UNIVERSIDAD DEL PAPALOAPAN

Campus Loma Bonita

INGENIERÍA EN COMPUTACIÓN

PROPUESTA DE ALGORITMOS DE ESTIMACIÓN DE DISTRIBUCIÓN PARA PROBLEMAS DE OPTIMIZACIÓN CONTINUA

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

SUSANA ESPINOZA PEREZ

DIRECTOR DE TESIS:

DR. EDUARDO SÁNCHEZ SOTO

CO-DIRECTOR DE TESIS:

DR. SERGIO IVVAN VALDEZ PEÑA

LOMA BONITA, OAXACA, 2017

UNIVERSIDAD DEL PAPALOAPAN

Campus Loma Bonita

LOMA BONITA, OAXACA, 2017

INGENIERÍA EN COMPUTACIÓN

LA PRESENTE TESIS TITULADA

PROPUESTA DE ALGORITMOS DE ESTIMACIÓN DE DISTRIBUCIÓN PARA PROBLEMAS DE OPTIMIZACIÓN CONTINUA

PRESENTADA POR LA SUSTENTANTE DE LICENCIATURA: **SUSANA ESPINOZA PEREZ** BAJO LA ASESORÍA DEL **DR. SERGIO IVVAN VALDEZ PEÑA** Y EL **DR. EDUARDO SÁNCHEZ SOTO**, HA SIDO REVISADA Y ACEPTADA POR EL COMITÉ DE EXAMINADORES PARA SER DEFENDIDA EN EL EXAMEN PROFESIONAL Y OBTENER EL TÍTULO DE **INGENIERO EN COMPUTACIÓN**.

DR. EDUARDO SÁNCHEZ SOTO
DIRECTOR

DR. SERGIO IVVAN VALDEZ PEÑA
CO-DIRECTOR

M.I. LUIS A. JUÁREZ BLANCO
SINODAL

DRA. BEATRIZ C. LUNA OLIVERA
SINODAL

Dedicado a mi madre y hermana.

Agradecimientos

Desde estas líneas quiero expresar mi gratitud a todas las personas que, desde diferentes aspectos, me han acompañado en esta interesante experiencia. Agradezco a Dios por la vida de todas ellas, por la oportunidad que tuve de rodearme de ellas, por las bendiciones y cuidados que ha tenido para conmigo.

Por ayudar a que este trabajo fuera posible quiero agradecer muy especialmente a mi director Dr. Eduardo Sánchez Soto y co-director Dr. Sergio Ivvan Valdez Peña. Gracias por compartir sus conocimientos conmigo y por confiar en mis capacidades. Gracias a CIMAT por la beca otorgada para el desarrollo de esta tesis y gracias a la UNPA por ser la casa de estudio donde adquirí, a través de mis profesores, conocimientos y agradables experiencias.

Agradezco a los profesores por ayudarme a crecer en lo académico, por sus consejos y conocimientos impartidos. Gracias a mis revisores de tesis: la Dra. Beatriz C. Luna Olivera y el M.I. Luis A. Juárez Blanco por su tiempo dedicado. Muy en especial gracias a la Dra. Beatriz porque fue ella junto con el Dr. Eduardo quienes siempre apoyaron mi interés por las matemáticas y me animaron a participar en eventos donde conocí a personas que se apasionan por la investigación en el cómputo y las matemáticas, gracias a ésto mi interés por estas ciencias se fue haciendo mas grande. No puedo olvidar el apoyo de mis compañeros, quienes me ayudaron a crecer en lo personal y académico. Gracias a todos.

En el plano personal quiero agradecer a mi familia, quienes me apoyaron en todo momento. A mi mamá Alberta por la confianza depositada en mi, por su amor y paciencia cuando tenía que dedicarme por completo a los estudios. Gracias a mi hermana Eunice por su amor, cuidado y confianza, por alegrarse de mis logros, y también gracias por convencer a mamá que debía hacer las salidas que hice para crecer en lo académico y lo profesional. No quiero tener por menos a mis dos hermanos, Diego y Luis, quienes con su compañía me hacían olvidar el estrés, gracias.

A mi nueva familia de León Gto., quienes me apoyaron en mi estadía en Guanajuato durante la realización de esta tesis, quiero darles las gracias porque en momentos de soledad ellos estaban ahí y me mostraban su amor, siempre me hicieron sonreír.

Resumen

Los algoritmos de optimización se presentan en la industria y la academia de manera cotidiana. Existen una gran cantidad de problemas de interés y no existe un algoritmo o un grupo de algoritmos que puedan resolver toda la variedad de problemas de manera eficiente y cierta. En particular, los problemas con múltiples mínimos y máximos, no derivables, o incluso sin conocimiento explícito de la función a optimizar, se presentan de forma cotidiana en diferentes campos, y no existe un algoritmo de optimización eficiente (de orden polinomial) que garantice la solución óptima. Estos problemas han sido ampliamente estudiados y se han propuesto muchos algoritmos para aproximar su solución, pero es un campo de estudio que continúa como un reto abierto a la comunidad académica.

Dado lo anterior, en esta tesis se estudian y presentan propuestas de algoritmos, que atacan problemas como los mencionados, dentro de una familia denominada: algoritmos de estimación de distribución. En estos algoritmos se estima una distribución de probabilidad de un subconjunto de la población de soluciones y se toman muestras para generar la siguiente población. Los algoritmos de estimación de distribución siguen los siguientes pasos: 1)Generar una población inicial P, 2) seleccionar un subconjunto D de la población P, 3) estimar la función de probabilidad $p(X = x)$ a partir del subconjunto D, 4) muestrear $p(X = x)$ para generar descendientes y 5) ir al paso 2 hasta que el criterio de paro se cumpla.

Las contribuciones de esta tesis se centran en 3 propuestas: el uso de direcciones de la búsqueda, los algoritmos de estado estable, y una aplicación de detección de una parábola en imágenes de la retina del ojo humano, usando el algoritmo propuesto de estado semi estable con estimadores pesados.

Abstract

Optimization algorithms are widely use in the industry and academy everyday. There are a lot of problems of interest and there is no algorithm or group of algorithms that can solve all variety of problems efficiently and in a certain way. In particular, the ploblems with multiple minimums and maximums, non-derivable, or even without explicit knowledge of mathematical model, are presented everyday in different fields, and there is no efficient optimization algorithm (of polynomial order) that guarantees the optimal solution. These problems have been extensively studied and many algorithms have been proposed to approximate their solution, nevertheless it is a field of study that continues as an open challenge to the academic community.

According to the above reason, this thesis are study and introduces novel algorithms, which tackle the mentioned issues, within a family called: estimation of distribution algorithms. In these algorithms a probability distribution, of a subset of the population with the best solutions, is estimated and new samples are generated to replace the population. Estimation of distribution algorithms follow the next steps: 1) Generate an initial population, 2) select a subset D of the population P, 3) Estimating the probability function $p(X = x)$ from the subset D, 4) sample $p(X = x)$ to generate offspring and 5) go to the step 2 until the stop criterion is met.

The contribution of this thesis are: the use of search directions, stable state algorithms, and EDA application for parabola detection in images of the retina of human eyes, using the proposed algoritm of semi stable state with weighted estimators.

Índice general

Capítulo 1

Introducción

Muchos problemas o sistemas en estudio que pueden ser modelados matemáticamente se pueden expresar o reescribir como problemas de optimización. Un problema de optimización se refiere a encontrar un punto x^* que es la posición del máximo o mínimo de una función $f(x)$. Estos problemas se encuentran tanto en la industria como en la academia, por ejemplo, en la industria los problemas de minimizar costos o maximizar ganancias son vistos como problemas de optimización. En la academia, resolver un sistema de ecuaciones lineales de la forma $Ax = b$, se puede expresar como la minimización de $(Ax - b)^2$ para encontrar los valores de x que solucionan el sistema.

La optimización es una herramienta de uso cotidiano en la ciencia de toma de decisiones y en el análisis de sistemas físicos. Para hacer uso de esta herramienta se necesita identificar un objetivo (función objetivo), que es una medida cuantitativa del rendimiento del sistema en estudio. El objetivo depende de ciertas características del sistema, llamadas variables. La meta es encontrar los valores de las variables que optimicen el sistema[21].

Los problemas de optimización se pueden categorizar de acuerdo al dominio donde se busca la solución. En algunos, las variables solo tienen sentido si toman valores enteros, son llamados de programación entera, el valor de la variable se extrae de un conjunto finito pero a menudo muy grande. En contraste, el conjunto factible para los problemas de optimización continua generalmente es infinito, los valores de la variable son números reales[21], estos problemas pueden ser lineales o no lineales, convexos o no convexos, separables o no separables. En los problemas de optimización no convexos, la función objetivo tiene un gran número de mínimos y máximos locales, por lo que es necesario encontrar un óptimo global y no solo un óptimo local, ya que el primero es

mejor que el segundo, en términos de la función objetivo planteada. La mayoría de los problemas asociados a aplicaciones de ingeniería y diseño se catalogan como problemas de optimización continua, puesto que las variables de optimización pueden tomar cualquier valor numérico real dentro de límites establecidos[12].

Actualmente no existe un algoritmo de optimización universal, sino más bien una colección de algoritmos, cada uno de los cuales esta adaptado a un tipo particular de problemas de optimización [21]. Del teorema de Non free lunch, introducido por David Wolpert y William Macready[29], se sabe que no se puede encontrar ningún optimizador que sea el mejor en comportamiento para cualquier problema. Por otra parte, conocer características específicas del problema que se quiere trabajar permite el diseño de algoritmos que pueden ser adecuados para la solución del problema.

Los Algoritmos de Estimación de Distribución EDA (por sus siglas en inglés Estimation of Distribution algorithms), introducidos por primera vez por Mühlenbein y PaaB en 1996[13], han sido ampliamente estudiados para su uso en problemas de optimización continua en busca de un óptimo global; emplean estrategias de búsquedas heurísticas basadas en poblaciones, modelan probabilísticamente el espacio de búsqueda a partir de las soluciones previamente evaluadas y generan nuevas soluciones mediante el modelo probabilístico. Los EDA se pueden categorizar de acuerdo a la complejidad del modelo probabilístico utilizado para la búsqueda: univariado o multivariado, el univarido considera que la función objetivo es separable y que se puede estimar una función de probabilidad para cada dimensión y la multiplicación de tales funciones genera la distribución conjunta, el multivariado considera que el problema no es separable. Los EDA que utilizan distribuciones multivariadas son: el EMNAglobal, EMNAadaptative, EMNAincremental y EGNAee, y los que usan un modelo univariado son: el UMDAc, SHCLVND y PBILc[13]. Cabe mencionar que los EDA son una importante linea de investigación sobre la óptimización de problemas black-box, donde la función objetivo es desconocida.

El paquete BBOB, introducido en 2009[23], contiene un conjunto de problemas de tipo black-box que los investigadores han utilizado para ver como se desempeñan diferentes algoritmos, y poder comparar unos con otros, pensando que este tipo de funciones son representativas de los problemas reales que se pueden encontrar. En la literatura[24, 23] se ha empleado el paquete BBOB para comparar distintos optimizadores, el resultado de estas comparaciones muestra que los algoritmos que estadisticamente se desempeñan mejor, hasta ahora, son el CMA-ES y el

AMaLGaM[24, 3]. La idea del enfoque CMA-ES es meter direcciones de búsquedas del óptimo, por medio de la matriz de covarianza[3]. Mientras que en el enfoque AMaLGaM la idea es escalar la matriz de covarianza cuando es necesario para prevenir la convergencia prematura en pendientes, para tratar de aproximar mejor el óptimo global[24].

Teniendo en cuenta lo anterior, en este trabajo se proponen dos EDA para dar solución a problemas no lineales de optimización continua, en los que se asume que se desconoce su función objetivo, no es ni convexa ni cóncava, y no esta sujeta a restricciones. Las propuestas surgen como resultado de la selección de ideas principales de las metaheurísticas de cómputo evolutivo: Algoritmos Evolutivos GA (por sus siglas en inglés Genetic algorithm), EDA y estrategias Evolutivas ES (por sus siglas en inglés Evolution Strategy), que actualmente han mostrado un mejor rendimiento en comparación con otros optimizadores. Con la primer propuesta se pretende resolver un caso real, utilizando pocas evaluaciones con un modelo probabilístico relativamente sencillo, que trata de usar toda la información generada en el proceso de optimización (idea general del enfoque de estado estable), a diferencia de los algoritmos comunes que reemplazan la población completa o en su mayor parte cada generación. En la segunda propuesta se considera introducir direcciones de búsqueda en modelos probabilísticos multivariados, esta propuesta se comparará con EDA clásicos.

Con el enfoque de la primer propuesta, llamado de estado semi estable debido a que la población crece, se quiere garantizar que las soluciones siempre o casi siempre aproximen el mismo óptimo global. Este enfoque no ha sido usado, en nuestro conocimiento actual, en los EDA.

Para observar el desempeño de la segunda propuesta, donde se emplean direcciones de búsqueda en modelos probabilísticos multivariados, se realizaron pruebas sobre un conjunto de funciones tomadas de la literatura consultada[13, 27] y del paquete BBOB. Los resultados obtenidos fueron comparados con los obtenidos por los algoritmos UMDA, EMNA y EEDA, reproducidos en este trabajo. Ésto con el fin de sustentar que la propuesta del algoritmo puede encontrar una aproximación al óptimo competitiva con otros algoritmos. Por último, el algoritmo de la primer propuesta se probó sobre un caso de estudio de detección de parábolas en imágenes médicas de la retina del ojo, útil para la detección de arcada temporal. Se ejecutaron una serie de pruebas para obtener los parámetros de entrada que aseguran siempre o casi siempre la aproximación al mismo óptimo

y se compara contra los resultados publicados en la literatura especializada[14].

El documento se organiza como sigue: En el capítulo 2 se introducen algunos términos que ayudarán a contextualizar las propuestas. La sección 3 es una descripción de los EDA, los diferentes algoritmos que han sido desarrollados bajo este enfoque. En la sección 4 se describen los dos enfoques de las nuevas propuestas de EDA. En el capítulo 5 se muestran experimentos y resultados de la segunda propuesta comparada con la implementación de los algoritmos UMDA, EMNA y EEDA, así como una descripción breve del conjunto de funciones utilizadas. El capítulo 6 es un caso de estudio para detectar parábolas en imágenes de la retina del ojo, en el mismo capítulo se muestran pruebas y resultados obtenidos con la primer propuesta. Por último el capítulo 7 muestra las conclusiones y trabajo futuro de esta tesis.

1.1. Objetivos

1.1.1. Objetivo general

Proponer Algoritmos de Estimación de Distribución para dar solución a problemas no lineales de optimización continua, de los que se asume se desconoce su función objetivo, puede tener múltiples máximos o mínimos y no esta sujeta a restricciones.

1.1.2. Objetivos específicos

Los objetivos específicos de esta tesis son los siguientes:

- Estudiar conceptos de álgebra lineal, métodos numéricos, probabilidad y optimización.
- Estudiar algoritmos de Estimación de Distribución.
- Seleccionar algunos algoritmos de Estimación de Distribución.
- Implementar algoritmos de Estimación de Distribución seleccionados.
- Proponer algoritmos de Estimación de Distribución.
- Realizar pruebas de los algoritmos implementados y los propuestos.
- Comparar resultados obtenidos de los algoritmos implementados de la literatura con los obtenidos por las propuestas.

- Programar el evaluador que determine la calidad del registro de una parábola a una imagen segmentada.
- Realizar detección de parábolas o formas parabólicas en imágenes de la retina del ojo con uno de los algoritmos propuestos.
- Escribir el documento de tesis.

Capítulo 2

Marco teórico

En este capítulo se introducen algunos términos que ayudarán a contextualizar las propuestas en esta tesis. Entre ellos los de algoritmos de optimización, términos probabilísticos y de álgebra lineal.

2.1. Optimización

La optimización es una herramienta poderosa en la ciencia de toma de decisiones y en el análisis de sistemas físicos[21]. Ésta se refiere al análisis y resolución de problemas en que se debe tomar una solución entre un conjunto de *soluciones factibles*. El objetivo es encontrar la mejor solución (no necesariamente única) y las elecciones se comparan de acuerdo a una cierta función, llamada *función objetivo* [18]. La Definición 1 introduce de manera formal el concepto de optimización.

Definición 1 *Hablando matemáticamente, optimizar es maximizar o minimizar una función sujeta a restricciones sobre sus variables[21]. Un problema de optimización generalmente se compone de:*

- x *es el vector de* ***variables****, también llamadas incognitas o parámetros;*
- f *es la* ***función objetivo****, medida cuantitativa del sistema a optimizar (maximizar o minimizar);*
- c_i ***función de restricción****, representa el conjunto de relaciones que ciertas variables estan obligadas a cumplir.*

2.1.1. Planteamiento de un problema de optimización

Usando la notación de la Definición 1, el problema de optimización puede ser escrito como sigue [21]:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \textit{Sujeto a} \quad \begin{array}{ll} c_i(x) = 0, & i \in \varepsilon \\ c_i(x) \geq 0, & i \in I \end{array} \tag{1}$$

Aquí ε e I son el conjunto de restricciones de igualdad y desigualdad, respectivamente.
Convención: Usualmente se habla de minimizar y no de maximizar, por lo que maximizar f puede obtenerse de minimizar $-f$ [17].

2.1.2. Clasificación de problemas de optimización

Un problema general de optimización puede ser clasificado de acuerdo a la naturaleza de la función objetivo y las restricciones (lineal, no lineal, convexa), el número de variables (grande o pequeño), etc. [21]. Dentro de éstas se encuentran las siguientes.

2.1.2.1. Basándose en la existencia de restricciones

- **Ploblemas de optimización sin restricciones**
  Para los cuales se tiene que $\varepsilon = I = \emptyset$ en la Ecuación 1. En la optimización sin restricciones se minimiza una función objetivo que depende de variables reales, sin restricciones sobre los valores de estas variables. La formulación matemática es [21]:

$$\min f(x) \tag{2}$$

  Donde $x \in \mathbb{R}^n$ es un vector real con $n \geq 1$ elementos y $f : \mathbb{R}^n \to \mathbb{R}$ es una función suave[1].

- **Problemas de optimización con restricciones**
  Surgen a partir de los modelos en los que las restricciones desempeñan un papel esencial. Estas restricciones pueden ser simples, tales como $0 \leq x \leq 100$, restricciones mas generales como $\sum_i x_i \leq 1$, o desigualdades que representan relaciones complejas entre las variables [21].
  La presencia de restricciones limita el espacio de búsqueda pero, al mismo tiempo, dificulta el encontrar la solución óptima porque se pueden perder algunos de los criterios de optimalidad [7].

[1] Una función es suave si todas sus derivadas parciales, de cualquier orden, existen. Toda función suave es continua [5].

2.1.2.2. Basándose en la naturaleza de las ecuaciones involucradas

- **Problemas lineales.**

  Una de las áreas más importantes y activas de la optimización es la *Programación lineal* (PL). Trata problemas basados en optimización de una función objetivo lineal, sujeta a una serie de *restricciones* lineales de igualdad o desigualdad [18].

  Matricialmente, un problema de PL en notación estándar (con igualdades) se puede expresar como:

$$\begin{aligned} \text{máx} \quad & z = cx \\ s.a \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3}$$

  Donde cx es la función objetivo a optimizar,$x \in \mathbb{R}^{+n}$ representa el vector de variables a determinar, $c \in \mathbb{R}^n$ es el valor de costos asociados a las variables, $A \in M_{m \times n}$ es la matriz de coeficientes y $b \in \mathbb{R}^{+m}$ el vector de términos independientes relativos a las restricciones. Es decir, la Ecuación 3 es equivalente a la Ecuación 4 en notación extendida:

$$\begin{aligned} \text{máx} \quad & z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ s.a \quad & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}xn = b_1 \\ & a_{21}x_2 + a_{22}x_2 + \cdots + a_{2n}xn = b_2 \\ & \cdots \\ & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}xn = b_m \\ & x_1, x_2, \cdots, x_n \geq 0 \end{aligned} \tag{4}$$

- **Problemas no lineales.**

  Un supuesto importante en programación lineal es que todas sus funciones (objetivo y restricciones) son lineales. Aunque, en esencia, este supuesto se cumple en el caso de muchos problemas prácticos, con frecuencia no es así. Por lo que es necesario abordarlos desde la programación no lineal (PNL) [18]. De manera general, un problema de PNL consiste en encontrar $x = (x_1, x_2, ..., x_n)$ tal que:

$$\begin{aligned} \text{máx} \quad & f(x) \\ s.a \quad & g_i(x) \leq 0, \forall i = 1, 2, ..., m \\ & x \geq 0 \end{aligned} \tag{5}$$

  donde $f(x)$ y $g_i(x)$ son funciones dadas de n variables de decisiones.

2.1.3. Optimización discreta y optimización continua

En algunos problemas de optimización las variables tienen sentido solo si se toman valores en números enteros ($\mathbb{Z}$) o binarios (0 o 1). Estos son llamados *problemas de programación entera*, los cuales son un tipo de *problemas de optimización discreta*. La característica que define un problema de optimización discreta es que la variable x se toma de un conjunto finito, a menudo muy grande. En contraste, el conjunto factible para *problemas de optimización continua* por lo general es infinito (no numerable), los elementos de x son números reales ($\mathbb{R}$) [21].

2.1.4. Óptimo global y óptimo local

Para denotar el **punto óptimo** se utiliza la notación $\overrightarrow{x}^* = x_1^*, x_2^*, \cdots, x_n^*$, y el valor correspondiente de $f(\overrightarrow{x}^*)$ se denomina el **valor óptimo** de la función objetivo. El par $\overrightarrow{x}^*$ y $f(\overrightarrow{x}^*)$ constituyen una solución óptima. Existen varias categorías de soluciones óptimas si la función objetivo no es **unimodal** [2], es decir, si tiene más de un extremo, como se ilustra en la función multimodal[3], de la figura siguiente:

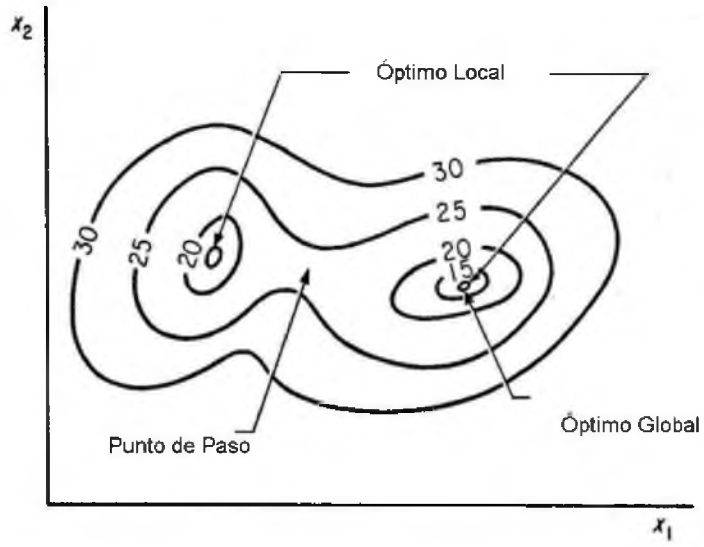


Figura 2.1: Función multimodal, ilustra los conceptos de óptimo global y local. Fuente:[6]

Un **óptimo global**, suponiendo minimización, es aquél que representa el valor más pequeño de

[2]Un sólo óptimo.
[3]Más de un óptimo.

$f(\vec{x})$, mientras que un **óptimo local** representa el valor más pequeño de $f(\vec{x})$ en la vecindad de algún vector $\vec{x}$. El óptimo global es el mejor de todos los óptimos locales [6].

2.1.5. Convexidad

El concepto de convexidad es fundamental en optimización. Muchos problemas prácticos poseen esta propiedad, lo cual generalmente los hace fáciles de resolver en la teoría y la práctica. El término "convexo" puede ser aplicado tanto a conjuntos como a funciones. Las funciones cóncavas y convexas representan un papel importante en la teoría de optimización ya que pueden garantizar la globalidad de los óptimos globales [21]. La Definición 2 introduce de manera formal el concepto de convexidad [11].

Definición 2 *Un conjunto $S \subset \mathbb{R}^n$ es convexo si al unir dos cualesquiera de sus puntos con un segmento éste no se sale del conjunto. Una función definida sobre un conjunto convexo es (estrictamente) convexa si al unir dos puntos cualesquiera de su gráfica con un segmento, éste queda (estrictamente) por encima de la gráfica. Una funcién es (estrictamente) céncava si al unir dos puntos cualesquiera de su gráfica con un segmento éste queda (estrictamente) por debajo de la gráfica (ver Figura 2.2).*

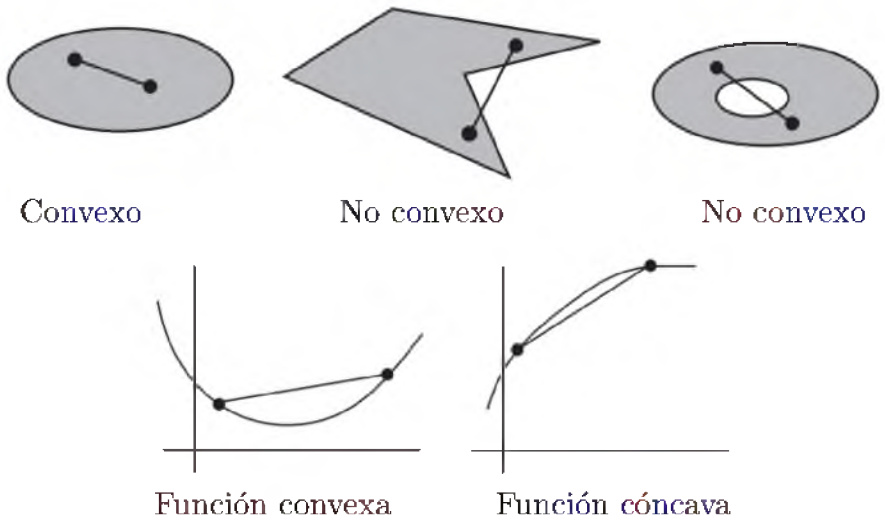


Figura 2.2: Representación del concepto de Convexidad. Fuente:[11]

2.1.6. Gradiente

Sea $f\colon \Omega \subset \mathbb{R}^n \to \mathbb{R}$ una función diferenciable (derivable) en $\overline{x_0}$. Entonces el vector cuyas componentes son las derivadas parciales de f en $\overline{x_0}$ se denomina **vector gradiente** y se denota por ∇f, es la función vectorial definida por [1]:

$$\nabla f(\overline{x_0}) = \left(\frac{\partial f(x_0)}{\partial x_1}, \frac{\partial f(x_0)}{\partial x_2}, \cdots, \frac{\partial f(x_0)}{\partial x_n} \right) \tag{6}$$

2.2. Conceptos básicos de probabilidad

La teoría de la probabilidad es la parte de las matemáticas que se encarga del estudio de los fenómenos o **experimentos aleatorios**. Un experimento aleatorio es todo aquel experimento que cuando se le repite bajo las mismas condiciones iniciales, el resultado que se obtiene no siempre es el mismo. El espacio muestral de un experimento aleatorio es el conjunto de todos los posibles resultados del experimento, y se denota generalmente por la letra griega Ω. Un evento es cualquier subconjunto del espacio muestral y se denotan por las primeras letras del alfabeto en mayúsculas: A, B, C, etc. [25].

2.2.1. Variable aleatoria

Una variable x evaluada numéricamente varía o cambia, dependiendo del resultado particular que se mida. Por ejemplo, se tira un dado y se mide x (número observado en la cara superior del dado). La variable x puede tomar cualquier de los valores: 1,2,3,4,5,6, dependiendo del resultado aleatorio del experimento. Por esta razón, la variable x es conocida como **variable aleatoria**. La idea intuitiva de una variable aleatoria es "un valor que depende del resultado de un experimento aleatorio"[19]. La Definición 3 introduce de manera formal el concepto de variable aleatoria [25]:

Definición 3 *Dado un experimento aleatorio cualquiera, una variable aleatoria es una transformación X del espacio de resultados Ω al conjunto de números reales, esto es, $X : \Omega \to \mathbb{R}$.*

Las variables aleatorias se clasifican en discretas y continuas [2] :

- **Discreta**: la variable X se dice que es discreta si los números asignados a los sucesos elementales Ω son puntos aislados. Sus posibles valores constituyen un conjunto finito o infinito numerable. Por ejemplo, lanzar tres veces una moneda no trucada; la variable aleatoria X (número de caras obtenidas en los tres lanzamientos) puede tomar valores finitos (0,1,2,3).

- **Continuas**: la variable aleatoria será continua si los valores asignados pueden ser cualesquiera, dentro de ciertos intervalos, es decir, puede tomar cualquier valor en los $\mathbb{R}$. Por ejemplo, medir el nivel de agua en un envase, la variable X (nivel del agua) puede tomar valores entre 0 y más infinito.

2.2.2. Función de probabilidad

Sea X una variable aleatoria discreta que toma los valores $x_0, x_1, \cdots$ con probabilidades:

$$\begin{aligned} p_0 &= P(X = x_0), \\ p_1 &= P(X = x_1), \\ p_2 &= P(X = x_2), \\ &\cdots . \end{aligned}$$

Esta lista de valores numéricos y de probabilidades puede ser finita o infinita, pero numerable. La función de probabilidad de X se define como aquella función que toma estas probabilidades como valores. La Definición 4 introduce de manera formal el concepto de función de probabilidad [25].

Definición 4 *Sea X una variable aleatoria discreta con valores $x_0, x_1, \cdots$ La **función de probabilidad** de X, se denota por $f(x) : \mathbb{R} \to \mathbb{R}$, se define como sigue:*

$$f(x) = \begin{cases} P(X = x) & \text{si } x = x_0, x_1, \cdots \\ 0 & \text{otro caso} \end{cases} \tag{7}$$

En palabras, la función de probabilidad es simplemente aquella función que indica la probabilidad en los distintos valores que toma la variable aleatoria. Toda función de la forma (7) se le llama función de probabilidad y cumple las siguientes dos propiedades [25]:

a) $f(x) \geq 0$ para todo $x \in \mathbb{R}$

b) $\sum_x f(x) = 1$

2.2.3. Función de densidad

Definición 5 *Sea X una variable aleatoria continua. Decimos que la función integrable y no negativa $f(x) : \mathbb{R} \to \mathbb{R}$ es la **función de densidad** de X si para cualquier intervalo $[a, b]$ de $\mathbb{R}$ se cumple la igualdad*

$$P(X \in [a, b]) = \int_a^b f(x)dx \tag{8}$$

Es decir, la probabilidad de que la variable tome un valor dentro del intervalo $[a, b]$ se puede calcular o expresar como el área bajo la función $f(x)$ en dicho intervalo. La Figura 2.3 muestra esta forma de calcular probabilidades. Toda función de densidad $f(x)$ de una variable aleatoria continua cumple las siguientes propiedades análogas al caso discreto [25]:

a) $f(x) \geq 0$, para toda $x \in \mathbb{R}$

b) $\int_{-\infty}^{\infty} f(x)dx = 1$

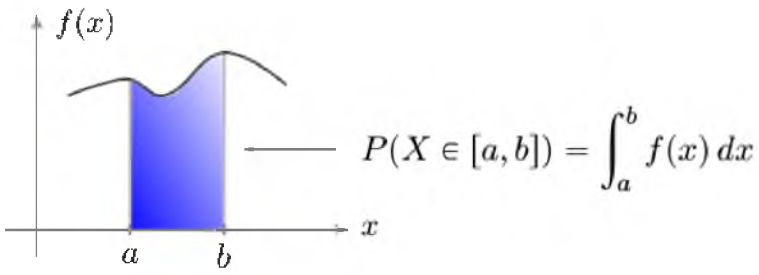


Figura 2.3: Representación gráfica de la función de densidad. Fuente: [25]

2.2.4. Función de distribución

Definición 6 *Sea X una variable aleatoria cualquiera. La **función de distribución** de X, denotada por $F(x) : \mathbb{R} \to \mathbb{R}$ se define como la probabilidad*

$$F(x) = P(X \leq x) \tag{9}$$

Esto es, la función de distribución $F(x)$ evaluada en el número x cualquiera es la probabilidad de que la variable aleatoria tome un valor menor o igual a x, o en otras palabras, que tome un valor en el intervalo $(-\infty, x]$. En el caso discreto suponiendo que $f(x)$ es la función de probabilidad de X, esta es la función de distribución acumulada y se calcula como sigue[25]:

$$F(x) = \sum_{u \leq x} f(u) \tag{10}$$

En el caso continuo, si $f(x)$ es la función de densidad de X, por función de distribución se tiene:

$$F(x) = \int_{-\infty}^{x} f(u)du \tag{11}$$

2.2.5. Distribución de probabilidad continua

Si la variable aleatoria es continua, hay infinitos valores posibles de la variable y entre cada dos de ellos se podrían definir infinitos valores. En estas condiciones no es posible deducir la probabilidad de un valor puntual de la variable como se puede hacer en el caso de las variables discretas, pero es posible calcular la probabilidad acumulada hasta cierto valor (**función de distribución**) y como cambia esta probabilidad acumulada en cada punto (**densidad de probabilidad**). Por lo tanto, cuando la variable aleatoria sea continua se habla de *función de densidad* [2].
Sea X una variable aleatoria continua, se llama *función de densidad* y se representa cono $f(x)$ a una función no negativa sobre la recta real, tal que para cualquier intervalo se verifica[2]:

$$\forall A \quad P(X \in A) = \int_A f(x)dx$$

2.2.6. Distribución uniforme discreta

Una variable aleatoria discreta X tiene una distribución uniforme discreta sobre el conjunto de n números $x_1, x_2, \cdots, x_n$ si la probabilidad de que X tome cualquiera de estos valores es constante: $1/n$. Esta distribución surge en espacios de probabilidad equiprobables, esto es, en situaciones en donde hay n resultados diferentes y todos ellos tienen la misma probabilidad de ocurrir. Se escribe $X \sim unif(x_1, x_2, \cdots, x_n)$ en donde el símbolo $\sim$ se lee " se distribuye como " [25]. La función de probabilidad de esta variable aleatoria es

$$f(x) = \begin{cases} \frac{1}{n} & \text{si } x = x_1, x_2, \cdots, x_n \\ 0 & \text{otro caso} \end{cases} \tag{12}$$

2.2.7. Distribución uniforme continua

Una variable aleatoria X tiene una distribución uniforme continua en el intervalo (a, b), y se escribe $X \sim unif(a, b)$, cuando su función de densidad es

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{si} x \in (a, b) \\ 0 & \text{otro caso.} \end{cases} \tag{13}$$

La gráfica general de esta función se muestra en la Figura 2.4(a). Integrando esta función de densidad desde menos infinito hasta un punto x cualquiera, puede encontrarse la función de distribución, la cual tiene la siguiente expresión (14) y cuya gráfica se muestra en la Figura 2.4(b)[25].

$$F(x) = \begin{cases} 0 & \text{si } x \leq a, \\ \frac{x-a}{b-a} & \text{si } a < x < b, \\ 1 & \text{si } x \geq b. \end{cases} \tag{14}$$

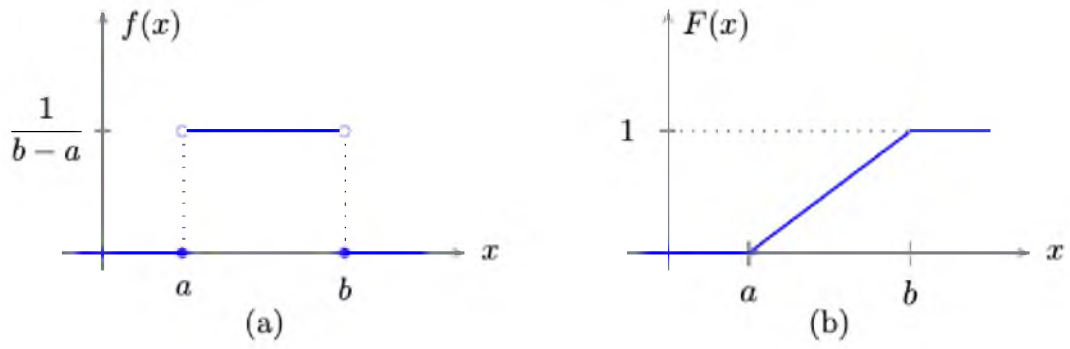


Figura 2.4: Distribución uniforme continua. Fuente: [25]

2.2.8. Distribución normal (o Gaussiana)

Esta es la distribución de probabilidad de mayor importancia. La variable aleatoria continua X tiene una *distribución normal* si su función de densidad esta dada por la siguiente expresión [25]:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{15}$$

en donde $\mu \in \mathbb{R}$ y $\sigma > 0$ son dos parámetros. La gráfica de esta función de densidad tiene forma de campana como se muestra en la Figura 2.5 y su correspondiente función de distribución es:

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{16}$$

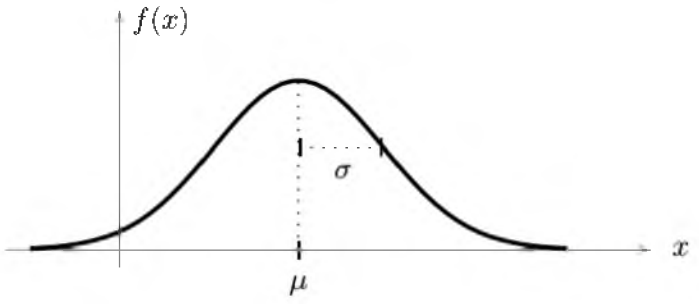


Figura 2.5: Función de densidad normal o Gaussiana. Se observa que la campana se abre o se cierra de acuerdo a la magnitud de la desviación estándar σ. Fuente: [25]

2.2.9. Esperanza

La esperanza de una variable aleatoria es un número que indica el promedio ponderado de los diferentes valores que la variable puede tomar. A la esperanza se le conoce también con los nombres de **media**, **valor esperado** o **valor promedio**. En general se usa también la letra griega μ (mu) para denotarla. La Definición 7 introduce de manera formal el concepto de esperanza [25].

Definición 7 *Sea X una variable aleatoria discreta con función de probabilidad $f(x)$. La **esperanza** de X se define como:*

$$E(X) = \sum_x x f(x). \tag{17}$$

Por otro lado, si X es continua con función de densidad $f(x)$, entonces la esperanza es

$$E(X) = \int_{-\infty}^{\infty} x f(x) dx. \tag{18}$$

2.2.10. Varianza

Otra característica numérica importante asociada a las variables aleatorias se llama **varianza**. La varianza es una medida del grado de dispersión de los diferentes valores tomados por la variable. Se denota regularmente por la letra σ^2. A la raíz cuadrada positiva de la varianza, esto es σ, se llama **desviación estándar**. La Definición 8 introduce de manera formal el concepto de varianza [25].

Definición 8 *Sea X una variable aleatoria discreta con función de probabilidad $f(x)$. La **varianza** de X se define como sigue*

$$Var(X) = \sum_x (x - \mu)^2 f(x), \tag{19}$$

cuando esta suma es convergente y en donde μ es la esperanza de X. Para una variable aleatoria continua X con función de densidad $f(x)$ se define

$$Var(x) = \int_{-\infty}^{\infty} (x - \mu)^2 f(x) dx. \tag{20}$$

2.3. Conceptos básicos de álgebra lineal

2.3.1. Espacios vectoriales

2.3.1.1. Espacio n dimensional

Si n es un entero positivo, entonces una **n-ada ordenada** es una sucesión de n números reales $(a_1, a_2, \cdots, a_n)$. El conjunto de todas las n-adas ordenadas se conoce como espacio n dimensional y se denota por $\mathbb{R}^n$ [4].

2.3.1.2. Combinación lineal

Se dice que un vector w es una combinación lineal de los vectores $v_1, v_2, \cdots, v_r$ si se puede expresar en la forma: $w = k_1v_1 + k_2v_2 + \cdots + k_rv_r$, donde $k_1, k_2, \cdots, k_r$ son escalares [4].

2.3.1.3. Conjuntos generadores

Si $v_1, v_2, \cdots, v_r$ son vectores en un espacio vectorial V y si todo vector en V es expresable como una combinación lineal de $v_1, v_2, \cdots, v_r$ entonces se dice que estos vectores generan a V [4].

2.3.2. Independencia lineal

Los conjuntos generadores resultan útiles en una gran diversidad de problemas ya que, a menudo, es posible estudiar un espacio vectorial V estudiando primero los vectores en un conjunto generador S y, a continuación, extendiendo los resultados hacia el resto de V. Por lo tanto, conviene mantener el conjunto generador S tan pequeño como sea posible. El problema de encontrar los conjuntos generadores más pequeños para un espacio vectorial depende de la noción de **independencia lineal**. Si $S = v_1, v_2, \cdots, v_r$ es un conjunto de vectores, entonces la ecuación vectorial

$$k_1v_1 + k_2v_2 + \cdots + k_rv_r = 0, \tag{21}$$

tiene al menos una solución, a saber, $k_1 = 0, k_2 = 0, \cdots, k_r = 0$. Si esta es la única solución, entonces S recibe el nombre de **conjunto linealmente independiente**. Si hay otras soluciones entonces se dice que S es un **conjunto linealmente dependiente**[4].

2.3.3. Bases y dimensiones

Definición 9 *Si V es cualquier espacio vectorial y $S = v_1, v_2, \cdots, v_r$ es un conjunto finito de vectores en V, entonces S se denomina una base para V si*

i) S es linealmente independientes, y

ii) S genera a V [4].

Ejemplo 1: [4] Sean $e_1 = (1,0,0,\cdots,0), e_2 = (0,1,0,\cdots,0), \cdots, e_n = (0,0,0,\cdots,1)$. $S = e_1, e_2, \cdots, e_n$ es un conjunto linealmente independiente en $\mathbb{R}^n$. Dado que cualquier vector $v = (v_1, v_2, \cdots, v_n)$ en $\mathbb{R}^n$ se puede escribir como $v = v_1e_1 + v_2e_2 + \cdots + v_ne_n$, S genera a $\mathbb{R}^n$, y por lo tanto, es una base. Esta se conoce como **base estándar** (o base natural, o base canónica) de $\mathbb{R}^n$.

Ejemplo 2: [15]

Una base de $\mathbb{R}^3$ distinta de la canónica: $(1,0,0), (1,1,0), (0,2,-3)$.

- Son linealmente independientes porque forman un determinante no nulo.
- Son sistemas generadores de $\mathbb{R}^3$ porque cualquier vector (a,b,c) se puede poner como combinación lineal de ellos. En efecto, dado (a,b,c), buscamos α, β, γ que satisfagan:
  $(a,b,c) = \alpha(1,0,0) + \beta(1,1,0) + \gamma(0,2,-3)$
  Se obtiene un sistema:

$$\begin{aligned} \alpha + \beta &= a, \\ \beta + 2\gamma &= b, \\ -3\gamma &= c. \end{aligned}$$

2.3.4. Eigenvectores y eigenvalores

La raíz **eigen** proviene del alemán y significa propio, por lo que los eigenvalores también reciben el nombre de **valores propios o valores característicos** de una transformación lineal o de una matriz, y los eigenvectores reciben el nombre de **vectores propios o vectores característicos** de una transformación lineal o una matriz. La Definición 10 introduce de manera formal los conceptos mencionados [15].

Definición 10 *Un **vector propio** de una matriz A de nxn es un vector **x** diferente de cero tal que A**x**=λ**x** para algún escalar λ. Un escalar λ se llama **valor propio** de A si existe una solución no trivial **x** de A**x**=λ**x**; una **x** como ésta se denomina vector propio correspondiente a λ.*

Ejemplo 1: [15]

Sea $A = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}$, **u**= $\begin{bmatrix} 6 \\ -5 \end{bmatrix}$ y **v**= $\begin{bmatrix} 3 \\ -2 \end{bmatrix}$ ¿son **u** y **v** vectores propios de A ?

Solución:

$$A_u = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix} \begin{bmatrix} 6 \\ -5 \end{bmatrix} = \begin{bmatrix} -24 \\ 20 \end{bmatrix} = -4 \begin{bmatrix} 6 \\ -5 \end{bmatrix} = -4u$$

$$A_v = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} -9 \\ 11 \end{bmatrix} \neq \lambda \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

Entonces **u** es un vector propio correspondiente a un valor propio (-4), pero **v** no es un vector propio de A porque A**v** no es un multiplo de **v**.

Ejemplo 2: [4]
Hállense los eigenvalores de la matriz

$$A = \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix}$$

Solución: Dado que

$$\lambda I - A = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} \lambda - 3 & -1 \\ 1 & \lambda \end{bmatrix},$$

el polinomio característico de A es

$$det(\lambda I - A) = det \begin{bmatrix} \lambda - 3 & -1 \\ 1 & \lambda \end{bmatrix} = \lambda^2 - 3\lambda + 2$$

y la ecuación característica de A es

$$\lambda^2 - 3\lambda + 2 = 0$$

Las soluciones de esta ecuación son $\lambda = 1$ y $\lambda = 2$; estos son los eigenvalores de A.

2.3.5. Producto interior, longitud y ortogonalización de vectores

2.3.5.1. Producto interior

Si **u** y **v** son vectores en $\mathbb{R}^n$, entonces **u** y **v** se consideran como matrices de $nx1$. u^T es una matriz de $1 \times n$ y el producto matricial $u^T v$ es una matriz de 1×1, la cual se escribe como un sólo número real (un escalar). A este número se le llama producto interior de **u** y **v**, y se escribe a menudo como **u**·**v**. También se conoce como **producto punto** [15].

2.3.5.2. La longitud de un vector

Si **v** esta en $\mathbb{R}^n$, con entradas $v_1, v_2, \cdots, v_n$, entonces la raíz cuadrada de **v**·**v** esta definida porque no es negativo. La Definición 11 introduce de manera formal el concepto de longitud de un vector [15].

Definición 11 *La longitud o norma de **v** es el escalar no negativo* $||v||$ *definido mediante*

$$||v|| = \sqrt{v \cdot v} = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}, \quad y \quad ||v||^2 = v \cdot v$$

2.3.5.3. Distancia en $\mathbb{R}^n$

Definición 12 *Para **u** y **v** en* $\mathbb{R}^n$*, la distancia entre **u** y **v**, escrita como dist(**u**, **v**), es la longitud del vector **u** - **v**, esto es [15]*

$$dist(u, v) = ||u - v||$$

2.3.5.4. Vectores ortogonales

Considere $\mathbb{R}^2$ y dos líneas que pasan por el origen determinadas mediante los vectores **u** y **v**. Las dos líneas que se muestran en la Figura 2.6 son geométricamente perpendiculares si, y sólo si, la distancia desde **u** hasta **v** es igual a la distancia desde **u** hasta **-v**. Esto es análogo a pedir que los cuadrados de las distancias sean iguales [15].

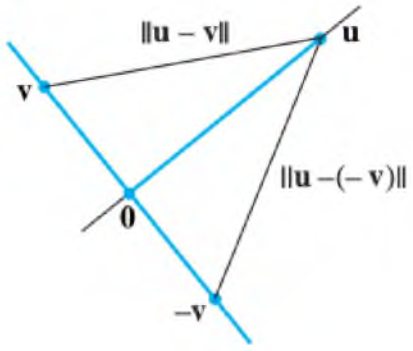


Figura 2.6: Representación gráfica del termino. Fuente: [15]

La Definición 13 generaliza a $\mathbb{R}^n$ esta noción de perpendicularidad (u *ortogonalidad*, como se le llama comúnmente en álgebra lineal).

Definición 13 *Dos vectores **u** y **v** en* $\mathbb{R}^n$ *son **ortogonales** (entre sí) si **u**·**v*** $= 0$.

2.3.5.5. Conjuntos ortogonales

Se dice que un conjunto de vectores $u_1, \cdots, u_p$ en $\mathbb{R}_n$ es un **conjunto ortogonal** si cada par de vectores distintos en el conjunto es ortogonal, esto es, si $u_i.u_j = 0$ siempre que $i \neq j$ [15].

Ejemplo 1: Muestre que $\{u_1, u_2, u_3\}$ es un conjunto ortogonal, donde

$$u_1 = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}, u_2 = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}, u_3 = \begin{bmatrix} -1/2 \\ -2 \\ 7/2 \end{bmatrix}.$$

Solución: Consideré los tres pares posibles de vectores, es decir, $\{u_1, u_2\}, \{u_1, u_3\}$ y $\{u_2, u_3\}$.

$$u_1.u_2 = 3(-1) + 1(2) + 1(1) = 0,$$
$$u_1.u_3 = 3(-1/2) + 1(-2) + 1(7/2) = 0,$$
$$u_2.u_3 = -1(-1/2) + 2(-2) + 1(7/2) = 0.$$

Cada par de vectores distintos es ortogonal, así que $\{u_1, u_2, u_3\}$ es un conjunto ortogonal

2.3.5.6. Conjuntos ortonormales

Un conjunto $\{u_1, \cdots, u_p\}$ es un **conjunto ortonormal** si es un conjunto ortogonal de vectores unitarios [15].

Ejemplo 1: Muestre que $\{v_1, v_2, v_3\}$ es una base ortonormal en $\mathbb{R}^3$, donde

$$v_1 = \begin{bmatrix} 3/\sqrt{11} \\ 1/\sqrt{11} \\ 1/\sqrt{11} \end{bmatrix}, v_2 = \begin{bmatrix} -1/\sqrt{6} \\ 2/\sqrt{6} \\ 1/\sqrt{6} \end{bmatrix}, v_3 = \begin{bmatrix} -1/\sqrt{66} \\ -4/\sqrt{66} \\ 7/\sqrt{66} \end{bmatrix}$$

Solución: Calcule

$$v_1.v_2 = -3/\sqrt{66} + 2/\sqrt{66} + 1/\sqrt{66},$$
$$v_1.v_3 = -3/\sqrt{726} - 4/\sqrt{726} + 7/\sqrt{726},$$
$$v_2.v_3 = 1/\sqrt{396} - 8/\sqrt{396} + 7/\sqrt{396}.$$

Entonces $\{v_1, v_2, v_3\}$ es un conjunto ortogonal. También,

$$v_1.v_1 = 9/11 + 1/11 + 1/11 = 1,$$
$$v_2.v_2 = 1/6 + 4/6 + 1/6 = 1,$$
$$v_3.v_3 = 1/66 + 16/66 + 46/66 = 1.$$

lo cual muestra que v_1, v_2, v_3 son vectores unitarios. Entonces $\{v_1, v_2, v_3\}$ es un conjunto ortonormal. Como el conjunto es linealmente independiente, sus tres vectores forman una base para $\mathbb{R}^3$.

2.3.6. El proceso Gram-Schmidt

2.3.6.1. Proyecciones ortogonales

Teorema 1 ***El teorema de la descomposición ortogonal***

Sea W un subespacio de $\mathbb{R}^n$. Entonces toda y en $\mathbb{R}^n$ puede escribirse únicamente en la forma

$$y = \hat{y} + z$$

donde $\hat{y}$ está en W y z en $W^{\perp}$[4]. De hecho, si $\{u_1, \cdots, u_p\}$ es cualquier base ortogonal de W, entonces:

$$\hat{y} = \frac{y \cdot u_1}{u_1 \cdot u_1} u_1 + \cdots + \frac{y \cdot u_p}{u_p \cdot u_p} u_p$$
$$y\ z = y - \hat{y}.$$

El vector $\hat{y}$ es la proyección ortogonal de **y** sobre W y a menudo se escribe como $proy_w y$ (ver la Figura 2.7).

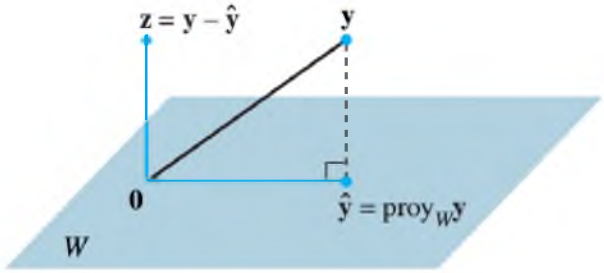


Figura 2.7: La proyección ortogonal de **y** sobre W. Fuente: [15]

2.3.6.2. El proceso Gram-Schmidt

El proceso Gram-Schmidt es un algoritmo sencillo para producir una base ortogonal u ortonormal para cualquier subespacio diferente de cero de $\mathbb{R}^n$ [15].

Teorema 2 ***El proceso Gram-Smichdt***

[4] Denota el conjunto de todos los vectores ortogonales a un subespacio W

Dada una base $\{x_1, \cdots x_p\}$ *para un subespacio W. en* $\mathbb{R}^n$*, defina*

$$
\begin{aligned}
v_1 &= x_1, \\
v_2 &= x_2 - \frac{x_2 \cdot v_1}{v_1 \cdot v_1} v_1, \\
v_3 &= x_3 - \frac{x_3 \cdot v_1}{v_1 \cdot v_1} v_1 - \frac{x_3 \cdot v_2}{v_2 \cdot v_2} v_2, \\
&\vdots \\
v_p &= x_p - \frac{x_p \cdot v_1}{v_1 \cdot u_1} u_1 - \frac{x_p \cdot v_2}{v_2 \cdot v_2} v_2 - \cdots - \frac{x_p \cdot v_{p-1}}{v_{p-1} \cdot v_{p-1}} v_{p-1}
\end{aligned}
$$

Entonces $\{x_1, \cdots x_p\}$ *es una base ortogonal para W. Además*

$$Gen\{v_1, \cdots v_k\} = Gen\{x_1, \cdots x_k\} \quad para \quad 1 \leq k \leq p$$

Ejemplo 1: Sean $x_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$, $x_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$, $x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$. Entonces. resulta claro que x_1, x_2, x_3 es linealmente independiente y, por lo tanto, es una base para el subespacio W en $\mathbb{R}^4$. Estructure una base ortogonal para w.

Solución:

Paso 1. Sean $v_1 = x_1$ y $W_1 = Genx_1 = Genv_1$.

Paso 2. Sea v_2 el vector producido al restar de x_2 su proyección sobre el subespacio W. Esto es, sea

$$
\begin{aligned}
v_2 &= x_2 - proy_{W_1} x_2 \\
&= x_2 - \frac{x_2.v_1}{v_1.v_1} v_1 \quad Puesto \quad que \quad v_1 = x_1 \\
&= \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{3}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -3/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}
\end{aligned}
$$

Paso 2'. Si es apropiado, escale v_2 para simplificar los cálculos posteriores.

Como v_2 tiene entradas fraccionarias, es conveniente calcularlo mediante un factor 4 y reemplazar a v_1, v_2 empleando la base ortogonal $v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, v_{2'} = \begin{bmatrix} -3 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

Paso 3. Sea v_3 el vector producido al restar de x_3 su proyección sobre el subespacio W_2. Utilice la base ortogonal $v_1, v_{2'}$ para calcular la proyección sobre W_2.

$$proy_{W_2}x_3 = \frac{x_3.v_1}{v_1.v_1}v_1 + \frac{x_3.v_{2'}}{v_{2'}.v_{2'}}v_{2'}$$
$$= \frac{2}{4}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} + \frac{2}{12}\begin{bmatrix}-3\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\2/3\\2/3\\2/3\end{bmatrix}$$

Entonces v_3 es la componente de x_3 ortogonal a W_2, a saber,

$$v_3 = x_3 - proy_{W_2}x_3 = \begin{bmatrix}0\\0\\1\\1\end{bmatrix} - \begin{bmatrix}0\\2/3\\2/3\\2/3\end{bmatrix} = \begin{bmatrix}0\\-2/3\\1/3\\1/3\end{bmatrix}$$

Entonces $v_1, v_{2'}, v_3$ es un subconjunto ortogonal de vectores diferentes de cero y, por lo tanto, un conjunto linealmente independiente en W.

Capítulo 3

Algoritmos de estimación de distribución para optimización en dominios continuos

El funcionamiento de algunos algoritmos de cómputo evolutivo tales como los algoritmos genéticos (GA) dependen de varios parámetros asociados a ellos (operadores de cruza y mutación, probabilidades de cruza y mutación, tamaño de la población, número de generaciones, etc.). Si el investigador no tiene experiencia en el uso de este tipo de enfoques para la solución de un problema de optimización, elegir los valores adecuados de los parámetros se convierte en si mismo en un problema de optimización. Esta razón ha motivado el nacimiento de un tipo de algoritmos conocidos como Algoritmos de Estimación de Distribuciones EDA (por sus siglas en inglés Estimation of Distribution Algorithms). Los EDA fueron introducidos en el campo del cómputo evolutivo por primera vez por Mühlenbein y PaaB (1996)[13].

En EDA no existen los operadores de cruza ni mutación, como en el caso de los GA. En su lugar la nueva población de individuos es muestreada de una distribución de probabilidad, la cual es estimada de una base de datos que contiene a los individuos seleccionados en la población anterior. Al mismo tiempo, mientras que en otras heurísticas de cómputo evolutivo la interrelación entre las variables que representan al individuo se mantienen en cuenta de manera implícita, en EDA las relaciones son representadas explícitamente a través de la distribución de probabilidad conjunta asociada con los individuos seleccionados en cada iteración[13].

3.1. Esquema general del EDA

Por lo general un EDA realiza los siguientes pasos[13]:

1. La población inicial D_0 de R individuos es generada. Los R individuos son generados asumiendo una distribución uniforme para cada variable.

2. Con el fin de que la $l-1^{th}$ población D_{l-1} evolucione hacia una siguiente D_l, un número de $N < R$ individuos son seleccionados desde D_{l-1} siguiendo un criterio. Denotado como D_{l-1}^N el conjunto de individuos seleccionados en la generación $l-1$.

3. El modelo probabilístico $n-$dimensional que mejor representa la interdependencia entre las n variables es inducido. Este paso también es conocido como el proceso de *aprendizaje*.

4. La nueva población D_l conformada por R nuevos individuos es obtenida por medio de una simulación de la distribución probabilística aprendida en el paso anterior. Usualmente se sigue un reemplazo elitista, por lo que el mejor individuo de la población D_{l-1}^N se mantiene en la población D_l.

Los pasos 2,3 y 4 son repetidos hasta que una condición de paro es verificada. Algunos ejemplos de condiciones de paro son: alcanzar un número fijo de poblaciones o un número fijo de evaluaciones de la función objetivo.

3.2. Clasificación de los EDA

Los EDA son clasificados usando una comparación análoga basada en la complejidad del modelo probabilístico usado para el aprendizaje de la interdependencia entre las variables de los individuos seleccionados. Para el caso del dominio continuo, la función de densidad será factorizada como un producto de n funciones de densidad condicional[13].

3.2.1. EDA sin dependencias

Los algoritmos no toman en cuenta las dependencias entre las variables. En este caso, la función de densidad conjunta es factorizada como un producto de n densidades unidimensional e independientes. Algunos EDA con este enfoque son[13]:

- Univariante Marginal Distribution Algorithm para el caso continuo ($UMDA_c$)[13].

- Stochastic Hill-Climbing with Learning by Vectors of Normal Distributions (SHCLVND)[26].
- Population-Based Incremental Learning for continuous domains ($PBIL_c$)[16].

3.2.2. EDA con múltiple interdependencia

En [13] se han propuesto varios enfoques de EDA en los cuales no hay restricción sobre el número de interdependencia entre variables a considerar en la función de densidad aprendida en cada generación. Las dependencias están codificadas en una matriz de covrianza o de precisión (inversa de la primera).

- Estimation of Multivariate Normal Algorithm (versiones: $EMNA_{global}$, $EMNA_{adaptative}$, $EMNA_{incremental}$)[13].
- Estimation of Gaussina Networks Algorithm (versiones: $EGNA_{ee}$, $EGNA_{BGe}$,$EGNA_{BIC}$)[13].

3.3. UMDAc- Univariate Marginal Distribution Algorithm

El Algoritmo de distribución marginal Univariado para dominios continuos $UMDA_c$ fue introducido por **Larrañaga et al** (1999-2000)[13]. En cada generación y para cada variable el $UMDA_c$ lleva a cabo pruebas estadísticas con el fin de encontrar la función de densidad que mejor se adapte a la variable. La factorización de la función de densidad conjunta esta dada por:

$$f_l(x;\theta^l) = \prod_{i=1}^{n} f_l(x_i;\theta_i^l) \tag{1}$$

Una vez que las densidades han sido identificadas, la estimación de los parámetros se realiza por sus estimaciones de máxima verosimilitud. Si todas las distribuciones univariadas son distribuciones normales, entonces los dos parámetros a ser estimados en cada generación y para cada variable son la media, μ_i^l, y la desviación estándar, σ_i^l. Sus respectivas estimaciones de máxima verosimilitud son:

$$\hat{\mu_i^l} = \overline{X_i^l} = \frac{1}{N}\sum_{r=1}^{N} x_{i,r}^l \tag{2}$$

$$\hat{\sigma_i^l} = \sqrt{\frac{1}{N}\sum_{r=1}^{N}(x_{i,r}^l - \overline{X_i^l})^2} \tag{3}$$

El Algoritmo 1 muestra el pseudocódigo para el aprendizaje de la función de densidad conjunta seguida por el $UMDA_c$.

Algoritmo 1 Algoritmo UMDAc.

1: $UMDA_c$
2: $*$ Aprendizaje de la función de densidad conjunta $*$
3: **while** $l = 1, 2, \cdots$ hasta que se verifique el criterio de paro **do**
4: **for** $i := 1$ to n **do**
5: Seleccionar por prueba de hipótesis la función de densidad $f_l(x_i; \theta_i^l)$ que mejor ajuste D_{l-1}^{Se,X_i}, en la proyección de los individuos seleccionados sobre la variable i^{th}.
6: Estimar la máxima verosimilitud para $\theta_i^l = (\theta_i^{l,k_i}, \cdots, \theta_i^{l,k_i})$ En cada generación la función de densidad conjunta aprendida es expresada como: $f_l(x; \theta^l) = \prod_{i=1}^{n} f_l(x_i; \hat{\theta_i^l})$.
7: **end for**
8: **end while**

3.4. EMNAglobal- Estimation multivariate normal algorithm

Está basado en la estimación de una función de densidad normal multivariada en cada generación. El pseudocódigo del $EMNA_{global}$ se muestran en el Algoritmo 2, como se observa, en cada generación se estima el vector de medias, $\mu_l = (\mu_{1,l}, \cdots, \mu_{n,l})$, y la matriz de varianza-covarianza $\sum_l$, cuyos elementos están denotados por $\sigma_{ij,l}^2$, con $i, j = 1, \cdots, n$. Esto significa que se necesita estimar $2n + \binom{n-1}{2}$ parámetros en cada generación: n medias, n varianzas y $\binom{n-1}{2}$ covarianzas.

Algoritmo 2 Algoritmo EMNAglobal.

1: $EMNA_{global}$
2: $D_0 \leftarrow$ Generar N individuos al azar (la población inicial)
3: **while** $l = 1, 2, \cdots$ hasta que se verifique el criterio de paro **do**
4: $D_{l-1}^N \leftarrow$ Seleccionar $N < R$ individuos, desde D_{l-1}, acorde al método de selección.
5: $f_l(x) = f(x|D_{l-1}^N) = N(x, \mu_l, \sum_l) \leftarrow$ Estimar la función de densidad normal Multivariada desde los individuos seleccionados.
6: $D_l \leftarrow$ Muestrear R individuos (la nueva población) a partir de $f_l(x)$.
7: **end while**

3.4.1. Reparación de matriz de covarianza

Matriz de covarianza mal planteada

Teoricamente se asegura que la matriz de covarianza ($\sum$) sea definida semi-positiva. Si existe un computador con una precisión infinita, entonces dado cualquier tipo de datos, siempre que se calcula la matriz de covarianza se obtiene un resultado definido semi-positivo. En la mayoria de los casos en simulación de datos reales, la matriz de covarianza es definida positiva. Si

$\exists i \in \{1, \cdots, n\}$, esto es, la fila i^{th} y la columna i^{th} de $\sum$ son todas cero, lo que significa que la búsqueda en la i^{th} dimensión ha convergido, entonces se pueden eliminar todos los ceros para formar una matriz de dimensión $n-1$ que sigue siendo positiva.
Es bien conocido que algunas veces la $\sum$ podría no ser definida semi-positiva (i.e. tiene eigenvalores negativos) porque la precisión del cálculo es finita. El error del cálculo aumenta cuando se calcula con muestras impares distribuidas, especialmente cuando el tamaño de la muestra es relativamente pequeño para el tamaño del problema [9].

Reparación de matriz de covarianza mal planteada (CMR) y eliminación de error acumulado

En EDA basados en la normal multivariada, la matriz de covarianza describe la forma de la distribución aprendida. Una forma de reparar la matriz de covarianza es añadiendo valores positivos a la diagonal de $\sum$. Se repara la matriz y se realiza al mismo tiempo modificaciones mínimas para preservar lo más posible la distribución previamente aprendida. Especialmente en la etapa convergente del EDA, la adición de un valor relativamente grande puede destruir la varianza casi convergida de una variable y dispersará los puntos de muestra en esa dimensión en la próxima generación. En [9] se toman los valores: valor absoluto del mínimo eigenvalor de $\sum$, multiplicado por un coeficiente k autoadaptado. El pseudocódigo de la matriz de covarianza reparada se muestra en el Algoritmo 3.

Algoritmo 3 Reparación de la matriz de covarianza.

CMR
$k \leftarrow 1$
while Condición **do**
  $\lambda \leftarrow$ Calcular el mínimo eigenvalor de la $\sum$
  if $\lambda \geq 0$ **then**
    $\sum$ es definida semi-positiva. Terminar ciclo.
  else
    Realizar una extensión como se definió anteriormente:
    $\sum \leftarrow \sum + |\lambda| \cdot k \cdot I$. Donde I es la matriz identidad.
  end if
  $k \leftarrow k \cdot \Delta$
end while

3.5. EEDA- Eigenspace estimation distribution algorithm

El algoritmo EMNAglobal convergerá prematuramente si no se inicializa con una distribución simétrica alrededor de la solución optima.
Suponiendo que se quiere minimizar la *funcion Sphere*, $F(x) = \sum_{i=1}^{n} x_i^2$, en n dimensiones. Considerando una distribución normal multivariada centrada lejos de la solución optima, se crea una población a partir de esta distribución y se selecciona una muestra de los mejores individuos respecto a la función objetivo. Se obtiene una muestra que es sesgada en el sentido de que hay poca variación en la dirección de minimización de la función objetivo. Una forma fácil de encontrar un conjunto finito de direcciones candidatas, con la cual se toma una dirección descendente, es calculando una descomposición de la matriz de covarianza de la muestra de los M mejores individuos [20].
EEDA también utiliza el modelo de la Gaussina multivariada, pero los mínimos eigenvalores de la matriz de covarianza deben ser reajustados al máximo eigenvalor, después de realizar la estimación de máxima verosimilitud[27].

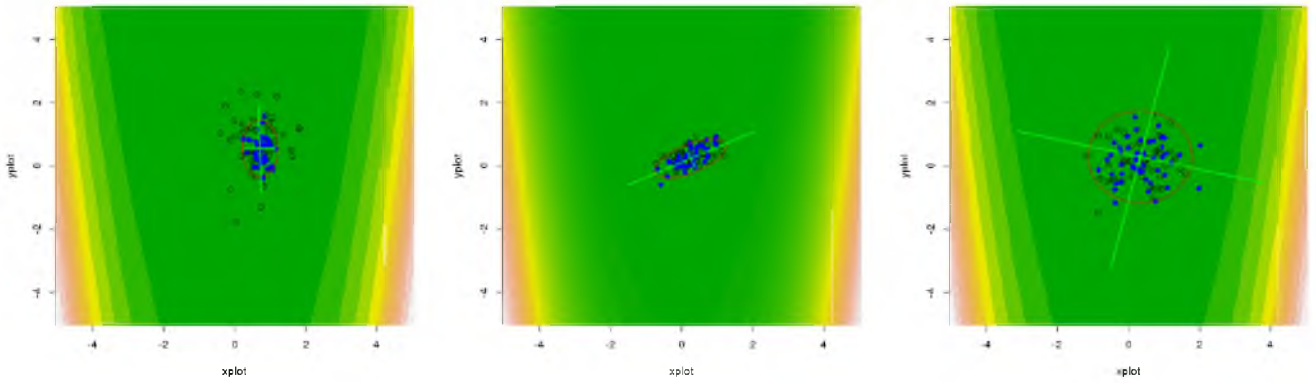
3.5.1. Modificación a EMNAglobal para crear EEDA

Una vez encontrado el eigenvector con el menor eigenvalor, se usa para modificar el EMNAglobal con el propósito de crear el Eigenspace EDA (EEDA). La mayor parte de la estructura del EMNAglobal se mantiene, solo se modifica el calculo la matriz de covarianza: **en cada iteración se redefine la matriz de covarianza muestreada extendiendo la original en dirección del eigenvector correspondiente al mínimo eigenvalor. En la descomposición de eigenvalores, se restablece el valor del mínimo eigenvalor al del máximo eigenvalor**. La descripción final de EEDA se muestra en el Algoritmo 4.

Algoritmo 4 Algoritmo EEDA.

1: $EEDA$
2: Generar una población de N individuos.
3: **while** $l = 1, 2, \cdots$ hasta que se verifique el criterio de paro **do**
4: En la generación $k+1$ seleccionar a los mejores $M < N$ individuos de la población de la generación anterior k.
5: Calcular la media μ^{k+1} y matriz de covarianza σ^{k+1} de los M individuos seleccionados.
6: **Calcular la descomposición de la matriz de covarianza** $\Sigma^{k+1} = V \Lambda V^T$.
7: **Realizar una extensión como se definio anteriormente:** $\Sigma^{k+1} = V \Lambda V^T + (\lambda_{max} - \lambda_{min}) v_{min} v_{min}^T$
8: Generar una nueva población de N individuos.
9: **end while**

En la Figura 3.1 se muestra el comportamiento de cada uno de los tres algoritmos: UMDA, EMNA y EEDA en el plano cartesiano. La matriz de covarianza describe la distribución general de esa forma elíptica en cada dirección. En el UMDA la función de probabilidad esta siempre alineada a las direcciones cartesianas, en el EMNA la matriz de covarianza ya se puede orientar, y el EEDA tiene orientación pero ademas se incrementa la búsqueda en cierta dirección (la del mínimo eigenvalor).



(a) UMDA (b) EMNA (c) EEDA

Figura 3.1: Imágenes del comportamiento de los algoritmos UMDA, EMNA y EEDA. Los puntos negros son los individuos a partir de los cuales se genera la matriz de covarianza (elipse roja) y los puntos azules son los nuevos individuos muestreados de la función de probabilidad.

3.6. Algoritmos de estado estable

Esta sección introduce el concepto de algoritmos de estado estable para contextualizar la nueva propuesta de un algoritmo de Estimación de Distribución con un enfoque de estado estable.

La implementación numérica de Algoritmos Evolutivos EA (por sus siglas en inglés, Evolutionary Algorithms) mantiene una población de posibles soluciones que evolucionan iterativamente según reglas de selección y otros operadores, como cruza y mutación. Como en cualquier técnica iterativa, un criterio de parada para terminar la implementación numérica del EA es obligatorio. En el caso de los métodos de optimización, el algoritmo debe detenerse en el momento en que ha alcanzado un estado estable en el que ya no puede mejorar los resultados[8].

El análisis de las propiedades de convergencia de EA aborda las condiciones que aseguran alcanzar el conjunto de individuos óptimos, independientemente de la población inicial. Dado los mecanismos estocásticos de los EA, tales propiedades generales deben darse en términos probabilísticos. En la práctica, cada ejecución del esquema iterativo de EA depende de una sola población inicial dada y consiste en una secuencia de estados de esta población. En este contexto y asumiendo que el estado de búsqueda es $\mathbb{R}^n$, la exploración del estado estable de la población de EA puede plantearse en términos de convergencia de secuencias de números reales[8].

Los Algoritmos Genéticos de estado estable difieren de los genéricos en que la selección de torneo no reemplaza a los individuos seleccionados en la población y en lugar de agregar a los hijos de los padres seleccionados a la siguiente generación, se añaden los dos mejores individuos entre los dos padres y dos hijos de nuevo a la población para que el tamaño de la población permanezca constante.

Una de las propuestas de esta tesis utiliza los conceptos anteriores, aunque no necesariamente funciona igual. Se propone un EDA de estado semi-estable, donde la generación cambia poco en cada generación y el algoritmo converge cuando difícilmente mejorará, en términos de probabilidad.

Capítulo 4

Propuestas de EDA

En este capítulo se describen dos propuestas de Algoritmos de Estimación de Distribución para dar solución a problemas no lineales de optimización continua, en los que se asume que se desconoce su función objetivo, no es ni convexa ni cóncava, y no esta sujeta a restricciones. La primer propuesta es un EDA univariado de estado semi estable, con estimadores pesados, y la segunda es un EDA multivariado que usa direcciones en la búsqueda del óptimo.

4.1. Propuesta 1: EDA univariado de estado semi estable con estimadores pesados

La propuesta en esta sección se denomina Algoritmo de Estimación de Distribución univariado de estado semi estable SSEDA (por sus siglas en inglés Steady-State Estimation of Distribution algorithm).

En esta propuesta no se considera la dependencia entre las variables, la función de densidad que se adapta para las variables es la normal (Gaussiana), y en cada iteración del algoritmo se calculan los estimadores pesados: media μ y desviación estándar σ. De acuerdo a la literatura (ver Sección 3.6), el enfoque de estado estable es usado en Algoritmos Genéticos pero, para EDA, aún no ha sido usado. En esta propuesta se usa este enfoque para utilizar toda la información de la población y de todas las muestras generadas durante todo el proceso de optimización, es decir, la población siempre va creciendo, para tener siempre mas información y tratar de aprovecharla al máximo para reducir el número de evaluaciones y mejorar la aproximación al óptimo. Una característica deseable del algoritmo es que la aproximación al óptimo sea suficientemente adecuada para resolver el problema y que las soluciones de varias ejecuciones sean similares.

En la industria existen problemas donde se requiere un número reducido de evaluaciones y en un tiempo relativamente corto, debido a que se quieren resultados en tiempo real no deberían de hacerse muchas evaluaciones. Por ejemplo, en el Capítulo 6 se muestra el caso de estudio de la detección de arcada temporal en la retina del ojo, donde se espera tener una respuesta en décima de segundos. Con esta propuesta se pueden resolver problemas de este tipo. Hacer pocas evaluaciones de la función objetivo significa hacer cientos o apenas millares de evaluaciones, todas las muestras son almacenadas (no son muchas) para aprovechar toda esta información en mejorar la estimación del óptimo. El Algoritmo 5 muestra el pseudocódigo de la nueva propuesta.

Algoritmo 5 EDA de estado semi estable(SSEDA) univariado.

1: $limites \leftarrow ObtenerLimites(f(x), k)$
2: $Pob^t \leftarrow IniciarPoblacion(f(x), k)$
3: $Fobj^t \leftarrow EvaluarPob(Pob, f(x))$
4: **while** $l = 1, 2, \cdots$ hasta que se verifique el criterio de paro **do**
5: $sort(Pob, Fobj)$
6: $Pi \leftarrow ProbTorneo()$
7: **if** $ValidarParo(Pi, Pob, limites)$ **then**
8: **BREAK**
9: **end if**
10: $(\mu, \sigma) = calcularEstimadores(Pi, Pob)$
11: $Pob^{t+1} \leftarrow Muestreo(n, \mu, \sigma, limites)$
12: $Fobj^{t+1} \leftarrow EvaluarPob(Pob_{new}, f(x))$
13: $Pob^t \leftarrow (Pob, Pob_{new})$
14: $Fobj^t \leftarrow (Fobj, Fobj_{new})$
15: **end while**

El algoritmo tiene cuatro componentes principales: *obtener limites*, que genera los espacios de búsqueda de cada variable continua del problema; *inicialización*, crea un conjunto inicial de soluciones; *estimación*, se establecen los valores para los estimadores: media y desviación estándar a partir de la población; y *muestreo*, genera nuevas soluciones a partir de muestras normales con la media y la desviación estándar estimadas. Los componentes mencionados están diseñados de la siguiente forma:

- *Obtener límites*: se genera el dominio de la solución.
- *Inicialización*: el algoritmo inicia generando k soluciones (guardadas en Pob^t) con una distribución uniforme, considerando los limites del dominio de búsqueda.

- *Estimación*: en cada iteración se genera un número n de individuos (Pob^{t+1}), a partir de población anterior (Pob^t). El proceso de estimación se realiza de la siguiente manera: para cada variable x_i, se calcula una probabilidad de selección Pi para cada individuo de la población a través de asignarles pesos (el cual es directamente proporcional a la posición después de ordenar Pob^t por valores crecientes de la función objetivo), y con ello se calculan los estimadores pesados: media y desviación estándar. Las respectivas estimaciones son:

$$\begin{aligned} P_i = P(k \epsilon S) = \frac{wi}{\sum N} = \frac{2w_i}{N(N+1)} \quad & w_i \quad es \quad el \quad indice \quad del \quad individuo \quad ordenado. \\ \mu &= \sum X_i P_i \\ \sigma^2 &= \sum P_i(\mu - X_i)^2 \\ \sigma &= \sqrt{\sigma^2} \end{aligned} \tag{1}$$

- *Muestreo*: son generados nuevos individuos a partir de muestras normales con la media y la desviación estándar dada anteriormente.

4.1.1. SSEDAw y SSEDAwe

En este apartado se introducen dos métodos para el calculo de los pesos usados en la línea 6 del Algoritmo 5. El primer método, llamado EDA univariado de estado semi estable con estimadores pesados (SSEDAw), consiste en asignar un peso directamente proporcional a la posición después de ordenar la población antes evaluada por valores crecientes de la función objetivo, el segundo método llamado EDA univariado de estado semi estable con estimadores pesado elevado a un exponente (SSEDAwe) consiste en elevar los pesos del primer método a un exponente igual al número de iteración del algoritmo.

4.2. Propuesta 2: EDA multivariado

Se propone un EDA multivariado, que utiliza una distribución normal multivariada para el paso de muestreo. La propuesta en esta sección se denomina Algoritmo de estimación de Distribución con Direcciones EDAD (por sus siglas en inglés Estimation of Distribution algorithm whit Directions). El Algoritmo 6 muestra el pseudocódigo de la nueva propuesta del algoritmo.

Algoritmo 6 Algoritmo EDAD.

1: $X^t \leftarrow$ generarPob(nPob,nVar)
2: $nselect \leftarrow nPob/2$
3: $F^t \leftarrow$ Eval(X^t)
4: **while** $l = 1, 2, \cdots$ hasta que se verifique el criterio de paro **do**
5: $[X_{best}, f_{best}] \leftarrow$ Elite(X^t,F^t)
6: $S^t \leftarrow$ Select(F^t)
7: $covarianza \leftarrow build\Sigma(S^t, X^t)$
8: $media \leftarrow X_{best}$
9: $X_{nselect} \leftarrow$ nBestSelect($nselect$,S^t)
10: $X_{new} \leftarrow$ DensMvNorm($nselect$,$media$,$covarianza$)
11: $X^{t+1} \leftarrow$ Replacement($X_{nselect}$, X_{new})
12: $F_{nselect} \leftarrow$ nBestSelect($nselect$, F^t)
13: $F_{new} \leftarrow$ Eval(X_{new})
14: $F^{t+1} \leftarrow$ Replacement($F_{nselect}$, F_{new})
15: **end while**

Descripción del algoritmo:

1. Generar una población de *nPob* individuos con una distribución uniforme, cada uno de los cuales tiene un número *nVar* de variables.

2. La variable declarada *nselect* es la cantidad de los mejores individuos a seleccionar.

3. **Eval** es una función que evalua a cada individuo de la población, los resultados se guardan en el vector F^t. Esta evaluación es conocida como la *función objetivo*.

4. **Elite** es una función que obtiene el mejor valor de la función evaluada f_{best} y al individuo X_{best} que generó ese f_{best}.

5. **Select** es la función que ordena de menor a mayor los valores de la función objetivo contenidos en F^t y obtiene las posiciones que tienen los individuos dentro de la población; S^t guarda en orden aleatorio las posiciones antes obtenidas.

6. $build\Sigma$ es la función que realiza la construcción de la matriz de covarianza. Para la construcción de esta matriz se consideran las distancias entre los primeros mejores D_{best} individuos y los siguientes mejores $D_{sigBest}$, donde:

   - D_{best} es un vector de tamaño nVar con los primeros mejores individuos y
   - $D_{sigBest}$ es un vector de tamaño nVar con los siguientes mejores individuos, tomados de (*nselect*-nVar) hacia atrás.

Como resultado de lo anterior se obtiene un vector de distancias el cual se normaliza y a partir de este se aplica el método de ortonormalización de GramSchmidt para obtener una base, que se tomaran como los eigenvectores (e) de la matriz de covarianza. Los eigenvalores se obtienen a partir del vector de distancias normalizado y un parámetro de distribución chi-cuadrada (chi^2). Así, la construcción de la matriz de covarianza esta dada por la multiplicación entre la matriz de eigenvectores, la matriz diagonal de eigenvalores (Λ) y la inversa de la matriz de eigenvectores ($\Sigma = e * \Lambda * e^{-1}$). La matriz de covarianza es de dimensión nVar por nVar.

7. La media (μ) esta definida como el mejor individuo X_{best}.

8. **nBestSelect** es una función que obtiene a partir del vector S^t a los $nselect$ mejores individuos y a los mejores $nselect$ valores de la función objetivo, guardándolos en $X_{nselect}$ y $F_{nselect}$ respectivamente.

9. **DensMvNorm** es una función que genera a los $nselect$ nuevos individuos simulando muestras de una normal multivariada con la media y la matriz de covarianza anteriormente calculadas.

10. **Replacement** es una función que sustituye la población actual con los nuevos individuos X^{t+1} y a partir de éstos se hacen nuevas evaluaciones de la función objetivo F_{t+1}.

Capítulo 5

Experimentos y resultados

En este capítulo se describen el conjunto de funciones tomadas como benchmark para comparar resultados obtenidos por la propuesta del Algoritmo 6 y los obtenidos con la implementación de los algoritmos UMDA, EMNA y EEDA.

5.1. Problemas de optimización

En [13] se citan algunas funciones de optimización continua utilizadas para probar el funcionamiento de algunos EDA. En este apartado estas mismas funciones serán útiles para probar el rendimiento de la propuestas del Algoritmo 6 y los algoritmos implementados,UMDA, EMNA y EEDA. A continuación se describen dichas funciones:

- *Sphere model*: generalmente conocido como un problema de minimización simple. Se define de modo que $-600 \leq x_i \leq 600, \quad i = 1, \cdots, n$, y

$$F(x) = \sum_{i=1}^{n} x_i^2 \tag{1}$$

- *Griewangk*: Problema de minimización propuesto por Ton y Zilinskas. La función es la siguiente:

$$F(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}). \tag{2}$$

  El rango de los componentes de los individuos es $-600 \leq x_i \leq 600, \quad i = 1, \cdots, n$, y el óptimo corresponde a un valor de 0.

- *Rosenbrock generalized*: Este es un problema de minimización propuesto en Rosenbrock (1960). Se define como:

$$F(x) = \sum_{i=1}^{n-1} 100 * (x_{i+1} - x_i^2)^2 + (1 - x_i)^2. \tag{3}$$

  El valor óptimo es 0. El rango de los valores es $-10 \leq x_i \leq 10, \quad i = 1, \cdots, n.$

- *Ackley*: Este problema de optimización fue propuesto en Ackley (1987), y es un problema de minimización en el cual el mejor valor es 0. Originalmente este problema fue definido para dos dimensiones, pero ha sido generalizado a n dimensiones (Bäck 1996). La función se calcula como sigue:

$$F(x) = -20.exp(-0.2\sqrt{\frac{1}{n}.\sum_{i=1}^{n} x_i^2} \quad) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 - exp(1). \tag{4}$$

En [20] las funciones empleadas para mostrar el funcionamiento del algoritmo EEDA se describen a continuación. Éstas mismas serán utilizadas para probar la implementación de EEDA en este trabajo.

1. Problema de maximización. El valor óptimo de la función es $f_1(x^*) = 10^7$.

$$f_1(x) = \frac{100}{10^{-5} + \sum_{i=1}^{n} |y_i|}, \tag{5}$$

   donde $y_i = (x_i - i + 1) + y_{i-1}$, $y_1 = x_1$.

2. Problema de minimización. El valor óptimo de la función es $f_2(x^*) = 0$

$$f_1(x) = \sum_{i=1}^{n} (x_i - i + 1)^2. \tag{6}$$

3. Problema de minimización. El valor óptimo de la función es $f_3(x^*) = 0$

$$f_1(x) = 1 + \sum_{i=1}^{n} (x_i - i + 1)^2 - \prod_{i=1}^{n} \cos(\frac{x_i - i + 1}{\sqrt{i+1}}). \tag{7}$$

5.2. Descripción de los experimentos

Los algoritmos fueron implementados en lenguaje **R**[1], la población inicial fue generada usando un proceso aleatorio con una distribución uniforme y los parámetros, para el UMDA, EMNA y la propuesta dos (llamada EDAD en las Tablas 5.3 y 5.2), son los siguientes:

[1] Lenguaje de programación para análisis estadístico y gráfico.

- Población inicial de 2000 individuos,
- una función objetivo,
- dimensión de 10 variables para cada individuo de la población,
- como criterios de paro: 300 mil evaluaciones de la función objetivo
- y desviación estándar de 10^{-6}.
- Para los casos del EEDA, los parámetros que cambian en el experimento son:
  - Población inicial de $40, 400, 4000$,
  - criterio de paro 100 mil evaluaciones de la función objetivo,

Otro experimento que se realizo fue sobre las funciones *Rosenbrock* y *Griewangk* con los algoritmos UMDA, EMNA y EEDA, para mostrar la convergencia al óptimo cuando la población se sitúa fuera del espacio donde se encuentra el óptimo. Se realizaron varias ejecuciones de los algoritmos para una población de 2000 individuos con 10 variables. En cada iteración se generaron n-1 nuevos individuos, ver resultados en las Tablas 5.4 y 5.5.

También se realizo una comparación entre resultados obtenidos con la implementación del EEDA y los de la literatura[20]. El algoritmo se ejecutó 20 veces, el espacio de búsqueda se situó alejado del óptimo ($100 < x < 150$). También se muestran resultados obtenidos con el Algoritmo 6 bajo las mismas condiciones de esta prueba, ver resultados en Tabla 5.3.

5.3. Resultados y comentarios

En las Tablas 5.1, 5.2 y 5.3 se muestran los resultados de la media y la desviación estándar de la mejor aproximación al óptimo de la función en diez ejecuciones de cada uno de los algoritmos para las funciones descritas en la sección 5.1. Los resultados experimentales muestran diferencias importantes entre los algoritmos dependiendo del problema de optimización.

En la Tabla 5.2 se observa que para una dimensión de 50 el Algoritmo 6 obtiene los peores resultados, por lo que no es comparable en esta dimensión contra los algoritmos UMDA, EMNA y EEDA. Por otra parte se observa que para las funciones Sphere model y Ackley obtiene los mejores resultado en dimensión 10 . En la Tabla 5.1 se hace una comparación entre los resultados de

la implementación de los algoritmos UMDA y EMNA con los resultados que se tienen en la literatura, donde se observa que los resultados de la implementación de los algoritmos es mejor que los de la literatura. Por último, en la Tabla 5.3 se muestran los resultados obtenidos con el algoritmo EEDA y el propuesto (EDAD), comparando los resultados del EEDA con los de la literatura. Para la tercer función los resultados del EEDA, los de la literatura y los del EDAD son muy parecidos en las dimensiones 40 y 400.

Con los resultados gráficos que se muestran en las Tablas 5.4 y 5.5, se observa que el algoritmo EEDA a diferencia del EMNA si converge a un óptimo y que en contraste con los resultados obtenidos con el UMDA, el EEDA converge con mayor rapidez. Ésto sucede cuando el espacio de búsqueda se sitúa fuera del espacio donde se encuentra el valor óptimo de las funciones Rosenbrock y Griewangk.

Funciones	Algoritmo	Dim.	implementación propia		Resultados de la literatura	
			Media Fobj	No. eval.	Media Fobj	No. eval.
Sphere model	**UMDAc**	10	**6.61219e-23**	300000	$7.7360e-06$	74163.9
	$UMDA_c$	50	$6.57797e-06$	300000	$8.9113e-06$	211495.2
	$EMNA_g$	10	5.56384e-10	300000	$7.3350e-06$	94353.8
	$EMNA_g$	50	$3.498321e-06$	300000	$8.5225e-06$	247477.2
Griewangk	**UMDAc**	10	**0**	300000	$6.0783e-02$	301850
	$UMDA_c$	50	8.27814e-07	300000	$8.9869e-06$	177912
	$EMNA_g$	10	1.09346e-10	300000	$5.1166e-02$	301850
	$EMNA_g$	50	1.14119e-07	300000	$8.7673-06$	216292
Rosenbrock generalized	$UMDA_c$	10	8.04592	300000	8.7204	301850
	$UMDA_c$	50	38.53004	300000	48.8949	301850
	$EMNA_g$	10	81.31921	300000	8.7201	289056.4
	$EMNA_g$	50	1475.91415	300000	49.7588	296252.8
Ackley	**UMDAc**	10	**1.94777e-13**	300000	$7.8784e-06$	114943.5
	$UMDA_c$	50	$5.28344e-05$	300000	$9.0848e-06$	296852.5
	$EMNA_g$	10	$3.00644e-05$	300000	$8.9265e-06$	119141.4
	$EMNA_g$	50	0.00010	300000	$9.5926E-06$	291255.3

Cuadro 5.1: Cuadro comparativo entre resultados obtenidos y los encontrados en la literatura[13], para los algoritmos UMDA y EMNA. En negritas se muestra el algoritmo que obtuvo en promedio los mejores resultados.

Función	Dim.	Algoritmo	Media Fobj	Ds Fobj	No. eval.
Sphere model	10	$UMDA_c$	$6.61219e-23$	$1.48327e-23$	300000
	50	$UMDA_c$	$6.57797e-06$	$6.66853e-07$	300000
	10	$EMNA_g$	$5.56384e-10$	1.04481	300000
	50	$EMNA_g$	$3.498321e-06$	$4.36585e-07$	300000
	10	$EEDA$	$7.61840e-21$	$1.74998e-21$	300000
	50	$EEDA$	0.00023	$3.36066e-05$	300000
	10	**EDAD**	**6.54773e − 28**	**2.08992e − 28**	**148000**
	50	$EDAD$	3398748.81472	354857.61270	300000
Griewangk	**10**	**$UMDA_c$**	**0**	**0**	**300000**
	50	$UMDA_c$	$8.27814e-07$	$6.28469e-08$	300000
	10	$EMNA_g$	$1.09346e-10$	1.98169	300000
	50	$EMNA_g$	$1.14119e-07$	$.12657e-08$	300000
	10	$EEDA$	$5.08394e-10$	$9.92142e-11$	300000
	50	$EEDA$	0.08846	0.01677	300000
	10	$EDAD$	0.57846	0.11533	300000
	50	$EDAD$	831.13467	111.63566	300000
Rosenbrock generalized	10	$UMDA_c$	8.04592	0.95497	300000
	50	$UMDA_c$	38.53004	2.46373	300000
	10	$EMNA_g$	81.31921	17.10853	300000
	50	$EMNA_g$	1475.91415	89.09993	300000
	10	**EEDA**	**0.02047**	**0.01150**	**300000**
	50	$EEDA$	7.4421	1.47289	300000
	10	$EDAD$	130.44064	337.29334	300000
	50	$EDAD$	4257706.83587	353150.92770	300000
Ackley	10	$UMDA_c$	$1.94777e-13$	$3.95631e-14$	300000
	50	$UMDA_c$	$5.28344e-05$	$3.18752e-06$	300000
	10	$EMNA_g$	$3.00644e-05$	$1.78516e-06$	300000
	50	$EMNA_g$	0.00010	$2.55304e-06$	300000
	10	$EEDA$	$2.75202e-12$	$3.91885e-13$	300000
	50	$EEDA$	0.00016	$1.18326e-05$	300000
	10	**EDAD**	**3.99680e − 15**	**0**	**300000**
	50	$EDAD$	13.30592	0.26938	300000

Cuadro 5.2: Resultados obtenidos con los algoritmos UMDAc, EMNAglobal, EEDA y el Algoritmo 6. En negritas se muestran los algoritmos con los que se obtuvieron, en promedio, los mejores resultados para cada función.

Funciones	Alg.	Pobl.	implementación propia		Resultados de la literatura	
			μ Fobj	μ Ds Fobj	μ Fobj	μ Ds Fobj
$f_1(x)$ 5	$EEDA$	40	2461.21323	3932.75679	36.230	5949.895
	$EEDA$	400	137208.177	221328.1196	1.772	1.053
	$EEDA$	4000	0.03377	0.00326	0.0207	$5.325e-05$
	$EDAD$	40	243846.91887	467342.02268	-	-
	$EDAD$	400	3131196.14815	10820146.96515	-	-
	$EDAD$	4000	11005.94524	23527.44234	-	-
$f_2(x)$ 6	$EEDA$	40	$4.49897-32$	$1.87029e-31$	0	$5.242e-34$
	$EEDA$	400	$1.41748e-31$	$2.57575e-31$	$1.856e-23$	$1.808e-22$
	$EEDA$	4000	8450.93357	939.55329	76143.3	343.002
	$EDAD$	40	$7.79822e-28$	$1.65807e-28$	-	-
	$EDAD$	400	$6.78074e-28$	$1.29165e-28$	-	-
	$EDAD$	4000	92.20243	286.05768	-	-
$f_3(x)$ 7	$EEDA$	40	0	0	0	0
	$EEDA$	400	0	0	0	0
	$EEDA$	4000	8563.10306	1088.11989	76080.8	295.580
	$EDAD$	40	0	0	-	-
	$EDAD$	400	0	0	-	-
	$EDAD$	4000	91.49851	282.26354	-	-

Cuadro 5.3: Tabla comparativa entre los resultados obtenidos con el EEDA y el Algoritmo 6, y los de la literatura.

5.4. Experimento con paquete Black-Box Optimization Benchmarking

En esta sección se comparan los algoritmos UMDA, EMNA y EEDA utilizando el paquete Black-Box Optimization Benchmarking (BBOB), con el fin de demostrar el desempeño de algoritmos que usan direcciones de búsqueda o que usan dependencias. El paquete BBOB proporciona un banco de pruebas de 24 funciones sin ruido. La descripción de dichas funciones se encuentran en el sitio oficial COCO[2] (Comparing Continuous Optimisers). En cada función y para cada dimensión n-pruebas son llevadas a cabo [23].

Entradas al algoritmo e inicialización

Para los algoritmos UMDA, EMNA y EEDA se usaron las siguientes parámetros:

- D es la dimensionalidad de espacio de búsqueda utilizada para todas las funciones. En la implementación se utilizan $D = \{2, 5\}$

[2]Plataforma para comparaciones sistémicas y ruidosas de optimizadores globales con parámetros reales.

- El dominio de búsqueda: todas las funciones de BBOB estan definidas en $\mathbb{R}^D$ y tienen su óptimo global en $[5, -5]^D$. La mayoría de las funciones BBOB tienen su óptimo global en el rango $[4, -4]^D$.

- Indicar el conjunto de datos sobre consideraciones, i.e. pueden ser usados distintos algoritmos y/o parámetros para el banco de pruebas libre de ruido. Se utilizaron el algoritmo UMDA, EMNA y EEDA.

- Delta-value $\Delta f = 10^{-8}$ es la precisión a alcanzar, es decir, una diferencia con menor valor posible de la función f_{opt}.

- $f_{target} = f_{opt} + \Delta f$ es valor de la función objetivo mas pequeño; sólo proporciona la terminación de las pruebas, a fin de reducir los requerimientos generales del CPU. El valor de la función objetivo puede no ser usada como entrada del algoritmo.

Las 24 funciones con las que cuenta el paquete BBOB se encuentran agrupadas de acuerdo a sus características en los siguientes grupos:

- Funciones separables: quiere decir pueden dividirse en suma de otras funciones que dependen de subconjuntos disjuntos de las variables, es decir, si puede escribir la función: $f(x_1, x_2, x_3) = g(x_1) + h(x_2, x_3)$, entonces $f(x_1, x_2, x_3)$ es separable. Estas funciones se consideran más fáciles de optimizar que las no separables porque podrían buscar por separado el mínimo de $g(x_1)$ y $h(x_2, x_3)$, y la suma de esos dos mínimos es el de $f(x_1, x_2, x_3)$.

- Funciones con acondicionamiento bajo o moderado: en estas funciones es necesario incluir dependencia entre las variables para hacer una búsqueda adecuada del óptimo.

- Funciones con acondicionamiento alto y unimodal: estas funciones tiene un solo mínimo y máximo global.

- Funciones multimodal con estructura global adecuada: estas funciones tienen muchos mínimos y máximos locales y un solo óptimo global.

- Funciones multimodal con estructura global débil: estas funciones tienen muchos mínimos y máximos locales, difiere de las anteriores en que el óptimo global es más difícil de encontrar.

5.4.1. Resultados y comentarios

En las Tablas 5.6 a 5.10 se muestran los resultados obtenidos con los algoritmos UMDA, EMNA y EEDA para las 24 funciones que proporciona el paquete BBOB. Para la función Sphere los tres algoritmos aproximan un mismo óptimo, esto se debe a que en esta función tiene un único óptimo global. Para las funciones donde las variables tienen dependencias entre sí, el EMNA y EEDA aproximan un mejor óptimo pero solo para 2 dimensiones. Para las funciones unimodales en las que hay dependencia entre sus variables, los tres algoritmos obtuvieron aproximaciones a un óptimo parecidas en la dimensión 2, pero para la dimensión 5 solo el algoritmo EMNA mostró mejores aproximaciones al óptimo para 3 de 5 funciones. En el caso donde las funciones son multimodales, para 6 de 10 funciones los tres algoritmos aproximan al óptimo, se puede decir que es debido a que el EMNA y EEDA consideran que sus variables tienen dependencia entre si. Con el UMDA la aproximación al óptimo es buena pero no mejor que para los otros dos para algunas funciones pertenecientes a este conjunto.

Como conclusión, los resultados muestran que los algoritmos UMDA, EMNA y EEDA tienen buenos resultados para funciones que tienen las características para las cuales fueron desarrollados los algoritmos.

5.5. Experimentos y pruebas para el EDAD

En esta sección se quiere probar la competitividad del Algoritmo 6 donde se propone el uso de direcciones en la búsqueda del óptimo, contra los algoritmos UMDA, EMNA y EEDA. Y encontrar una función que describa el tamaño adecuado de la población con respecto al número de variables, para garantizar una aproximación al óptimo.

5.5.1. Descripción de los Experimentos

Para probar el rendimiento del Algoritmo 6 se llevaron a cabo las siguientes pruebas:

Prueba 1 Esta prueba consiste en aproximar el óptimo de las funciones descritas en la Sección 5.1 con el nuevo algoritmo, para observar que los resultados obtenidos se asemejen al de los algoritmos UMDA, EMNA y EEDA implementados en este trabajo. En las Tablas 5.2 y 5.3 se muestran los resultados.

Prueba 2 Consiste en ejecutar el nuevo algoritmo 5 veces con distintos tamaños de población y distintos números de variables. De las cinco ejecuciones se obtuvieron: la media, la mediana, el mínimo y máximo valor de la función objetivo, los resultados se muestran en la Tabla 5.11. El objetivo de esta prueba es encontrar el tamaño de la población adecuada en función del número de variables; lo anterior no garantiza que siempre converja al óptimo debido a su base estocástica. Para encontrar una función que describa el tamaño adecuado de la población con respecto del número de variables se implemento el método de regresión lineal por mínimos cuadrados. La Figura 5.1 muestra esta relación. Los parámetros usados para esta prueba son los siguientes:

- Las funciones a minimizar: función Rosenbrock y Griewangk,
- el número de variables varía, dimensión=2,4,8,10,16 y 20,
- el tamaño de la población=10*dimensión, 20*dimensión, 40*dimensión, 60*dimensión, 80*dimensión y 100*dimensión,
- y como criterios de paro: 300 mil evaluaciones de la función objetivo y una tolerancia de $1e^{-20}$.

Dims.	Pobs.	μ Fobj	μ_c Fobj	Min. Fobj	Max. Fobj
2	80	4.37853e-21	2.97146e-21	1.20937e-21	9.27981e-21
4	80	7.46098e-01	2.70909e-13	1.81489e-15	3.70143e+00
6	60	1.63150e+00	4.34507e-05	2.97574e-05	4.18346e+00
8	80	9.01149e-01	3.76237e-05	1.61435e-05	4.50559e+00
10	100	1.70355e+01	1.42002e-04	1.29119e-04	8.51613e+01
16	160	9.87028e+00	2.17372e-01	1.52095e-01	4.85231e+01
20	200	2.38398e+01	1.05055e+01	7.44742e+00	8.11986e+01

Cuadro 5.11: Valores de la función objetivo obtenidos en la **prueba 2** para la función Rosenbrock. Se muestran el valor objetivo de cada dimensión con sus respectivos tamaños de población.

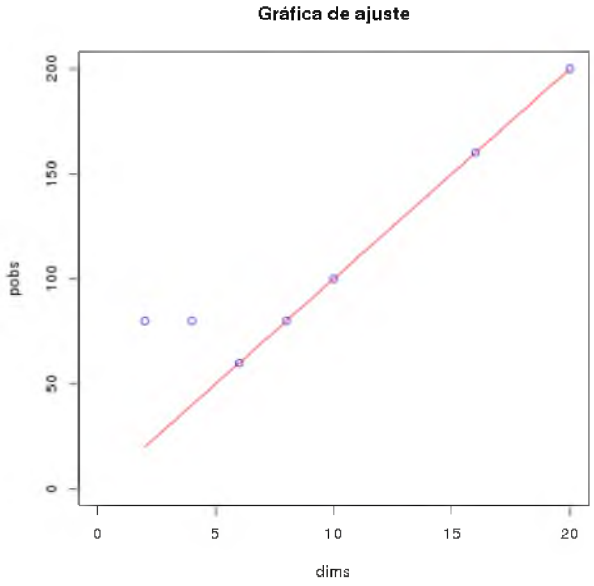
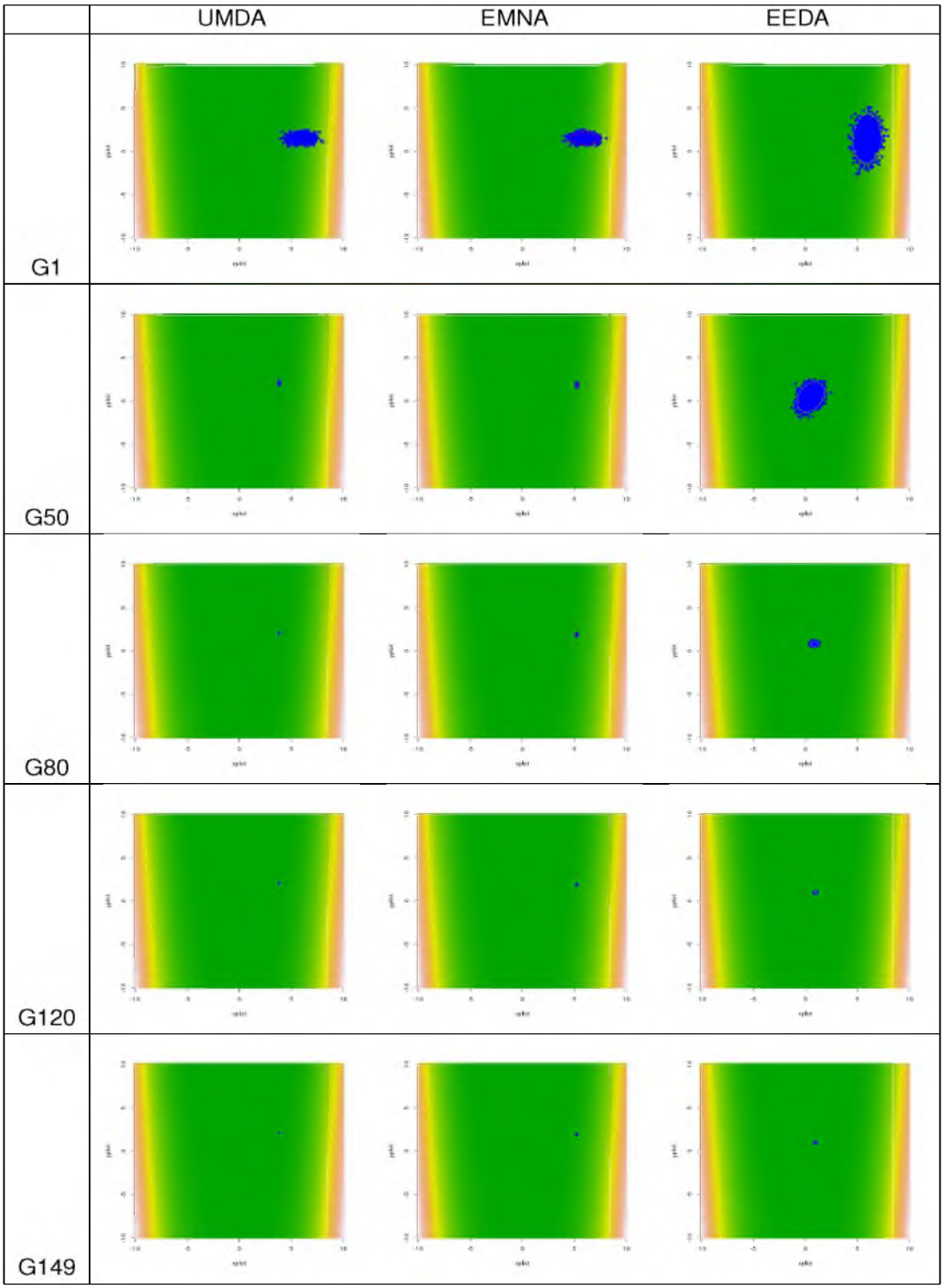
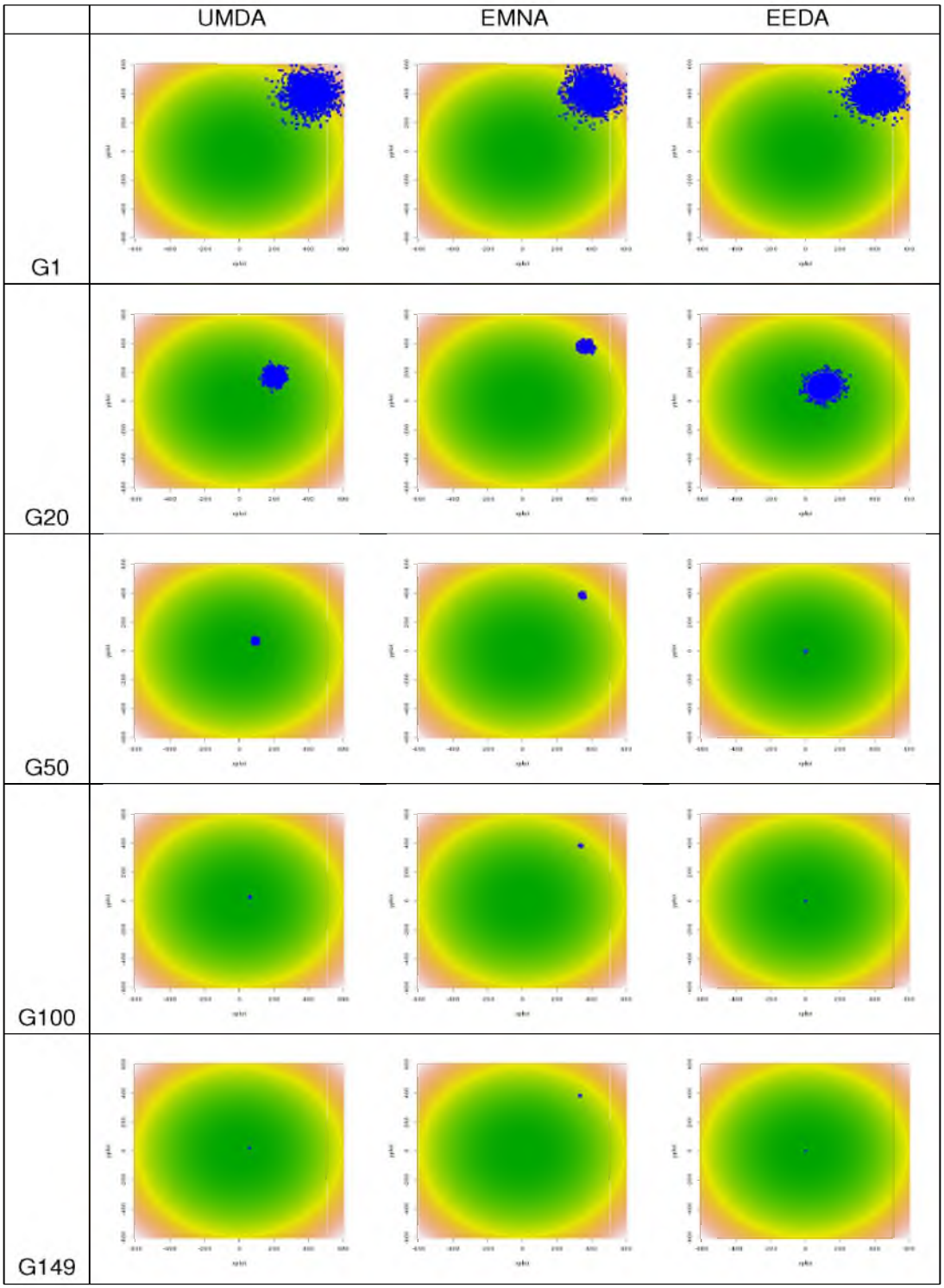


Figura 5.1: Ajuste lineal tamaño de población-variables. Se considera la mediana de los valores de la función objetivo en cada dimensión de la Tabla 5.11. La función que describe es lineal por lo que se puede argumentar que con los datos de dimensión y el tamaño de la población calculada, el algoritmo obtiene los mejores resultados de acuerdo a la evidencia estadística.



Cuadro 5.4: Gráficas de la función *Rosenbrock*, generaciones 1,50,80,120 y 149. Cada variable del individuo se generó en los espacios inferior={5,1,1,3,4,6,5,1,2,6} y superior={7,2,5,10,5,8,7,2,7,10}, todos lejos del óptimo y el espacio de observación entre -10 a 10.



Cuadro 5.5: Gráficas de la función *Griewangk*, generaciones 1,20,50,100 y 149. Las 10 variables de cada individuo fueron generadas lejos del óptimo, entre inferior=300 y superior=550, y el espacio de observación entre -600 a 600.

Funciones separables				
Función	Dimensión	Algoritmo	Media Fobj	No. evaluaciones
Sphere	2	$UMDA_c$	$-1.0e-08$	99951
	5	$UMDA_c$	$-1.0e-08$	99951
	2	$EMNA_{global}$	$-1.0e-08$	97952
	5	$EMNA_{global}$	$-1.0e-08$	97952
	2	$EEDA$	$-1.0e-08$	97952
	5	$EEDA$	$-1.0e-08$	97952
Ellipsoidal	2	$UMDA_c$	$-1.0e-08$	99951
	5	$UMDA_c$	$-9.9e-09$	99951
	2	$EMNA_{global}$	$-6.8e-09$	97952
	5	$EMNA_{global}$	$6.5e-06$	101952
	2	$EEDA$	$-1.0e-08$	97952
	5	$EEDA$	$3.2e+02$	101952
Rastrigin	2	$UMDA_c$	$1.8e-03$	101951
	5	$UMDA_c$	$1.7e+00$	101951
	2	$EMNA_{global}$	$1.2e-03$	101952
	5	$EMNA_{global}$	$1.6e+00$	101952
	2	$EEDA$	$1.9e-03$	101952
	5	$EEDA$	$3.3e+00$	101952
Buche-Rastrigin	2	$UMDA_c$	$1.6e-02$	101951
	5	$UMDA_c$	$6.2e+00$	101951
	2	$EMNA_{global}$	$2.6e-02$	101952
	5	$EMNA_{global}$	$1.0e+01$	101952
	2	$EEDA$	$4.2e-02$	101952
	5	$EEDA$	$5.4e+00$	101952
Linear Slope	2	$UMDA_c$	$-1.0e-08$	99951
	5	$UMDA_c$	$-1.0e-08$	99951
	2	$EMNA_{global}$	$-1.0e-08$	97952
	5	$EMNA_{global}$	$1.2e+01$	101952
	2	$EEDA$	$-1.0e-08$	97952
	5	$EEDA$	$-1.0e-08$	97952

Cuadro 5.6: Resultados de tres instancias de los algoritmos UMDAc y EMNAglobal sobre las funciones proporcionadas por el paquete BBOB. El máximo número de evaluaciones para cada función fue de 100,000.

Funciones con acondicionamiento bajo o moderado				
Función	Dimensión	Algoritmo	Media Fobj	No. evaluaciones
Attractive Sector	2	$UMDA_c$	$2.9e-02$	101951
	5	$UMDA_c$	$5.3e+00$	101951
	2	$EMNA_{global}$	$1.7e-02$	101952
	5	$EMNA_{global}$	$4.1e+00$	101952
	2	$EEDA$	$8.0e-03$	101952
	5	$EEDA$	$9.9e-01$	101952
Step Ellipsoidal	2	$UMDA_c$	$5.1e-03$	101951
	5	$UMDA_c$	$4.5e-01$	101951
	2	$EMNA_{global}$	$-1.0e-08$	97952
	5	$EMNA_{global}$	$-1.0e-08$	97952
	2	$EEDA$	$-1.0e-08$	97952
	5	$EEDA$	$5.0e-01$	101952
Rosenbrock, original	2	$UMDA_c$	$2.7e-03$	101951
	5	$UMDA_c$	$2.7e+00$	101951
	2	$EMNA_{global}$	$2.2e-03$	101952
	5	$EMNA_{global}$	$2.5e+00$	101952
	2	$EEDA$	$6.0e-03$	101952
	5	$EEDA$	$2.7e+00$	101952
Rosenbrock, rotated	2	$UMDA_c$	$8.1e-03$	101951
	5	$UMDA_c$	$3.4e+00$	101951
	2	$EMNA_{global}$	$2.6e-02$	101952
	5	$EMNA_{global}$	$2.7e+00$	101952
	2	$EEDA$	$7.5e-03$	101952
	5	$EEDA$	$2.8e+00$	101952

Cuadro 5.7: Resultados de tres instancias de los algoritmos UMDAc y EMNAglobal sobre las funciones proporcionadas por el paquete BBOB. El máximo número de evaluaciones para cada función fue de 100,000.

Funciones con acondicionamiento alto y unimodal				
Función	Dimensión	Algoritmo	Media Fobj	No. evaluaciones
Ellipsoidal	2	$UMDA_c$	$5.2e-02$	101951
	5	$UMDA_c$	$6.7e+01$	101951
	2	$EMNA_{global}$	$-7.3e-09$	101952
	5	$EMNA_{global}$	$2.4e-06$	101952
	2	$EEDA$	$-1.0e-08$	97952
	5	$EEDA$	$1.0e+03$	101952
Discus	2	$UMDA_c$	$3.8e-01$	101951
	5	$UMDA_c$	$8.5e-01$	101951
	2	$EMNA_{global}$	$-4.2e-09$	97952
	5	$EMNA_{global}$	$1.8e-06$	101952
	2	$EEDA$	$-1.0e-08$	97952
	5	$EEDA$	$5.7e+00$	101952
Bent Cigar	2	$UMDA_c$	$6.4e-01$	101951
	5	$UMDA_c$	$4.8e+00$	101951
	2	$EMNA_{global}$	$7.0e-04$	101952
	5	$EMNA_{global}$	$3.8e+04$	101952
	2	$EEDA$	$3.5e-03$	101952
	5	$EEDA$	$1.5e+03$	101952
Sharp Ridger	2	$UMDA_c$	$8.9e-01$	101951
	5	$UMDA_c$	$3.9e+00$	101951
	2	$EMNA_{global}$	$1.1e-06$	101952
	5	$EMNA_{global}$	$9.1e+01$	101952
	2	$EEDA$	$-1.0e-08$	97952
	5	$EEDA$	$1.1e+01$	101952
Different Powers	2	$UMDA_c$	$-8.2e-10$	101951
	5	$UMDA_c$	$5.1e-06$	101951
	2	$EMNA_{global}$	$-1.4e-09$	97952
	5	$EMNA_{global}$	$1.7e-07$	101952
	2	$EEDA$	$7.7e-07$	101952
	5	$EEDA$	$9.3e-04$	101952

Cuadro 5.8: Resultados de tres instancias de los algoritmos UMDAc y EMNAglobal sobre las funciones proporcionadas por el paquete BBOB. El máximo número de evaluaciones para cada función fue de 100,000.

Funciones multimodal con estructura global adecuada				
Función	Dimensión	Algoritmo	Media Fobj	No. evaluaciones
Rastrigin	2	$UMDA_c$	$-1.0e-08$	99951
	5	$UMDA_c$	$2.6e+00$	101951
	2	$EMNA_{global}$	$-1.0e-08$	97952
	5	$EMNA_{global}$	$4.1e-01$	101952
	2	$EEDA$	$6.2e-04$	101952
	5	$EEDA$	$4.1e+00$	101952
Weierstrass	2	$UMDA_c$	$1.9e-04$	101951
	5	$UMDA_c$	$1.3e+00$	101951
	2	$EMNA_{global}$	$1.9e-09$	97952
	5	$EMNA_{global}$	$1.2e+00$	101952
	2	$EEDA$	$4.6e-03$	101952
	5	$EEDA$	$1.7e+00$	101952
Schaffers F7	2	$UMDA_c$	$7.5e-07$	101951
	5	$UMDA_c$	$2.3e-04$	101951
	2	$EMNA_{global}$	$1.1e-06$	101952
	5	$EMNA_{global}$	$1.6e-04$	101952
	2	$EEDA$	$1.1e-03$	101952
	5	$EEDA$	$1.7e-02$	101952
SchaffersF7, moderately ill-conditioned	2	$UMDA_c$	$2.6e-01$	101951
	5	$UMDA_c$	$2.1e-01$	101951
	2	$EMNA_{global}$	$5.5e-06$	101952
	5	$EMNA_{global}$	$5.2e-04$	101952
	2	$EEDA$	$6.1e-02$	101952
	5	$EEDA$	$5.2e-01$	101952
Composite Griewank-Rosenbrock F8F2	2	$UMDA_c$	$2.7e-06$	101951
	5	$UMDA_c$	$8.9e-02$	101951
	2	$EMNA_{global}$	$3.9e-07$	101952
	5	$EMNA_{global}$	$2.4e-02$	101952
	2	$EEDA$	$3.1e-05$	101952
	5	$EEDA$	$2.4e-01$	101952

Cuadro 5.9: Resultados de tres instancias de los algoritmos UMDAc sobre las funciones proporcionadas por el paquete BBOB. El máximo número de evaluaciones para cada función fue de 100,000.

Funciones ultimodal con estructura global débil				
Función	Dimensión	Algoritmo	Media Fobj	No. evaluaciones
Schwefel	2	$UMDA_c$	$4.8e-03$	101951
	5	$UMDA_c$	$1.6e+00$	101951
	2	$EMNA_{global}$	$3.1e-02$	101952
	5	$EMNA_{global}$	$1.6e+00$	101952
	2	$EEDA$	$7.0e-02$	101952
	5	$EEDA$	$1.5e+00$	101952
Gallagher's Gaussian 101-me Peaks	2	$UMDA_c$	$6.5e-06$	101951
	5	$UMDA_c$	$9.0e-03$	101951
	2	$EMNA_{global}$	$7.3e-06$	101952
	5	$EMNA_{global}$	$-1.0e-08$	97952
	2	$EEDA$	$4.1e-07$	101952
	5	$EEDA$	$3.7e-02$	101952
Gallagher's Gaussian 21-hi Peaks	2	$UMDA_c$	$4.1e-05$	101951
	5	$UMDA_c$	$1.8e-01$	101951
	2	$EMNA_{global}$	$2.2e-04$	101952
	5	$EMNA_{global}$	$4.0e-01$	101952
	2	$EEDA$	$5.5e-07$	101952
	5	$EEDA$	$8.5e-02$	101952
Katsuura	2	$UMDA_c$	$1.0e-01$	101951
	5	$UMDA_c$	$9.0e-01$	101951
	2	$EMNA_{global}$	$2.8e-01$	101952
	5	$EMNA_{global}$	$9.1e-01$	101952
	2	$EEDA$	$6.2e-02$	101952
	5	$EEDA$	$6.1e-01$	101952
Lunacek bi-Rastrigin	2	$UMDA_c$	$1.4e+00$	101951
	5	$UMDA_c$	$7.7e+00$	101951
	2	$EMNA_{global}$	$1.3e+00$	101952
	5	$EMNA_{global}$	$7.1e+00$	101952
	2	$EEDA$	$4.3e-01$	101952
	5	$EEDA$	$8.0e+00$	101952

Cuadro 5.10: Resultados de tres instancias de los algoritmos UMDAc sobre las funciones proporcionadas por el paquete BBOB. El máximo número de evaluaciones para cada función fue de 100,000.

Capítulo 6

Caso de estudio: Detección rápida de parábola usando la propuesta del SSEDA

6.1. Introducción

La diabetes, la hipertensión, la arteriosclerosis y la retinopatía de la prematuridad (ROP) afectan la estructura de los vasos sanguíneos en la retina mediante la modificación de su anchura, forma y tortuosidad. El reconocimiento de bajos niveles de tortuosidad y dilatación es difícil incluso para los observadores expertos, como lo indican Freedman et al.[10]. Considerando que los cambios en la tortuosidad se han observado para ser coherente con la gravedad de la enfermedad [10], la cuantificación de los cambios en el ancho de los vasos puede ser difícil debido al hecho de que tales cambios son comparables a la resolución de la imagen[28]. Debido a que las enfermedades mencionadas anteriormente cambian el ángulo de inserción de la arcada temporal, también cambia la apertura de la arcada temporal principal. En consecuencia, la detección de la arcada temporal principal y el análisis cuantitativo de su apertura podrían ayudar a diagnosticar y monitorear las etapas de la retinopatía, así como a determinar los efectos de la terapia[22].

La arcada temporal puede ser modelada geométricamente como parábolas, así, el problema es encontrar un parábola que se ajuste a la imagen de la retina del ojo; entonces el objetivo es encontrar los parámetros de la forma parabólica que mejor se ajuste a la imagen de la retina del ojo.

En la literatura se han propuesto diferentes métodos para detectar parábolas usando la transformación de Hough (Hough Transform) como base. Oloumi y Rangayyan [22], aplican la trans-

formación de Hough para detectar parábolas usando tres parámetros: el vértice (x_0, y_0) y a para describir la apertura de la parábola. La principal aplicación reportada de este método es la detección de arcada temporal en imágenes de fondo de retina, donde las parábolas se ajustan a los vasos de la retina. Jesus Guerrero I. et al.[14] presentan un método basado en EDA para detectar formas parabólicas en imágenes sintéticas y médicas (retina del ojo y planta del pie). Ellos muestran que al usar un EDA se tiene un menor costo en tiempo computacional comparado con método tradicionales de detección de formas parabólicas en imágenes sintéticas y médicas.

Este capítulo presenta la propuesta de un EDA de estado semi estable (SSEDA) para el problema de detección rápida de parábolas en imágenes de la retina del ojo. Actualmente no se conoce el uso de estado semi estable en EDA, esto se hace con el fin de aprovechar la información de todas las parábolas generadas en el proceso de optimización, y reducir el número de evaluaciones o mejorar la calidad de la detección. En las siguientes secciones se describe la propuesta.

6.2. Metodología propuesta

El método consiste en generar una parábola virtual y por medio del producto Hadamard obtener la función objetivo que se utiliza en el Algoritmo 5. Las siguientes secciones describen la propuesta de un EDA de estado semi estable para detectar formas parabólicas en imágenes de la retina del ojo.

6.2.1. Espacio de búsqueda

Una solución candidata esta definida por 4 variables (x_0, y_0, a, θ), es decir, el algoritmo de optimización se utiliza en 4 dimensiones. Para generar los límites, se toman N muestras de coordenadas, pixeles que pertenecen al vaso de la retina. Cada muestra es de M pixeles. Con los M pixeles de cada muestra se ajusta una parábola por mínimos cuadrados sin rotar. Entonces, se tienen N conjuntos de parámetros de valores de (x_0, y_0, a), para estos se calculan las medias ($\mu(x_0)$, $\mu(y_0)$, $\mu(a)$) y desviación estándar ($\sigma(x_0), \sigma(y_0), \sigma(a)$) para cada conjunto de parámetros, y

los límites se definen como:

$$\begin{aligned} x_{inf} &= \mu x_0 - 2.5 * \sigma(x_0), \\ y_{inf} &= \mu y_0 - 2.5 * \sigma(y_0), \\ a_{inf} &= \mu a - 2.5 * \sigma(a), \\ x_{sup} &= \mu x_0 + 2.5 * \sigma(x_0), \\ y_{sup} &= \mu y_0 + 2.5 * \sigma(y_0) \quad y \\ a_{sup} &= \mu a + 2.5 * \sigma(a). \end{aligned} \tag{1}$$

Los límites del ángulo de rotación se definen, por conveniencia y en grados, como sigue:

$$\begin{aligned} \theta_{inf} &= -10 \quad y \\ \theta_{sup} &= 10. \end{aligned} \tag{2}$$

6.2.2. Representación del individuo

La línea 2 del Algoritmo 5 consiste en muestrear a partir de los limites obtenidos en el paso anterior, Ecuación 1 y 2, con una distribución normal los nuevos individuos.

La ecuación paramétrica de la parábola es requerida para representar una solución potencial para encontrar la forma parabólica en una imágen. En el sistema de coordenadas cartesianas, una parábola puede ser descrita usando su formula general:

$$f(x) = y = Ax^2 + Bx + C, \tag{3}$$

donde A,B y C representan constantes desconocidas, las cuales son determinada usando una coordenada (x, y), en el dominio de la imagen de entrada hace referencia a un pixel, y una apertura a. Por lo que cada individuo de la población esta representado por los parámetros: x, y, a, θ.

6.2.3. Función objetivo

Para evaluar la función objetivo, línea 3 del Algoritmo 5, se genera una imagen virtual (I_{vs}) del mismo tamaño que la imagen de entrada (f_{binary}), la imagen de entrada tiene Nc columnas y Nr filas. Los valores de los pixeles son ceros, entonces la parábola rotada es calculada usando los valores de la constante A,B y C obtenidas con los parámetros: x, y, a, θ del individuo. Así, los puntos de la parábola son almacenados en I_{vs} poniendo a los pixeles correspondientes un valor 1. Finalmente, es calculado el producto Hadamard entre la imagen de entrada y la imagen virtual como sigue:

$$Hd = f_{binary} \odot I_{vs}. \tag{4}$$

La imagen resultante Hd contiene solo los pixeles con valor de 1 donde f_{binary} y I_{vs} coinciden, por lo que la función objetivo que se usa para medir la calidad de la solución potencial es el número de pixeles con valor 1 resultantes del producto Hadamard. La Figura 6.1 muestra un ejemplo en [14] de una parábola formada por un individuo, una imagen de entrada y el resultado Hd.

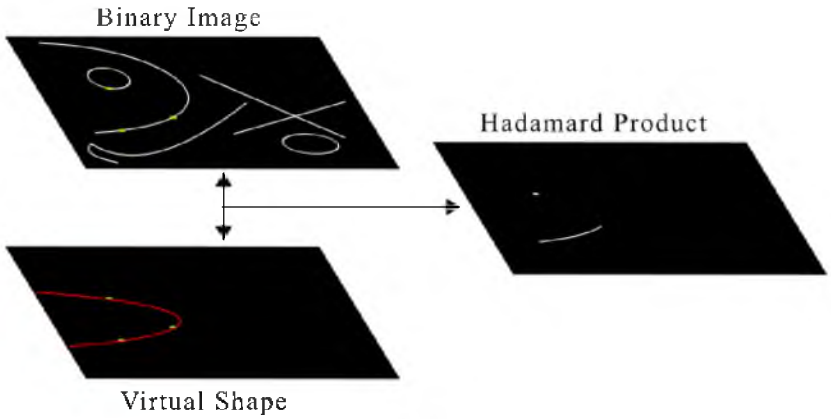


Figura 6.1: Producto Hadamard entre la imagen de entrada y la virtual . Fuente: [14]

Dados los parámetros x, y, a, θ de un individuo, el cálculo de la parábola rotada se realiza de la siguiente manera:

a) Se emplean la ecuación general de la parábola y los parámetros x, y, a para calcular las constantes B y C,

b) teniendo las constantes B y C se calcula el vértice $V = (\frac{-b}{2a}, f(\frac{-b}{2a}))$ para rotar la imagen.

c) La rotación de la parábola con respecto al vértice (x_r, y_r) y dado el ángulo θ se calcula como sigue:

$$X = x_r + (x - x_r)\cos(\theta) - (y - y_r)\sin(\theta)$$
$$Y = y_r + (x - x_r)\sin(\theta) + (y - y_r)\cos(\theta)$$

d) Para dibujar la parábola en la imagen virtual se recorren las Nr filas de ésta, pero no se recorren de 1 en 1, ya que recorrerla de 1 en 1 puede generar una parábola con pixeles desconectados, un incremento adecuado para asegurar que la parábola se dibuje bien se calcula como sigue: se toma en cuenta el valor de la pendiente $m = abs(2 * a * x + b) + 1$, con lo que los incrementos en x se hacen de $\frac{1}{m}$.

e) Se verifica que las coordenadas de la nueva parábola se encuentren dentro de los límites de la imagen de entrada, esto es, que $0 < (x, y) < (Nr, Nc)$.

6.3. Experimentos y resultados

En esta sección se presentan los experimentos y resultado del caso de estudio. Para los experimentos se tomaron las imágenes médicas de la retina del ojo usadas en la referencia[14], ver Figura 6.2 de donde para este trabajo se utilizaron las imágenes binarizadas. Los parámetros empleados en cada experimento se describen a continuación junto con los resultados obtenidos.

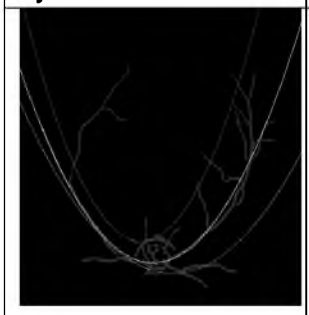
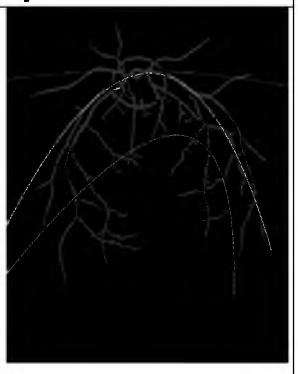
(a) (b)

Figura 6.2: 6.2a Imagen de la retina en la base de datos DRIVE, 565 x 584 pixeles. 6.2b Imagen binarizada.

6.3.1. Descripción y resultados del experimento 1

El algoritmo de optimización se ejecutó 5 veces para tres imágenes diferentes. Los parámetros del algoritmo son los siguientes: una población de 15 individuos y se detiene si los elementos del individuo convergen a una desviación estándar de $5e^{-3}$. Se proponen dos métodos para el calculo de los pesos usados en la línea 3 del Algoritmo 5.

La Tabla 6.1 muestra los resultados en imágenes y datos. Los métodos 1 y 2 se describen en la Sección 4.1.1. Donde se observa que los mejores resultados vienen dados por el método 2, por lo que elevar a un exponente los estimadores pesados resulta mejor que dejarlos como estimadores pesados con exponente uno. Se esperaría que metiendo un ángulo de rotación se tendrían mejores resultados de la función objetivo, pero vemos que no es así; habría que llevar las imágenes a un experto en el estudio de detección de arcada temporal en la retina del ojo para evaluar si el ángulo de rotación de la parábola está dando mejores aproximaciones.

Método 1	Método 1 con ángulo	Método 2	Método 2 con ángulo
			
fob best=376 **No. generación= 15** **Ejecución= 4**	**fob best=270** **No. generación= 24** **Ejecución= 3**	**fob best=387** **No. generación= 8** **Ejecución= 2**	**fob best=376** **No. generación= 9** **Ejecución= 4**
			
fob best=552 **No. generación=17** **Ejecución= 4**	**fob best=519** **No. generación= 25** **Ejecución=2**	**fob best=537** **No. generación=8** **Ejecución=3**	**fob best=514** **No. generación= 11** **Ejecución= 1**
			
fob best=441 **No. generación=13** **Ejecución= 3**	**fob best=400** **No. generación= 28** **Ejecución= 2**	**fob best=482** **No. generación=10** **Ejecución= 2**	**fob best=384** **No. generación= 11** **Ejecución= 3**

Cuadro 6.1: Resultados del experimento 1. Se muestran las imágenes obtenidas con los mejores valores de la función objetivo obtenidos con el Algoritmo 5.

6.3.2. Experimento 2: Búsqueda de los parámetros del SSEDA

El propósito de esta sección es encontrar los parámetros que proporcionen mejores resultados para el SSEDA. El Algoritmo 5 se ejecuto 15 veces para las combinaciones de: número de población inicial (de 90, 100 y 120 individuos) y muestras (de 6, 8 y 10 individuos) que formarán parte de las iteraciones futuras. Los criterios de paro son los siguientes: se detiene si el valor de los elementos están por debajo de una desviación estándar de $5e^{-3}$, o si sobrepasa las 1780 evaluaciones de la función objetivo.
La Tabla 6.2 muestra los resultados obtenidos. Se muestran los parámetros: número de individuos de la población inicial, número de individuos a ser muestreados, número de evaluaciones e iteraciones, con los que se obtuvo una aproximación al valor óptimo. De acuerdo a los datos que se muestran, se observa que el algoritmo aproxima un óptimo cuando se tiene una población inicial considera entre 100 y 120 individuos, y que los nuevos individuos a ser muestreados se encuentran entre 6, 8 y 10.

Imagen	Pob. Inicial	Muestras	Fobj	Evaluaciones Fobj.	Generaciones
1	120	6	560	1176	177
2	100	6	399	1174	180
3	100	8	524	1172	135
4	100	6	410	1174	180
5	120	10	328	1170	106
6	100	10	433	1170	108
7	90	6	426	1176	182

Cuadro 6.2: Resultados del experimento 2. Mejores soluciones de la función objetivo para siete imágenes distintas de la retina del ojo obtenidas con el Algoritmo 5. Fobj hace referencia al valor de la función objetivo.

6.3.3. Comparación entre resultados del artículo[14] y los propios

El objetivo de este experimento es comparar la propuesta de un EDA Univariado de estado semi estable con estimadores pesados , el Algoritmo 5, con una propuesta reciente en el estado del arte. La comparación se realizó de dos formas: 1) un experimento en donde el Algoritmo 5 se detuviera con un criterio de paro de una desviación estándar no mayor a $5e^{-3}$, el algoritmo realiza mas evaluaciones que el de la referencia[14] pero se espera que entregue mejores resultados, y 2) un segundo experimento donde se ocupa el mismo criterio de paro que en [14], para ver si bajo las mismas condiciones la propuesta del Algoritmo 5 es competitiva, a través de comparaciones

justas. Para ambos experimentos se hicieron 15 ejecuciones de los algoritmos sobre 5 imágenes de la retina del ojo con una población inicial de 30 individuos y muestras de 6 individuos por iteración.

Del primer experimento, la Tabla 6.3 muestra los resultados comparativos entre el Algoritmo 5 y implementado de [14]. En este experimento las evaluaciones de la función objetivo se encuentran entre 374 y 668 evaluaciones, son mucho más que en los de [14], pero la aproximación al óptimo es mejor.

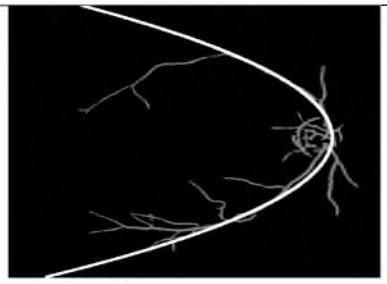
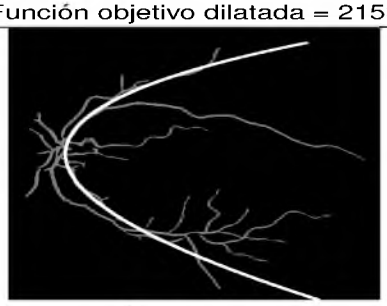
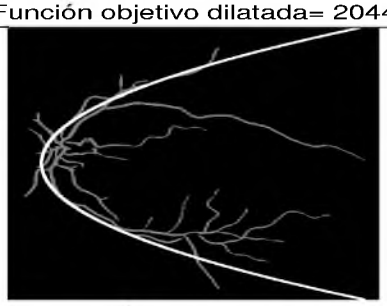
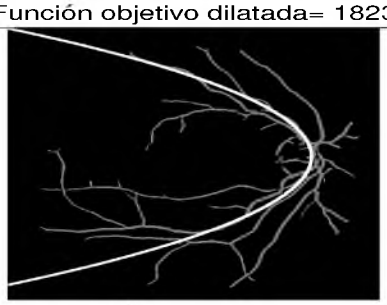
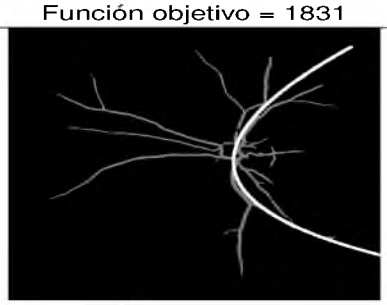
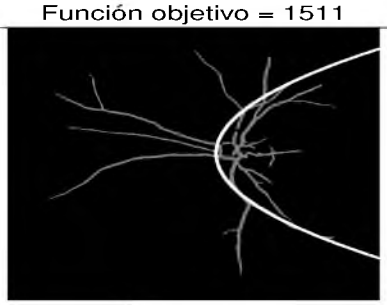
Imagen	Algoritmo 5				Algoritmo utilizado en[14]			
	Fobj	Fobj Dilatada	T	Evals.	Fobj	Fobj Dilatado	T	Evals.
1	540	2229	33.22	668	247	2044	3.72	60
2	358	1729	32.38	668	197	1461	1.58	40
3	511	2433	40.29	668	206	1823	3.24	60
4	391	1799	34.99	668	189	1511	2.74	60
5	267	1139	23.91	668	178	1255	3.61	80

Cuadro 6.3: Resultados comparativos entre el Algoritmo 5 y el algoritmo de la referencia[14]. El primer algoritmo se detiene con el criterio de paro $\sigma \leq 5e^{-3}$, mientras que el segundo se detiene en 30 iteraciones.

Por último se hizo una comparación entre resultados obtenidos de la implementación del algoritmo de la referencia[14] y los obtenidos para el Algoritmo 5, donde para ambos algoritmos se utilizo como criterio de paro 30 iteraciones. Los resultados comparativos se muestran en la Tabla 6.4, se observa que la aproximación del óptimo es mejor con el Algoritmo 5 que con el de la referencia[14], pero se tiene un mayor tiempo, que se cree puede ser reducido si se paraleliza el Algoritmo 5. En la Tabla 6.5 se muestran los resultados de las 5 imágenes, donde se observa un mejor ajuste de la parábola con los resultados del Algoritmo 5.

Imagen	Algoritmo 5						Algoritmo utilizado en[14]					
	Fobj	Dilatado	T	Evals.	Media	SD	Fobj	Dilatado	T	Evals.	Media	SD
1	519	2153	8.82	200	429.46	42.44	247	2044	3.72	60	176.33	43.70
2	319	1716	9.63	200	292.2	27.59	197	1461	1.58	40	120.13	38.86
3	427	2362	11.73	200	380.06	39.59	206	1823	3.24	60	140.4	41.60
4	354	1831	6.78	200	281.93	36.30	189	1511	2.74	60	125.86	38.06
5	269	1373	6.78	200	214.26	25.32	178	1255	3.61	80	93.2	53.83

Cuadro 6.4: Resultados comparativos entre el Algoritmo 5 y el utilizado en [14]. Se muestran las mejores soluciones de la función objetivo para siete imágenes distintas de la retina del ojo.

SSEDA	Algoritmo utilizado en[14]
	
Función objetivo = 519 Función objetivo dilatada = 2153	Función objetivo = 247 Función objetivo dilatada= 2044
	
Función objetivo = 319 Función objetivo dilatada= 1716	Función objetivo = 197 Función objetivo dilatada= 1461
	
Función objetivo = 427 Función objetivo dilatada= 2362	Función objetivo = 206 Función objetivo dilatada= 1823
	
Función objetivo = 354 Función objetivo = 1831	Función objetivo = 189 Función objetivo = 1511
	
Función objetivo = 269 Función objetivo = 1373	Función objetivo = 178 Función objetivo = 1255

Cuadro 6.5: Imágenes de la retina del ojo. Se muestran los resultados de 5 imágenes diferentes para los dos algoritmos comparativos.

Capítulo 7

Conclusiones y trabajo a futuro

7.1. Conclusiones

El estudio de los algoritmos de estimación de distribución: UMDA, EMNA y EEDA, y el caso de detección de arcada temporal en la retina del ojo que se hizo en esta tesis sirve como base para comparar los resultados obtenidos de las dos propuestas, las cuales introducen las ideas de un algoritmo multivariado con direcciones en la búsqueda del óptimo y un algoritmo univariado de estado semi estable con el que se hacen pocas evaluaciones de la función objetivo usando toda la información del proceso de optimización.

El uso de direcciones en las búsqueda del óptimo que se empleó en el EDAD que se propone, un EDA multivariado, muestra (como se menciona en la literatura especializada en estrategias evolutivas[24, 3]) que el algoritmo aproxima con menor error el óptimo que otros EDA univariados y multivariados que no utilizan direcciones de búsqueda. De los problemas que se plantean en el Capítulo 5, se obtuvieron los mejores resultados para las funciones Sphere y Ackley para una dimensión de 10, la primer función tiene un solo mínimo global mientras que la segunda tiene múltiples mínimos locales. Así, se muestra que el EDAD aproxima el óptimo global mejor que los otros algoritmos implementados, la segunda función cumple con algunas de las características de la función para las que se plantea esta propuesta (tiene múltiples mínimos locales, es continua y no lineal). Por otra parte el que los resultados comparados con el EEDA sean similares e incluso mejores, lleva a la conclusión de que la propuesta con direcciones en la búsqueda del óptimo esta similar al EEDA, ya que este es un algoritmo que también introduce direcciones en la búsqueda del óptimo. Se puede decir, que sin embargo no esta a la altura del CMA-ES o el AMaLGaM que son el estado del arte, pero se ve que trabajar con algoritmos que utilizan direcciones es muy

promisorio.

En este trabajo se propuso el uso del concepto de estado semi estable en algoritmos de estimación de distribución y se aplicó al caso de detección de arcada temporal en la retina del ojo, lo cual consiste en ajustar una parábola a la imagen de la retina del ojo, y como resultado se obtuvo una mejor aproximación del óptimo en comparación con los resultados en un artículo reciente del estado del arte [14]. En cuanto al tiempo computacional no se obtuvieron los mejores tiempos pero si similares a los obtenidos con métodos tradicionales de detección de parábolas en imágenes[22]. Dado que el algoritmo es paralelizable, se podría reducir el tiempo utilizando cómputo paralelo. Al tener una muestra mas grande para estimar los parámetros de la función de probabilidad, el algoritmo es más robusto, unas pocas muestras no van a modificar sustancialmente el valor de los parámetros. Los pesos en los estimadores de los parámetros provocan que se le de más importancia a las soluciones en las mejores regiones, o con mejor valor objetivo, al ir incrementando el valor de los pesos de las mejores soluciones encontradas lleva a que el algoritmo converja a las regiones donde están estas soluciones.

En la industria existen problemas donde se requiere un número reducido de evaluaciones y en un tiempo relativamente rápido, como por ejemplo en el caso de estudio de la detección de arcada temporal en la retina del ojo se espera tener una respuesta en tiempo real. Con el SSEDA se pueden resolver problemas de este tipo. En el caso de estudio de detección de parábolas en imágenes de la retina del ojo, donde los resultados esperados deben ser de forma casi inmediata y la detección debe ser la misma o muy similar cuando se pone al mismo paciente, el SSEDA demostró un buen desempeño, ya que se pudieron almacenar todas las evaluaciones por ser pocas y con ello se redujo el tiempo de cómputo, y las aproximaciones al óptimo fueron similares para una misma imagen (un mismo paciente) ejecutando 15 veces el algoritmo para la misma imagen. Obtener resultados similares para la misma imagen es importante, porque en el mundo real, se espera que para el mismo paciente siempre se detecte la misma parábola o una muy similar.

Lo que se puede destacar, es que tenemos dos propuestas, de la primera se mostró que el uso de direcciones tiene un impacto importante en el diseño de algoritmo ya que se ve que el desempeño del algoritmo va casi en el orden: UMDA, cuya función de probabilidad esta siempre alineada a las direcciones cartesianas, EMNA, que ya se puede orientar con la matriz de covarianza, EEDA que

tiene orientaciones pero que ademas incrementa la búsqueda en cierta dirección (la del mínimo eigenvalor), y finalmente la propuesta de EDAD, que inserta las direcciones que van de las peores soluciones a las mejores en la función de probabilidad. Por otro lado se presenta la propuesta del primer algoritmo de estimación de distribución de estado semi estable (SSEDA), que usa toda la información de todas las soluciones muestreadas durante el proceso de optimización para mejorar la estimación del óptimo, y disminuir la varianza de la estimación. La mayoría de los EDA y otros algoritmos evolutivos reemplazan la mayoría de la población, en estos reemplazos es muy probable que se pierda información importante sobre regiones promisorias donde puede estar el óptimo, o que requiera estar continuamente muestreando zonas que ya han sido muestreadas para obtener la información suficiente para construir un buen estimador del óptimo, estas dos desventajas se eliminan o reducen con el SSEDA, donde se aprovecha toda la información de todas las muestras obtenidas durante el proceso de optimización.

7.2. Trabajo a futuro

La propuesta del EDA con estado semi estable tiene como una característica particular que es altamente paralelizable por lo que en un futuro se podría paralelizar. Al paralelizar el algoritmo, el tiempo que se obtuvo en el caso de estudio de detección de arcada en la retina del ojo puede mejorar mucho y ya se ha mostrado que con el algoritmo se tiene una aproximación al óptimo muy buena comparada con lo que existe en el estado del arte.

El Algoritmo 6, llamado EDAD, puede ser mejorado considerando un cambio en la selección y el calculo de la matriz de covarianza. En el EDAD se propone como estimador de la media el mejor de los individuos, se podría proponer como media un valor mas próximo hacia la dirección del óptimo, es decir, poner la media sobre la dirección del eigenvalor mas grande. También se podría utilizar la información de las direcciones en pasos anteriores para mejorar las direcciones actuales en la matriz de covarianza.

Bibliografía

[1] Propiedades del gradiente. http://sistemas.fciencias.unam.mx/~erhc/calculo3/gradiente1.pdf/.

[2] Variables aleatorias y función de distribución. http://www.ugr.es/~eues/webgrupo/Docencia/MonteroAlonso/estadisticaII/tema2.pdf/.

[3] *Theory and Principled Methods for the Design of Metaheuristics*, chapter Principled Design of Continuous Stochastic Search: From Theory to. Springer, 2014.

[4] Howard Anton. *Introducción al álgebra lineal*. Limusa, 3 edition, 1994.

[5] CIMAT. *Topología de variables diferenciales, notas núm. 1*, ene-jun 2013.

[6] Dr. Carlos Coello Coello. Optimización en ingeniería. Technical report, Departamento de computación, CINVESTAV-IPN, 2009.

[7] Prof. Cesar de Prada. optimización con restricciones. Technical report, ISA-UVA.

[8] Agnés Borrás Jesús Giraldo Debora Gil, David Roche. Terminating evolutionary algorithms at their steady state. *Springer*, 2015.

[9] Weishan Dong and Xin Youn. Covariance matrix repairing in gaussian based edas. *IEEE*, 2007.

[10] Hall JG Freedman SF, Kylstra JA and Capowski JJ. *Plus disease in retinopathy of prematurity - Photographic evaluation by an expert panel*, chapter Investigative Ophthalmology and Visual Science. 1995.

[11] Carlos Ivorra. matemáticas ii, apuntes de teoría. Technical report, Facultad de economia, Universidad de Valencia, 2012.

[12] A. Bonilla Petriciolet J. A. Fernández Varga. Desarrollo de un algoritmo de optimización global en colonias de hormigas con selección de región factible para espacios continuos. *ELSEVIER*, 2014.

[13] Jose A. Lozano (eds.) J. A. Lozano (auth.), Pedro Larrañaga. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Genetic Algorithms and Evolutionary Computation 2. Springer US, 1 edition, 2002.

[14] et.al. Jesus Guerrero T., Ivan Cruz A. Fast parabola detection using estimation of distribution algorithms. September 2-6, 2009.

[15] David C. Lay. *Álgebra lineal y sus aplicaciones*. Pearson, 3 edition, 2007.

[16] Sebag M. and Ducoulombier A. Extending population-based incremental learning to continuos search spaces. *Springer-Verlag*, 1998.

[17] María Cristina Maciel. Introducción a la optimización numérica. Technical report, Departament in Mathematical Science, Rice University, Houston, Texas, 2000.

[18] María Merino Maestre. Técnicas clásicas de optimización. Technical report, Facultad de ciencia y tecnología, Departamento de matemáticas aplicadas y estadística e investigación de operaciones, UPV.

[19] Ricardo A. Maronna. probabilidad y estadística elementales para estudiante de ciencia.

[20] Marc Schoenauer Michael Wagner, Anne Auger. Eeda, a new robust estimation of distribution algorithms. Technical report, INRIA, 2004.

[21] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Science+Business Media, 2006.

[22] Faraz Oloumi and Rangaraj M. Rangayyan. Detection of the tamporal arcade in fundus images of the retina using the hough transform. *IEEE*, September 2-6, 2009.

[23] COmparing Continuous Optimisers. Comparing continuous optimisers: Coco. http://coco.gforge.inria.fr/, 2009.

[24] Dierk Thierens Peter A. N. Bosman, Jorn Grahl. *Benchmarking Parameter-Free AMaLGaM on Function With and whithout Noise*, volume 21, chapter Avolutionary computation, pages 445–469. by the Massachusetts Institute of Technology, 2012.

[25] Luis Rincón. Una introducción a la probabilidad y estadística. 2006.

[26] Rudlof S. and Koppen M. Stochastic hill climbing by vectors of normal distributions. Technical report, Fraunhofer-Institute for Production Systems and Design Technology (IPK), 1997.

[27] Xin Youn Weishan Dong. Unified eigen analysis on multivariate gaussian based estimation of distribution algorithms. *ELSEVIER*, 2008.

[28] Moseley MJ Paterson C et. al. Wilson CM, Cocker KD. *Computerized analysis of retinal vessel width and tortuosity in premature infants*, chapter Investigative Ophthalmology and Visual Science. 2008.

[29] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE*, 1997.