88E+00	4.88E+00	3.68E-08
	F04				F05				F06			
DELOCOS-DG2	**100 %**	**0.00E+00**	**3.35E-31**	1.16E-30	**100 %**	**3.00E+00**	**3.00E+00**	1.38E-15	**100 %**	**4.00E+00**	**4.00E+00**	**1.78E-15**
DELOCOS-DVIIC	**100 %**	**0.00E+00**	2.10E-31	**8.55E-31**	**100 %**	**3.00E+00**	**3.00E+00**	**1.76E-15**	**100 %**	**4.00E+00**	**4.00E+00**	1.61E-15
MDDE	**100 %**	1.85E-53	8.58E-29	3.79E-28	**100 %**	**3.00E+00**	**3.00E+00**	2.11E-08	**100 %**	**4.00E+00**	**4.00E+00**	4.93E-06
CCSaNSDE-DG2	**100 %**	6.45E-30	3.03E-27	6.25E-27	**100 %**	2.94E+00	2.94E+00	2.09E-08	**100 %**	3.90E+00	3.90E+00	4.31E-06
	F07				F08				F09			
DELOCOS-DG2	**100 %**	**3.18E-264**	**5.30E-149**	**2.65E-148**	**100 %**	**0.00E+00**	**2.96E-33**	**5.37E-33**	**100 %**	**2.00E+00**	**2.00E+00**	**4.53E-16**
DELOCOS-DVIIC	**100 %**	1.23E-32	2.84E-19	1.04E-18	**100 %**	6.17E-247	6.28E-30	2.88E-29	**100 %**	**2.00E+00**	**2.00E+00**	**4.53E-16**
MDDE	**100 %**	4.84E-18	8.83E-15	4.31E-14	**100 %**	1.28E-29	3.58E-26	1.07E-25	**100 %**	**2.00E+00**	**2.00E+00**	**4.53E-16**
CCSaNSDE-DG2	**100 %**	2.09E-22	7.48E-21	8.72E-21	**100 %**	3.43E-35	4.31E-30	1.53E-29	**100 %**	1.94E+00	1.94E+00	2.49E-13
	F10				F11				F12			
DELOCOS-DG2	**100 %**	**0.00E+00**	**2.91E-298**	**0.00E+00**	**100 %**	**3.00E+00**	**3.00E+00**	**7.85E-16**	**100 %**	**4.00E+00**	**4.00E+00**	**1.49E-15**
DELOCOS-DVIIC	**100 %**	1.37E-292	2.35E-21	5.88E-21	**100 %**	**3.00E+00**	**3.00E+00**	5.78E-11	**100 %**	**4.00E+00**	**4.00E+00**	1.86E-04
MDDE	**100 %**	1.09E-18	2.86E-16	5.51E-16	**100 %**	**3.00E+00**	**3.00E+00**	4.01E-04	**100 %**	**4.00E+00**	4.01E+00	2.20E-02
CCSaNSDE-DG2	**100 %**	7.80E-24	1.38E-01	6.87E-01	**100 %**	2.94E+00	2.94E+00	4.14E-03	**100 %**	3.90E+00	3.90E+00	3.99E-09
	F13				F14				F15			
DELOCOS-DG2	**100 %**	**4.89E-22**	**1.87E-14**	**9.17E-14**	**100 %**	**3.00E+00**	**3.00E+00**	**3.58E-09**	**100 %**	**4.00E+00**	**4.00E+00**	**1.26E-06**
DELOCOS-DVIIC	**100 %**	9.30E-19	3.96E-08	1.58E-07	**100 %**	**3.00E+00**	**3.00E+00**	3.54E-03	**100 %**	**4.00E+00**	4.01E+00	1.75E-02
MDDE	**100 %**	9.99E-08	7.41E-07	1.60E-06	**100 %**	**3.00E+00**	3.01E+00	2.35E-02	**100 %**	**4.00E+00**	4.02E+00	2.08E-02
CCSaNSDE-DG2	**100 %**	3.13E-07	7.55E+00	3.24E+01	**100 %**	2.94E+00	2.94E+00	4.90E-03	**100 %**	**4.00E+00**	**4.00E+00**	1.09E-02
	F16				F17				F18			
DELOCOS-DG2	**100 %**	**8.17E+01**	**1.12E+02**	2.78E+01	**100 %**	**1.37E+02**	1.39E+02	6.24E+00	**100 %**	**1.37E+02**	**1.37E+02**	4.19E-02
DELOCOS-DVIIC	**100 %**	**8.17E+01**	1.34E+02	**1.10E+01**	**100 %**	**1.37E+02**	**1.37E+02**	**3.34E+00**	**100 %**	**1.37E+02**	**1.37E+02**	**2.86E-02**
MDDE	**100 %**	1.37E+02	1.40E+02	4.14E+00	**100 %**	1.39E+02	1.43E+02	3.24E+00	**100 %**	**1.37E+02**	**1.37E+02**	3.05E-01
CCSaNSDE-DG2	**100 %**	1.64E+02	3.41E+07	1.70E+08	**48 %**	1.99E+02	9.17E+04	3.15E+05	**36 %**	3.72E+02	5.95E+05	1.62E+06

Tabla. 7.9.: Resultados estadísticos para la dimensión 500. FR porcentaje de factibilidad. En negrita se presentan los mejores resultados para los algoritmos cuando alcanzan el 100 % de factibilidad.

	F01				F02				F03			
	FR	Mejor	Mediana	std	FR	Mejor	Mediana	std	FR	Mejor	Mediana	std
DELOCOS-DG2	**100 %**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**100 %**	**6.00E+00**	**6.00E+00**	**7.19E-14**	**100 %**	**1.00E+01**	**1.00E+01**	**9.63E-13**
DELOCOS-DVIIC	**100 %**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**100 %**	**6.00E+00**	**6.00E+00**	3.07E-13	**100 %**	**1.00E+01**	**1.00E+01**	1.96E-12
MDDE	**100 %**	8.26E-145	2.38E-143	3.38E-143	**100 %**	**6.00E+00**	**6.00E+00**	4.85E-13	**100 %**	**1.00E+01**	**1.00E+01**	3.46E-12
CCSaNSDE-DG2	**100 %**	1.65E-105	1.62E+04	2.37E+04	**100 %**	5.92E+00	1.24E+02	5.03E+02	**100 %**	**1.00E+00**	1.16E+00	4.02E-01
	F04				F05				F06			
DELOCOS-DG2	**100 %**	**2.91E-15**	**8.65E-08**	**2.61E-07**	**100 %**	**6.00E+00**	**6.00E+00**	**9.05E-06**	**100 %**	**9.00E+00**	**9.00E+00**	**4.68E-03**
DELOCOS-DVIIC	**100 %**	9.67E-13	5.11E-04	2.55E-03	**100 %**	**6.00E+00**	**6.00E+00**	1.14E-03	**100 %**	**9.00E+00**	9.15E+00	3.23E-01
MDDE	**100 %**	9.83E-03	7.25E-02	9.25E-02	**100 %**	6.49E+00	8.21E+00	1.45E+00	**100 %**	9.61E+00	1.66E+01	6.37E+00
CCSaNSDE-DG2	**100 %**	9.70E-05	1.79E+01	7.41E+01	**100 %**	5.93E+00	3.99E+01	1.62E+02	**100 %**	9.21E+00	3.16E+02	1.53E+03
	F07				F08				F09			
DELOCOS-DG2	**100 %**	1.93E+02	3.75E+02	1.28E+02	**100 %**	2.35E+00	4.62E+01	3.90E+01	**100 %**	**4.00E+00**	**4.14E+01**	**4.36E+01**
DELOCOS-DVIIC	**100 %**	**8.68E-02**	9.57E+01	9.01E+01	**100 %**	3.39E-02	4.00E+01	5.29E+01	**100 %**	4.08E+00	8.43E+01	7.50E+01
MDDE	**100 %**	**2.24E-03**	**1.54E-01**	**2.40E-01**	**100 %**	**6.71E-05**	**1.01E-01**	**1.71E-01**	**100 %**	4.07E+00	6.80E+01	5.78E+01
CCSaNSDE-DG2	**100 %**	1.02E+02	1.81E+03	1.80E+03	**100 %**	9.63E+00	5.73E+02	1.57E+03	**100 %**	3.70E+01	1.95E+03	3.19E+03
	F10				F11				F12			
DELOCOS-DG2	**100 %**	1.18E+02	1.54E+02	2.30E+01	**100 %**	**7.40E+01**	1.34E+02	**3.55E+01**	**100 %**	**7.60E+01**	**1.04E+02**	**2.54E+01**
DELOCOS-DVIIC	**100 %**	**6.80E+01**	**1.38E+02**	3.55E+01	**100 %**	**7.40E+01**	**1.14E+02**	3.83E+01	**100 %**	**7.60E+01**	1.32E+02	3.92E+01
MDDE	**100 %**	1.18E+02	1.69E+02	**2.13E+01**	**100 %**	1.24E+02	1.82E+02	2.78E+01	**100 %**	1.26E+02	1.79E+02	3.20E+01
CCSaNSDE-DG2	**100 %**	2.61E+02	2.70E+09	1.30E+10	**100 %**	2.28E+02	3.07E+02	1.05E+02	**100 %**	2.30E+02	1.58E+09	7.92E+09
	F13				F14				F15			
DELOCOS-DG2	**100 %**	5.06E+01	**5.06E+01**	5.54E-13	**100 %**	5.66E+01	6.10E+01	2.16E+01	**100 %**	6.02E+01	**6.02E+01**	**6.53E-11**
DELOCOS-DVIIC	**100 %**	5.06E+01	**5.06E+01**	**2.79E-13**	**100 %**	5.66E+01	**5.66E+01**	**9.37E-12**	**100 %**	6.02E+01	**6.02E+01**	1.74E-10
MDDE	**100 %**	5.06E+01	1.49E+02	8.49E+01	**100 %**	1.65E+02	2.61E+02	3.20E+01	**100 %**	2.24E+02	2.66E+02	2.61E+01
CCSaNSDE-DG2	92 %	**5.05E+01**	7.73E+09	3.32E+10	**100 %**	**5.63E+01**	3.54E+07	1.77E+08	**100 %**	**5.94E+01**	2.66E+02	9.87E+02
	F16				F17				F18			
DELOCOS-DG2	**100 %**	**5.17E+02**	**5.72E+02**	3.60E+01	**100 %**	**5.99E+02**	6.38E+02	4.66E+01	**100 %**	**6.01E+02**	6.07E+02	**2.82E+01**
DELOCOS-DVIIC	**100 %**	5.72E+02	6.45E+02	6.05E+01	**100 %**	**5.99E+02**	**6.23E+02**	**4.08E+01**	**100 %**	**6.01E+02**	**6.49E+02**	5.59E+01
MDDE	**100 %**	6.90E+02	7.19E+02	**2.61E+01**	**100 %**	6.90E+02	7.23E+02	2.16E+01	**100 %**	6.84E+02	6.86E+02	6.94E+00
CCSaNSDE-DG2	96 %	7.33E+02	2.48E+10	3.03E+10	3 %	1.09E+03	9.51E+09	1.64E+10	20 %	8.87E+03	2.04E+10	3.08E+10

Tabla. 7.10.: Resultados estadísticos para la dimensión 1000. FR porcentaje de factibilidad. En negrita se presentan los mejores resultados para los algoritmos cuando alcanzan el 100 % de factibilidad.

	F01				F02				F03			
	FR	**Mejor**	**Mediana**	**std**	**FR**	**Mejor**	**Mediana**	**std**	**FR**	**Mejor**	**Mediana**	**std**
DELoCoS-DG2	**100 %**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**100 %**	**9.00E+00**	**9.00E+00**	1.40E-12	**100 %**	**1.50E+01**	**1.50E+01**	**2.20E-12**
DELoCoS-DVIIC	**100 %**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**100 %**	**9.00E+00**	**9.00E+00**	2.15E-12	**100 %**	**1.50E+01**	**1.50E+01**	1.06E-11
MDE	**100 %**	1.51E-117	1.05E-116	3.30E-116	48 %	9.00E+00	9.00E+00	0.00E+00	28 %	1.50E+01	1.50E+01	0.00E+00
CCSaNSDE-DG2	**100 %**	1.02E+04	4.43E+04	1.78E+04	**100 %**	1.37E+04	3.57E+04	1.33E+04	**100 %**	2.25E+04	4.77E+04	1.61E+04
	F04				F05				F06			
DELoCoS-DG2	**100 %**	**3.65E-06**	4.61E-01	1.33E+01	**100 %**	**9.00E+00**	9.22E+00	6.42E+00	96 %	1.40E+01	1.43E+01	4.55E+00
DELoCoS-DVIIC	**100 %**	1.03E-05	**4.57E-04**	**6.48E-04**	**100 %**	9.01E+00	**9.05E+00**	**1.75E-01**	**100 %**	**1.41E+01**	**1.56E+01**	**4.49E+00**
MDE	**100 %**	5.23E+01	1.33E+02	5.99E+01	36 %	1.30E+02	1.51E+02	5.03E+01	20 %	1.40E+02	2.68E+02	1.21E+02
CCSaNSDE-DG2	**100 %**	1.21E+04	4.53E+04	1.62E+04	**100 %**	3.37E+04	6.02E+04	2.01E+04	**100 %**	3.33E+04	6.71E+04	1.75E+04
	F07				F08				F09			
DELoCoS-DG2	**100 %**	2.07E+02	3.19E+02	8.80E+01	**100 %**	**5.44E-24**	1.18E+02	1.72E+02	**100 %**	**5.00E+00**	**5.00E+00**	**2.85E+01**
DELoCoS-DVIIC	**100 %**	**1.06E-02**	**1.54E+01**	**8.26E+01**	**100 %**	2.62E-05	**7.85E-01**	**6.33E+01**	**100 %**	5.01E+00	2.28E+02	1.75E+02
MDE	**100 %**	4.31E+02	6.27E+02	1.25E+02	36 %	1.37E+02	4.44E+02	1.82E+02	32 %	5.19E+02	6.38E+02	1.76E+02
CCSaNSDE-DG2	**100 %**	2.53E+05	3.36E+05	4.41E+04	**100 %**	1.74E+05	2.67E+05	4.74E+04	**100 %**	1.93E+05	2.78E+05	3.80E+04
	F10				F11				F12			
DELoCoS-DG2	**100 %**	**1.41E+02**	1.99E+02	4.02E+01	**100 %**	**1.47E+02**	1.70E+02	2.20E+01	**100 %**	**1.54E+02**	**1.77E+02**	**1.88E+01**
DELoCoS-DVIIC	**100 %**	**1.41E+02**	**1.41E+02**	3.21E+01	**100 %**	**1.47E+02**	**1.47E+02**	2.19E+01	**100 %**	**1.54E+02**	**1.77E+02**	2.23E+01
MDE	**100 %**	**1.41E+02**	1.50E+02	**2.35E+01**	52 %	1.47E+02	1.53E+02	1.20E+01	28 %	1.54E+02	1.65E+02	1.84E+01
CCSaNSDE-DG2	**100 %**	4.59E+02	1.76E+04	8.79E+03	**100 %**	3.65E+03	2.00E+04	1.08E+04	**100 %**	6.55E+02	1.72E+04	1.61E+07
	F13				F14				F15			
DELoCoS-DG2	**100 %**	1.84E+02	2.75E+02	3.17E+01	**100 %**	**2.20E+02**	2.82E+02	2.54E+01	**100 %**	**2.53E+02**	3.17E+02	3.80E+01
DELoCoS-DVIIC	**100 %**	1.84E+02	2.75E+02	**2.65E+01**	**100 %**	**2.20E+02**	2.81E+02	**2.15E+01**	**100 %**	**2.53E+02**	3.14E+02	**2.33E+01**
MDE	**100 %**	2.83E+02	3.23E+02	2.70E+01	32 %	2.82E+02	2.85E+02	1.36E+01	36 %	3.21E+02	3.42E+02	2.33E+01
CCSaNSDE-DG2	**100 %**	1.35E+03	1.66E+04	9.82E+08	**100 %**	9.66E+02	1.55E+04	2.44E+07	**100 %**	5.12E+02	1.15E+04	1.26E+05
	F16				F17				F18			
DELoCoS-DG2	**100 %**	**1.14E+03**	**1.14E+03**	**7.56E+00**	**100 %**	**1.24E+03**	1.43E+03	7.60E+01	**100 %**	1.32E+03	**1.46E+03**	7.55E+01
DELoCoS-DVIIC	**100 %**	**1.14E+03**	1.39E+03	1.08E+02	**100 %**	1.28E+03	1.45E+03	1.08E+02	**100 %**	**1.28E+03**	1.55E+03	1.30E+02
MDE	**100 %**	1.38E+03	1.43E+03	3.54E+01	56 %	1.37E+03	1.41E+03	3.58E+01	24 %	1.43E+03	1.45E+03	4.04E+01
CCSaNSDE-DG2	**100 %**	8.88E+10	1.72E+11	3.63E+10	**100 %**	8.96E+10	1.69E+11	3.12E+10	**100 %**	1.26E+11	1.71E+11	4.54E+10

7.1.3 Conclusiones

Se llevaron a cabo tres experimentos que son los siguientes: (1) evaluar el efecto de la descomposición en la fase local del algoritmo memético que se propone, (2) determinar el método de descomposición más adecuado entre DG2 y DVIIC, y (3) comparar la versión con mejor desempeño contra un algoritmo del *estado del arte* basado en Co-evolución Cooperativa.

Los resultados indican que agregar el proceso de descomposición dentro de la fase de búsqueda local de un algoritmo mémetico simple (utilizando una variante de DE y un método de búsqueda directa bien conocido como buscador local) mejora significativamente el rendimiento del algoritmo en problemas de alta dimensión con espacios restringidos. Además, DELoCoS es capaz de trabajar tanto con el método de descomposición DVIIC como con DG2, pero el uso de DG2 es conveniente si el problema es altamente restringido, es decir, DG2 es conveniente ante problemas con más restricciones.

Ambas versiones de DELoCoS superan a un algoritmo de *estado del arte* que utiliza una variante de DE más compleja como buscador global. De hecho, la version más simple del algoritmo memético (sin descomposición) fue competitiva contra CCSAnSDE en el mismo conjunto de funciones de prueba.

Como conclusión general, en este capítulo, se comprobó que agregar la información de descomposición a la fase local de un MA es capaz de mejorar los resultados, aun cuando el buscador global y local son algoritmos muy simples.

Como futuro trabajo, se plantea respecto al MA, el uso de diferentes algoritmos de búsqueda local y probar otros algoritmos de EA, con la finalidad de probar sus capacidades cuando el problema es dividido en sub-problemas. Otro enfoque de investigación es mejorar DVIIC y DG2 para problemas de búsqueda con espacios restringidos.

7.2 Buscadores locales utilizando DG2 y DVIIC

El algoritmo propuesto en este trabajo es un algoritmo memético, tal como se vio en la Sección 5.2.1, la característica principal es el uso de algún método de optimización clásico o heurístico como buscador local para mejorar las soluciones durante el proceso de un optimizador global. La propuesta DELoCoS inicialmente fue probada utilizando el algoritmo de Hooke-Jeeves como buscador local guiado por la descomposición del problema. En esta sección se pretende observar de manera empírica el efecto de incluir la descomposición en tres algoritmos clásicos. Estos son Hooke-Jeeves, Nelder-Mead y Random Walk, los cuales fueron descritos en el Capítulo 2.

7.2.1 Diseño experimental

Los algoritmos Hooke-Jeeves, Nelder-Mead y Random Walk, fueron probados en el conjunto de funciones de prueba de alta-dimensión con restricciones [37]. Cada uno de los algoritmos fue probado con los tres métodos de descomposición utilizados durante el desarrollo del esté trabajo (VIIC, DVIIC y DG2), y la version de cada algoritmo sin descomposición.

Con la finalidad de realizar una comparación justa y poder determinar el desempeño de los algoritmos se realizó la experimentación bajo el siguiente diseño:

- Se realizaron 25 ejecuciones independiente por cada función de prueba en cada una de la dimensiones (100, 500 y 1000).

- Se utilizó el mismo esquema de descomposición obtenido por los métodos de descomposición para cada versión de los algoritmos. Es decir, el resultado obtenido de VIIC, DVIIC y DG2, fue utilizado para las versiones correspondientes de Hooke-Jeeves, Nelder-Mead y Random Walk.

- Del mismo modo, para cada ejecución del problema se generó una solución aleatoria como punto inicial para los algoritmos, los tres algo-

ritmos utilizaron esta misma solución. Esta decisión se basa en evitar el sesgo que podría ocasionar obtener una buena solución inicial.

- Para comparar el desempeño de los algoritmos se utilizó la prueba de Wilcoxon con una confianza del 95 %.

Parámetros

A continuación se describen el ajuste de parámetros para cada uno de los algoritmos. Los parámetros no fueron calibrados y fueron utilizados los propuestos en la literatura de cada uno de los algoritmos.

- DG2: es libre de parámetros, requiere $(D(D+1)/2)+1$ evaluaciones para obtener la descomposición el problema.

- VIIC: número de subgrupos $m = 2$, el máximo número de evaluaciones fue ajustada al requerido por DG2.

- DVIIC: tamaño de población $popSize = 6$, probabilidad de cruza $pc = 0{,}9$, probabilidad de mutación $pm = 0{,}1$, máximo número de evaluaciones igual a las requeridas por DG2.

- Hooke-Jeeves: tamaño de paso $\Delta = 0{,}5$, el criterio de parada es $\epsilon = 1{,}0$e-4, y el factor de reducción del tamaño de paso es $\alpha = 2$.

- Nelder-Mead: distancia para generar el simplex $\alpha = 1$, contracción $beta = 0{,}5$, expansión $\gamma = 1{,}5$, criterio de parada es $\epsilon = 1{,}0$e-4.

- Random Walk: número de pasos aleatorios $\lambda = 100$, criterio de parada es $\epsilon = 1{,}0$e-4.

7.2.2 Resultados

La combinación de cada buscador local con un método descomposición nos da el total de nueve algoritmos probados (HJ DVIIC, HJ VIIC, HJ DG2, NM DVIIC, NM VIIC, NM DG2; RW DVIIC, RW VIIC y RW DG2), más las versiones sin descomposición que son referidos en las tablas y resultados con “NoDes”.

Los resultados numéricos se muestran en al Apéndice B para cada una de las dimensiones, en las tablas se describen los valores correspondientes al valor de la mediana y la desviación estandar obtenido de las 25 ejecuciones de cada función de prueba. Para el análisis de los resultados se utilizó el conteo de los resultados obtenidos por la prueba estadística de Wilcoxon. Cabe mencionar que para este análisis se tomó la Sumatoria de Violación de restricciones como indicador, ya que ningún algoritmo logro obtener resultados factibles, un comportamiento esperado por la naturaleza de las funciones de prueba.

Resultados Dimensión 100

Primero se analizó el desempeño obtenido por cada uno de los algoritmos de búsqueda local respecto al método de descomposición utilizado.

En la Tabla 7.11, se muestran los resultados de la prueba de Wilcoxon, para el algoritmo de Hooke-Jeeves y todas sus variantes. Tomando como base HJ DVIIC, se puede observar que en dimensión 100 el rendimiento entre los diferentes esquemas de descomposición es similar. En la mayoría de las funciones se puede considerar que hay un efecto positivo al aplicar DVIIC respecto a no descomponer el problema, pues en 9 de las 18 funciones HJ DVIIC obtuvo diferencias significativas a favor.

Tabla. 7.11.: Resultados de la prueba de Wilcoxon para la dimensión 100, de Hooke Jeeves, usando como base el algoritmo HJ DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

HJ DVIIC	HJ DG2	HJ VIIC	HJ NoDes
Favor	5	1	9
SdS	**7**	**14**	**8**
En contra	6	3	1

Los resultados para el algoritmo de Nelder-Mead y sus variantes se muestran en la Tabla 7.12. De la misma forma que con Hooke-Jeeves, entre los métodos de descomposición no hay diferencias significativas, incluso sin aplicar descomposición los resultados obtenidos por Nelder-Mead son similares en dimensión 100.

Tomando como base el algoritmo de Random Walk con DVIIC, en la Tabla 7.13 se observa que esta versión del algoritmo se ve superada por su similar que

Tabla. 7.12.: Resultados de la prueba de Wilcoxon para la dimensión 100, de Nelder Mead, usando como base el algoritmo NM DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

NM DVIIC	NM DG2	NM VIIC	NM NoDes
Favor	5	1	3
Sin diferencia	**9**	**13**	**12**
En contra	4	4	3

utiliza DG2 en 11 de las 18 funciones de prueba. Sin embargo, contra las versiones de VIIC y NoDes en la mayoría de las funciones no existe diferencia significativa.

Tabla. 7.13.: Resultados de la prueba de Wilcoxon para la dimensión 100, de Random Walk, usando como base el algoritmo RW DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

RW DVIIC	RW DG2	RWVIIC	RW NoDes
Favor	2	2	5
Sin diferencia	5	**16**	**11**
En contra	**11**	0	2

Para comparar el desempeño de los algoritmos HJ, NM y RW, de los resultados antes descritos se tomaron aquellos que muestran “un mejor desempeño”, estos son HJ DVIIC, NM DVIIC y RW DG2. En la Tabla 7.14 se muestra el conteo del resultado de la prueba estadística de Wilcoxon. Con la finalidad de identificar que característica de los problemas está influenciando los resultados, en la Tabla 7.14 se encuentran agrupados los problemas por función objetivo, la cual determina la separabilidad general del problema.

Como algoritmo base de comparación se utilizó HJ DVIIC. Comparando éste contra NM DVIIC, se puede observar que sólo para las funciones F1 a F3, HJ DVIIC se ve superado por NM DVIIC, estas funciones son completamente separables. Sin embargo, en los demás problemas el desempeño de ambos algoritmos es similar. Por otra parte, al comparar HJ DVIIC contra RW DG2 se observa claramente cómo la versión de Random Walk supera a HJ en 16 problemas, y sólo en las funciones F16 a F18, HJ DVIIC muestra ser competitivo aunque sólo en dos de las tres funciones.

Tabla. 7.14.: Resultados de la prueba de Wilcoxon para la dimensión 100 agrupados por función objetivo, de las tres mejores versiones de los algoritmos, usando como base el algoritmo HJ DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

	HJ DVIIC vs NM DVIIC			HJ DVIIC vs RW DG2		
	Favor	**SdS**	**En contra**	**Favor**	**SdS**	**En contra**
F1-F3	0	1	**2**	0	0	**3**
F4-F6	0	**2**	1	0	0	**3**
F7-F9	0	**2**	1	0	0	**3**
F10-F12	0	**2**	1	0	0	**3**
F13-F15	0	**2**	1	0	0	**3**
F16-F18	0	**2**	1	1	1	1
Total	0	**11**	7	1	1	**16**

Resultados Dimensión 500

En dimensión 500 respecto a Hooke-Jeeves, los resultados de la prueba de Wilcoxon (Tabla 7.15) muestran que la versión de HJ DG2 se desempeña mejor que las otras versiones. HJ DG2 tiene un mejor rendimiento en 7/18 funciones de prueba respecto a HJ DVIIC, contra su versión sin descomposición, HJ DG2 es mejor en 9/18 funciones, mientras que con HJ VIIC no hubo diferencia significativa en 8 funciones de prueba.

Tabla. 7.15.: Resultados de la prueba de Wilcoxon para la dimensión 500, de Hooke Jeeves, usando como base el algoritmo HJ DG2, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

HJ DG2	**HJ DVIIC**	**HJ VIIC**	**HJ NoDes**
Favor	**7**	4	**9**
Sin diferencia	5	**8**	5
En contra	6	6	4

Respecto a los resultados de Nelder-Mead en dimensión 500, es evidente que la mejor versión es aquella que utiliza DG2 como esquema de descomposición (Tabla 7.16), la cual muestra mejor desempeño en 10 funciones respecto a cualquiera de las otras versiones. Únicamente la versión NM DVIIC muestra ser competitiva, pues en 7 funciones es mejor que NM DG2.

En la Tabla 7.17 se muestran los resultados de la prueba de Wilcoxon para los algoritmos de Random Walk. Los resultados indican que la versión RW DG2 es mejor en 11, 10 y 14 funciones con respecto a RW DVIIC, RW VIIC y RW

Tabla. 7.16.: Resultados de la prueba de Wilcoxon para la dimensión 500, de Nelder Mead, usando como base el algoritmo NM DG2, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

NM DG2	NM DVIIC	NM VIIC	NM NoDes
Favor	**10**	**10**	**10**
Sin diferencia	1	5	6
En contra	7	3	2

NoDes, respectivamente. Sólo las versiones basadas en VIIC logran competir en 6 funciones de prueba.

Tabla. 7.17.: Resultados de la prueba de Wilcoxon para la dimensión 500, de Random Walk, usando como base el algoritmo RW DG2, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

RW DG2	RW DVIIC	RW VIIC	RW NoDes
Favor	**11**	**10**	**14**
Sin diferencia	1	2	4
En contra	6	6	0

De acuerdo a los resultados de cada uno de los algoritmos, es evidente que aquellos que utilizan DG2 para descomponer el problema son los que tienen mejor desempeño en dimensión 500. La Tabla 7.18 muestra el conteo del resultado de la prueba de Wilcoxon tomando como base el algoritmo de HJ DG2 y comparándolo contra NM DG2 y RW DG2.

El algoritmo NM DG2 es mejor que HJ DG2 en 12/18 funciones de prueba. Ya que ambos utilizan el mismo esquema de descomposición, podemos argumentar que los resultados están directamente relacionados al buscador local. HJ DG2 tiene un desempeño similar a NM DG2 en funciones parcialmente separables como son las funciones F4 a F6, y supera a NM DG2 en las funciones F16 a F18 que son funciones no separables con variables empalmadas y con superposición. Respecto a RW DG2, este fue superado por HJ DG2 en 11 funciones, en su mayoría en aquellas que son parcialmente separables, y al igual que contra NM DG2, en las funciones F16 a F18.

Tabla. 7.18.: Resultados de la prueba de Wilcoxon para la dimensión 500 agrupados por función objetivo, de las tres mejores versiones de los algoritmos, usando como base el algoritmo HJ DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

	HJ DG2 vs NM DG2			HJ DG2 vs RW DG2		
	Favor	**SdS**	**En contra**	**Favor**	**SdS**	**En contra**
F1-F3	1	0	**2**	**2**	0	1
F4-F6	1	1	1	**2**	0	1
F7-F9	0	0	**3**	**2**	0	1
F10-F12	0	0	**3**	1	1	1
F13-F15	0	0	**3**	1	1	1
F16-F18	**3**	0	0	**3**	0	0
Total	5	1	**12**	**11**	2	5

Resultados Dimensión 1000

Ahora se compara lo obtenido en dimensión 1000. En la Tabla 7.19 se muestran los resultados de la prueba de Wilcoxon para los algoritmos de Hooke-Jeeves tomando como base a HJ DVIIC, el cual supera en desempeño a HJ DG2 y HJ NoDes en 9 funciones de prueba, pero se ve superado por HJ VIIC en 8 funciones. Tomando el total de funciones se puede observar que en general el número de diferencias significativas a favor y en contra no permiten catalogar alguna de las versiones de HJ como la mejor. Sin embargo, HJ DVIIC se tomara como el referente en los siguientes resultados.

Tabla. 7.19.: Resultados de la prueba de Wilcoxon para la dimensión 1000, de Hooke Jeeves, usando como base el algoritmo HJ DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

HJ DVIIC	HJ DG2	HJ VIIC	HJ NoDes
Favor	**9**	5	**9**
Sin diferencia	1	5	2
En contra	8	**8**	7

Los resultados para los algoritmos basados en Nelder-Mead se muestran en la Tabla 7.20. Comparando NM DVIIC contra los otros algoritmos, se puede observar que supera en 12 y 11 funciones a NM VIIC y NM NoDes respectivamente. Sin embargo, NM DVIIC se ve superado por NM DG2 en 10 funciones de prueba.

Tabla. 7.20.: Resultados de la prueba de Wilcoxon para la dimensión 1000, de Nelder Mead, usando como base el algoritmo NM DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

NM DVIIC	NM DG2	NM VIIC	NM NoDes
Favor	7	**12**	**11**
Sin diferencia	1	6	7
En contra	**10**	0	0

Respecto a Random Walk en dimensión 1000, los resultados se muestran en la Tabla 7.21 tomando como base el algoritmo RW DVIIC. Este algoritmo supera a RW NoDes en 16 funciones, sin embargo, su desempeño es superado en 10 funciones respecto a RW DG2 y en 8 funciones por RW VIIC. A diferencia de Hooke-Jeeves, con RW se pude determinar que la versión RW DG2 es la que mejor comportamiento muestra considerando el conjunto de funciones de prueba total.

Tabla. 7.21.: Resultados de la prueba de Wilcoxon para la dimensión 1000, de Random Walk, usando como base el algoritmo RW DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

RW DVIIC	RW DG2	RW VIIC	RW NoDes
Favor	8	4	**16**
Sin diferencia	0	6	2
En contra	**10**	**8**	0

Tal como se describió anteriormente, las versiones de cada uno de los algoritmos de búsqueda local que mejores resultados arrojaron son: HJ DVIIC, NM DG2 y RW DG2. En la Tabla 7.22 se muestran los resultados comparando estos tres algoritmos tomando como base de la prueba estadística al algoritmo HJ DVIIC. En general, HJ DVIIC supera a NM DG2 en 12 funciones prueba. A diferencia de los resultados en dimensión 100 y 500, en dimensión 1000, HJ DVIIC muestra tener mejor rendimiento en las funciones totalmente separables (F1-F3), en funciones parcialmente no separables que involucren o no empalmado o superposición de variables (F4-F6, F10-F12 y F16-F18). El desempeño de HJ DVIIC es mejor que RW DG2 en 17 funciones, siendo sólo en una función totalmente separable donde ambos algoritmos muestran el mismo comportamiento.

En la siguiente sección se presentan las conclusiones a las que se llegaron de acuerdo a estos resultados obtenidos.

Tabla. 7.22.: Resultados de la prueba de Wilcoxon para la dimensión 1000 agrupados por función objetivo, de las tres mejores versiones de los algoritmos, usando como base el algoritmo HJ DVIIC, cantidad de funciones con diferencia estadística a Favor, Sin diferencia Significativa (SdS), y En Contra

	HJ DVIIC vs NM DG2			HJ DVIIC vs RW DG2		
	Favor	**SdS**	**En contra**	**Favor**	**SdS**	**En contra**
F1-F3	**3**	0	0	2	1	0
F4-F6	**2**	1	0	**3**	0	0
F7-F9	1	0	2	**3**	0	0
F10-F12	**2**	1	0	**3**	0	0
F13-F15	1	2	0	**3**	0	0
F16-F18	**3**	0	0	**3**	0	0
Total	**12**	4	2	**17**	1	0

7.2.3 Conclusiones

Con la finalidad de comparar el desempeño de diferentes buscadores locales añadiendo la descomposición del problema, se probaron tres algoritmos directos (libre de derivadas): Hooke-Jeeves, Nelder-Mead y Random Walk. Por otro lado, los métodos de descomposición, fueron los que hemos estudiado a lo largo de este trabajo (VIIC, DVIIC, DG2). Se realizó un experimento empírico y se utilizó la prueba estadística de Wilcoxon para comparar el comportamiento de los algoritmos función a función. Aunque los algoritmos no lograron obtener resultados factibles, la sumatoria de violación de restricciones es considerada como función de aptitud, ya que minimizar este valor indica un acercamiento a la región factible.

Los resultados en dimensión 100 muestran que la inclusión de la descomposición del problema en los algoritmos de búsqueda no muestra una mejora significativa en los resultados finales. Sin embargo, comparando los algoritmos que utilizan descomposición (Tabla 7.14) se concluye que en dimensión 100 la combinación de Random Walk con DG2 tiene el mejor rendimiento en la mayoría de las funciones de prueba.

Respecto a la dimensión 500, utilizar el método DG2 mostró mejorar el rendimiento general de los algoritmos de búsqueda. Se observó que los algoritmos con mejor rendimiento en esta dimensión fueron Hooke-Jeeves y Nelder-Mead dejando atrás a Random Walk, sobre todo en funciones que son parcialmente separables.

Los resultados en dimensión 1000, mostraron claramente que el algoritmo HJ DVIIC obtiene mejor rendimiento respecto a NM DG2 y RW DG2. Estos resultados llevan a concluir que conforme la dimensión del problema crece, el algoritmo de Hooke-Jeeves lleva a obtener mejores resultados. Del mismo modo se puede observar que mientras en dimensión 100 la descomposición no muestra influir en los resultados finales, cuando la dimensión se incrementa a 500 el método DG2 predomina al mejorar el resultado en los tres algoritmos. Sin embargo, en dimensión 1000, DG2 logra mejorar el rendimiento de Nelder-Mead y Random Walk, pero DVIIC en conjunto con Hooke-Jeeves obtienen mejores resultados sobre todo en las funciones que tienen variables que deben aparecer en más de un grupo.

En general, se comprueba que incluir la descomposición del problema en algoritmos de búsqueda local mejora el rendimiento para acercar las soluciones a la zona factible, y este efecto se enfatiza al incrementar la dimensión de los problemas. Respecto a DVIIC y DG2, que son los métodos de descomposición que predominan en los resultados, se puede concluir que DVIIC, al ser un método estocástico que utiliza un algoritmo genético para la búsqueda de la agrupación de las variables, es capaz de encontrar esquemas de descomposición viables para resolver los problemas. En contraste, para DG2, que es un método determinista, es decir, para un problema siempre entrega el mismo esquema, se comprueba la dificultad de encontrar una descomposición viable en problemas altamente no separables.

A partir de estas conclusiones, como trabajo futuro se propone estudiar el efecto de diseñar algoritmos basados en el ensamble, tanto de buscadores locales como de métodos de descomposición. Estos ensambles pueden incluirse en la propuesta principal de este trabajo que es el esquema de Búsqueda Local Cooperativa (LoCoS).

8 Conclusiones y Trabajo futuro

8.1 Conclusiones

Actualmente, los problemas de optimización a resolver cada vez incrementan el número de variables que los describen, convirtiéndose en problemas de alta dimensión. Los algoritmos evolutivos han mostrado ser capaces de resolver problemas de optimización, con o sin restricciones, de manera satisfactoria. Sin embargo, su rendimiento se ve mermado cuando la dimensión del problema crece, cayendo en la "maldición de la dimensionalidad", pues el incremento de variables de diseño conlleva a problemas como el crecimiento exponencial del espacio de búsqueda, que el paisaje de este espacio cambie conforme la dimensión crece y que el costo de evaluar una solución sea elevado.

Para dar frente a estos problemas desde, la perspectiva del cómputo evolutivo, diversos algoritmos se han desarrollado tales como algoritmos evolutivos con operadores especiales, algoritmos meméticos y el esquema de la Co-evolución Cooperativa. Este último tipo de algoritmos están basados en la premisa de "divide y vencerás"; requieren un proceso de descomposición del problema, para después dar paso a la optimización de cada subproblema generado. Todos estos algoritmos han sido ampliamente evaluados en problemas de optimización sin restricciones, dejando un área de oportunidad en problemas con restricciones.

Por lo anterior, en este trabajo de tesis se planteó un esquema basado en la co-evolución, pero enfocado en algoritmos meméticos, en el cual la fase de búsqueda local es guiada por la descomposición del problema en subproblemas. Este algoritmo fue diseñado para resolver problemas de alta dimensión con restricciones. Basado en este esquema se propusieron dos algoritmos llamados DE-LoDeS y DE-LoCoS (Local Decomposition Search "LoDeS" y Local Cooperative Search "LoCoS") utilizan la descomposición del problema de manera distinta, por un lado LoDeS, guia la búsqueda local por el orden de las variables que resulta de la descomposición, mientras que

LoCoS se basa en la cooperación de optimizar cada subgrupo de variables de manera separada.

DE-LoDeS y DE-LoCoS, utilizan Evolución Diferencial como buscador global y el algoritmo de Hooke-Jeeves como método de búsqueda local. Durante la primera fase de experimentación ambos algoritmos trabajaron con el método de descomposición VIIC, que es el primer método enfocado en problemas con restricciones. Ambos algoritmos fueron probados en un conjunto de funciones de prueba diseñado para las dimensiones 100, 500 y 1000.

VIIC, considerando la literatura especializada, es el primer método de descomposición para problemas con restricciones. Sin embargo, se detectaron áreas de oportunidad para la mejora en el rendimiento de este algoritmo. Se presentaron tres modificaciones al algoritmo de VIIC que fueron: (1) utilizar la información de la sumatoria de violación de restricciones en la evaluación de la descomposición, (2) incluir estrategias de vecindario para generar nuevos esquemas de descomposición basados en el actual esquema (VIIC realiza la búsqueda de manera aleatoria), (3) incluir el uso de la heurística de recocido simulado para guiar la búsqueda, y (4) abordar la descomposición del problema como un problema de optimización combinatoria, por lo que se incluyó el uso de un algoritmo genético para mejorar el desempeño. Estas mejoras llevaron al desarrollo del algoritmo que llamamos DVIIC (Dynamic VIIC).

Por último, se realizó nueva experimentación utilizando DVIIC y comparándolo con un método altamente competitivo de la literatura (DG2) y contra la contraparte del algoritmo memético DE-LoCoS sin uso de descomposición. Finalmente, los resultados fueron comparados contra tres algoritmos del estado del arte: DEVIIC, DERG y CCSaNSDE-DG2.

Los resultados de la experimentación llevaron a las siguientes conclusiones:

- Agregar el proceso de descomposición dentro de la fase de búsqueda local de un algoritmo mémetico simple (utilizando una variante de DE y un método de búsqueda directa bien conocido como buscador local) mejora significativamente el desempeño del algoritmo en problemas de alta dimensión con espacios restringidos.

- La factibilidad de DE-LoCoS y DE-LoDeS no se afectó con el incremento de la dimensión. Sin embargo, la calidad de sus soluciones obtenidas por DE-LoDes sí se vio afectada por el aumento de dimensiones.

- Los resultados remarcan que la cooperación de los sub-grupos (LoCoS) de variables mejora el rendimiento respecto a sólo utilizar el orden de las variables (LoDeS).

- El método de Hooke-Jeeves es bueno en problemas separables, lo que permitió que su combinación con el esquema LoCoS llevara a una exploración adecuada de la vecindad de cada uno de los sub-grupos. Las gráficas de convergencia sustentan esta observación, las cuales muestran convergencia rápida pero sin llegar a ser prematura o mostrar algún efecto de estancamiento.

- De acuerdo al método empírico para medir la complejidad, DE-LoCoS y DE-LoDeS tienen un costo computacional más elevado respecto a los algoritmos DEVIIC y DERG. Ello es un efecto esperado, pues se conoce que los algoritmos meméticos pueden ser más costosos que los algoritmos evolutivos tradicionales.

- DE-LoCoS es capaz de trabajar tanto con el método de descomposición DVIIC como con DG2, pero el uso de DG2 es conveniente si el problema es altamente restringido, es decir, DG2 parece ser más robusto ante problemas con mayor número de restricciones. Ambas versiones de DE-LoCoS superan a un algoritmo de *estado del arte* que utiliza una variante de DE más compleja como buscador global. De hecho, la versión más simple del algoritmo memético (sin descomposición) fue competitiva contra CCSAnSDE en el mismo conjunto de funciones de prueba.

- Respecto a las mejoras planteadas para VIIC se encontró lo siguiente:

    - A pesar de que todos los algoritmos se ven afectados de manera negativa en su desempeño cuando la dimensión del problema crece: (1) el evaluar sólo un arreglo de variables no afecta e incluso mejora los resultados finales para encontrar un arreglo de variables adecuado, (2) las dos estrategias de vecindario para generar arreglos de variables a partir del ya existente mejora el desempeño de VIIC, (3) es interesante que al combinar estas estrategias con la

versión voraz de VIIC se obtienen mejores resultados que cuando se aplica una heurística como recocido simulado.

- DVIIC mostró ser más rápido que las versiones que no utilizan un algoritmo genético. Además DVIIC no requiere determinar el número de subgrupos de variables en los que se quiere descomponer el problema.

Posterior al análisis del comportamiento del esquema de Local Cooperative Search (LoCoS), se decidió analizar el comportamiento de aplicar los métodos de descomposición en diferentes buscadores locales fuera del contexto del algoritmo memético. Los métodos que se probaron son algoritmos directos (libre de derivadas): Hooke-Jeeves, Nelder-Mead y Random Walk, y los métodos de descomposición aplicados, son los que se han estudiado a lo largo de este trabajo (VIIC, DVIIC, DG2). Las conclusiones a las que se llegaron en esta experimentación son:

- Los algoritmos no lograron obtener resultados factibles, sin embargo, la sumatoria de violación de restricciones fue considerada como función de aptitud, ya que minimizar este valor indica el acercamiento a la región factible.

- En dimensión 100 la inclusión de la descomposición del problema a los algoritmos de búsqueda no muestra una mejora significativa en los resultados finales.

- Respecto a los algoritmos usando descomposición, en dimensión 100 la combinación de Random Walk con DG2 tiene el mejor rendimiento en la mayoría de las funciones de prueba.

- DG2 mejoró el rendimiento general de los algoritmos de búsqueda en dimensión 500. Los algoritmos con mejor rendimiento en ésta dimensión fueron Hooke-Jeeves y Nelder-Mead dejando atrás a Random Walk, sobre todo en funciones que son parcialmente separables.

- En dimensión 1000, se observó que el algoritmo HJ DVIIC obtiene el mejor rendimiento a comparación de NM DG2 y RW DG2. Esto es, conforme la dimensión del problema crece, el algoritmo de Hooke-Jeeves lleva a obtener mejores resultados.

- DVIIC en conjunto con Hooke-Jeeves obtuvieron los mejores resultados en las funciones que tienen variables que deben aparecer en más de un grupo (superposición y empalme de variables).

- En general, se comprobó que incluir la descomposición del problema en algoritmos de búsqueda local mejora su desempeño para acercar las soluciones a la zona factible, y este efecto se enfatiza al incrementar la dimensión de los problemas.

Se puede concluir de manera general que el esquema propuesto (LoCoS) satisface la hipótesis planteada en este trabajo de investigación. Se observó de manera empirica mediante los resultados númericos que aplicar la descomposición de problemas fuera del contexto de la co-evolución cooperativa permite a los algoritmos mejorar el rendimiento en problemas de alta dimensión con restricciones, dando pie a una nueva linea de investigación para responder a la pregunta **¿ Cómo diseñar algoritmos de cómputo evolutivo para problemas de alta dimensión con restricciones utilizando la descomposición fuera del contexto de la co-evolución cooperativa?**. A partir de ésta pregunta se crea un área de oportunidad nueva dentro de este campo de alta dimensionalidad. En la siguiente sección referente al trabajo futuro, se describen algunas ideas que encajan con esta conclusión.

8.2 Trabajo Futuro

Una de las desventajas evidentes y propias de los algoritmos meméticos es el incremento en la complejidad del algoritmo, como trabajo futuro se propone estudiar cómo reducir esta complejidad a partir de mejorar el control de la frecuencia de aplicación de la búsqueda local.

Por otro lado, se probaron diferentes buscadores locales y los resultados sugieren que dependiendo de las características del problema la combinación de buscador local y método de descomposición cambian. Por lo tanto, investigar un esquema de ensamble de buscadores locales (multi-meme) para realizar la búsqueda sobre cada uno de los subgrupos de variables con un determinado algoritmo de busqueda local.

Dadas las claras ventajas y desventajas de cada uno de los algoritmos de descomposición estudiados, se propone el estudio de una posible hibridación o ensamble de métodos de descomposción. Por un lado DVIIC es un algoritmo de optimización de separabilidad de variables y por el otro DG2 un algoritmo libre de parametros y determinista, que adolece de requerir una gran cantidad de evaluaciones. Sin embargo DG2 muestra un desempeño bueno, y puede ser que una combinación de las caracteristicas heuristicas (DVIIC) y deterministas (DG2) podria ampliar el rendimiento en problemas restringidos.

LoCoS, es un marco de trabajo que determina una forma de desarrollar los algoritmos, estrictamente basado en un algoritmo memético. Partiendo de esto, en este trabajo se probó la propuesta utilizando Evolución Diferencial como algoritmo de búsqueda global. Sin embargo el estudio del rendimiento de LoCoS utilizando otros algoritmos de la literatura, ya sean de Cómputo Evolutivo (JADE, SADE, SaNSDE, Estrategias Evolutivas) o de inteligencia colectiva (ABC, PSO), es un trabajo que no solo se propone, si no que debe ser realizado. Ya que cada uno de esto algoritmos presentara ventajas en las diferentes caracteristicas de los problemas de alta dimensionalidad con restricciones.

Una parte importante de este trabajo es haber probado que la descomposición del problema puede ayudar a mejorar los resultados fuera del contexto de la Cooperación Coevolutiva, por tanto, un trabajo futuro directamente relacionado con esto es el estudio y aplicación de métodos de descomposción fuera de su contexto habitual, diseñar nuevos operadores de variación para los algoritmos evolutivos guiados por los esquemas de descomposición o incluso por que no, el desarrollo de manejadores de restricciones especiales para alta dimensión.

Bibliografía

[1] A. E. Aguilar-Justo y E. Mezura-Montes. „Towards an improvement of variable interaction identification for large-scale constrained problems“. En: *2016 IEEE Congress on Evolutionary Computation (CEC)*. 2016, págs. 4167-4174 (vid. pág. 81).

[2] A. E. Aguilar-Justo, E. Mezura-Montes, S. M. Elsayed y R. A. Sarker. „Decomposition of large-scale constrained problems using a genetic-based search“. En: *2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. 2016, págs. 1-6 (vid. págs. 90, 96).

[3] M. J. Box. „A New Method of Constrained Optimization and a Comparison With Other Methods“. En: *The Computer Journal* 8.1 (1965), págs. 42-52. eprint: `http://comjnl.oxfordjournals.org/content/8/1/42.full.pdf+html` (vid. págs. 19, 20).

[4] Zijian Cao, Lei Wang, Yuhui Shi y col. „An effective cooperative coevolution framework integrating global and local search for large scale optimization problems“. En: *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2015, págs. 1986-1993 (vid. pág. 2).

[5] Wenxiang Chen, Thomas Weise, Zhenyu Yang y Ke Tang. „Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning“. English. En: *Parallel Problem Solving from Nature, PPSN XI*. Vol. 6239. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, págs. 300-309 (vid. págs. 2, 43).

[6] Kenneth A. De Jong. *Evolutionary Computation: A Unified Approach*. Cambridge, MA, USA: MIT Press, 2016 (vid. pág. 26).

[7] K. DEB. *Optimization for Engineering Design: Algorithms and Examples*. PHI Learning, 2012 (vid. págs. 10, 18, 19).

[8] Kalyanmoy Deb. „An efficient constraint handling method for genetic algorithms“. En: *Computer Methods in Applied Mechanics and Engineering* 186 (2000), págs. 311-338 (vid. págs. 34, 60).

[9] Saber M. Elsayed, Ruhul A. Sarker y Daryl L. Essam. „On an evolutionary approach for constrained optimization problem solving“. En: *Applied Soft Computing* 12.10 (2012), págs. 3208 -3227 (vid. págs. 55, 66, 95).

[10] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006, págs. 1-4 (vid. pág. 28).

[11] L.J. Fogel, A.J. Owens y M.J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, 1966 (vid. pág. 26).

[12] David E. Goldberg y John H. Holland. „Genetic Algorithms and Machine Learning“. En: *Machine Learning* 3.2 (1988), págs. 95-99 (vid. pág. 28).

[13] Robert Hooke y T. A. Jeeves. „Direct Search Solution of Numerical and Statistical Problems“. En: *J. ACM* 8.2 (1961), págs. 212-229 (vid. pág. 12).

[14] Dervis Karaboga y Bahriye Basturk. „A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm“. English. En: *Journal of Global Optimization* 39.3 (2007), págs. 459-471 (vid. pág. 30).

[15] J. Kennedy y R. Eberhart. „Particle swarm optimization“. En: *Neural Networks, 1995. Proceedings., IEEE International Conference on*. Vol. 4. 1995, 1942-1948 vol.4 (vid. pág. 29).

[16] A. LaTorre, S. Muelas y J. M. Peña. „Multiple Offspring Sampling in Large Scale Global Optimization“. En: *2012 IEEE Congress on Evolutionary Computation*. 2012, págs. 1-8 (vid. pág. 3).

[17] Hong qi Li y Li Li. „A Novel Hybrid Particle Swarm Optimization Algorithm Combined with Harmony Search for High Dimensional Optimization Problems“. En: *The 2007 International Conference on Intelligent Pervasive Computing, 2007. IPC.* 2007, págs. 94-97 (vid. pág. 3).

[18] Xiaodong Li y Xin Yao. „Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms“. En: *IEEE Congress on Evolutionary Computation, 2009. CEC '09.* 2009, págs. 1546-1553 (vid. pág. 2).

[19] Xiaodong Li, Ke Tang, Mohammad N. Omidvar, Zhenyu Yang y Kai Qin. *Benchmark Functions for the CEC2013 Special Session and Competition on Large-Scale Global Optimization*. 2013 (vid. págs. 1, 39, 45, 54).

[20] JJ Liang, Thomas Philip Runarsson, Efren Mezura-Montes y col. „Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization“. En: *Journal of Applied Mechanics* 41.8 (2006) (vid. pág. 53).

[21] H. Liu, Y. Wang, X. Liu y S. Guan. „Empirical study of effect of grouping strategies for large scale optimization“. En: *2016 International Joint Conference on Neural Networks (IJCNN)*. Jul. de 2016, págs. 3433-3439 (vid. pág. 3).

[22] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari y Thomas Stützle. „The irace package: Iterated racing for automatic algorithm configuration“. En: *Operations Research Perspectives* 3 (2016), págs. 43 -58 (vid. págs. 63, 95).

[23] Xinran Ma y Jinliang Ding. „A preferred learning based adaptive differential evolution algorithm for large scale optimization“. En: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dic. de 2016, págs. 1-8 (vid. pág. 2).

[24] Rammohan Mallipeddi y Ponnuthurai Nagaratnam Suganthan. „Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization“. En: *Nanyang Technological University, Singapore* (2010) (vid. págs. 53, 63).

[25] D. Molina, M. Lozano y F. Herrera. „MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization“. En: *IEEE Congress on Evolutionary Computation (CEC), 2010*. Jul. de 2010, págs. 1-8 (vid. pág. 3).

[26] D. Mosk-Aoyama y D. Shah. „Fast Distributed Algorithms for Computing Separable Functions“. En: *Information Theory, IEEE Transactions on* 54.7 (jul. de 2008), págs. 2997-3007 (vid. pág. 46).

[27] S. Muelas, A. La Torre y J. Pea. „A Memetic Differential Evolution Algorithm for Continuous Optimization“. En: *Ninth International Conference on Intelligent Systems Design and Applications, 2009. ISDA '09*. Sep. de 2009, págs. 1080-1084 (vid. pág. 3).

[28] John A Nelder y Roger Mead. „A simplex method for function minimization“. En: *The computer journal* 7.4 (1965), págs. 308-313 (vid. pág. 11).

[29] Ferrante Neri y Carlos Cotta. „Memetic algorithms and memetic computing optimization: A literature review“. En: *Swarm and Evolutionary Computation* 2.0 (2012), págs. 1 -14 (vid. pág. 57).

[30] M. N. Omidvar, X. Li, Y. Mei y X. Yao. „Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization“. En: *IEEE Transactions on Evolutionary Computation* 18.3 (jun. de 2014), págs. 378-393 (vid. págs. 2, 43, 45, 83).

[31] M. N. Omidvar, M. Yang, Y. Mei, X. Li y X. Yao. „DG2: A Faster and More Accurate Differential Grouping for Large-Scale Black-Box Optimization“. En: *IEEE Transactions on Evolutionary Computation* PP.99 (2017), págs. 1-1 (vid. págs. 2, 43, 45, 46, 93, 96).

[32] M.N. Omidvar, Xiaodong Li, Zhenyu Yang y Xin Yao. „Cooperative Co-evolution for large scale optimization through more frequent random grouping“. En: *IEEE Congress on Evolutionary Computation (CEC), 2010*. Jul. de 2010, págs. 1-8 (vid. págs. 1, 39, 44, 60, 83).

[33]M.N. Omidvar, Xiaodong Li y Xin Yao. „Cooperative Co-evolution with delta grouping for large scale non-separable function optimization“. En: *IEEE Congress on Evolutionary Computation (CEC), 2010.* Jul. de 2010, págs. 1-8 (vid. pág. 2).

[34]MitchellA. Potter y KennethA. De Jong. „A cooperative coevolutionary approach to function optimization“. English. En: *Parallel Problem Solving from Nature PPSN III*. Vol. 866. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1994, págs. 249-257 (vid. págs. 1, 39, 40).

[35]T. Ray y Xin Yao. „A cooperative coevolutionary algorithm with Correlation based Adaptive Variable Partitioning“. En: *IEEE Congress on Evolutionary Computation, 2009. CEC '09.* Mayo de 2009, págs. 983-989 (vid. págs. 2, 43).

[36]E. Sayed, D. Essam y R. Sarker. „Dependency Identification technique for large scale optimization problems“. En: *Evolutionary Computation (CEC), 2012 IEEE Congress on*. Jun. de 2012, págs. 1-8 (vid. págs. 2, 3, 83).

[37]Eman Sayed, Daryl Essam, Ruhul Sarker y Saber Elsayed. „Decomposition-based evolutionary algorithm for large scale constrained problems“. En: *Information Sciences* 316 (2015). Nature-Inspired Algorithms for Large Scale Global Optimization, págs. 457 -486 (vid. págs. 3, 46, 54, 56, 60, 62, 64, 65, 83, 86, 95, 109, 133, 137).

[38]Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. Ene. de 1975 (vid. pág. 27).

[39]Wen Shi, Wei-Neng Chen, Qiang Yang y Jun Zhang. „A multi-optimizer cooperative coevolution method for large scale optimization“. En: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dic. de 2016, págs. 1-7 (vid. pág. 2).

[40]Rainer Storn y Kenneth Price. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. Vol. 3. ICSI Berkeley, 1995 (vid. pág. 30).

[41]Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang y col. „Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization“. En: *KanGAL report* 2005005 (2005) (vid. págs. 54, 66, 87, 89).

[42]T. Takahama, S. Sakai y N. Iwane. „Solving Nonlinear Constrained Optimization Problems by the epsilon Constrained Differential Evolution“. En: *2006 IEEE International Conference on Systems, Man and Cybernetics*. 2006 (vid. pág. 35).

[43]Ke Tang, Xiaodong Li, P. N. Suganthan, Zhenyu Yang y Thomas Weise. *Benchmark functions for the cec'2010 special session and competition on large-scale global optimization*. Inf. téc. Nature Inspired Computation y Applications Laboratory, 2009 (vid. pág. 45).

[44] Guohua Wu, R Mallipeddi y PN Suganthan. „Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization“. En: *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report* (2016) (vid. pág. 54).

[45] Zhenyu Yang, Ke Tang y Xin Yao. „Large scale evolutionary optimization using cooperative coevolution“. En: *Information Sciences* 178.15 (2008). Nature Inspired Problem-Solving, págs. 2985 -2999 (vid. págs. 2, 44).

[46] Zhenyu Yang, Ke Tang y Xin Yao. „Self-adaptive differential evolution with neighborhood search“. En: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. Jun. de 2008, págs. 1110-1116 (vid. págs. 93, 95).

[47] Kaibo Zhang y Bin Li. „Cooperative Coevolution with global search for large scale global optimization“. En: *2012 IEEE Congress on Evolutionary Computation*. Jun. de 2012, págs. 1-7 (vid. pág. 2).

[48] S.Z. Zhao, J.J. Liang, P.N. Suganthan y M.F. Tasgetiren. „Dynamic multi-swarm particle swarm optimizer with local search for Large Scale Global Optimization“. En: *IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)*. Jun. de 2008, págs. 3845-3852 (vid. pág. 3).

[49] H. Zille, H. Ishibuchi, S. Mostaghim e Y. Nojima. „Mutation operators based on variable grouping for multi-objective large-scale optimization“. En: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2016, págs. 1-8 (vid. pág. 4).

A Funciones de prueba

En esta sección se presentan de manera breve la estructura de las funciones de prueba propuestas por Sayed [37]. El conjunto de funciones esta compuesto por seis funciones objetivo y tres restricciones.

A.1 Funciones Objetivo

Objetivo 1: este objetivo es totalmente separable en cualquier dimensión que se pruebe, siendo esta la función de la esfera.

$$Obj1 = \sum_{i}^{D} x_i^2$$

Objetivo 2: en la función objetivo 2 el 90 % de las variables son separables mientras que el 10 % restante son no separables divididas en grupos de $longitud = 5$ variables.

$$Obj2 = \sum_{c=1}^{C}(\sum_{i}^{n1} x^2 + \sum_{j}^{n2-1} [100 \times (x_j^2 - x_{j+1})^2 + (x_j - 1)^2] + \sum_{l}^{n3} x_l^2)$$

donde $\gamma = 0{,}10$, $longitud = 5$, $C = (c \times D)/longitud$, $i = [(c-1) \times longitud/\gamma] + 1$, $n1 = [(c-1) \times longitud/\gamma] + longitud$, $j = [(c-1) \times longitud/\gamma] + longitud + 1$, $n2 = [(c-1) \times longitud/\gamma] + (1/\gamma)$, $l = [(c-1) \times longitud/\gamma] + (1/\gamma) + 1$, y $n3 = count \times (longitud/\gamma)$

Objetivo 3: tiene 90 % de variables separables usando la función de la esfera, y el 10 % restante son pares de variables no separables que utilizan la función de Rosenbrock.

$$Obj3 = \sum_{c=1}^{C}(\sum_{i}^{n1}[100 \times (x_i^2 - x_{i+(longitud/\gamma)})^2 + (x_i)^2] + \sum_{j}^{n2}(x_j^2 + x_{j+(longitud/\gamma)}^2))$$

donde $\gamma = 0{,}10$, $longitud = 5$, $C = (\gamma \times D)/(2 \times longitud)$, $i = [2 \times (c-1) \times longitud/\gamma] + 1$, $n1 = [2 \times (c-1) \times longitud/\gamma] + longitud$, $j = [2 \times (c-1) \times longitud/\gamma] + longitud + 1$, $n2 = [2 \times (c-1) \times longitud/\gamma] + (longitud/\gamma)$.

Objetivo 4: este objetivo tiene variables no separables y algunas de ellas está superpuestas. Dado está superposición de grupos de variables no separadas, la optimización debe realizarse por medio de un grupo grande de variables.

$$\begin{aligned} Obj4 = \sum_{c=1}^{C}(\sum_{i}^{n-1}([100 \times (x_i^2 - x_{i+1})^2 + (x_i - 1)^2] \\ +[100 \times (x_{i+\delta}^2 - x_{i+\delta+1} - 1)^2 + (x_{i+\delta} - 1)^2])) \\ + \sum_{iovr}^{n1_{ovr}-1}([100 \times (x_{iovr}^2 - x_{iovr+1})^2 + (x_{iovr} - 1)^2] \\ +[100 \times (x_{iovr+\delta}^2 - x_{iovr+\delta+1})^2 + (x_{iovr+\delta} - 1)^2]) \\ + \sum_{j}^{n2} x_j^2 + \sum_{l}^{n3} x_l^2 \end{aligned}$$

donde $\gamma = 0{,}20$, $C = 2$, $i = ((c-1) \times \gamma/4 \times D) + 1$, $n1 = \gamma \times D/4$, $\delta = 0{,}6 \times D$, $i_{ovr} = \gamma \times N/4$, $n1_{ovr} = (\gamma \times D/4) + (\gamma \times D/10)$, $j = (\gamma \times D/2) + 1$, $n2 = 3 \times \gamma \times D$, $l = 0{,}7 \times D + 1$, y $n3 = D$. δ es la posición del segundo grupo de variables no separables, por lo regular la segunda mitad.

Objetivo 5: Este objetivo busca empalmar las variables no separables de forma que no sean adyacentes entre si.

$$
\begin{aligned}
Obj5 = \sum_{i}^{n1}([100 \times (x_{2i-1}^2 + x_{2i+1} - 1)^2 + (x_{2i-1} - 1)^2] \\
+[100 \times (x_{2i-2}^2 + x_{2i+2} - 1)^2 + (x_{2i-1} - 1)^2] \\
+[100 \times (x_{2i+\delta-1}^2 + x_{2i+\delta+1} - 1)^2 + (x_{2i+\delta-1} - 1)^2] \\
+[100 \times (x_{2i+\delta-2}^2 + x_{2i+\delta+2} - 1)^2 + (x_{2i+\delta-1} - 1)^2]) \\
+ \sum_{iovr}^{n1_{ovr}-1} ([100 \times (x_{iovr}^2 - x_{iovr+1})^2 + (x_{iovr} - 1)^2] \\
+[100 \times (x_{iovr+\delta}^2 - x_{iovr+1+\delta})^2 + (x_{iovr+\delta} - 1)^2]) \\
+ \sum_{j}^{n2} x_j^2 + \sum_{l}^{n3} x_l^2
\end{aligned}
$$

donde $n1 = 0{,}1 \times D/2$, $i = 1$, $\delta = 0{,}6 \times D$, $i_{ovr} = 1$, $n1_{ovr} = 0{,}04 \times D - 1$, $j = (\gamma \times D/2) + 1$, $n2 = 0{,}6 \times D$, $l = 0{,}7 \times D + 1$, $n3 = D$, donde δ es utilizado como control de la dispersión de las variables no separables cuando la dimensión del problema cambia.

Objetivo 6: Este objetivo es completamente no separable, y tiene dos grupos no separables empalmados y pares de variables en superposición.

$$
\begin{aligned}
Obj6 = \sum_{i}^{n1}([100 \times (x_{2i-1}^2 + x_{2i+1})^2 + (x_{2i-1} - 1)^2] \\
+[100 \times (x_{2i}^2 + x_{2i+2} - 1)^2 + (x_{2i} - 1)^2]) \\
+ \sum_{j}^{n2} \sum_{ov}^{OV} [100 \times (x_{ov+\tau}^2 - x_{ov+\tau+1})^2 + (x_{ov+\tau} - 1)^2]
\end{aligned} \tag{A.1}
$$

donde $i = 1$, $n1 = D/2$, $n2 = 0{,}1 \times D$, $j = 1$, $K = 2$, $\tau = (j-1) \times (0{,}1 \times D)$, ov es el numero de variables en superposición; τ es el parametro que controla la ocurrencia de las variables en superposición indicado por su índice.

A.2 Restricciones

Restricción 1 (g1): contiene 1 % de las variables separables en grupos de 5 variables.

$$g1_{D=100} = \sum_{j=1}^{5} x_{j+\phi}^2, \phi = 25$$

$$g1_{D=500} = \sum_{i=1}^{5}\sum_{j=1}^{5} x_{j+\phi}^2, \phi = \{25, 125, 225, 325, 425\}$$

$$g1_{D=1000} = \sum_{i=1}^{10}\sum_{j=1}^{5} x_{j+\phi}^2, \phi = 25 + ((n1 - 1) \times 100), n1 = [1 : 10]$$

Restricción 2 (g2): tiene 1, 2 o 3 grupos de variables de acuerdo a la dimensión 100, 500 o 1000.

$$g2_{D=100} = \sum_{j=1}^{3-1}[100(x_{j+\phi}^2 - x_{j+\phi+1})^2 + (x_{j+\phi} - 1)^2], \phi = \{50\}$$

$$g2_{D=500} = \sum_{i=1}^{2}\sum_{j=1}^{3-1}[100(x_{j+\phi}^2 - x_{j+\phi+1})^2 + (x_{j+\phi} - 1)^2], \phi = \{50, 450\}$$

$$g2_{D=500} = \sum_{i=1}^{3}\sum_{j=1}^{3-1}[100(x_{j+\phi}^2 - x_{j+\phi+1})^2 + (x_{j+\phi} - 1)^2], \phi = \{50, 450, 850\}$$

Restricción 3 (g3): es la restricción más dura de las tres, esta tiene pares de variables no separables que se encuentran empalmadas.

$$g3_{D=100} = 100(x_{10}^2 - x_{90})^2 + (x_{10} - 1)^2$$

$$g3_{D=500} = 100(x_{10}^2 - x_{490})^2 + (x_{10} - 1)^2 + 100(x_{16}^2 - x_{496})^2 + (x_{16} - 1)^2$$

$$\begin{aligned} g3_{D=1000} &= 100(x_{10}^2 - x_{890})^2 \\ &+(x_{10} - 1)^2 + 100(x_{16}^2 - x_{896})^2 + (x_{16} - 1)^2 \\ &+100(x_{22}^2 - x_{902})^2 + (x_{22} - 1)^2 \end{aligned}$$

A.3 Conjunto de prueba y características

Tabla. A.1.: Resumen de las funciones objetivo y restricciones para el conjunto de prueba Sayed [37]. Los 18 problemas restringidos son generados a a partir de la combinación de una de las 6 funciones objetivo, con una, dos o tres restricciones.

Función	Descripción		Objetivo	g1	g2	g3		Objetivo	g1	g2	g3
Obj1	Completamente separable	**F1**	**Obj1**	✓			**F10**	**Obj4**	✓		
Obj2	Parcialmente no separable	**F2**		✓	✓		**F11**		✓	✓	
Obj3	Parcialmente no separable	**F3**		✓	✓	✓	**F12**		✓	✓	✓
Obj4	Parcialmente no separable, sobrelapado	**F4**	**Obj2**	✓			**F13**	**Obj5**	✓		
Obj5	Empalmada no separable, sobrelapado	**F5**		✓	✓		**F14**		✓	✓	
Obj6	Empalmada no separable, sobrelapado	**F6**		✓	✓	✓	**F15**		✓	✓	✓
g1	Separable en grupos de 6 variables	**F7**	**Obj3**	✓			**F16**	**Obj6**	✓		
g2	No separable en grupos de 3 variables	**F8**		✓	✓		**F17**		✓	✓	
g3	Empalmada en pares no separables	**F9**		✓	✓	✓	**F18**		✓	✓	✓

B

Tablas de resultados de Buscadores Locales

Tabla. B.1.: Resultados numéricos para la dimensión 100. Funciones 01 a 09. En negrita se presenta el mejor resultado obtenido por la variante especifica del algoritmo. En gris el mejor resultado obtenido de todos los algoritmos probados.

	F	median±Desv.Std.	F	median±Desv.Std.	F	median±Desv.Std.
HJ-DG2		2.46E-05 ± 4.76E-06		1.48E+03 ± 1.32E+04		1.14E+05± 1.13E+06
HJ-DVIIC		3.61E-04 ± 4.88E-04		**1.57E+02 ± 1.30E+06**		**7.36E+02± 1.62E+06**
HJ-VIIC		**8.21E-05 ± 1.72E-05**		6.85E+03 ± 1.23E+06		2.32E+03± 7.85E+05
HJ-ND		3.15E+00 ± 5.31E-01		1.66E+03 ± 4.91E+03		1.06E+04± 1.10E+04
NM-DG2		2.86E-07 ± 9.08E+03		4.15E+01 ± 5.59E+03		**2.63E+01± 7.18E+03**
NM-DVIIC		**9.09E-13 ± 3.05E+03**		**2.36E-03 ± 5.61E+06**		2.11E+03± 1.58E+09
NM-ND	F01	2.86E-11 ± 7.12E+03	F02	2.42E+07 ± 4.05E+09	F03	2.08E+04± 2.25E+09
NM-VIIC		6.81E-13 ± 1.31E+03		3.08E+03 ± 1.23E+09		1.52E+04± 2.96E+07
RW-DG2		3.95E-12 ± 2.17E+01		**4.34E+01 ± 2.53E+01**		**7.20E+01± 4.52E+01**
RW-DVIIC		**8.03E-12 ± 2.28E-11**		4.68E+03 ± 8.47E+03		1.13E+04± 1.38E+05
RW-ND		3.74E-12 ± 1.98E-12		1.51E+04 ± 5.96E+03		1.85E+04± 8.06E+03
RW-VIIC		1.08E-11 ± 6.81E-12		1.11E+04 ± 3.42E+05		1.49E+04± 5.81E+05
HJ-DG2		2.36E-05 ± 4.87E-06		1.47E+03 ± 1.59E+04		8.09E+04± 1.41E+05
HJ-DVIIC		3.76E-04 ± 1.10E-03		1.46E+05 ± 1.73E+06		2.52E+05± 1.15E+06
HJ-VIIC		**8.51E-05 ± 1.64E-05**		**5.98E+02 ± 9.25E+05**		**1.92E+03± 1.09E+06**
HJ-ND		3.07E+00 ± 5.99E-01		7.79E+02 ± 3.94E+03		1.12E+04± 1.23E+05
NM-DG2		3.96E-12 ± 8.52E+03		**1.79E-07 ± 2.90E+03**		**4.53E+01± 2.00E+03**
NM-DVIIC		**9.02E-13 ± 8.82E+02**		1.85E+03 ± 1.84E+09		2.08E+04± 1.40E+09
NM-ND	F04	1.56E-11 ± 6.24E+03	F05	2.45E+04 ± 3.30E+09	F06	1.62E+04± 2.51E+09
NM-VIIC		2.39E-13 ± 2.85E+03		9.82E+03 ± 4.34E+08		4.51E+03± 2.96E+09
RW-DG2		5.29E-12 ± 2.63E-12		**1.88E+01 ± 2.11E+01**		**3.73E+01± 4.35E+01**
RW-DVIIC		7.79E-12 ± 3.20E-11		9.65E+03 ± 5.95E+05		3.54E+04± 9.43E+05
RW-ND		4.12E-12 ± 4.96E-01		1.43E+04 ± 5.17E+03		1.59E+04± 7.61E+03
RW-VIIC		**9.13E-12 ± 9.07E-12**		1.07E+04 ± 9.86E+05		1.88E+04± 7.41E+05
HJ-DG2		**2.33E-05 ± 3.78E-06**		**1.85E+02 ± 5.58E+01**		1.58E+05± 1.30E+06
HJ-DVIIC		4.61E-04 ± 3.74E+00		1.93E+05 ± 1.99E+06		7.89E+04± 8.14E+05
HJ-VIIC		8.60E-05 ± 1.89E-05		3.25E+04 ± 9.43E+05		2.17E+05± 1.68E+06
HJ-ND		2.95E+00 ± 7.65E-01		2.01E+03 ± 7.85E+03		**1.37E+04± 1.41E+04**
NM-DG2		2.44E-09 ± 7.65E+03		**8.45E-08 ± 2.44E+03**		**3.62E+01± 5.58E+03**
NM-DVIIC		**7.61E-13 ± 3.07E+03**		8.57E+01 ± 1.58E+09		2.15E+04± 3.67E+06
NM-ND	F07	2.03E-05 ± 8.16E+03	F08	2.37E+04 ± 1.56E+09	F09	2.03E+04± 2.70E+09
NM-VIIC		6.70E-13 ± 7.14E+03		3.19E+03 ± 2.29E+08		1.37E+04± 1.22E+09
RW-DG2		5.86E-12 ± 1.45E+01		**4.58E+01 ± 2.32E+01**		**8.92E+01± 3.83E+01**
RW-DVIIC		6.62E-12 ± 6.35E-11		1.13E+04 ± 3.95E+05		1.72E+04± 1.58E+06
RW-ND		5.07E-12 ± 1.86E-12		1.43E+04 ± 5.54E+03		1.66E+04± 8.19E+04
RW-VIIC		**8.89E-12 ± 6.62E-12**		8.44E+03 ± 2.58E+05		2.12E+04± 1.38E+06

Tabla. B.2.: Resultados numéricos para la dimensión 100. Funciones 10 a 18. En negrita se presenta el mejor resultado obtenido por la variante especifica del algoritmo. En gris el mejor resultado obtenido de todos los algoritmos probados.

	F	mediana±Desv.Std.	F	mediana±Desv.Std.	F	mediana±Desv.Std.
HJ-DG2		3.16E-04 ± 1.40E-04		2.31E+03 ± 9.58E+03		**2.24E+03± 1.01E+04**
HJ-DVIIC		4.30E-04 ± 9.27E-01		1.45E+05 ± 1.50E+06		1.90E+05± 1.44E+06
HJ-VIIC		**8.39E-05 ± 2.02E-05**		4.20E+03 ± 1.05E+06		3.93E+03± 8.75E+05
HJ-ND		3.02E+00 ± 5.26E-01		**1.49E+03 ± 3.48E+03**		7.19E+03± 1.37E+04
NM-DG2		1.16E-12 ± 6.94E+03		**3.53E-01 ± 5.69E+03**		**6.88E+01± 3.34E+03**
NM-DVIIC	F10	9.41E-13 ± 2.23E+03	F11	5.82E+03 ± 4.03E+08	F12	7.56E+03± 1.48E+09
NM-ND		**4.47E-15 ± 6.17E+03**		1.12E+04 ± 1.96E+09		1.78E+04± 2.24E+09
NM-VIIC		4.57E-13 ± 2.62E+03		8.63E+03 ± 2.75E+08		9.12E+03± 1.74E+09
RW-DG2		5.83E-12 ± 2.24E-12		**2.10E+01 ± 2.84E+01**		**6.33E+01± 3.03E+01**
RW-DVIIC		5.69E-12 ± 1.79E-11		1.50E+04 ± 1.52E+06		1.92E+04± 2.24E+05
RW-ND		5.55E-12 ± 2.95E+01		1.33E+04 ± 7.49E+03		1.77E+04± 5.46E+03
RW-VIIC		**7.83E-12 ± 8.97E-12**		1.60E+04 ± 8.00E+05		1.88E+04± 1.40E+06
HJ-DG2		3.37E-04 ± 1.51E-04		1.60E+03 ± 4.32E+03		**1.80E+03± 1.11E+04**
HJ-DVIIC		3.55E-04 ± 1.82E-02		4.38E+05 ± 3.08E+06		8.13E+04± 2.14E+06
HJ-VIIC		**7.99E-05 ± 1.50E-05**		1.05E+05 ± 1.30E+06		3.32E+04± 7.98E+05
HJ-ND		3.15E+00 ± 5.87E-01		**1.45E+03 ± 1.90E+03**		5.20E+03± 1.26E+04
NM-DG2		1.05E-14 ± 4.66E+03		**1.88E-06 ± 3.46E+03**		**6.97E+01± 6.24E+03**
NM-DVIIC	F13	8.24E-13 ± 3.58E+03	F14	4.23E+04 ± 5.36E+08	F15	3.17E+04± 6.56E+07
NM-ND		7.91E-11 ± 8.77E+03		2.66E+04 ± 2.29E+09		2.04E+06± 2.17E+09
NM-VIIC		**6.33E-14 ± 2.21E+03**		1.12E+04 ± 6.55E+05		2.36E+04± 6.67E+07
RW-DG2		**6.93E-12 ± 3.96E-12**		**1.97E+01 ± 2.52E+01**		**6.56E+01± 3.74E+01**
RW-DVIIC		6.08E-12 ± 1.02E-11		1.32E+04 ± 1.38E+06		2.06E+04± 1.29E+06
RW-ND		4.40E-12 ± 1.46E-12		1.85E+04 ± 6.80E+03		1.78E+04± 4.88E+03
RW-VIIC		1.09E-11 ± 8.09E-12		1.44E+04 ± 6.60E+05		2.22E+04± 8.39E+05
HJ-DG2		2.43E-05 ± 4.60E-06		**1.46E+02 ± 7.00E+01**		**2.75E+02± 2.41E+02**
HJ-DVIIC		3.11E-04 ± 6.83E-04		5.87E+05 ± 2.71E+06		3.11E+05± 1.44E+06
HJ-VIIC		**8.10E-05 ± 2.09E-05**		1.55E+04 ± 1.61E+06		9.85E+02± 2.45E+05
HJ-ND		2.99E+00 ± 5.45E-01		2.28E+03 ± 3.89E+03		9.60E+03± 1.20E+04
NM-DG2		4.58E-11 ± 7.23E+03		1.60E+04 ± 3.51E+09		1.50E+04± 2.34E+09
NM-DVIIC	F16	1.05E-12 ± 1.07E+02	F17	**4.25E+03 ± 1.89E+09**	F18	4.67E+05± 7.33E+08
NM-ND		3.06E-13 ± 5.80E+03		3.08E+04 ± 2.55E+09		1.32E+04± 2.21E+09
NM-VIIC		**3.26E-13 ± 1.39E+03**		4.96E+03 ± 1.44E+09		**1.11E+04± 6.93E+05**
RW-DG2		3.75E-12 ± 2.04E-12		1.28E+04 ± 6.72E+03		1.55E+04± 7.02E+03
RW-DVIIC		1.18E-11 ± 4.60E-11		1.21E+04 ± 1.08E+06		2.22E+05± 7.45E+05
RW-ND		**4.30E-12 ± 1.69E+02**		1.24E+04 ± 7.52E+03		**1.43E+04± 6.41E+03**
RW-VIIC		1.01E-11 ± 8.29E-12		**9.13E+03 ± 2.38E+05**		1.58E+04± 6.92E+05

Tabla. B.3.: Resultados numéricos para la dimensión 500. Funciones 01 a 09. En negrita se presenta el mejor resultado obtenido por la variante especifica del algoritmo. En gris el mejor resultado obtenido de todos los algoritmos probados.

	F	median±Desv.Std.	F	mediana±Desv.Std.	F	mediana±Desv.Std.
HJ-DG2		2.46E-05 ± 4.76E-06		1.48E+03 ± 1.32E+04		1.14E+05± 1.13E+06
HJ-DVIIC		3.61E-04 ± 4.88E-04		**1.57E+02 ± 1.30E+06**		**7.36E+02± 1.62E+06**
HJ-VIIC		**8.21E-05 ± 1.72E-05**		6.85E+03 ± 1.23E+06		2.32E+03± 7.85E+05
HJ-ND		3.15E+00 ± 5.31E-01		1.66E+03 ± 4.91E+03		1.06E+04± 1.10E+04
NM-DG2		7.34E+04 ± 1.31E+04		**7.42E+04 ± 3.00E+04**		**7.50E+04± 2.96E+04**
NM-DVIIC	F01	**2.96E+04 ± 2.70E+04**	F02	1.22E+09 ± 3.90E+09	F03	4.28E+09± 9.21E+09
NM-ND		3.09E+04 ± 2.98E+04		3.87E+09 ± 5.06E+09		8.17E+09± 5.63E+09
NM-VIIC		7.49E+04 ± 2.73E+04		6.73E+09 ± 4.89E+09		9.18E+09± 5.69E+09
RW-DG2		9.17E+03 ± 4.51E+03		**1.08E+04 ± 6.05E+03**		**1.10E+04± 4.39E+03**
RW-DVIIC		**2.55E-11 ± 1.78E+03**		3.74E+04 ± 1.31E+06		5.36E+04± 1.62E+06
RW-ND		1.18E-11 ± 4.64E-12		9.17E+04 ± 1.24E+06		8.98E+04± 8.15E+05
RW-VIIC		1.19E+04 ± 5.59E+03		7.53E+04 ± 1.24E+04		7.52E+04± 1.65E+04
HJ-DG2		2.36E-05 ± 4.87E-06		1.47E+03 ± 1.59E+04		8.09E+04± 1.41E+05
HJ-DVIIC		3.76E-04 ± 1.10E-03		1.46E+05 ± 1.73E+06		2.52E+05± 1.15E+06
HJ-VIIC		**8.51E-05 ± 1.64E-05**		**5.98E+02 ± 9.25E+05**		**1.92E+03± 1.09E+06**
HJ-ND		3.07E+00 ± 5.99E-01		7.79E+02 ± 3.94E+03		1.12E+04± 1.23E+05
NM-DG2		7.74E+04 ± 3.22E+04		**7.30E+04 ± 3.72E+04**		**7.40E+04± 3.70E+04**
NM-DVIIC	F04	**1.41E+04 ± 2.88E+04**	F05	2.96E+09 ± 3.97E+09	F06	2.09E+09± 4.46E+09
NM-ND		4.83E+04 ± 3.21E+04		3.85E+09 ± 4.66E+09		**1.03E+10± 5.16E+09**
NM-VIIC		7.09E+04 ± 3.46E+04		8.01E+09 ± 4.69E+09		1.11E+10± 6.59E+09
RW-DG2		7.41E+03 ± 3.75E+03		**8.02E+03 ± 4.58E+03**		**6.45E+03± 4.66E+03**
RW-DVIIC		**3.18E-11 ± 4.80E+03**		1.65E+05 ± 1.73E+06		3.14E+05± 1.15E+06
RW-ND		1.42E-11 ± 4.80E-12		7.18E+04 ± 9.43E+05		8.74E+04± 1.10E+06
RW-VIIC		1.04E+04 ± 6.75E+03		7.65E+04 ± 1.33E+04		7.77E+04± 1.42E+04
HJ-DG2		2.33E-05 ± 3.78E-06		**1.85E+02 ± 5.58E+01**		1.58E+05± 1.30E+06
HJ-DVIIC		4.61E-04 ± 3.74E+00		1.93E+05 ± 1.99E+06		**7.89E+04± 8.14E+05**
HJ-VIIC		**8.60E-05 ± 1.89E-05**		3.25E+04 ± 9.43E+05		2.17E+05± 1.68E+06
HJ-ND		2.95E+00 ± 7.65E-01		2.01E+03 ± 7.85E+03		1.37E+04± 1.41E+04
NM-DG2		7.17E+04 ± 3.78E+04		**7.68E+04 ± 3.96E+04**		**7.47E+04± 3.37E+04**
NM-DVIIC	F07	**2.29E+04 ± 2.69E+04**	F08	4.86E+09 ± 5.23E+09	F09	4.67E+09± 5.31E+09
NM-ND		4.30E+04 ± 2.95E+04		4.59E+09 ± 4.71E+09		8.87E+09± 5.63E+09
NM-VIIC		7.79E+04 ± 3.14E+04		3.50E+09 ± 4.31E+09		8.58E+09± 5.43E+09
RW-DG2		7.84E+03 ± 5.23E+03		**8.94E+03 ± 4.17E+03**		**7.12E+03± 5.35E+03**
RW-DVIIC		**2.73E-11 ± 2.94E+03**		2.55E+05 ± 1.98E+06		1.45E+05± 8.88E+05
RW-ND		1.30E-11 ± 4.17E-12		1.14E+05 ± 9.70E+05		3.13E+05± 1.67E+06
RW-VIIC		1.20E+04 ± 4.57E+03		7.40E+04 ± 1.58E+04		7.95E+04± 6.47E+04

Tabla. B.4.: Resultados numéricos para la dimensión 500. Funciones 10 a 18. En negrita se presenta el mejor resultado obtenido por la variante especifica del algoritmo. En gris el mejor resultado obtenido de todos los algoritmos probados.

	F	mediana±Desv.Std.	F	mediana±Desv.Std.	F	mediana±Desv.Std.
HJ-DG2		3.16E-04 ± 1.40E-04		2.31E+03 ± 9.58E+03		**2.24E+03± 1.01E+04**
HJ-DVIIC		**4.30E-04 ± 9.27E-01**		1.45E+05 ± 1.50E+06		1.90E+05± 1.44E+06
HJ-VIIC		8.39E-05 ± 2.02E-05		4.20E+03 ± 1.05E+06		3.93E+03± 8.75E+05
HJ-ND		3.02E+00 ± 5.26E-01		**1.49E+03 ± 3.48E+03**		7.19E+03± 1.37E+04
NM-DG2		4.91E+04 ± 2.01E+04		**4.64E+04 ± 1.64E+04**		**5.04E+04± 3.99E+07**
NM-DVIIC	F10	**2.82E+04 ± 1.96E+04**	F11	2.84E+09 ± 3.85E+09	F12	4.77E+09± 4.45E+09
NM-ND		3.15E+04 ± 2.57E+04		4.44E+09 ± 4.65E+09		1.01E+10± 6.84E+09
NM-VIIC		7.21E+04 ± 3.60E+04		7.24E+09 ± 5.49E+09		1.10E+10± 6.31E+09
RW-DG2		3.32E-11 ± 4.37E+02		**1.03E+02 ± 4.20E+02**		**1.38E+04± 4.11E+03**
RW-DVIIC		2.21E-11 ± 8.80E-12		2.16E+05 ± 1.56E+06		2.53E+05± 1.43E+06
RW-ND		**1.26E-11 ± 3.30E-12**		6.24E+04 ± 1.05E+06		1.01E+05± 9.15E+05
RW-VIIC		1.12E+04 ± 7.11E+03		7.80E+04 ± 1.54E+04		8.41E+04± 1.31E+04
HJ-DG2		3.37E-04 ± 1.51E-04		1.60E+03 ± 4.32E+03		**1.80E+03± 1.11E+04**
HJ-DVIIC		3.55E-04 ± 1.82E-02		4.38E+05 ± 3.08E+06		8.13E+04± 2.14E+06
HJ-VIIC		**7.99E-05 ± 1.50E-05**		1.05E+05 ± 1.30E+06		3.32E+04± 7.98E+05
HJ-ND		3.15E+00 ± 5.87E-01		**1.45E+03 ± 1.90E+03**		5.20E+03± 1.26E+04
NM-DG2		4.70E+04 ± 2.03E+04		**4.67E+04 ± 1.49E+04**		**5.64E+04± 3.16E+09**
NM-DVIIC	F13	**2.41E+04 ± 2.36E+04**	F14	1.50E+09 ± 4.55E+09	F15	5.37E+09± 6.48E+09
NM-ND		4.20E+04 ± 2.60E+04		5.17E+09 ± 3.46E+09		1.03E+10± 6.21E+09
NM-VIIC		7.94E+04 ± 3.76E+04		9.20E+09 ± 5.52E+09		9.00E+09± 5.94E+09
RW-DG2		3.16E-11 ± 1.26E+03		**7.57E+01 ± 6.84E+02**		**1.40E+04± 5.52E+03**
RW-DVIIC		2.80E-11 ± 1.61E+03		4.70E+05 ± 3.08E+06		1.23E+05± 2.13E+06
RW-ND		**1.41E-11 ± 3.64E-12**		1.68E+05 ± 1.31E+06		1.18E+05± 8.12E+05
RW-VIIC		1.22E+04 ± 5.34E+03		8.16E+04 ± 1.10E+04		7.96E+04± 1.08E+04
HJ-DG2		**2.43E-05 ± 4.60E-06**		**1.46E+02 ± 7.00E+01**		**2.75E+02± 2.41E+02**
HJ-DVIIC		3.11E-04 ± 6.83E-04		5.87E+05 ± 2.71E+06		3.11E+05± 1.44E+06
HJ-VIIC		8.10E-05 ± 2.09E-05		1.55E+04 ± 1.61E+06		9.85E+02± 2.45E+05
HJ-ND		2.99E+00 ± 5.45E-01		2.28E+03 ± 3.89E+03		9.60E+03± 1.20E+04
NM-DG2		7.30E+04 ± 3.47E+04		8.88E+09 ± 6.63E+09		8.25E+09± 6.13E+09
NM-DVIIC	F16	**4.52E+01 ± 2.31E+04**	F17	**2.90E+09 ± 3.37E+09**	F18	**6.06E+09± 5.30E+09**
NM-ND		3.07E+04 ± 3.42E+04		5.99E+09 ± 4.95E+09		9.09E+09± 7.56E+09
NM-VIIC		7.30E+04 ± 3.96E+04		7.81E+09 ± 5.57E+09		8.00E+09± 7.31E+09
RW-DG2		1.02E+04 ± 5.74E+03		**7.45E+04 ± 1.36E+04**		8.00E+04± 4.38E+05
RW-DVIIC		2.44E-11 ± 8.63E-12		6.21E+05 ± 2.71E+06		3.56E+05± 1.44E+06
RW-ND		**1.43E-11 ± 4.39E-12**		1.06E+05 ± 1.61E+06		**7.72E+04± 2.49E+05**
RW-VIIC		1.08E+04 ± 5.70E+03		6.95E+04 ± 1.43E+04		8.49E+04± 1.51E+04

Tabla. B.5.: Resultados numéricos para la dimensión 1000. Funciones 01 a 09. En negrita se presenta el mejor resultado obtenido por la variante especifica del algoritmo. En gris el mejor resultado obtenido de todos los algoritmos probados.

	F	mediana±Desv.Std.	F	mediana±Desv.Std.	F	mediana±Desv.Std.
HJ-DG2		1.34E-05 ± 1.37E-06		2.53E+03 ± 2.03E+04		1.30E+05± 1.76E+06
HJ-DVIIC		3.88E-04 ± 4.19E-04		**2.50E+02 ± 2.46E+06**		1.05E+04± 1.16E+06
HJ-VIIC		**4.66E-05 ± 7.08E-06**		2.11E+05 ± 1.69E+06		8.92E+04± 1.76E+06
HJ-ND		1.81E+00 ± 1.91E-01		2.02E+03 ± 7.51E+03		**7.45E+03± 1.48E+04**
NM-DG2		1.65E+05 ± 4.17E+04		**1.61E+05 ± 3.12E+04**		**1.67E+05± 1.82E+04**
NM-DVIIC	F01	**6.92E+04 ± 4.56E+04**	F02	9.04E+09 ± 6.52E+09	F03	1.23E+10± 7.59E+09
NM-ND		1.35E+05 ± 5.71E+04		9.75E+09 ± 6.95E+09		1.57E+10± 7.14E+09
NM-VIIC		1.69E+05 ± 1.69E+04		1.16E+10 ± 6.76E+09		1.43E+10± 7.51E+09
RW-DG2		5.16E+04 ± 1.24E+04		**4.64E+04 ± 1.03E+04**		**4.96E+04± 1.01E+04**
RW-DVIIC		7.20E-11 ± 1.02E+02		9.49E+04 ± 2.48E+06		1.57E+05± 1.16E+06
RW-ND		3.00E-11 ± 9.77E-12		3.81E+05 ± 1.70E+06		2.95E+05± 1.81E+06
RW-VIIC		5.52E+04 ± 9.85E+03		1.53E+05 ± 6.49E+04		1.67E+05± 8.31E+06
HJ-DG2		1.32E-05 ± 1.55E-06		**3.99E+03 ± 8.84E+03**		1.71E+05± 1.98E+05
HJ-DVIIC		5.58E-04 ± 3.74E-04		2.39E+05 ± 2.66E+06		9.30E+05± 2.18E+06
HJ-VIIC		**4.68E-05 ± 6.98E-06**		2.70E+05 ± 1.67E+06		1.14E+05± 8.55E+05
HJ-ND		1.77E+00 ± 1.87E-01		1.59E+03 ± 4.33E+03		**1.40E+04± 1.38E+04**
NM-DG2		1.64E+05 ± 3.94E+04		**1.59E+05 ± 4.03E+04**		**1.60E+05± 2.62E+04**
NM-DVIIC	F04	**4.50E+04 ± 4.02E+04**	F05	7.21E+09 ± 5.29E+09	F06	1.07E+10± 6.25E+09
NM-ND		1.46E+05 ± 5.31E+04		1.02E+10 ± 5.68E+09		1.33E+10± 6.76E+09
NM-VIIC		1.59E+05 ± 3.76E+04		9.71E+09 ± 5.34E+09		1.42E+10± 8.89E+09
RW-DG2		4.18E+04 ± 1.22E+04		**3.97E+04 ± 1.13E+04**		**3.90E+04± 1.13E+04**
RW-DVIIC		**8.27E-11 ± 1.06E+04**		3.43E+05 ± 2.61E+06		1.10E+06± 2.21E+06
RW-ND		3.32E-11 ± 4.22E+01		5.30E+05 ± 1.68E+06		2.85E+05± 8.61E+05
RW-VIIC		5.19E+04 ± 1.00E+04		1.60E+05 ± 1.91E+04		1.71E+05± 8.60E+06
HJ-DG2		1.30E-05 ± 1.36E-06		**2.67E+02 ± 7.02E+01**		1.41E+05± 1.49E+06
HJ-DVIIC		3.87E-04 ± 1.92E-01		7.53E+05 ± 1.88E+06		7.49E+05± 2.29E+06
HJ-VIIC		**4.85E-05 ± 5.10E-06**		3.78E+05 ± 1.53E+06		2.52E+05± 2.07E+06
HJ-ND		1.70E+00 ± 2.50E-01		1.90E+03 ± 3.84E+03		**5.53E+03± 1.14E+04**
NM-DG2		1.56E+05 ± 2.16E+04		**1.61E+05 ± 3.84E+04**		**1.55E+05± 3.68E+04**
NM-DVIIC	F08	**5.88E+04 ± 3.08E+04**	F08	7.13E+09 ± 6.01E+09	F09	9.23E+09± 8.42E+09
NM-ND		1.28E+05 ± 4.56E+04		8.16E+09 ± 7.63E+09		1.59E+10± 6.73E+09
NM-VIIC		1.61E+05 ± 2.29E+04		1.21E+10 ± 5.50E+09		1.74E+10± 7.28E+09
RW-DG2		4.20E+04 ± 1.06E+04		**4.28E+04 ± 9.35E+03**		**3.83E+04± 1.04E+04**
RW-DVIIC		**6.94E-11 ± 5.64E-03**		8.47E+05 ± 1.88E+06		8.51E+05± 2.34E+06
RW-ND		3.81E-11 ± 1.60E-01		5.34E+05 ± 1.54E+06		3.89E+05± 2.07E+06
RW-VIIC		5.01E+04 ± 1.24E+04		1.61E+05 ± 1.81E+04		1.61E+05± 6.90E+06

Tabla. B.6.: Resultados numéricos para la dimensión 1000. Funciones 10 a 18. En negrita se presenta el mejor resultado obtenido por la variante especifica del algoritmo. En gris el mejor resultado obtenido de todos los algoritmos probados.

	F	mediana±Desv.Std.	F	mediana±Desv.Std.	F	mediana±Desv.Std.
HJ-DG2		4.08E-05 ± 9.74E-05		**1.23E+03 ± 7.05E+03**		**3.47E+03± 1.11E+04**
HJ-DVIIC		**9.28E-05 ± 5.47E-04**		1.15E+06 ± 1.70E+06		9.91E+05± 2.13E+06
HJ-VIIC		3.52E-05 ± 4.75E-05		3.20E+05 ± 1.51E+06		2.25E+04± 3.37E+06
HJ-ND		1.43E+00 ± 1.73E+00		1.69E+03 ± 6.40E+03		7.52E+03± 8.87E+03
NM-DG2		1.03E+05 ± 1.34E+05		**1.62E+05 ± 1.99E+09**		**1.43E+05± 1.42E+09**
NM-DVIIC	F10	**1.35E+00 ± 7.10E+04**	F11	9.62E+09 ± 6.43E+09	F12	9.84E+09± 6.59E+09
NM-ND		2.54E+04 ± 1.47E+05		9.02E+09 ± 8.72E+09		2.09E+10± 9.62E+09
NM-VIIC		1.16E+05 ± 1.57E+05		1.21E+10 ± 5.80E+09		1.61E+10± 8.26E+09
RW-DG2		6.56E+03 ± 1.92E+04		**2.39E+04 ± 7.83E+03**		**3.11E+04± 8.50E+03**
RW-DVIIC		**3.73E-11 ± 6.15E-11**		1.32E+06 ± 1.71E+06		8.35E+05± 2.36E+06
RW-ND		2.16E-11 ± 3.90E-11		4.05E+05 ± 1.52E+06		1.66E+05± 3.41E+06
RW-VIIC		2.64E+04 ± 4.70E+04		1.61E+05 ± 1.83E+04		1.67E+05± 8.53E+06
HJ-DG2		**9.92E-05 ± 4.55E-05**		**9.75E+02 ± 3.32E+03**		**1.60E+03± 1.20E+04**
HJ-DVIIC		3.98E-04 ± 1.40E+00		8.57E+05 ± 2.60E+06		9.99E+05± 1.58E+06
HJ-VIIC		4.79E-05 ± 6.77E-06		3.15E+05 ± 1.47E+06		1.65E+03± 1.92E+06
HJ-ND		1.71E+00 ± 2.10E-01		1.63E+03 ± 6.51E+03		6.32E+03± 4.97E+03
NM-DG2		1.33E+05 ± 2.32E+04		1.61E+05 ± 1.88E+09		**1.54E+05± 4.43E+09**
NM-DVIIC	F13	**8.14E+04 ± 3.91E+04**	F14	7.48E+09 ± 6.97E+09	F15	1.08E+10± 6.18E+09
NM-ND		1.02E+05 ± 4.31E+04		1.20E+10 ± 6.58E+09		1.82E+10± 7.46E+09
NM-VIIC		1.63E+05 ± 2.04E+04		9.77E+09 ± 5.70E+09		1.53E+10± 7.80E+09
RW-DG2		1.58E+04 ± 6.62E+03		**2.65E+04 ± 6.99E+03**		**3.35E+04± 8.14E+03**
RW-DVIIC		**6.50E-11 ± 1.64E+03**		1.02E+06 ± 2.77E+06		1.16E+06± 1.55E+06
RW-ND		3.32E-11 ± 1.10E-11		4.81E+05 ± 1.48E+06		1.70E+05± 1.93E+06
RW-VIIC		4.96E+04 ± 1.21E+04		1.63E+05 ± 2.09E+04		1.75E+05± 3.32E+06
HJ-DG2		1.34E-05 ± 1.57E-06		**2.13E+02 ± 1.21E+02**		**4.19E+02± 2.92E+02**
HJ-DVIIC		3.27E-04 ± 2.93E-02		8.77E+05 ± 1.76E+06		1.35E+06± 1.90E+06
HJ-VIIC		**4.85E-05 ± 8.42E-06**		4.52E+05 ± 9.97E+05		1.44E+05± 2.21E+06
HJ-ND		1.66E+00 ± 1.60E-01		1.69E+03 ± 3.46E+03		5.85E+03± 7.22E+03
NM-DG2		1.64E+05 ± 2.03E+04		8.62E+09 ± 6.98E+09		1.42E+10± 7.05E+09
NM-DVIIC	F16	**9.23E+04 ± 4.50E+04**	F17	**7.09E+09 ± 5.74E+09**	F18	**1.03E+10± 8.06E+09**
NM-ND		1.45E+05 ± 4.64E+04		7.29E+09 ± 5.59E+09		1.54E+10± 6.10E+09
NM-VIIC		1.64E+05 ± 2.33E+04		8.46E+09 ± 6.14E+09		1.63E+10± 7.21E+09
RW-DG2		4.99E+04 ± 1.14E+04		1.62E+05 ± 2.31E+04		1.70E+05± 4.67E+06
RW-DVIIC		**8.19E-11 ± 2.12E+03**		9.99E+05 ± 1.87E+06		1.08E+06± 1.99E+06
RW-ND		3.45E-11 ± 1.29E+01		5.76E+05 ± 9.96E+05		2.91E+05± 2.29E+06
RW-VIIC		5.13E+04 ± 1.25E+04		**1.54E+05 ± 1.54E+04**		**1.70E+05± 1.44E+06**

Índice de figuras

Lista de Tablas

Lista de algoritmos

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.