Una propuesta de Algoritmo Evolutivo de Inspiración Cuántica para Representación Real usando Filtro de Partículas

Josimar Edinson Chire Saire

Orientador: Dr. Yván Jesús Túpac Valdivia

Jurado:
Dr. Heitor Silvério Lopes – Universidade Federal Tecnológica do Paraná – Brasil
Dr. José Ochoa Luna – Universidad Católica San Pablo – Perú
Dr. Alexander Benavides – Universidade Federal do Rio Grande do Sul – Brasil
Dr. Alex Cuadros – Universidad Católica San Pablo – Perú

Tesis presentada al
Centro de Investigación e Innovación en Ciencia de la Computación (RICS)
como parte de los requisitos para obtener el grado de
Maestro en Ciencia de la Computación.

Universidad Católica San Pablo – UCSP
Marzo de 2017 – Arequipa – Perú

A Dios, porque gracias a él ingresé al programa y ha trazado mi camino como investigador. A mi familia por sus consejos y confianza puesta en mí.

Abreviaturas

ABC *Artificial Bee Colony*

BA *Bat Algorithm*

CGA *Classical Genetic Algorithm*

DE *Differential Evolution*

fda *función de distribución acumulada*

FP *Filtro de partículas*

FP-QIEA-R *Quantum Inspired Evolutionary Algorithm with Real Representation using Filter Particle*

GNA *Generador de Números Aleatorios*

GQA *Genetic Quantum Algorithm*

GSA *Gravitational Search Algorithm*

PSO *Particle Swarm Optimization*

QIEA-B *Quantum Inspired Evolutionary Algorithm with Binary Representation*

QIEA-R *Quantum Inspired Evolutionary Algorithm with Real Representation*

Agradecimientos

Primeramente a Dios porque gracias a él ingresé, tenía un propósito conmigo en esta universidad. Me impulsó y me permitió continuar hasta el final.

A mis padres Celso y Matilde, porque me animaron, aconsejaron durante estos años a pesar de todo.

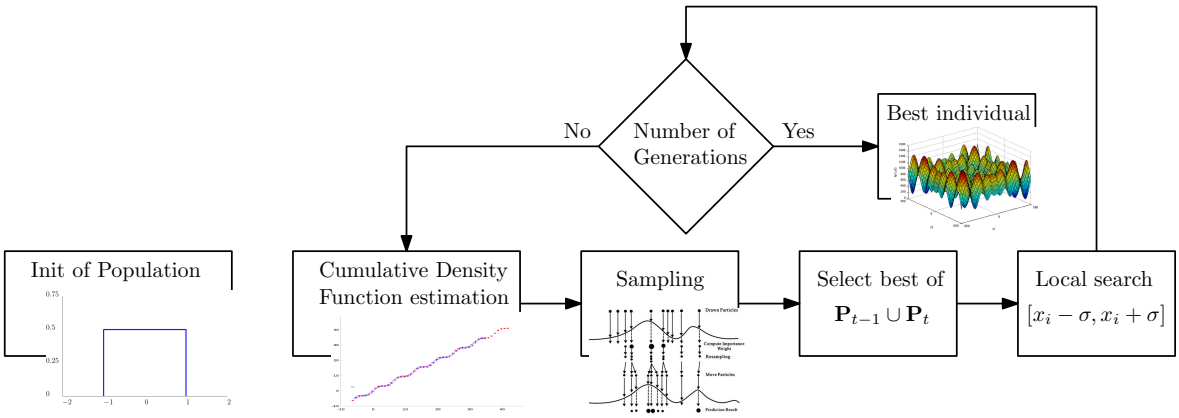
A mis hermanos, Yadira y Joel, porque me animaron con sus sonrisas, palabras y oraciones.

A mis amigos que aunque pocos, son valiosos, por compartir tiempo, vivencias y consejos.

Al profesor Yván Túpac por su ayuda en los papers y en la investigación.

Deseo agradecer de manera especial al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC) y al Fondo Nacional de Desarrollo Científico, Tecnológico e Innovación Tecnológica (FONDECYT-CIENCIACTIVA), que mediante Convenio de Gestión UCSP-FONDECYT Nº 011-2013, han permitido la subvención y financiamiento de mis estudios de Maestría en Ciencia de la Computación en la Universidad Católica San Pablo (UCSP).

Abstract



This work proposes, implements and evaluates the FP-QIEA-$\mathbb{R}$ model; it uses classical generation of model QIEA-$\mathbb{R}$; and proposes classical generation using a mechanism inspired in particle filter, function approximation methods, reward criterion, sampling using probability density function for global search and centroid for local search. During the progress of this work many function approximation methods were evaluated such as: unidimensional(splines, akima interpolation), multidimensional(multilinear regression, parzen window) to estimate modified cumulative density function(using reward criterion). The evaluation of the model were performed using benchmark functions (Ackley, Rastrigin, Rosenbrock, Schwefel, Sphere) with dimensionality of 30 and 100. Some real applications were performed such as: initialization of multilayer perceptron network to help convergence, to find the angles in protein folding problem. In the first experiments, there was a comparison of all the models using statistical measurements(mean, standard deviation), execution time. According to the results, the most robust model was the model that used Akima interpolation, and the addition of the best individuals during iteration to the whole generation. The results showed that the proposal has better performance than initial QIEA-$\mathbb{R}$ when applied in numerical optimization problems.

Keywords: Quantum Inspired Evolutionary Algorithm, Evolutionary Computation, Optimization, Probability Density Function .

Resumen

En este trabajo se propone, implementa y evalúa el modelo *Quantum Inspired Evolutionary Algorithm with Real Representation using Filter Particle* (FP-QIEA-R); este modelo usa la generación clásica del modelo *Quantum Inspired Evolutionary Algorithm with Real Representation* (QIEA-R) (uso de función de distribución de probabilidad uniforme) y propone la generación clásica usando un mecanismo inspirado en filtro de partículas, aproximación de funciones, recompensa de los mejores individuos y muestreo usando funciones de distribución de probabilidad para la búsqueda global y centroides para la búsqueda local. Durante el progreso de este trabajo fueron evaluados varios métodos de estimación de funciones: uni-dimensionales (*splines*, interpolación de akima), multi-dimensionales (regresión multilineal, *parzen window*) para estimar la función de distribución acumulada(modificada usando el criterio de recompensa). Para evaluar el modelo, se realizaron experimentos con funciones *benchmark* (Ackley, Rastrigin, Rosenbrock, Schwefel, Sphere) usando una dimensionalidad de 30 y 100. Algunas aplicaciones reales fueron evaluadas: la inicialización de una red perceptrón multicapa para ayudar la convergencia(reducir el número de épocas), encontrar los ángulos en el problema de desdoblamiento de proteínas. En los primeros experimentos, todos los modelos fueron comparados usando medidas estadísticas(media,desviación estándar), tiempo de ejecución y de acuerdo a los resultados obtenidos el modelo más robusto fue el modelo que usa interpolación de akima y añade durante las generaciones a los mejores individuos. Los resultados obtenidos mostraron que la propuesta tiene el mejor desempeño tratando diversos problemas de optimización numérica comparado con el modelo existente QIEA-$\mathbb{R}$.

Palabras clave: Algoritmo Evolutivo de Inspiración Cuántica, Computación Evolutiva, Optimización, Función de Distribución de Probabilidad.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación y Contexto

Es común enfocarnos en la maximización de ingresos, beneficios, productividad, etc. y también la minimización de gastos, tiempo de operación, uso de recursos en la industria, educación y todas las áreas relacionadas a las actividades de hoy en día, en general en maximizar el beneficio. Este tipo de problemas son tratados por una de las ramas de la matemática aplicada: Optimización Global (Arnold, 2006).

Desde una visión simplificada, estos problemas se reducen a encontrar los valores óptimos en una función n-dimensional $f(x_1, \ldots, x_n)$ donde cada dimensión x_i tiene un dominio específico y puede estar sujeta a restricciones.

Existen los métodos tradicionales que logran encontrar soluciones exactas, requieren que se cumplan ciertas condiciones para ser utilizados, como crear el modelo matemático que incluye una función objetivo, restricciones y dominios para cada variable. Estas condiciones pueden cumplirse o no de acuerdo al problema. Todo esto puede significar un alto coste computacional y representar prolongados tiempos de espera para obtener una solución.

Por otro lado, existen los métodos heurísticos(métodos especializados en un problema en particular) y metaheurísticas(métodos de propósito general), que pueden hallar una solución de este tipo de problemas de forma aproximada, con menor coste computacional e inclusive sin exigir condiciones de optimalidad.

Entre las metaheurísticas, los algoritmos evolutivos están basados en el modelo de evolución de Darwin (van der Hauw, 1996). Los algoritmos evolutivos se pueden clasificar en tres grupos (Back et al., 1997): Programación evolutiva (Fogel et al., 1966), Estrategias evolutivas (Schwefel, 1993), Algoritmos genéticos (Bremermann, 1962),(Holland, 1975a).

Los algoritmos evolutivos usan una población (conjunto de individuos) y opera-

dores inspirados en el paradigma Neodarwiniano de la Evolución como la supervivencia del más apto y la herencia genética para "evolucionar" la población aproximándola hacia soluciones óptimas. Aunque estos algoritmos son ampliamente empleados en problemas de optimización, pueden presentar algunos problemas porque se necesita validar que las soluciones obtenidas estén sujetas a las restricciones del problema, y muchas veces esta tarea incrementa notablemente el costo computacional del proceso, impactando negativamente en su desempeño. También, se pueden presentar dificultades cuando existen problemas con características de *epistasia* (bajo grado de separabilidad de las variables) (da Cruz, 2007).

Un tipo más especializado de algoritmos evolutivos son los algoritmos evolutivos de inspiración cuántica, que traen al mundo evolutivo algunas ideas de la computación cuántica, como la superposición de estados. El primer modelo formal encontrado en la literatura es el *Genetic Quantum Algorithm* (GQA) (Han y Kim, 2000), que utiliza el q-bit como unidad de información, basándose en la representación binaria de los Algoritmos Genéticos Canónicos e incluyendo la superposición de estados. Usando como base este modelo, se pueden encontrar en la literatura disversos trabajos que se agrupan en 3 áreas: modificación en los operadores para mejorar el desempeño, híbridos y aplicaciones.

Cabe destacar que cualquier modelo basado en el modelo de Han mantendrá las mismas desventajas inherentes a la codificación binaria como: **aumento de la dimensionalidad** y complejidad en el proceso de **codificación y decodificación**.

Estos problemas son eliminados en el modelo QIEA-R (da Cruz et al., 2010), que se basa también en el concepto de q-bit y la superposición de estados, utiliza una representación real más parecida a las Estrategias Evolutivas. Este modelo es probado con funciones benchmark(funciones para evaluar robustez de los modelos) y comparado con otros algoritmos de búsqueda poblacional *Particle Swarm Optimization* (PSO), *Differential Evolution* (DE) mostrando un desempeño superior en general, pero también presentando dificultades con algunas funciones benchmark(funciones que son propuesta para evaluar este tipo de algoritmos) específicas. Se realizaron algunos trabajos de aplicación utilizando el modelo para el entrenamiento de redes neuronales (Escovedo et al., 2013, 2014).

El modelo usado por DaCruz **mantiene** la limitación de utilizar funciones de distribución de probabilidad uniforme y deja abierta la posibilidad de utilizar funciones de distribución de probabilidad que se adapten mejor a la naturaleza del problema. Pese a la limitación mencionada, el trabajo en el área es aún incipiente y abre la posibilidad para investigar.

1.2. Planteamiento del Problema

El modelo QIEA-R utiliza una representación real y tiene tres etapas: la generación de población cuántica, la generación de la población clásica, la actualización de la

población cuántica. Durante la generación de la población cuántica se inicializa a cada individuo cuántico utilizando una función de distribución de probabilidad uniforme por cada dimensión, dándole a cada elemento del dominio la misma probabilidad de ser seleccionado. A pesar de superar a los algoritmos PSO, DE; al utilizar la función de distribución acumulada(una recta) para la generación de individuos clásicos, el modelo está **limitado** por la naturaleza de la función de distribución de probabilidad uniforme y presenta dificultades si se tratan problemas con características multimodales o con características de epistasía. Esta situación nos lleva a pensar en la necesidad de contar con métodos más eficientes en la adaptación del comportamiento probabilístico(función de distribucion acumulada) para la generación de los individuos clásicos.

1.3. Objetivos

Proponer, implementar y evaluar un modelo de algoritmo de inspiración cuántica con representación real usando un mecanismo inspirado en filtro de partículas y métodos de aproximación de funciones para representar la función de distribución de probabilidad acumulada modificada usando el criterio de recompensa para la generación de individuos clásicos.

1.3.1. Objetivos Específicos

- Analizar el modelo QIEA-$\mathbb{R}$.
- Modelar un mecanismo eficiente de generación de individuos clásicos basado en filtro de partículas y métodos de aproximación de funciones.
- Implementar, realizar pruebas con funciones *benchmark* y aplicaciones reales.
- Analizar y evaluar los resultados obtenidos comparado con el modelo QIEA-$\mathbb{R}$.

1.4. Organización de la tesis

Este trabajo de tesis está organizado de la siguiente manera:

- El capítulo 2, contiene una revisión del marco teórico, los modelos existentes y el estado del arte.
- El capítulo 3, aborda la propuesta FP-QIEA-$\mathbb{R}$ inspirada en QIEA-$\mathbb{R}$, Filtro de Partículas y sus variantes utilizando diferentes métodos de aproximación de funciones.

- El capítulo 4, describe los experimentos, sus resultados y la discusión.
- El capítulo 5, finaliza con las conclusiones, limitaciones y trabajos futuros.

Capítulo 2

Marco teórico

A continuación se describen los conceptos básicos que serán utilizados en los siguientes capítulos que son parte de la investigación realizada y los trabajos relacionados a la investigación.

2.1. Optimización Global

Optimización global es la rama de la matemática aplicada y análisis numérico que se enfoca en la optimización. El objetivo es encontrar los mejores posibles elementos x^* de un conjunto X de acuerdo a un conjunto de criterios (funciones objetivo) $F = \{f_1, f_2, \ldots, f_n\}$ (Weise, 2008).

Definición 1 *Una función objetivo $f : X \mapsto Y \subseteq R$ es una función a optimizar. Su codominio Y y su rango deben ser un subconjunto ($Y \subseteq R$). El dominio de X puede ser cualquier tipo de elemento (números, listas,etc.), de acuerdo al problema de optimización. Las funciones objetivos no son necesariamente expresiones matemáticas pueden ser algoritmos complejos que involucran numerosas simulaciones.*

2.2. Heurísticas

Definición 2 *Se denomina heurística a un proceso que puede resolver un problema específico, pero que no ofrece ninguna garantía de lograrlo.*

Una definición más precisa y adecuada es la proporcionada por (Reeves, 1993). "Una heurística es una técnica que busca soluciones buenas (es decir, casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u optimalidad de las mismas. En algunos casos, ni siquiera puede determinar cuán cerca del óptimo se encuentra una solución factible en particular."

2.2.1. Metaheurísticas

Son algoritmos de propósito general, a diferencia de las heurísticas que son utilizados para problemas específicos. Son utilizados para resolver problemas complejos, en los últimos años han surgido como alternativa a los métodos clásicos (Bianchi et al., 2009).

2.2.2. Computación Evolutiva

Son algoritmos de optimización metaheurísticos basados en población y usan mecanismos inspirados en la biología como mutación, cruzamiento, selección natural y sobrevivencia del más apto para refinar un conjunto de soluciones candidatas iterativamente (Dortmund, 1995; Baeck et al., 1997).

2.2.2.1. Los principios básicos de la naturaleza

En 1859, Charles Darwin publicó su libro "El origen de las especies" e identificó los principios de *selección natural* y *sobrevivencia del más apto* como las fuerzas que dirigen la evolución biológica. Su teoría puede ser resumida en 10 observaciones y deducciones:

1. Los individuos de las especies tienen gran fertilidad y producen más descendencia que puede crecer en la adultez.
2. Durante la ausencia de influencias externas(desastres naturales, seres humanos, etc.), el tamaño de la población aproximadamente permance constante.
3. Si no hay influencias externas, los recursos alimenticios son limitados pero estables en el tiempo.
4. Cuando los individuos compiten por esos recursos limitados, una lucha empieza.
5. Especialmente en especies con reproducción sexual, no hay dos individuos iguales.
6. Algunas de las variaciones entre los individuos afectará su aptitud y su habilidad para sobrevivir.
7. Muchas de estas variaciones son heredables.
8. Los individuos menos aptos tienen menor posibilidad de reproducirse en contraste, los más aptos tienen mayor probabilidad de sobrevivencia y de reproducirse.
9. Los individuos que sobreviven y se producen pasarán sus características a su descendencia.
10. Las especies lentamente cambiarán y adaptarán más y más al ambiente durante este proceso y al final pueden resultar en nuevas especies.

Los algoritmos evolutivos se basan en el proceso biológico e introducen un cambio en la semántica por ser *dirigidos por el objetivo*. Las soluciones candidatas de cierto problema juegan el rol de individuos. Su aptitud es medida de acuerdo a las funciones objetivos del problema de optimización y dirige la evolución. La ventaja de los algoritmos evolutivos es que sólo se necesitan pocos supuestos sobre la función de aptitud para alcanzar una solución y por tanto tienen un buen desempeño en diferentes clases de problemas.

2.2.3. Algoritmos genéticos

En los años 70, gracias a Holland (Holland, 1975b) comenzaron las ideas de los algoritmos que después Goldberg aprovechó para presentar su modelo de algoritmo genético (Goldberg, 1989), basado en conceptos de biología y genética. Una población inicial experimenta procesos biológicos (mutación, cruzamientos) de forma aleatoria, además de un proceso de selección natural dónde el más apto sobrevive durante las generaciones.

El algoritmo tiene el siguiente proceso:

- Comienza con un conjunto de individuos con representación binaria de acuerdo al problema a resolver y luego son evaluados para determinar quién es mejor
- luego se realizan operaciones de cruzamiento siguiendo algún criterio de cruce, esperando obtener un mejor individuo
- luego se realizan operaciones de mutación, dónde un 0 cambia 1 o viceversa
- finalmente se obtiene una nueva población y según un criterio de selección natural(sobrevivencia del más fuerte) los más aptos pasan a la siguiente generación e iteran hasta alcanzar una solución o un criterio de parada.

Podemos notar que la representación binaria es insuficiente para problemas de toda clase, entonces uno de los desafíos en estos algoritmos es plantear una representación adeacuada al tipo de problema sea de dominio binario, real, etc.

Además este tipo algoritmos puede quedar atrapado en óptimos locales(soluciones no globales) y para tal problema, utiliza operaciones de cruce y mutación para realizar una mejor búsqueda en el espacio de soluciones.

Debemos recordar que algoritmos genéticos es una metaheurística(algoritmo de propósito general) y que durante las generaciones, sólo nos presentará una solución al problema que no podría cumplir la optimalidad(mejor solución existente en todo el dominio).

A pesar de las limitaciones mencionadas ha sido usado ampliamente en diversos problemas y áreas relacionadas o no a la computación.

2.2.4. Computación Cuántica

Un computador cuántico es un dispositivo que hace uso de ciertos fenómenos de mecánica cuántica para realizar operaciones con datos (Spector, 2006). El término *computación cuántica* es usado para describir procesos computacionales que se basen en estos fenómenos, son capaces de disminuir el esfuerzo y la complejidad computacional para resolver determinados problemas. Lo principal de esta área es el poder de procesamiento y la afirmación "las posibilidades cuentan, aunque no ocurran" (Spector, 2006).

Definición 3 *El bit es la menor unidad de información en un computador clásico, cuyos valores pueden ser '0' o '1'. En un computador cuántico, la unidad de información básica ser llama q-bit. Puede tener los estados '0', '1' o una sobreposición de los dos y puede ser representada:*

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

Donde $|\psi\rangle$ es el estado del q-bit y α, β son números complejos que determinan las amplitudes de probabilidad de los estados correspondientes.

Y $|\alpha^2|$ indica la probabilidad del q-bit de ser encontrado en el estado "0", $|\beta^2|$ indica la probabilidad del qubit de ser encontrado en el estado "1". La normalización garantiza:

$$\left|\alpha^2\right| + \left|\beta^2\right| = 1 \tag{2.2}$$

Si hay un sistema de m q-bits, el sistema puede representar 2^m estados al mismo tiempo. Sin embargo, en la observación de un estado cuántico se limita a un sólo estado.

2.2.5. Algoritmos con inspiración cuántica

A pesar de la ventaja existente al usar computadores cuánticos existen dos limitaciones: la dificultad de implementar un computador cuántico y crear algoritmos que aprovechen la capacidad de estos computadores. Por otro lado, Moore plantea aprovechar los conceptos de la física cuántica en lugar de usar computadores cuánticos, para mejorar el desempeño de los algoritmos clásicos y este es el origen del término *inspiración cuántica*. Moore presenta la siguiente metodología para la formulación de un algoritmo de inspiración cuántica:

1. El problema debe tener una representación numérica.
2. La configuración inicial debe ser definida(parámetros).

3. Definir una condición de parada.
4. El problema debe ser divisible en sub-problemas menores.
5. El número de universos(dominio por cada variable) debe ser identificado.
6. Cada sub-problema debe estar asociado a uno de los universos.
7. Los cálculos de los diversos universos deben ser independientes.
8. Alguna forma de interacción entre los universos debe existir.

2.2.6. QIEA-$\mathbb{B}$

Es el primer algoritmo evolutivo de inspiración cuántica que fue bien definido, que en la literatura original se denomina QEA (Han y Kim, 2000) pero que denominaremos QIEA-$\mathbb{B}$ de ahora en adelante. QIEA-$\mathbb{B}$ es inspirado en el concepto de computación cuántica, el modelo está diseñado con una representación q-bit, un operador de actualización Q-gate y observación. Se explica la representación y el algoritmo a continuación:

2.2.6.1. Representación

Diferentes representaciones pueden ser usadas para codificar las soluciones de los individuos de computación evolutiva. Las representaciones pueden ser clasificadas como: binaria, numérica y simbólica (Hinterding, 1999). El modelo usa una representación usando q-bits para la representación probabilística basada en el concepto de q-bit.

Definición 4 (q-bit) *Un Q-bit es la unidad de información más pequeña en QIEA-$\mathbb{B}$, y es definido como un par de parámetros (α, β)*

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

donde se cumple $|\alpha|^2+|\beta|^2 = 1$. $|\alpha^2|$ indica la probabilidad del q-bit de encontrarse en el estado "0", $|\beta^2|$ indica la probabilidad del q-bit de encontrase en el estado "1". Un Q-bit puede estar en el estado "0", "1", o una superposición lineal de los dos.

Definición 5 (Individuo cuántico) *Un individuo cuántico es una cadena de m q-bits, y es definido como:*

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix}$$

donde se cumple $|\alpha_i|^2 + |\beta_i|^2 = 1$, $\forall i = 1, 2, \dots, m$.
Con esta definición, se logra que cada individuo quántico represente una superposición de individuos clásicos formados por m Q-bits(genes).

Definición 6 *Operador Q-gate*

La actualización de la población es realizada por el operador Q-gate, es definida como una matriz de rotación.

$$U(\Delta\theta_i) = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \tag{2.3}$$

$U(\Delta\theta_i)$ multiplica cada columna del individuo, cada par de valores α y β, son tratadas como un vector de 2 dimensiones y serán operados usando $U(\Delta\theta_i)$, como se muestra en la imagen 2.1.

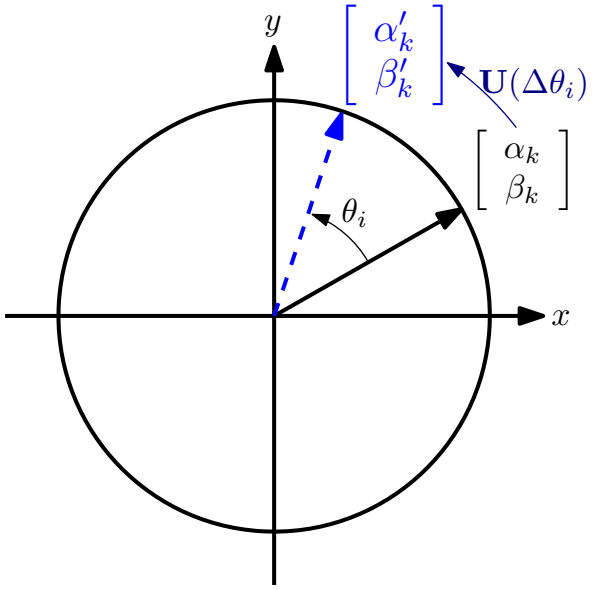


Figura 2.1: Rotación de un q-bit usando Q-gate

La actualización de este individuo es realizada por cada gen, de acuerdo con la ecuación:

$$\begin{bmatrix} \alpha'_i \\ \beta'_i \end{bmatrix} = U(\Delta\theta_i) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{2.4}$$

$\Delta\theta_i$ es definido de acuerdo al problema que se desee resolver, modificando α y β para dirigir a los individuos a soluciones más óptimas (Han y Kim, 2000).

2.2.6.2. Algoritmo: QIEA-$\mathbb{B}$

El procedimiento del QIEA-$\mathbb{B}$ propuesto por (Han y Kim, 2000), es descrito en el algoritmo 1:

Algorithm 1 Algoritmo evolutivo cuántico

```
t ← 0
Inicializar Q(t)
Generar P_t observando los estados de Q_t
Evaluar P_t
Guardar las mejores soluciones de P_t en b_t
while condición de parada no sea alcanzada do
    t ← t + 1
    Generar P_t observando los estados de Q_{t−1}
    Evaluar P_t
    Guardar la mejor solución de P_t en b_t
    b_t ← mejor entre b_t y b_{t−1}
    Actualizar Q_t usando Q-gates
end while
```

donde: Q_t es la población, P_t es una población de individuos clásicos generada en cada iteración. En la inicialización de Q_t, sus individuos son inicializados con α_i y β_i con $\frac{1}{\sqrt{2}}$, $\forall i = 1, 2, \ldots, m$ para tener la misma probabilidad de generar los estados "0"ó "1". Y b_t es usado para guardar los mejores en cada iteración.

El proceso del modelo es el siguiente:

Figura 2.2: Modelo QIEA-$\mathbb{B}$

2.2.7. QIEA-$\mathbb{R}$

El modelo QIEA-$\mathbb{R}$ utiliza codificación real, fue propuesto por (da Cruz, 2007) y fue la primera en utilizar representación real en lugar de binaria (Han y Kim, 2000). Tiene los siguientes pasos: generación de población cuántica, generación de población clásica al observar la población cuántica y actualización de la población cuántica.

Algorithm 2 QIEA-R

Iniciar

1: $t \leftarrow 1$
2: Generar población cuántica Q(t) con N individuos con G genes.
3: **while** $t \leq T$ **do**
4: E(t) $\leftarrow$ generar individuos clásicos observando individuos cuánticos
5: **if** t = 1 **then**
6: C(t) $\leftarrow$ E(t)
7: **else**
8: E(t) $\leftarrow$ recombinación entre E(t) y C(t)
9: evaluar E(t)
10: C(t) $\leftarrow$ K mejores individuos de [E(t) $\cup$ C(t)]
11: **end if**
12: Q(t+1) $\leftarrow$ actualiza Q(t) usando N mejores individuos de C(t)
13: $t \leftarrow t + 1$
14: **end while**

2.2.7.1. Algoritmo QIEA-R

El presente algoritmo se detallará en las secciones siguientes.

2.2.7.2. Representación Cuántica

El individuo cuántico representa la superposición de los posibles estados. En este modelo, el conjunto de estados observables es continuo.

- Sean $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ los individuos cuánticos de la población $\mathbf{Q}_t$ en la generación t,
- Cada individuo cuántico $\mathbf{q}_i$ está compuesto por n genes, $\mathbf{q}_{ij} = \{q_{i1}, \ldots, q_{in}\}$,
- Cada gen q_{ij} está definido por una función de distribución de probabilidad $p_{ij}(x)$. Esta función de distribución de probabilidad es inicializada cubriendo todo el dominio de q_{ij}
- Basado en esta definición, un individuo cuántico puede ser representado como:

$$\mathbf{q}_i = \{p_{i1}(x), p_{i2}(x), \ldots, p_{in}(x)\} \tag{2.5}$$

La función de distribución de probabilidad utilizada en (da Cruz et al., 2010) es la función de densidad de probabilidad uniforme. Esta función se define por la ecuación:

$$p_{ij}(x) = \begin{cases} \frac{1}{U_{ij} - L_{ij}} & si \quad L_{ij} \leq x \leq U_{ij} \\ 0 & otro \end{cases} \tag{2.6}$$

Donde L_{ij} y U_{ij} son los límites inferior y superior respectivamente del intervalo que el i-ésimo individuo cuántico puede asumir en la dimensión j cuando es observado.

Un ejemplo de un gen cuántico formado por un pulso cuadrado es mostrado a continuación:

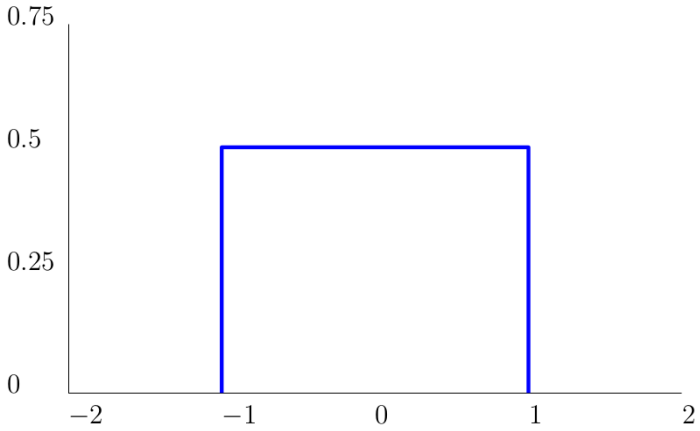


Figura 2.3: Ejemplo de un gen cuántico en QIEA-R, dónde el eje x es el dominio de la variable y el eje y es la probabilidad (da Cruz, 2007)

En este caso los límites están definidos por -1 y 1.

Da cruz propone dos formas de representación (da Cruz, 2007):

- Crear pulsos cuadrados con un ancho $\frac{U_{ij}-L_{ij}}{N}$, con sus centros distribuidos uniformente a lo largo de todo el dominio, donde N es el número de individuos cuánticos.
- Crear los pulsos con límite superior e inferior iguales a los límites del dominio.

En las siguiente figuras se pueden apreciar ejemplos de ambas opciones, considerando una población de tamaño 5, 2 dimensiones y un dominio de -100 a 100 por cada dimensión :

$$Q(0) = \begin{bmatrix} q_1 = [(\mu = -80, \sigma = 40); (\mu = -80, \sigma = 40)] \\ q_2 = [(\mu = -40, \sigma = 40); (\mu = -40, \sigma = 40)] \\ q_3 = [(\mu = 0, \sigma = 40); (\mu = 0, \sigma = 40)] \\ q_4 = [(\mu = 40, \sigma = 40); (\mu = 40, \sigma = 40)] \\ q_5 = [(\mu = 80, \sigma = 40); (\mu = 80, \sigma = 40)] \end{bmatrix} \quad Q(0) = \begin{bmatrix} q_1 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_2 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_3 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_4 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_5 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \end{bmatrix}$$

Figura 2.4: Propuestas de inicialización (da Cruz, 2007)

Una vez inicializada la población cuántica $\mathbf{Q}_0$(población inicial cuántica), el ciclo principal del proceso evolutivo empieza y continua de la siguiente forma:

2.2.7.3. Observación

La generación de individuos clásicos observando individuos cuánticos usando la función de distribución de probabilidad $p_{ij}(x)$, probabilidades acumuladas P_{ij} y un generador de números aleatorios $\mathbf{U}(0,1)$ en el dominio de 0 a 1 siguiendo el procedimiento presentado a continuación:

1: **Generar** $r \sim \mathbf{U}(0,1)$
2: **Encontrar** x tal que

$$P_{ij}(x) = \int_{-\infty}^{\infty} p_{ij}(\tau)d\tau \tag{2.7}$$

$$x = P_{ij}^{-1}(r) \tag{2.8}$$

3: **Asignar** $x_{ij}(t) \in \mathbf{x}_i \leftarrow x$

Si se usan pulsos (μ_{ij} , σ_{ij}) y una realización $r_{ij} \sim U(0,1)$, el valor del gen clásico se obtiene por:

$$x_{ij} = r_{ij} * \sigma_{ij} + \left(\mu_{ij} - \frac{\sigma_{ij}}{2}\right) \tag{2.9}$$

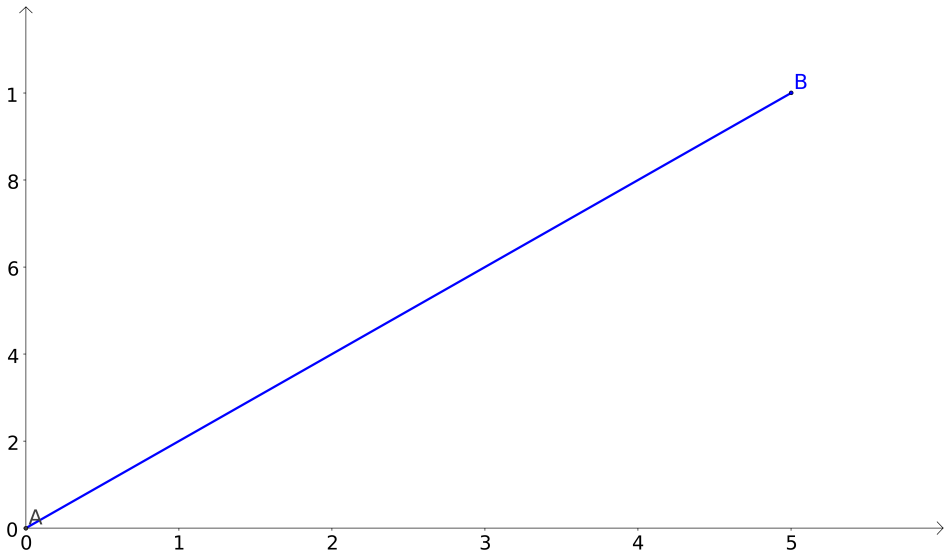


Figura 2.5: Ejemplo de función de distribución acumulada con $\mu = 2{,}5$ y $\sigma = 2{,}5$ dónde el eje x es el dominio de la variable y el eje y es la probabilidad de la función acumulada

2.2.7.4. Actualización

La actualización tiene dos etapas: ajuste del ancho del pulso y movimiento del centro del pulso.

- Ajuste ancho del pulso
  QIEA-$\mathbb{R}$ puede reducir o incrementar el espacio de búsqueda de cualquier individuo cuántico de acuerdo a la aptitud de la población clásica durante el proceso de observación y usando la regla de 1/5"(Rechenberg, 1973). Si menos del 20 % de la población clásica de la actual generación tiene una aptitud mejor que la generación anterior, el ancho del gen es reducido, si es mayor que 20 %, el ancho es incrementado y si es igual, el ancho no es modificado, la ecuación 2.10, muestra la regla de 1/5".

$$\sigma_{ij} = \begin{cases} \sigma ij * \delta & \varphi < 1/5 \\ \sigma_{ij}/\delta & \varphi > 1/5 \\ \sigma_{ij} & \varphi = 1/5 \end{cases} \tag{2.10}$$

  donde φ representa el porcentaje de mejores individuos de la población en comparación a la población anterior.

- Movimiento del centro del pulso
  Escoger los mejores individuos clásicos para actualizar la población cuántica. Por ejemplo, suponer que el centro del gen cuántico en la generación t es $\mu_{ij}(t)$ y el valor del mejor gen clásico es x_{ij} entonces la nueva posición del gen cuántico es calculada para la generación $t+1$ por la ecuación 2.11:

$$\mu_{ij}(t+1) = \mu_{ij}(t) + \lambda(x_{ij} - \mu_{ij}(t)) \tag{2.11}$$

  donde $\lambda \in [0,1]$, es el porcentaje de movimiento en dirección al gen clásico.

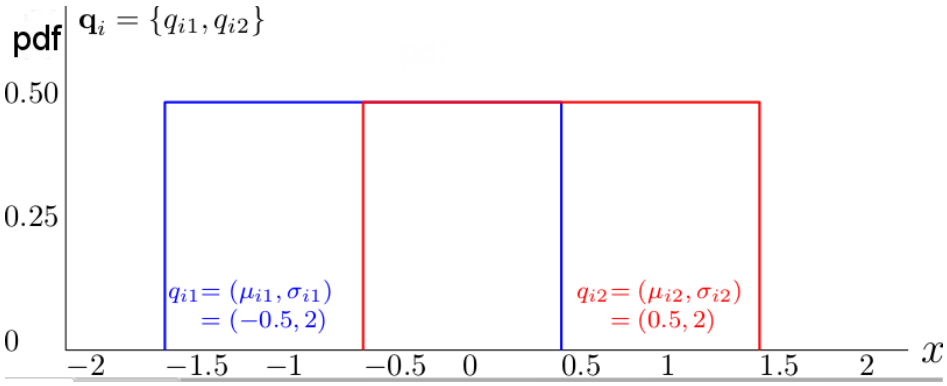


Figura 2.6: Ejemplo de movimiento de gen cuántico en una dimensión donde el eje x es el dominio de la variable y el eje y es la probabilidad

2.3. Aproximación de funciones

Es la rama de análisis numérico que investiga como ciertas funciones conocidas pueden ser aproximadas por una clase de funciones específica (funciones polinomiales) que tienen propiedades de cálculo de bajo coste, continuidad y otros. Y en un segundo caso, en lugar de tener una fórmula sólo tener un conjunto de puntos de la forma $(x, f(x))$ utilizar técnicas de interpolación, extrapolación, regresión y ajuste de curvas.

2.3.1. *Splines*

Splines es una función numérica definida en partes por funciones polinomiales que son suavizadas en los lugares donde las piezas polinomiales se conectan por puntos (Judd, 1998).

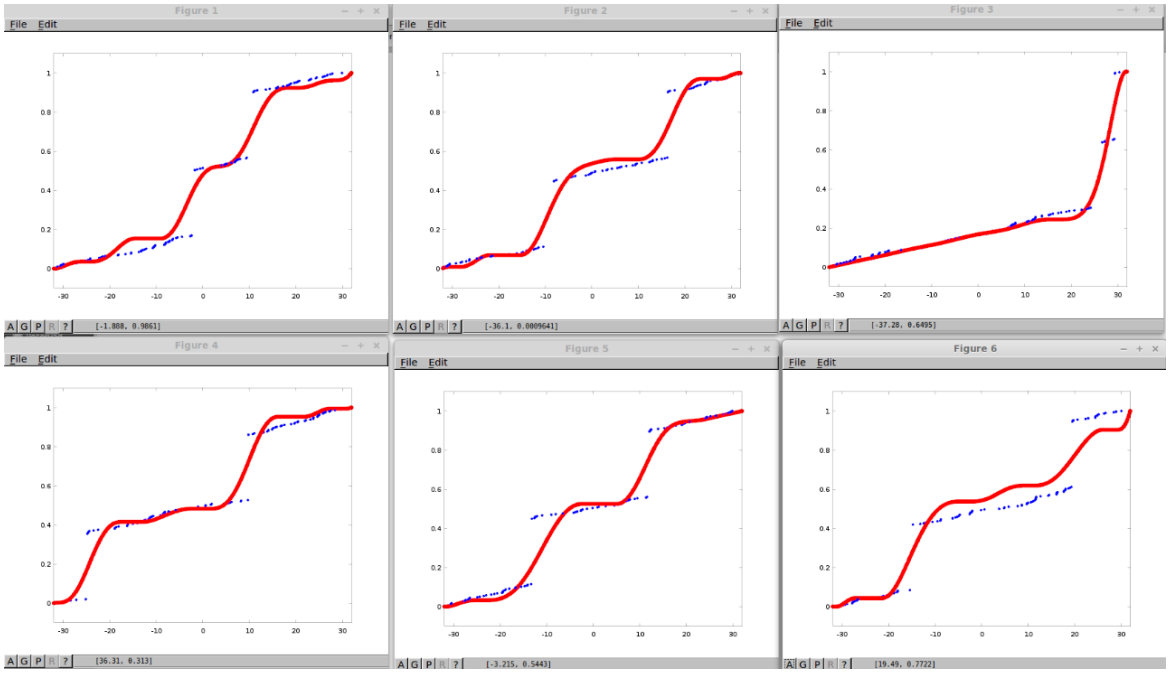


Figura 2.7: Aproximación de función de distribución acumulada usando *splines*

2.3.2. Akima

La interpolación de Akima es una interpolación sub-spline continua y diferenciable. Es construida por partes a partir de polinomios de tercer grado. Sólo los datos de los puntos vecinos próximos son usados para determinar los coeficientes del polinomio de interpolación. No hay necesidad de sistemas de ecuaciones y por tanto es computacionalmente más eficiente. Al usar un número reducido de puntos se evitan curvas extrañas en la curva estimada (Akima, 1970).

Para un conjunto de 4 puntos:

$$s_i = s(x_i), 1 \leq i \leq k, \tag{2.12}$$

La función de interpolación es definida como:

$$s(x) = a_0 + a_1(x - x_i) + a_2(x - x_i)^2 + a_3(x - x_i)^3 x_i \leq x \leq x_{i+1} \tag{2.13}$$

Para determinar los coeficientes a_0, a_1, a_2 y a_3 del polinomio de interpolación para cada intervalo $[x_i, x_{i+1}\ ]$, los valores de la función s_i y s_{i+1} y las primeras derivadas s_i' y s_{i+1}' en los puntos finales del intervalo son usados.

La primera derivada s_i' de la función de interpolación en x_i es estimada desde los datos de este punto y los siguientes dos puntos a cada lado de x_i. Usando las proporciones:

$$d_j = \frac{s_{j+1} - s_j}{x_{j+1} - x_j}, j = i-2, i-1, i, i+1 \tag{2.14}$$

y los coeficientes de ponderación

$$w_{i-1} = |d_{i+1} - d_i|, w_i = |d_{i-1} - d_{i-2}| \tag{2.15}$$

la derivada estimada s_i' es definada como:

$$s'_i = \frac{w_{i-1}d_{i-1} + w_i d_i}{w_{i-1} + w_i} \tag{2.16}$$

Diferentes casos especiales para s_i' deben ser considerados.

$$\begin{aligned} s'_i = d_{i-1}, d_{i-2} = d_{i-1}, d_i \neq d_{i+1} \\ s'_i = d_i, d_i = d_{i+1}, d_{i-2} \neq d_{i-1} \\ s'_i = d_{i-1} = d_i, d_{i-1} = d_i \\ s'_i = \frac{d_{i-1} + d_i}{2}, d_{i-2} = d_{i-1} \neq d_i = d_{i+1} \end{aligned} \tag{2.17}$$

Para usar 2.3.2 para el cálculo de las derivadas s'_1, s'_2, s'_{k-1} y s_k' ratios adicionales d_{k-1}, d_0, d_k y d_{k+1} deben ser estimados.

$$\begin{aligned} d_{k-1} = 2d_0 - d_1 \\ d_0 = 2d_1 - d_2 \\ d_k = 2d_{k-1} - d_{k-2} \\ d_{k+1} = 2d_k - d_{k-1} \end{aligned} \tag{2.18}$$

El grado de la función de interpolación se reduce a 2 para estos intervalos.

A continuación podemos observar algunos ejemplos:

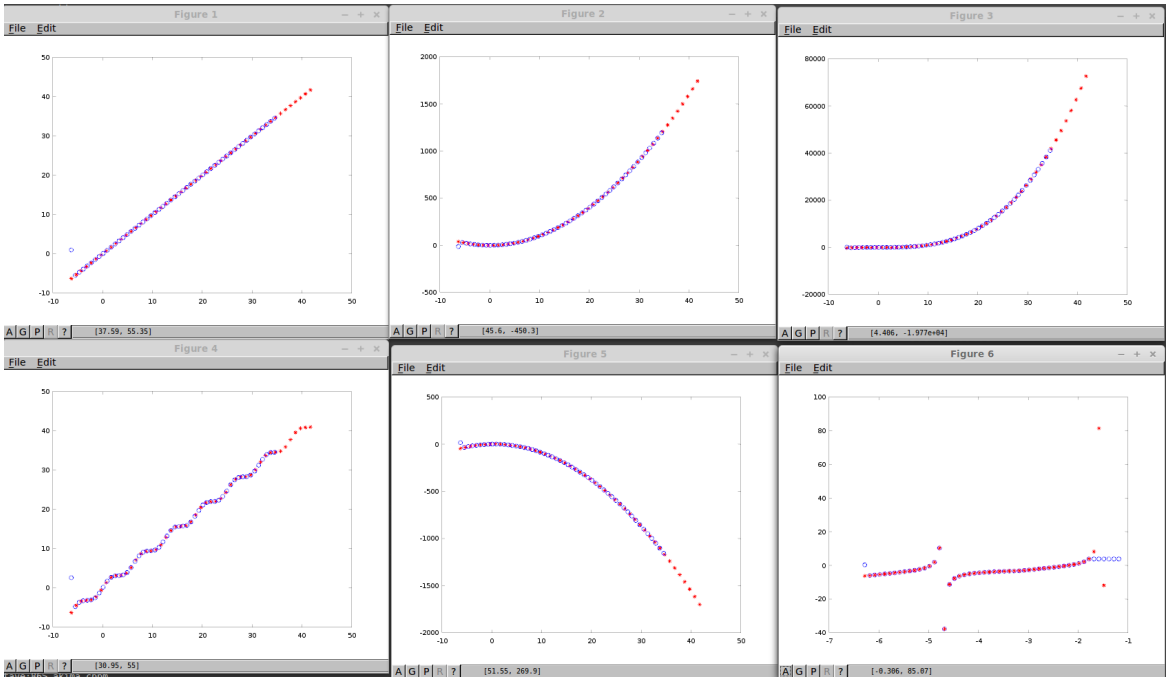


Figura 2.8: Aproximación de funciones usando interpolación de akima

2.4. Aproximación Multidimensional

2.4.1. Regresión multilineal

Regresión lineal es un enfoque para modelar la relación entre una variable dependiente y una o más variables independientes. En caso de una variable es llamado regresión lineal simple, para más de una variable es llamado regresión multilineal.

Sean p variables independientes relacionadas por la ecuación 2.19:

$$y_i = \beta_1 x_{i1} + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} \tag{2.19}$$

donde x_{ij} es una observación de la variable independiente y_i y asume valor 1 para la primera variable independiente, para cada i.

Las ecuaciones normales en su forma matricial son:

$$(X^T X)\widehat{\beta} = X^T Y \quad \widehat{\beta} = (X^T X)^{-1} X^T Y \tag{2.20}$$

2.4.2. Parzen-window Density Estimation

El modelo ventana de Parzen (Duda et al., 2000) es utilizado para estimar densidades y asume que la region R_n es un hipercubo d-dimensional. Si h_n es la longitud de arista del hipercubo, entonces su volumen está dado por:

$$V_n = h_d^d \tag{2.21}$$

Podemos obtener una expresión para k_n, el número de muestras que estén dentro del hipercubo y definimos la siguiente función ventana:

$$\varphi(u) = \begin{cases} 1 & |u_j| \geq 1/2 \quad j = 1, \ldots, d \\ 0 & otro \quad caso \end{cases} \tag{2.22}$$

De esta forma, $\varphi(u)$ define un hipercubo con centro en el origen.

Y $\varphi((x - x_i)/h_n)$ es igual a la unidad si x_i está dentro del hipercubo de volumen V_n con centro en x y zero en otro caso. El número de muestras en este hipercubo está dado por:

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{x - x_i}{h_n}\right) \tag{2.23}$$

y la probabilidad está dada por:

$$p_n(x) = \frac{k_n/n}{V_n} \tag{2.24}$$

Reemplazando la ecuación 2.24 en 2.23 obtenemos:

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi(\frac{x - x_i}{h_n}) \tag{2.25}$$

La función ventana es usada para interpolación y debe cumplir con lo siguiente:

$$\varphi(x) \leq 0 \tag{2.26}$$

y

$$\int \varphi(u) du = 1, \tag{2.27}$$

y si mantenemos la relación $V_n = h_n^d$ entonces $p_n(x)$ satisface estas condiciones.

Examinemos el efecto del ancho de la ventana h_n que tiene en $p_n(x)$. Si definimos la función $\delta_n(x)$ por:

$$\sigma_n(x) = \frac{1}{V_n}\varphi\left(\frac{x}{h_n}\right) \tag{2.28}$$

entonces podemos escribir $p_n(x)$ como el promedio:

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\delta_n(x - x_i) \tag{2.29}$$

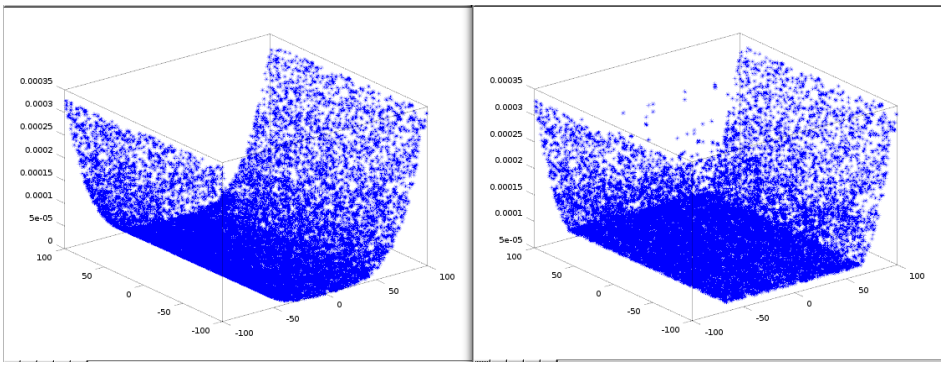


Figura 2.9: Aproximación usando window-parzen de función Rosenbrock, donde los ejes x, y son las variables y el eje z es la probabilidad

2.5. Funciones *benchmark*

Son funciones utilizadas para evaluar las siguientes características de algoritmos de optimización:

- velocidad de convergencia
- precisión
- robustez
- desempeño general

Pueden presentar uno o varios óptimos globales(mejor solución en todo el dominio) y varios óptimos locales(mejor solución en un subdominio).

2.5.1. Ackley

Esta función se caracteriza por una región casi plana y un gran agujero en el centro. La función tiene una dificultad para los algoritmos de optimización basado en hill climbing(algoritmo de búsqueda local) que pueden quedarse atrapados en un de sus óptimos locales (Ackley, 1987).

En la Fig. 2.10 podemos apreciar la función en 2 dimensiones.

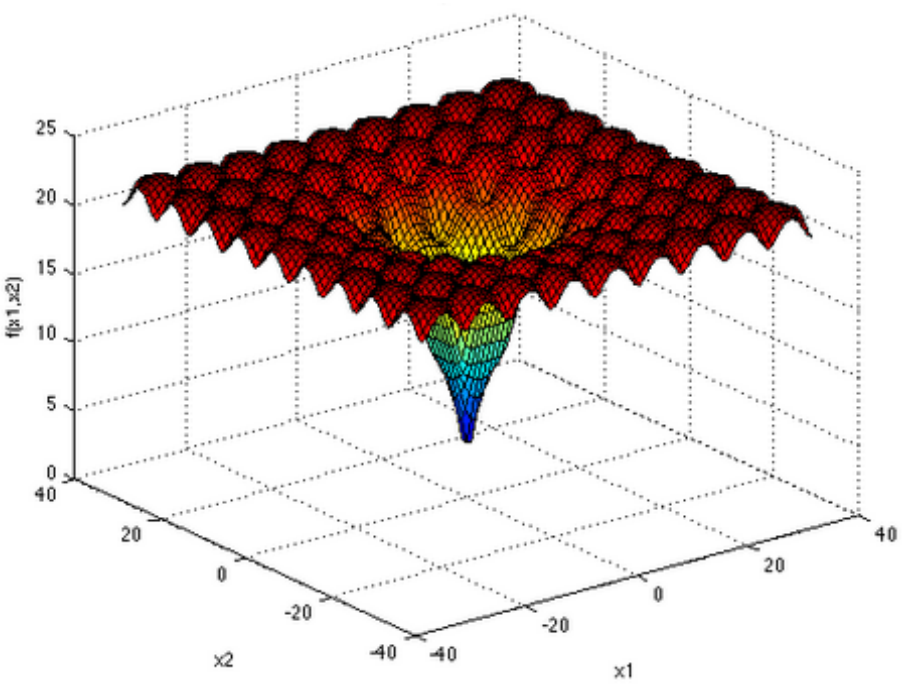


Figura 2.10: Función Ackley

La ecuación de la función Ackley:

$$f_1(x) = -a * \exp\left(-b \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1) \qquad (2.30)$$

Dominio y mínimo global

$x_i \in [-32{,}768, 32{,}768] \quad , \quad \forall i = 1, \ldots, d \qquad f_1(x*) = 0 \quad , \quad x* = (0, \ldots, 0)$

2.5.2. Rastrigin

La función Rastrigin es una función no convexa, tiene varios mínimos locales y es multimodal (Rastrigin, 1974).

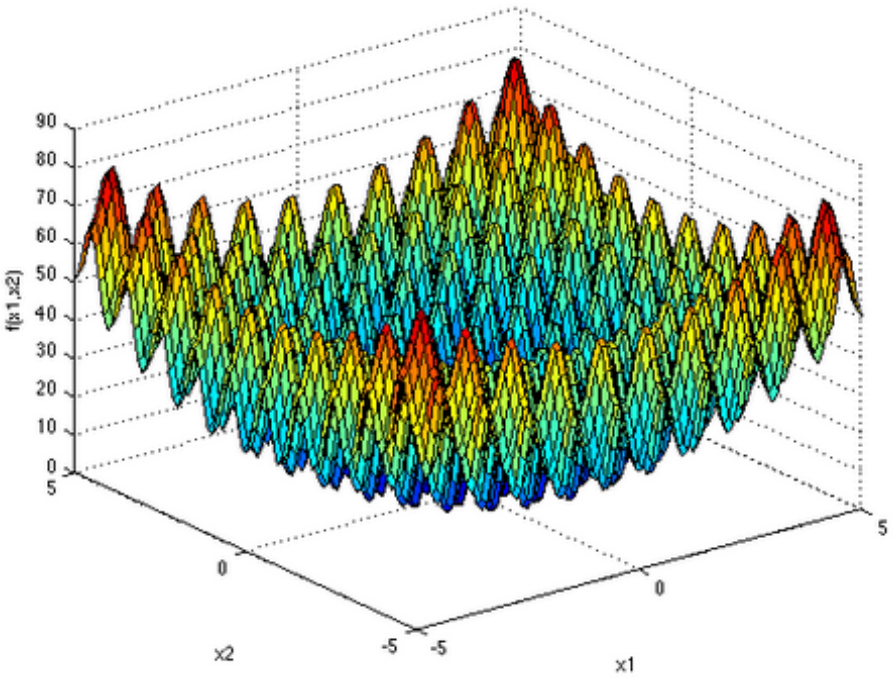


Figura 2.11: Función Rastrigin

La ecuación de la función Rastrigin:

$$f_2(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)] \tag{2.31}$$

Dominio

$x_i \in [-5{,}12, 5{,}12] \quad , \quad \forall i = 1, \ldots, d \qquad f_1(x*) = 0 \quad , \quad x* = (0, \ldots, 0)$

2.5.3. Rosenbrock

Esta función es unimodal y el mínimo global está en un valle estrecho similar a una parábola. Sin embargo, es fácil encontrar este valle pero converger al mínimo es difícil (Rosenbrock, 1960).

La ecuación de la función Rosenbrock:

$$f_3(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{2.32}$$

Dominio

$x_i \in [-5, 10] \quad , \quad \forall i = 1, \ldots, d \qquad f_1(x*) = 0 \quad , \quad x* = (1, \ldots, 1)$

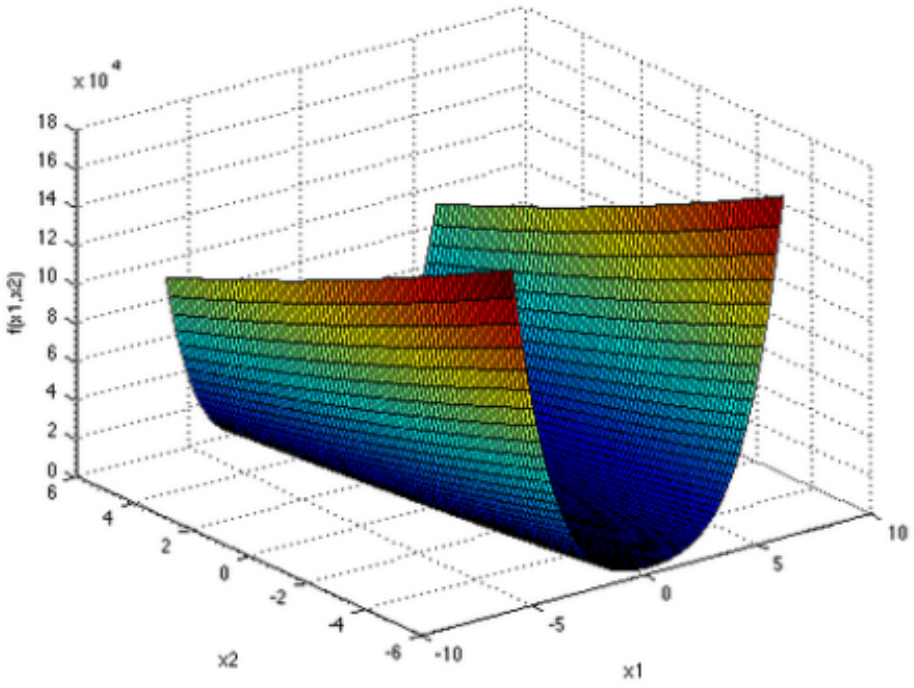


Figura 2.12: Función Rosenbrock

2.5.4. Función Schwefel

La función Schwefel es muy compleja con muchos mínimos locales (Schwefel, 1981). La figura 2.13 muestra la función en el dominio [-500, 500].

Un ejemplo de la función en 2 dimensiones puede ser apreciada en la Fig. 2.13.

La ecuación de la función Schwefel:

$$f_4(x) = 418{,}9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{|\ x_i\ |}) \tag{2.33}$$

Dominio

$x_i \in [-500, 500] \quad , \quad \forall i = 1, \ldots, d \qquad f_1(x*) = 0 \quad , \quad x* = (420{,}9687, \ldots, 420{,}9687)$

2.5.5. Función Sphere

Esta función es continua, convexa y unimodal (Dixon y Szego, 1978).

En la Fig. 2.14 podemos apreciar la función en 2 dimensiones.

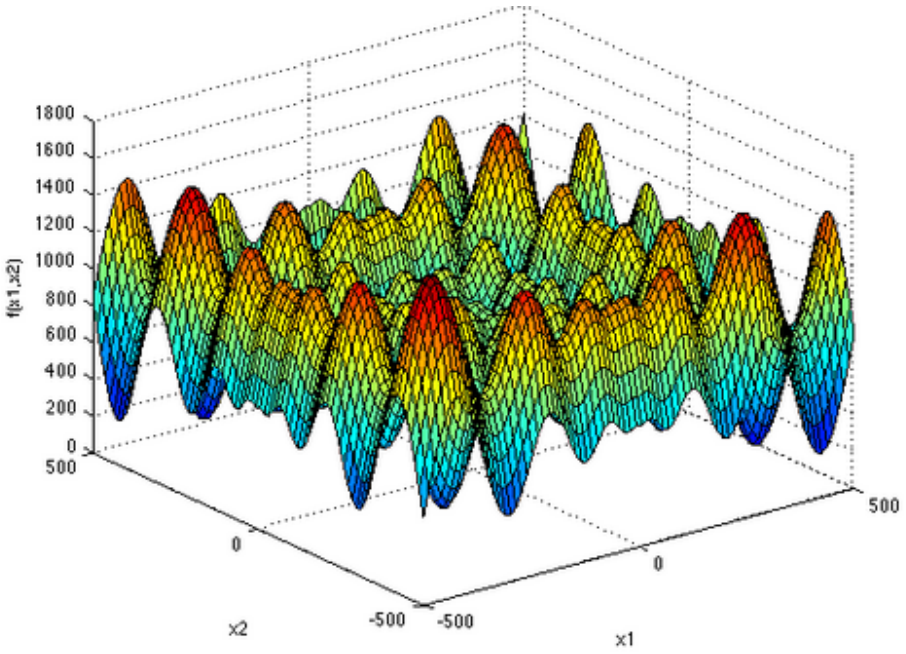


Figura 2.13: Función Schwefel

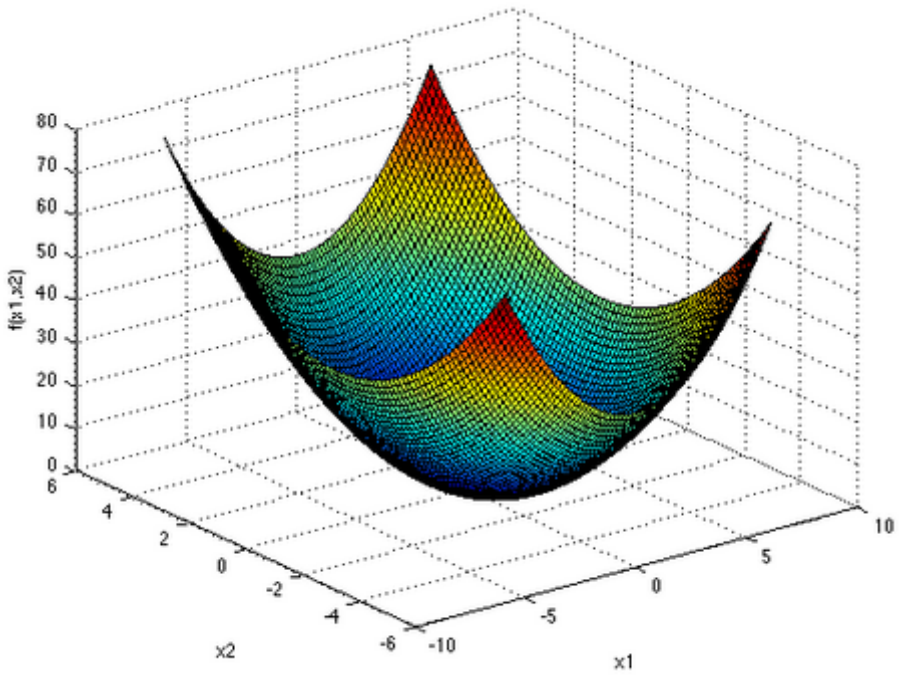


Figura 2.14: Función Sphere

La ecuación de la función Sphere:

$$f_5(x) = \sum_{i=1}^{d} x_i^2 \tag{2.34}$$

Dominio

$x_i \in [-5{,}12, 5{,}12] \quad , \quad \forall i = 1, \ldots, d \qquad f_1(x*) = 0 \quad , \quad x* = (0, \ldots, 0)$

2.6. Trabajos Relacionados

A continuación se presenta un resumen del estado del arte del algoritmo utilizado.

GQA (Han y Kim, 2000) es la primera propuesta inspirada en el concepto y principios de la computación cuántica usando *q-bits* y superposición de estados de la mecánica cuántica. Un *q-bit* puede estar en el estado '1' o '0' con determinada probabilidad. La suma de las probabilidades debe ser igual a 1. Un sistema de m *q-bits* puede representar 2^m estados al mismo tiempo. Un *q-bit* presenta características de exploración y explotación. En GQA, las etapas de observación y actualización son presentadas, los experimentos fueron realizados utilizando el problema de la mochila y los resultados muestran que QGA tiene la habilidad de búsqueda global por la representación probabilística que presenta y tiene una mejor convergencia que *Classical Genetic Algorithm* (CGA).

Podemos apreciar los resultados obtenidos:

Cuadro 2.1: Resultados de experimentos realizados por Han (Han y Kim, 2000) comparando algoritmos genéticos clásicos(CGA) y su propuesta con 100 y 250 items con 30 experimentos

		CGA	**QGA**
100	b	602.2	612.7
	m	593.6	609.5
	w	582.6	607.6
	σ	4.958	2.404
	t	0.786	0.2030

		CGA	**QGA**
250	b	1472.5	1525.2
	m	1452.4	1518.7
	w	1430.1	1515.2
	σ	10.324	2.910
	t	1.804	0.558

dónde b= best(mejor), m=median(mediana) y w = worst(peor), luego σ es la desviación estándar de las soluciones y t el tiempo de ejecución.

Podemos observar que la propuesta de Han supera al algoritmo genético clásico en sus medidas estadísticas y tiempo de ejecución.

Varios trabajos (Zhang y Gao, 2007; Liu y Liu, 2013; Iriyama y Ohya, 2014), (Le Gall, 2014) han sido presentados utilizando esta representación binaria y proponiendo mejoras de los operadores para mejorar el desempeño del modelo de Han. Existen propuestas híbridas utilizando PSO (Hossain et al., 2009),(Hao y Shiyong, 2012), (Zhao et al., 2013), utilizando criterios de sistemas inmunes (Hongjian et al., 2009), colonia de abejas (Bouaziz et al., 2013), utilizando inicialización basada en teoría del caos (Yanguang

et al., 2010). El modelo de Han es aplicado en diversas áreas: procesamiento de señales (Wu et al., 2009) clustering (Tsai et al., 2013), clasificación fuzzy (Nunes et al., 2013), problema clique (Das y Khan, 2015) entre otras.

A pesar de exitir varios trabajos después de la propuesta de Han, todos mantienen la representación binaria y preservan las desventajas relacionadas como: **mayor dimensionalidad**, **codificación y decodificación**.

El modelo QIEA-R (da Cruz et al., 2010) supera esta limitación, en este modelo el individuo cuántico está basado en una función de distribución de probabilidad uniforme y los individuos clásicos son arreglos de dimensión N. Una etapa de observación es propuesta de forma similar a *Quantum Inspired Evolutionary Algorithm with Binary Representation* (QIEA-B), se usa función de distribución de probabilidad uniforme para cada dimensión. Los pasos de actualización son: ampliar o reducir el pulso uniforme usando la regla 1/5"(Rechenberg, 1973) y movimiento del centro del pulso por cada dimensión. Los experimentos fueron realizados usando funciones *benchmark* (Griewank, Schwefel, Rosenbrock, Sphere, Ackley, Rastrigin y otras). Los resultados demostraron el buen desempeño del algoritmo comparado con QIEA-B.

Apreciamos en la siguiente tabla los resultados obtenidos (da Cruz et al., 2010), se extrajeron sólo las medidas estadísticas para una mejor comprensión.

Cuadro 2.2: Resultados de experimentos realizados por Abs da Cruz comparando algoritmos genéticos clásicos, QIEA-$\mathbb{B}$ y su propuesta con 25 experimentos y 30 dimensiones

Function		GA	QIEA-$\mathbb{B}$	QIEA-$\mathbb{R}$
Sphere	m	4.71	1.8E-04	**1.99E-06**
	σ	n.a.	1.3E-04	9.50E-06
Ackley	m	4.71E-01	2.5E-03	**4.26E-05**
	σ	n.a.	8.1E-04	1.39E-04
Griewank	m	1.03	3.6E-02	**9.42E-06**
	σ	n.a.	3.2E-02	1.55E-05
Rastrigin	m	4.40E-01	3.9E-02	**1.65E-11**
	σ	n.a.	1.9E-01	3.39E-11
Schwefel	m	1.77	3.8E-04	**8.1E-09**
	σ	n.a.	3.0E-09	0
Rosenbrock	m	14.15	11.73	**2.52**
	σ	n.a.	18.36	4.43

dónde m=*median* (mediana) y σ es la desviación estándar de las soluciones.

El algoritmo es mejor en medidas estadísticas comparando con algoritmos genéticos y la propuesta de Han.

Pero si consideramos la siguiente tabla dónde compara con PSO, DE, dónde se siguió el criterio anterio de visualización(sólo medidas estadísticas).

Cuadro 2.3: Resultados de experimentos realizados por Abs da Cruz comparando algoritmos genéticos clásicos, QIEA-$\mathbb{B}$ y su propuesta con 25 experimentos y 30 dimensiones

Function		DE	PSO	QIEA-$\mathbb{R}$
Ackley	ae	2.121E+01	**2.118E+01**	2.119E+01
	σ	6.801E-02	7.595E-02	6.070E-02
Rastrigin	ae	2.761E+02	**1.040E+02**	1.533E+02
	σ	2.175E+01	2.698E+01	2.069E+01

dónde *ae=average error* (error medio) y σ es la desviación estándar de las soluciones.

Los resultados no fueron los mejores pero si próximos al mejor y con una menor desviación estándar. Esto da indicios que el el uso de función de distribución uniforme no es suficiente para representar el comportamiento de las soluciones en problemas multimodales como Rastrigin y uni-modales como Ackley pero con alto número de óptimos locales.

Capítulo 3

Propuestas FP-QIEA-$\mathbb{R}$

En este capítulo se detallarán las propuestas realizadas:

- la primera utilizando el criterio de centroides para la generación de partículas con regresión multilineal para la estimación de la función de distribución acumulada
- la segunda utilizando *splines* para representar la función de distribución acumulada de cada dimensión para la generación de las partículas utilizando este criterio
- luego una tercera, utilizando la interpolación de Akima
- una cuarta utilizando *parzen window* para la estimación de la función de distribución acumulada
- finalmente una modificación de la tercera propuesta que usa interpolación de akima añadiendo los mejores individuos a la función de distribución acumulada.

Es importante mencionar que todas las propuestas usan una modificación de la función de distribución acumulada de acuerdo a los mejores individuos y no una uniforme como el modelo de Abs da Cruz.

3.1. Iteraciones combinadas FP-QIEA-$\mathbb{R}$ con aproximación multilineal

Este es el primer modelo propuesto, intercala el uso de la representación cuántica del modelo QIEA-$\mathbb{R}$ de la sección 2.2.7 y usa la representación de Filtro de Partículas, regresión multi-lineal, centroides, remuestreo y frecuencia relativa para obtener una nueva función de distribución de probabilidad para generar una población clásica y combinarla con la población obtenida por QIEA-R y obtener los n mejores individuos. El procedimiento es explicado a continuación:

3.1.1. Representación cuántica

Similar al modelo QIEA-ℝ, el individuo cuántico representa la superposición de posibles estados que una variable puede tomar. El conjunto de estados observables es continuo.

- Sea $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ individuos cuánticos de una población $\mathbf{Q}_t$ en la generación t,
- Cada inviduo cuántico $\mathbf{q}_i$ está compuesto por n genes, $\mathbf{q}_{ij} = \{q_{i1}, \ldots, q_{in}\}$,
- Cada gen q_{ij} está definido por una función de distribución de probabilidad $p_{ij}(x)$. Esta función de distribución de probabilidad es inicializada como un pulso uniforme cubriendo todo el dominio de q_{ij}.
- Bajo esta definición, un individuo cuántico puede ser representado po la ecuación 3.1

$$\mathbf{q}_i = \{p_{i1}(x), p_{12}(x), \ldots, p_{in}(x)\} \tag{3.1}$$

Una vez inicializada la población cuántica $\mathbf{Q}_0$, el ciclo principal del proceso evolutivo empieza y continua de la siguiente forma:

3.1.2. Observación

En esta etapa se escogió utilizar la generación de clásicos del modelo de Filtro de Partículas en cada 4 iteraciones debido a que su tiempo de ejecución aumentaba el tiempo de ejecución global.

3.1.2.1. Iteraciones que son múltiplos de 4

En esta iteraciones se hace la generación de individuos clásicos observando individuos cuánticos usando la función de distribución acumulada uniforme $p_{ij}(x)$, probabilidades acumuladas P_{ij} y un generador de números aleatorios $\mathbf{U}(0, 1)$ como en la sección 2.2.7.3.

3.1.2.2. Iteraciones que no son múltiplos de 4

La nueva población clásica es generada usando una nueva función de distribución acumulada creada por *Filtro de partículas* (FP). Un generador de números aleatorios es utilizado para obtener valores reales en $[0, 1]$. Este valor es usado para seleccionar una partícula aplicando un mecanismo similar a la selección proporcional de algoritmos evolutivos (Bäck, 1996), explicado abajo.

- De la nueva distribución obtenemos las probabilidades p_i de los individuos x_i .
- Calcular la probabilidad acumulada a de todos los individuos usando:

  $a_1 = p_1$
  for $i = 2$ to N **do**
  $a_i = a_{i-1} + p_i$
  end for

- Generar un número aleatorio $r = \mathbf{U}(0, 1)$ y seleccionar el primer individuo con probabilidad acumulada mayor a r r.

3.1.3. Actualización

Por la misma razón expresada en el la etapa de Observación, el modelo FP-QIEA-ℝ se actualiza cada 4 iteraciones.

3.1.3.1. Iteraciones que son múltiplos de 4

Utiliza el mismo criterio de actualización que QIEA-R.

3.1.4. Generación usando función de distribución de probabilidad

La propuesta de este trabajo usa FP, regresión multilineal, centroides y frecuencia relativa para generar una nueva función de distribución de probabilidad como se explica abajo.

Sean $\{x^i_{0:N}, w^i\}_{i=1}^{NP}$ donde: $\{x^i_{0:N}\}$ con $i = 0, \ldots, NP$, es un conjunto de partículas de dimensión N de tamaño NP con pesos asociados $\{w^i, i = 0, \ldots, NP\}$(representan la probabilidad de cada partícula).

Se evalúa las partículas utilizando la función a optimizar $f(x_i)$ y se recompensa al mejor valor, luego los pesos son normalizados para satisfacer $\sum_i w^i_k = 1$.

Para obtener una aproximación eficiente de la función de distribución de probabilidad generada en el paso anterior, se propone utilizar regresión multilineal y se obtiene los coeficientes de la regresión multilineal para la función de distribución de probabilidad, la regresión multilineal se describe a continuación:

Sean p variables independientes relacionadas por la ecuación 3.2:

$$y_i = \beta_1 x_{i1} + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} \tag{3.2}$$

donde x_{ij} es una observación de la variable independiente y_j asume valor 1 para la primera variable independiente, para cada i.

Buscamos las NB partículas con mejor fitness y obtenemos una media por cada dimensión, se almacena en avg.

A continuación iteramos T veces de la siguiente forma:

Generamos nuevas partículas dentro del dominio $-avg$ y avg, utilizando los coeficientes calculados usando la regresión multilineal, calculamos el fitness correspondiente de las partículas generadas. Se normaliza los pesos y son seleccionados las NB partículas con mejor fitness y obtenemos una media por cada dimensión, se actualiza avg.

Al término de las T iteraciones se genera la función de distribución acumulada para la generación de números aleatorios y se obtiene la nueva población clásica de acuerdo a la función de distribución acumulada.

El proceso se muestra en el algoritmo 3 a continuación:

El modelo realiza una estimación de la *función de distribución acumulada* (fda) multi-dimensional y la operación matricial tiene un costo computacional considerable que incrementa el costo global. Por esta razón se exploran métodos de aproximación uni-dimensionales, que siguen el siguiente esquema:

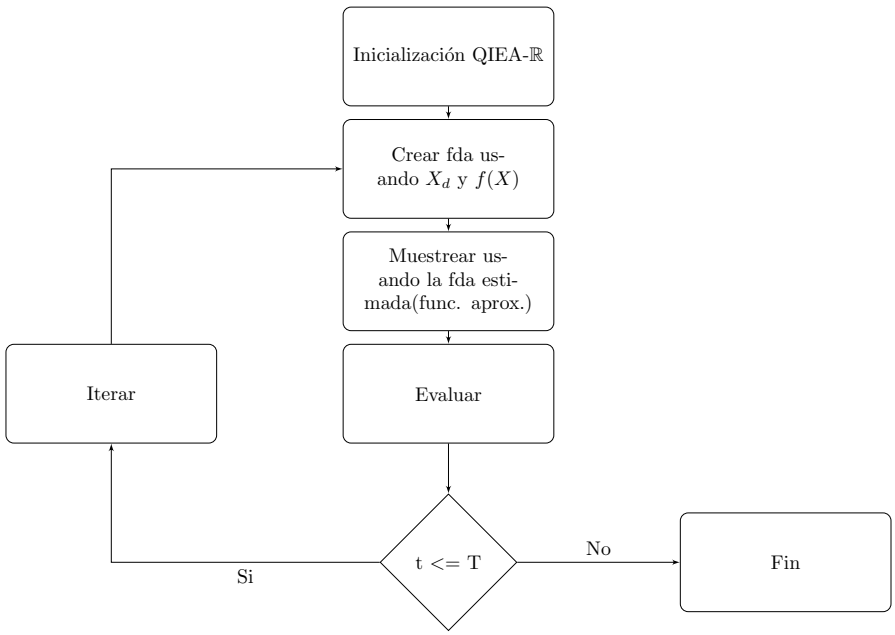


Figura 3.1: Esquema de modelos

dónde se comienza con una inicialización usando el modelo QIEA-ℝ, luego se crea una función de distribución acumulada y se muestrea utilizando un método de aproxi-

Algorithm 3 Algoritmo Evolutivo de Inspiración Cuántica con Codificación Real usando Filtro de Partículas

Inicialización

1: $t \leftarrow 1$
2: Generar población cuántica $Q(t)$ con N individuos con G genes.
3: **while** $t \leq T$ **do**
4: **if** t % 4 != 0 **then**
5: $E(t) \leftarrow$ generar individuos clásicos observando individuos cuánticos
6: **else**
7: $E(t) \leftarrow$ generar individuos clásicos usando nueva función de distribución de probabilidad de la nueva función de distribución de probabilidad obtenida en el paso 28.
8: **end if**
9: **if** t = 1 **then**
10: $C(t) \leftarrow E(t)$
11: **else**
12: **if** t % 4 = 0 **then**
13: Evaluate $E(t)$
14: $E(t) \leftarrow$ reemplazar los n peores individuos clásicos de $C(t)$ con los n mejores individuos de la nueva población $E(t)$ generada por QIEA-ℝ.
15: **else**
16: Evaluate $E(t)$
17: $E(t) \leftarrow$ reemplazar los n peores de $C(t)$ con n aleatorios de la nueva población $E(t)$ generada a partir de la nueva distribución.
18: **end if**
19: **end if**
Partículas
20: $p \leftarrow 1$
21: Generar M partículas usando el generador de números aleatorios
22: Calcular los pesos de las M partículas usando la función objetivo $F(x_i)$
23: Normalizar los pesos
24: Obtener coeficientes usando regresión multilineal
25: Obtener las NB partículas con mejor fitness y obtener la media de sus valores por cada dimensión avg
26: **while** $p \leq T_P$ **do**
27: Generar las nuevas partículas en el dominio [-avg,avg]
28: Obtener los pesos usando los coeficientes obtenidos en el paso anterior.
29: Obtener las NB partículas con mejor fitness y obtener la media de sus valores por cada dimensión y actualizar avg
30: **end while**
31: Remuestrear usando la función $(x^i, f(p))$ y obtener la nueva función de distribución de probabilidad usando frecuencia relativa
32: **if** t % 4 = 0 **then**
33: $Q(t+1) \leftarrow$ Actualizar $Q(t)$ usando la regla de 1/5
34: **end if**
35: $t \leftarrow t + 1$
36: **end while**

mación de funciones, se evalúan los individuos e itera hasta el criterio de parada(número de iteraciones).

Se experimentó con un método de interpolación *splines* que se explica a continuación:

3.2. Fitness recompensando y aproximación con *splines*(FP-SP-QIEA-$\mathbb{R}$)

Utilizar un mecanismo inspirado en filtro de partículas y definir la probabilidad del individuo x_i utilizando la función *benchmark*, recompensando a x_j por tener un mejor fitness y definir la función de distribución de probabilidad utilizando *splines* para la generación de una nueva población clásica utilizando el modelo de ruleta para cada dimensión (Rechenberg, 1973).

3.2.1. Inicialización de partículas

Sean $\{x^i_{0:N}, w^i\}_{i=1}^{NP}$ donde: $\{x^i_{0:N}\}$ con $i = 0, \ldots, NP\}$, es un conjunto de partículas de dimensión N de tamaño NP con pesos asociados $\{w^i, i = 0, \ldots, NP\}$(representan la probabilidad de cada partícula). Los valores de las partículas $x^i_{0:N}$ son inicializados usando un *Generador de Números Aleatorios* (GNA) en el dominio de cada dimensión. Los pesos asociados w^i son calculados usando la función a optimizar. A continuación se normaliza , para asegurar la influencia de los mejoras partículas les damos una recompensa en su probabilidad y normalizamos una vez más.

3.2.2. Aproximación de función de distribución de probabilidad usando *splines*

Almacenamos la dimensión d de las partículas $x_{0:N}$ con sus pesos respectos w^i en un vector y ordenamos en orden decreciente de acuerdo al valor x_d y en este paso hemos creado la función de distribución de probabilidad correspondiente a la dimensión d, usamos *splines* para tener una mejor representación de esta función. Se crea una tabla de acuerdo a un intervalo por ejemplo: 4.0, 4.5, 5.0 y calculamos sus probabilidades de acuerdo a la interpolación sp-line. Corregimos la función obtenida por *splines* recordando que las probabilidades deben ser mayores que 0 y deben sumar 1.

3.2.3. Muestreo utilizando función de distribución acumulada

Calculamos la función de distribución acumulada, utilizamos un RNG para generar números en el dominio $[0, 1]$ y usamos el criterio de ruleta (Rechenberg, 1973) para seleccionar valores para cada partícula en cada dimensión.

3.2.4. Actualización de función de distribución de probabilidad

En la primera iteración guardamos el mejor valor *min_global* y la población inicial X, en cada iteración siguiente guardamos un mejor local *min_local*. Si el mínimo local es mejor que el mínimo global primero actualizamos el valor del mínimo global y luego reemplazamos el peor de la población X con este mejor y normalizamos.

El algoritmo es presentado a continuación:

Algorithm 4 Fitness recompensando y aproximación con *splines*(FP-SP-QIEA-ℝ)

Inicialización

1: Creamos la población X usando el modelo QIEA-ℝ.
2: Evaluamos la población generada X y la guardamos en Y.
3: Normalizamos *y_global*, recompensamos al mejor y normalizamos.
4: **while** $t \leftarrow 1 \leq T$ **do**
5: **while** $d \leq D$ **do**
6: Creamos la función de distribución de probabilidad *x_y* con los valores de X y Y.
7: Ordenamos en orden creciente los valores de la primera columna de *x_y*.
8: Calculamos la función de distribución acumulada utilizando la interpolación de Akima y muestreamos utilizando el criterio de ruleta.
9: **end while**
10: Evaluamos la población generada *x_est* y almacenamos en *y_est*.
11: Calculamos el mínimo y actualizamos X de acuerdo a 3.2.4.
12: **end while**
13: **while** $d \leq D$ **do**
14: Creamos una función de distribución de probabilidad con los valores de *x_est* con su respectiva probabilidad *x_y*.
15: Ordenamos en orden creciente los valores de la primera columna de *x_y*.
16: Calculamos la función de distribución acumulada y muestramos de acuerdo al criterio de ruleta, almacenamos en X.
17: **end while**
18: Evaluamos X y almacenamos en Y.

Este modelo utiliza interpolación sp-line para estimar la función de distribución acumulada, está limitado a la tabla que se debe crear y al tamaño del intervalo, además debemos reparar la función generada por sp-line pues presenta muchas oscilaciones. De-

bemos recordar que la función acumulada debe ser creciente, por esta razón exploramos otro método de interpolación a continuación.

3.3. Fitness recompensando utilizando interpolación Akima y centroides (FP-AK-QIEA-$\mathbb{R}$)

Esta propuesta consiste en reemplazar la interpolación sp-line por la interpolación de Akima para aproximar la función de distribución acumulada y continuamos usando el recompensa de los mejores individuos. Y es añadido un paso que utiliza centroides para obtener una mejor población.

3.3.1. Centroides

Si consideramos que el centroide es el punto medio en un conjuntos de puntos. Entonces, luego de obtener la población final después de las iteraciones; calculamos el centroide de cada dimensión y muestreamos alrededor utilizando una variación *sigma* para mejorar la población existente. La propuesta es presenta en el algoritmo 5.

Esta propuesta mejoró la estimación de la función de distribución acumulada y eliminó la necesidad de tener una tabla y repara la función. Debemos recordar que las últimas dos propuestas son métodos de estimación de funciones uni-dimensionales, se planteó explorar un método de estimación multi-dimensional y se propuso el siguiente algoritmo, esperando obtener mejores resultados.

3.4. Uso de *parzen window* para aproximación de función de distribución de probabilidad

Este modelo mantiene el criterio utilizado en FP-SP-QIEA-$\mathbb{R}$ y FP-AK-QIEA-$\mathbb{R}$, en lugar de realizar una aproximación de función de distribución de probabilidad por cada dimensión, la propuesta utiliza un método de aproximación multidimensional. La propuesta es presentada en el algoritmo 6.

La propuesta no pudo suficiente para representar la función de distribución acumulada multi-dimensional y es necesario encontrar el tamaño de la ventana para la estimación. Por esta razón se retomó el modelo planteado anteriormente que utiliza la interpolación de Akima, con una modificación que es presentada a continuación.

Algorithm 5 Fitness recompensando utilizando interpolación Akima y centroides (FP-AK-QIEA-$\mathbb{R}$)

Inicialización

1: Generar pulsos uniformes para cada dimensión utilizando criterio similar a QIEA-$\mathbb{R}$
2: Muestreo utilizando los pulsos uniformes por cada dimensión
3: Evaluación de la población generada utilizando la función *benchmark* y normalizar
4: Usa criterio de recompensa y normalizar para crear función de distribución de probabilidad
5: $t \leftarrow 1$
6: **while** $t \leq T$ **do**
7: $d \leftarrow 1$
8: **while** $d \leq D$ **do**
9: Crear una función de distribución de probabilidad para cada dimensión d y generar función de distribución acumulada
10: Crear un vector de probabilidades *prob*
11: Utilizando la función de distribución acumulada y su respectivo x_d, utilizando la interpolación de Akima para generar el respectivo x_{estim} relacionado a *prob*
12: Almacenar x_{estim} en X_{est} por cada dimensión
13: **end while**
14: Evaluar la población X_{est} y almacenar el mejor en X_{global}
15: **end while**
16: Calcular los centroides por cada dimensión y almacenar en avg
17: $sigma \leftarrow 0{,}01$
18: **while** $d \leq D$ **do**
19: Muestrear utilizando el centroide en el dominio [avg-$sigma$, avg + $sigma$] y almacenar en X_{global}
20: **end while**
21: Retornar el mejor de X_{global}

Algorithm 6 Uso de *parzen window* para aproximación de función de distribución de probabilidad

Inicialización

1: Generar pulsos uniformes para cada dimensión utilizando criterio similar a QIEA-ℝ
2: Muestreo utilizando los pulsos uniformes por cada dimensión
3: Evaluación de la población generada utilizando función *benchmark* y normalizar
4: Usar criterio de 1 - p(x) y normalizar
5: Ordenar de acuerdo al fitness y recompensar al 10 % de los mejores individuos, normalizar
6: Crear función de distribución acumulada
7: Crear una segunda población aleatoria y utilizando la aproximación *parzen window* estimar su probabilidad
8: Normalizar el vector de probabilidades generado por *parzen window*
9: Crear la función de distribución acumulada
10: Crear un vector de probabilidades entre 0 y 1, usando criterio de la ruleta generar una nueva población X
11: Retornar el mejor de X

3.5. Fitness recompensando utilizando interpolación Akima (FP-AK-QIEA-ℝ 2)

Esta propuesta es similar a su versión anterior pero se añade el mejor individuo a la población actual y por consiguiente la función de distribución de probabilidad es modificada y a continuación muestrear utilizando la función de distribución de probabilidad y generar una nueva mejor población durante las iteraciones. Esta propuesta es presentada en el algoritmo 7.

El añadir el mejor individuo a la población actual guía al algoritmo hacia las zonas más promisorias.

3.5.1. Modificación para problema de *Protein folding*

Protein folding es un problema de optimización que tiene muchos óptimos locales entonces para superar este problema se realizó una modificación en el algoritmo anterior y se añadió un operador de mutación, el algoritmo 8 muestra la propuesta.

Algorithm 7 Fitness recompensando utilizando interpolación Akima (FP-AK-QIEA-$\mathbb{R}$ 2)

Inicialización

Generar pulsos uniformes para cada dimensión utilizando criterio similar a QIEA-$\mathbb{R}$
Muestreo utilizando los pulsos uniformes por cada dimensión
Evaluación de la población generada utilizando función *benchmark* y normalizar
Usa criterio de recompensa y normalizar para crear función de distribución de probabilidad
$t \leftarrow 1$
while $t \leq T$ **do**
  $d \leftarrow 1$
  while $d \leq D$ **do**
    Crear una función de distribución de probabilidad para cada dimensión d y generar función de distribución acumulada
    Crear un vector de probabilidades *prob*
    Utilizando la función de distribución acumulada y su respectivo x_d, utilizando la interpolación de Akima para generar el respectivo x_{estim} relacionado a *prob*
    Almacenar x_{estim} en X_{est} por cada dimensión
  end while
  Evaluar la población X_{est} y añadir el mejor a la función de distribución de probabilidades
end while
Retornar el mejor de X_{global}

Algorithm 8 FP-AK-QIEAR con mutación para búsqueda local

Initialization

Generar población cuántica usando QIEA-$\mathbb{R}$
Generar población clásica utilizando QIEA-$\mathbb{R}$
Evaluar, normalizar, recompensar los mejores individuos y normalizar
$t \leftarrow 1$
while $t \leq T$ **do**
  while $d \leq D$ **do**
    Create la función de distribución de probabilidad y calcular la función acumulada, generar el vector de probabilidad *prob* para muestrear
    Utilizar función de distribución acumulada y el respectivo valor de X para para estimar (x_{estim}) de acuerdo *prob* utilizando interpolación de Akima
    Guardar x_{estim} en X_{est}
  end while
  Evaluar X_{est}
  Seleccionar los mejores de X y X_{est} para actualizar X y Y.
  Encontrar el mejor individuo y aplicar mutación a cada dimensión de acuerdo $prob_{mutation} \leq 0.1$ y guardar en X_{mut}
  Encontrar el peor individuo en X y reemplazarlo por X_{mut}
  Evaluar X y guardar en Y, normalizar, recompensar, normalizar.
end while

Capítulo 4

Pruebas y Resultados

Para evaluar las diversas propuestas de este trabajo, algunos experimentos son realizados usando las siguientes funciones *benchmark*: Ackley, Rastrigin, Rosenbrock, Schwefel y Sphere para comparar con la propuesta original de QIEA-$\mathbb{R}$. Una de las características de estas funciones *benchmark* son su configuración para variables de dimensión $n > 2$.

Para minimizar el efecto de aleatoreidad inherente en este tipo de algoritmos, 100 experimentos fueron realizados y promediados para estar próximo al comportamiento real. Se pone a prueba el modelo FP-AK-QIEA-$\mathbb{R}$ 2 que mostró buenos resultados en medidas estadísticas y tiempos de ejecución en la optimización de pesos de una red perceptrón multicapa y el problema de *Protein folding*.

Todos los experimentos fueron realizados en una computadora con las siguientes características: procesador core i7 y una memoria Ram de 8gb.

Y se usó un modelo simple de paralismo usando c++11(threads) para enviar varias ejecuciones en paralelo y obtener respuestas más rápido para su análisis.

4.1. Plan Experimental

4.1.1. Uso de funciones *benchmark*

4.1.1.1. Configuración

Las tablas 4.1, 4.2 y 4.3 presentan los parámetros que fueron usados en los experimentos, para los modelos QIEA-$\mathbb{R}$, FP-QIEA-$\mathbb{R}$, FP-SP-QIEA-$\mathbb{R}$ y FP-AK-QIEA-$\mathbb{R}$. Los parámetros como población cuántica, dimensión, número de iteraciones y población clásica son comunes en todos los modelos.

Cuadro 4.1: Parámetros usados en QIEA-$\mathbb{R}$ y FP-QIEA-$\mathbb{R}$

Población cuántica	PC	Indice de reemplazo de los peores	IRP
Dimensión	D	Frecuencia de actualización	FA
Iteraciones en QIEA-$\mathbb{R}$	IQ	Iteraciones de FP	IFP
Población clásica	PCL	Número de partículas	P
Frecuencia de actualización	FA	Partículas remuestreadas	PRP

Cuadro 4.2: Parámetros usados en FP-SP-QIEA-$\mathbb{R}$ y FP-AK-QIEA-$\mathbb{R}$

Población cuántica	PC
Dimensión	D
Iteraciones en FP-AK-QIEA-$\mathbb{R}$	IQ
Población clásica	PCL
Nro mejores	NM
Nro partículas	P

El modelo FP-WP-QIEA-$\mathbb{R}$ utiliza los parámetros FP-SP-QIEA-$\mathbb{R}$ y FP-AK-QIEA-$\mathbb{R}$, la diferencia es el método de aproximación; entonces se añade un parámetro.

Cuadro 4.3: Parámetros usados en FP-WP-QIEA-$\mathbb{R}$

Longitud arista	h_n

4.1.1.2. Parámetros básicos

El trabajo de Da Cruz (da Cruz et al., 2010) usó la siguiente configuración:

- Ackley, Rastrigin and Sphere

Cuadro 4.4: Parámetros usados en los experimentos de Ackley, Rastrigin y Sphere

QP	D	IQ	CP	UF
5	30	25	100	1

- Rosenbrock and Schwefel

Cuadro 4.5: Parámetros usados en los experimentos de Rosenbrock y Schwefel

QP	D	IQ	CP	UF
5	30	25	100	10

4.1.2. Experimentos

Cada una de las siguientes tablas muestran los parámetros de los experimentos realizados por cada función *benchmark*. Para QIEA-$\mathbb{R}$, los parámetros usados son los mismos que se usaron en los experimentos de DaCruz (da Cruz et al., 2010) como se menciona en las tablas 4.4, 4.5, cuyos parámetros fueron ajustados después de diversos experimentos para lograr el mejor desempeño.

4.1.2.1. Ackley

Cuadro 4.6: Parámetros - Ackley

	PC	**D**	**IRP**	**IQ**	**PCL**	**NM**	**FA**	**IFP**	**P**	**PRP**	H_d
QIEA-R	5	30	-	75	100	-	1	-	-	-	
FP-QIEA-R	5	30	5	25	100	-	1	3	35	40	
FP-SP-QIEA-R	500	30	-	500	30	1	-	-	50	-	
FP-AK-QIEA-R	1000	30	-	30	1000	5	-	-	50	-	
FP-PW-QIEA-R	1000	30	-	50	1000	5	-	-	50	-	1.11722
FP-AK-QIEA-R 2	1000	30	-	30	1000	5	-	-	50	-	

4.1.2.2. Rastrigin

Las tablas muestran los parámetros usado para los experimentos con la función Rastrigin.

Cuadro 4.7: Parámetros - Rastrigin

	PC	**D**	**IRP**	**IQ**	**PCL**	**NM**	**FA**	**IFP**	**P**	**PRP**	H_d
QIEA-R	5	30	-	75	100	-	1	-	-	-	
FP-QIEA-R	5	30	5	25	100	-	1	3	33	50	
FP-SP-QIEA-R	500	30	-	500	30	1	-	-	50	-	
FP-AK-QIEA-R	1000	30	-	30	1000	5	-	-	50	-	
FP-PW-QIEA-R	1000	30	-	50	1000	5	-	-	50	-	1.1933
FP-AK-QIEA-R 2	1000	30	-	30	1000	5	-	-	50	-	

4.1.2.3. Rosenbrock

Las tablas muestran los parámetros usados para los experimentos con la función Rosenbrock.

Cuadro 4.8: Parámetros - Rosenbrock

	PC	**D**	**IRP**	**IQ**	**PCL**	**NM**	**FA**	**IFP**	**P**	**PRP**	H_d
QIEA-R	5	30	-	75	100	-	10	-	-	-	
FP-QIEA-R	5	30	5	25	100	-	1	3	31	80	
FP-SP-QIEA-R	500	30	-	500	30	1	-	-	50	-	
FP-AK-QIEA-R	1000	30	-	30	1000	5	-	-	50	-	
FP-PW-QIEA-R	1000	30	-	50	1000	5	-	-	50	-	1.074855
FP-AK-QIEA-R 2	1000	30	-	30	1000	5	-	-	50	-	

4.1.2.4. Schwefel

Las tablas muestran los parámetros aplicados a los experimentos con la función Schwefel.

Cuadro 4.9: Parámetros - Schwefel

	PC	**D**	**IRP**	**IQ**	**PCL**	**NM**	**FA**	**IFP**	**P**	**PRP**	H_d
QIEA-R	5	30	-	75	100	-	10	-	-	-	
FP-QIEA-R	5	30	5	25	100	-	1	3	33	30	
FP-SP-QIEA-R	500	30	-	500	30	1	-	-	50	-	
FP-AK-QIEA-R	1000	30	-	30	1000	5	-	-	50	-	
FP-PW-QIEA-R	1000	30	-	50	1000	5	-	-	50	-	1.072485
FP-AK-QIEA-R 2	1000	30	-	30	1000	5	-	-	50	-	

4.1.2.5. Sphere

Las tablas muestran los parámetros usados para los experimentos con la función Sphere.

Cuadro 4.10: Parámetros - Sphere

	PC	**D**	**IRP**	**IQ**	**PCL**	**NM**	**FA**	**IFP**	**P**	**PRP**	H_d
QIEA-R	5	30	-	75	100	-	1	-	-	-	
FP-QIEA-R	5	30	5	25	100	-	1	3	31	60	
FP-SP-QIEA-R	500	30	-	500	30	1	-	-	50	-	
FP-AK-QIEA-R	1000	30	-	30	1000	5	-	-	50	-	
FP-PW-QIEA-R	1000	30	-	50	1000	5	-	-	50	-	1.072265
FP-AK-QIEA-R 2	1000	30	-	30	1000	5	-	-	50	-	

4.2. Aplicación: Optimizar pesos de una red perceptrón multicapa

Las redes neuronales perceptrón multicapa son ampliamente conocidas en el mundo de Inteligencia Artificial y tienen dos etapas:

- Entrenamiento: se realiza el ajuste de pesos de acuerdo a los patrones.
- Prueba: en base a los pesos calculados en el entrenamiento se realiza la clasificación.

Por los resultados mostrados en 4.12, se escogió el modelo FP-AK-QIEAR para inicializar los pesos de una red perceptrón multicapa y lograr una mejor convergencia o menor número de etapas en el entrenamiento.

4.3. Aplicación: *Protein folding*

Las proteinas son estructuras básicas de todos los seres vivos (Hunter, 1993), compuestas de una cadena de aminoácidos enlazadas por enlaces peptídicos. *Protein folding* es el proceso donde la cadena es transformada en un estructura compacta que realiza alguna función biológica.

Un modelo de representación para las estructuras de las proteinas fue introducido por (Stillinger y Head-Gordon, 1995) basadas en los ángulos que la estructura puede tener. Para codificar las variables la propuesta de (Parpinelli et al., 2014) fue usada. *Protein folding* tiene muchos óptimos locales entonces para superar este problema se realizó una modificación en el algoritmo FP-AK-QIEAR 2 que se detalló en la sección anterior.

Se experimentó con las siguientes problemas artificiales:

Cuadro 4.11: Secuencias artificiales utilizadas para experimentación

Tamaño	**Secuencias**
13	ABBABBABABBAB
21	BABABBABABBABBABABBAB
34	ABBABBABABBABBABABBABABBABBABABBAB
55	BABABBABABBABBABABBABABBABBABABBAB
	BABABBABABBABBABABBAB

4.4. Resultados obtenidos

A continuación se muestran los resultados obtenidos luego de los experimentos realizados con las funciones *benchmark*, la aplicación del modelo en la optimización del entrenamiento de una red perceptrón multicapa y la aplicación en el problema de *Protein folding*.

Se utiliza un gráfico llamado diagrama de cajas y en la gráficas se pueden apreciar los siguiente símbolos: 'o', valor que está fuera de 3 veces el rango del intercuartil y '+', para puntos entre 1.5 y 3 veces el rango del intercuartil. El rango de intercuartil es una medida de variabilidad basada en dividir un conjunto de datos en cuartiles(dividir en 4 partes iguales), equivalente a la diferencia del tercer y primer cuartil.

4.4.1. Funciones *benchmark*

Para la comparación entre los modelos se usaron medidas estadísticas(media de los mejores y desviación estándar) (da Cruz et al., 2010) y una comparación cuantitativa.

A continuación en la tabla 4.12 se muestran el mínimo obtenido y la desviación estándar de los experimentos realizados por cada función *benchmark*.

Cuadro 4.12: Resultados comparativos entre QIEA-R, FP-QIEA-R, FP-SP-QIEAR, FP-AK-QIEAR, FP-AK-QIEAR 2 y M = media de las soluciones y DS = desviación estándar

Función	**QIEAR**	**FP-QIEAR**	**FP-SP-QIEAR**	**f***
Ackley	6.90169e-05 ± 0.153807	4.69737e-06 ± 0.000108147	0.0672064 ± 0.366029	0
Rastrigin	3.40506e-08 ± 10.7567	1.16864e-09 ± 1.01112e-05	0.103937 ± 0.984452	0
Rosenbrock	86250.5 ± 70620	28.7711 ± 0.0969862	38.0677 ± 63.1627	0
Schweffel	264.575 ± 99.796	1388.98 ± 40.4354	1.3761 ± 0.604725	0
Sphere	1.02857e-07 ± 6.89456	3.4288e-10 ± 7.05949e-07	0.0994436 ± 1.308	0

Función	**FP-AK** QIEAR	**PW**	**FP-AK-QIEAR 2**	**f***
Ackley	0.223239 ± 0.042933	20.6243 ± 0.0178	4.44089e-16 ± 1.47946	0
Rastrigin	7.90374 ± 1.25134	442.447 ± 0.211	0 ± 0.0896933	0
Rosenbrock	121.85 ± 24.2877	2.36158e+10 ± 0.5129	29 ± 0	0
Schweffel	0.0428689 ± 940.325	12085.1 ± 0.351	11.4743 ± 122.079	0
Sphere	0.0620225 ± 0.157783	57911.4 ± 0.2901	4.69125e-34 ± 0.00104732	0

Como se observa en la tabla 4.12 el modelo que utiliza *parzen window* no tiene buenos resultados y se dejará de usar en los siguientes experimentos.

A continuación se presentan el diagrama de caja(*box-plot*) en la Fig. 4.1 para analizar la distribución de las soluciones obtenidas en los experimentos. Para ayudar

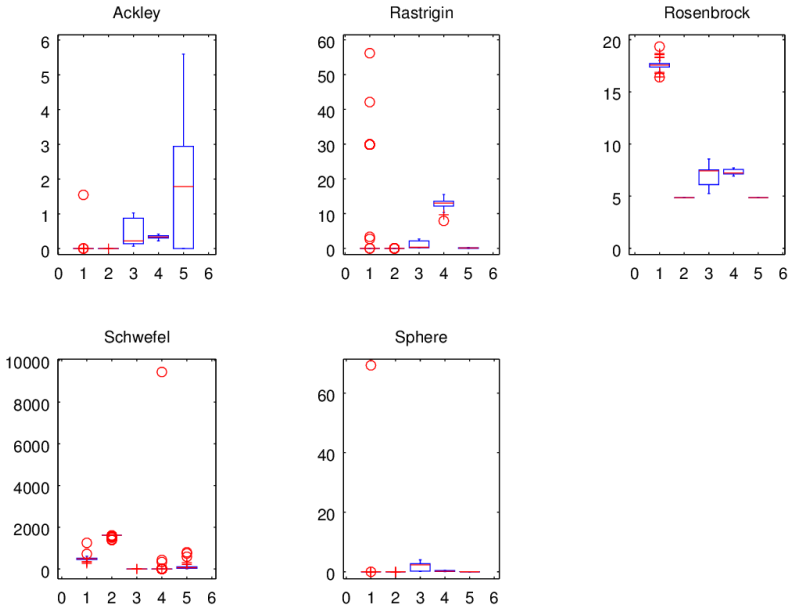


Figura 4.1: Diagrama de caja de experimentos con dimensionalidad 30, dónde el eje x son las propuestas en orden(QIEAR, FP-QIEAR, FP-SP-QIEAR, FP-AK-QIEAR, FP-AK-QIEAR 2 y el eje y es el valor de solución

en el análisis visual en el caso de la función Rosenbrock se utilizó una escala logarítmica.

Los tiempos de ejecución son mostrados en la Fig. 4.2.

Observando la tabla 4.12y las figuras Fig. 4.1, 4.2 podemos notar que el modelo FP-AK-QIEA-$\mathbb{R}$ 2 tiene un mejor comportamiento en comparación a todas las propuestas realizadas(FP-SP-QIEAR, FP-AK-QIEAR) a excepción de la función Ackley. Además presenta un comportamiento similar a FP-QIEAR a excepción de la función Schwefel. Al comparar el modelo original de daCruz, la propuesta FP-AK-QIEA-$\mathbb{R}$ 2 supera a QIEAR a excepción de la función Ackley. Podemos afirmar que el modelo FP-AK-QIEAR 2 tiene un balance entre desempeño y tiempo de ejecución.

4.4.2. Aumento de la dimensionalidad en las funciones *benchmark*

Este experimento consiste en evaluar los modelos con las mismas funciones *benchmark* pero aumentando la dimensionalidad de 30 a 100.

En la tabla 4.13 se muestran el mínimo obtenido y la desviación estándar de los experimentos realizados por cada función *benchmark* con una dimensionalidad de 100:

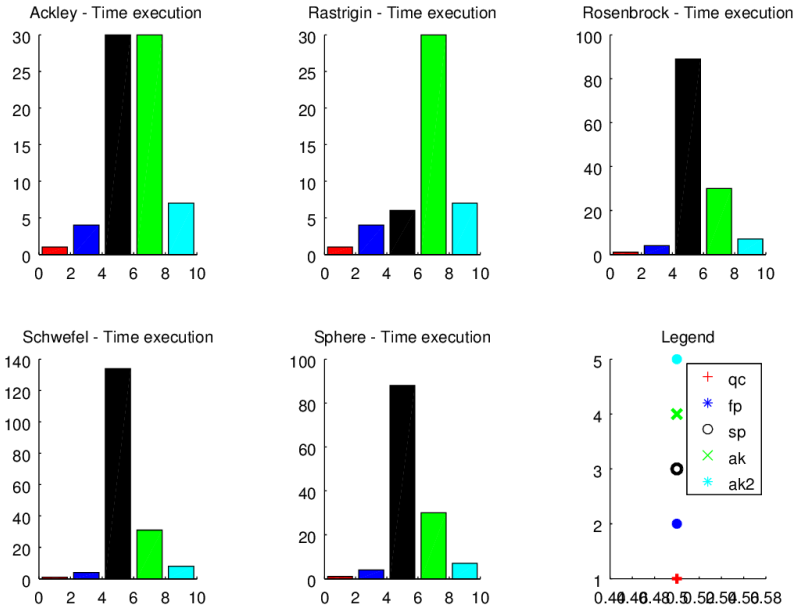


Figura 4.2: Tiempos de ejecución de experimentos con dimensionalidad 30

Cuadro 4.13: Resultados comparativos entre QIEA-R, FP-QIEA-R, FP-SP-QIEAR, FP-AK-QIEAR, FP-AK-QIEAR 2 y M = media de las soluciones y DS = desviación estándar

Función	**QIEAR**	**FP-QIEAR**	**FP-SP-QIEAR**	**f***
Ackley	0.00024217 ± 0.646307	1.8158e-05 ± 0.000164685	0.202298 ± 0.494792	0
Rastrigin	1.33097e-06 ± 55.9562	4.28784e-08 ± 1.22719e-05	15.3224 ± 9.5327	0
Rosenbrock	753920 ± 2.42686e+07	98.7622 ± 0.0624988	333.367 ± 6368.59	0
Schweffel	1896.18 ± 837.701	5380.48 ± 1.00044e-11	443.526 ± 244.843	0
Sphere	2.62333e-06 ± 14.0878	3.00546e-09 ± 4.61292e-06	1.65366 ± 10.0522	0

Función	**FP-AK QIEAR**	**FP-AK-QIEAR 2**	**f***
Ackley	0.333576 ± 0.10987	4.44089e-16 ± 1.45092	0
Rastrigin	43.4482 ± 2.83427	0 ± 0.0275965	0
Rosenbrock	584.161 ± 2707.02	99 ± 0	0
Schweffel	9.76339 ± 40.0847	56.4231 ± 224.711	0
Sphere	0.405037 ± 2.40594	2.23807e-34 ± 0.000310734	0

A continuación se presentan el diagrama de caja(*box-plot*) en la Fig. 4.3 para analizar la distribución de las soluciones obtenidas en los experimentos. Dónde el eje x son las propuestas en orden(QIEAR, FP-QIEAR, FP-SP-QIEAR, FP-AK-QIEAR, FP-AK-QIEAR 2 y el eje y es el valor de solución. Para ayudar en el análisis visual en el caso de la función Rosenbrock se utilizó una escala logarítmica.

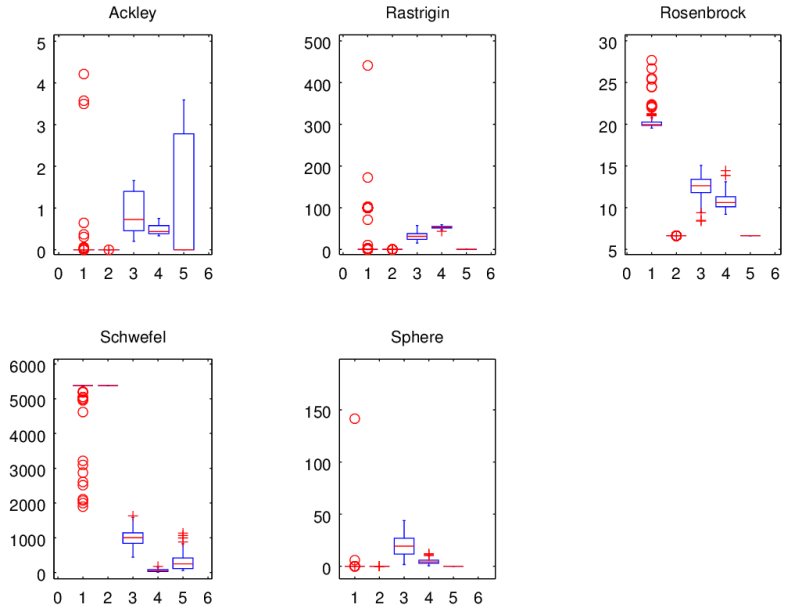


Figura 4.3: Diagrama de caja de experimentos con dimensionalidad 100

Los tiempos de ejecución son mostrados en la Fig. 4.4.

Observando la tabla 4.13 y las figuras Fig. 4.3, 4.4 podemos notar que el modelo FP-AK-QIEA-$\mathbb{R}$ 2 tiene un comportamiento similar a los experimentos realizados con 30 dimensiones. Podemos notar que la diferencia entre el modelo QIEAR y FP-AK-QIEAR 2 persisten.

En base a los resultados de los experimentos anteriores con funciones *benchmark* podemos observar que el modelo FP-AK-QIEA-$\mathbb{R}$ 2 tiene un buen desempeño en términos de medidas estadísticas y tiempos de ejecución. Por esta razón, llamaremos al modelo FP-AK-QIEA-$\mathbb{R}$ 2 sólo FP-AK-QIEA-$\mathbb{R}$ para los próximos experimentos y pondremos a prueba su desempeño considerando la dificultad de los problemas siguientes.

4.4.3. Pesos de una perceptrón multicapa

En la etapa de entrenamiento los pesos son ajustados hasta que converga la red y sea inferior al error mínimo permitido. Usualmente la inicialización de los pesos es aleatoria entonces se planteó encontrar eso pesos como si fuese un problema de optimización cuya función de aptitud es back-propagation, la cual se usó en el procesamiento de entrenamiento general.

A continuación la configuración del modelo usado:

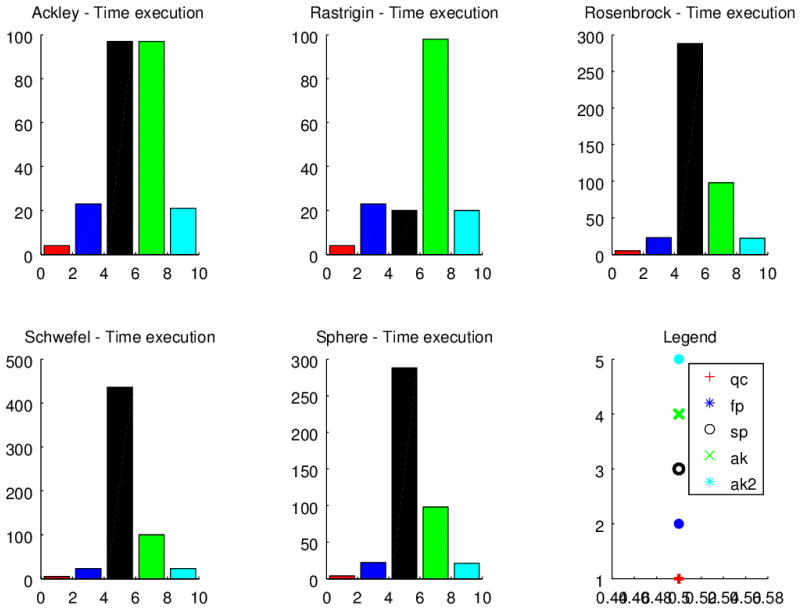


Figura 4.4: Tiempos de ejecución de experimentos con dimensionalidad 100

Cuadro 4.14: Configuración de FP-AK-QIEAR

Parámetros	**Valor**
PC	1000
IQ	30
PCL	1000
NM	0.005*PCL
P	50
Dominio	[-1,1]

Se planteó usar el modelo en tres tipos de problemas: regresión, patrones y clasificación que son detallados a continuación.

4.4.3.1. Pruebas

1. Función seno:

La arquitectura de la red es la siguiente:

Cuadro 4.15: Parámetros - seno

Parámetros	**Valor**
Nodos Capa Entrada	1
Nodos Capa Media	30
Nodos Capa Salida	1
Training	0.5
Minimal Error	0.01

obteniéndose una dimensionalidad (cantidad de pesos a optimizar) igual a 91. La dimensionalidad se calcula utilizando:

$$Dimensin = out\_num + hid\_num + hid\_num * inp\_num + out\_num * hid\_num \tag{4.1}$$

Por ejemplo, el siguiente patrón es el valor de la función del ángulo en radianes.

$$\begin{bmatrix} seno(x) & \text{radianes} \\ 0 & 5 \\ 0{,}0871 & 10 \\ 0{,}1736 & 15 \end{bmatrix}$$

Cuadro 4.16: Número de etapas en 10, 50 y 100 experimentos - seno

Seno	**10**	**50**	**100**
Random Init.	64703	45395	48571
FP-AK-QIEAR	**13505**	**12102**	**11890**
QIEAR	21948	20472,74	18854,14

2. Patrones de números en binario:

   La arquitectura de la red es la siguiente:

Cuadro 4.17: Parámetros - patrones de números en binario

Parámetros	**Value**
Nodos Capa Entrada	15
Nodos Capa Media	6
Nodos Capa Salida	4
Training	0.5
Minimal error	0.01

resultando una dimensionalidad de cada inviduo clásico de 124.

Por ejemplo el siguiente patrón equivale al número 0.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Cuadro 4.18: Número de etapas en 10, 50 y 100 experimentos - patrones de números en binario

Patrones	**10**	**50**	**100**
Random Init.	5034	5593	7839
FP-AK-QIEAR	3452	7679	6995
QIEAR	**3162,1**	**5187,88**	**4530,38**

3. Detección de cáncer pulmonar

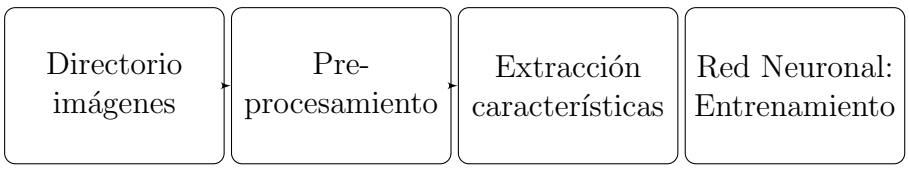
   La arquitectura de la red es la siguiente:

Cuadro 4.19: Parámetros - detección cáncer pulmonar

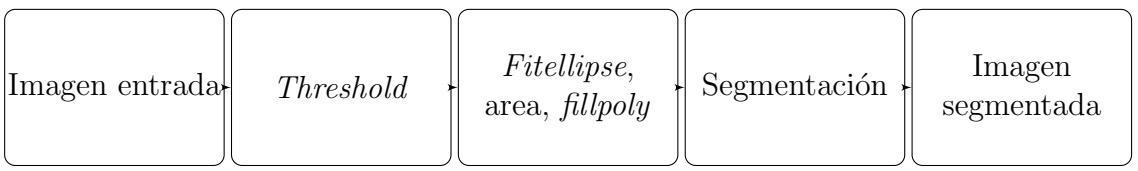
Parámetros	**Valor**
Nodos Capa Entrada	8
Nodos Capa Media	17
Nodos Capa Salida	1
Training	0.5
Minimal error	0.01

Y la dimensionalidad de cada inviduo clásico es 171.

El esquema seguido para el entrenamiento es:



Y el esquema para pre-procesamiento es:



El esquema para la clasificación es similar al esquema de entrenamiento.

El pre-procesamiento se realiza de la siguiente forma: Comenzamos con la imagen de entrada, luego utilizamos las siguientes funciones de OpenCv: *findContours*, *fitellipse* para encontrar los bordes, luego encontrar elipses que se ajusten a los bordes. Utilizar la función minAreaRect, que encuentra el rectángulo rotado de menor área que encierra un conjunto de puntos en 2D, cálcular el área utilizando *contourArea* y sólo consideramos áreas mayores a 118 y menores 5000(valores obtenidos durante la experimentación). Utilizando los bordes, usamos *fillpoly* para llenar el área del rectángulo calculado

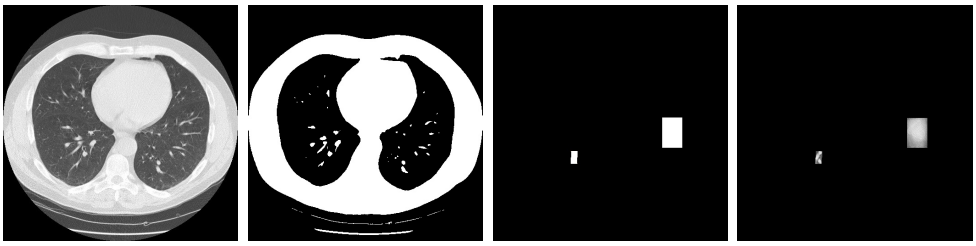


Figura 4.5: Pre-procesamiento

Utilizamos Patrón binario local pero con la siguiente modificación:

$$s(x) = \begin{cases} 1 & if x > 0 \\ 0 & if x <= 0 \end{cases} \tag{4.2}$$

La imagen resultante es:

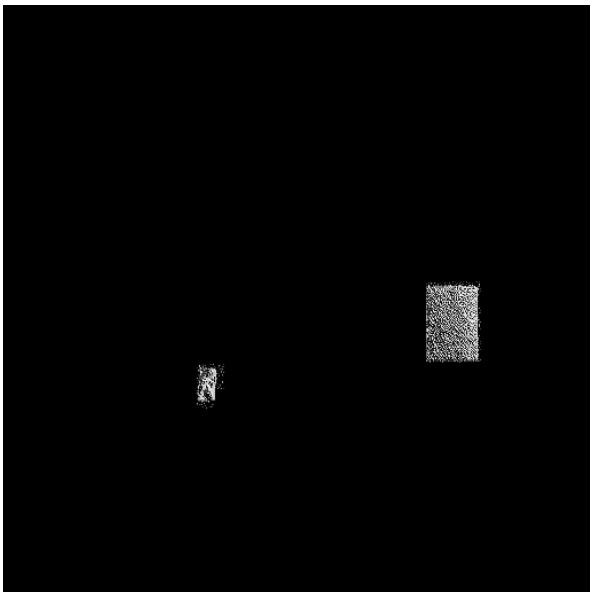


Figura 4.6: Imagen generada al aplicar Patrón binario local

La extracción de características utiliza medidas estadísticas como media, varianza, energía, entropía, curtosis, sesgo, homogeneidad y momento de diferencia inverso.

Utilizando las medidas de una imagen mencionadas antes, obtenemos un vector de 8 valores:

2.206 403.816 9.322 87.357 1.071e+08 -4.337e+06 3.898e-05 46.807

Se realiza la extracción de características de un conjunto de imágenes, luego se realiza una normalización y se guardan dichos valores en un archivo de texto con el siguiente formato:

Cuadro 4.20: Ejemplo archivo de patrones

```
2
0.036 0.044 0.012 0.005 0.044 0.038 -8.213E-005 8.197E-005
0
0.005 0.005 0.031 0.049 0.003 0.003 -6.045E-007 2.652E-005
1
```

La primera línea indica el número de patrones, la siguiente el patrón y a continuación su valor esperado. Y luego se realiza el entrenamiento guiado por los patrones definidos.

Cuadro 4.21: Número de etapas en 10, 50 y 100 experimentos - detección cáncer pulmonar

Cáncer pulmonar	**10**	**50**	**100**
Random Init.	55127.1	55239.5	56266.12
FP-AK-QIEAR	**29439.6**	**30663.22**	**30820.72**
QIEAR	30759,3	35982,54	34976,36

A continuación podemos observar una comparación entre los tiempos de ejecución en cada caso usando 100 experimentos:

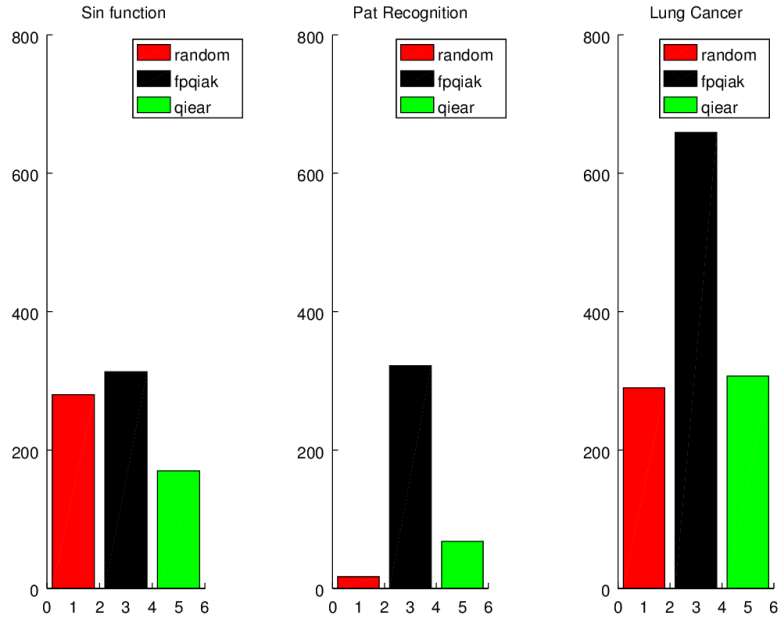


Figura 4.7: Tiempos de ejecución de experimentos con red perceptrón multicapa

Para realizar un mejor análisis presentamos los diagramas de cajas comparando los modelos en la figura Fig. 4.8.

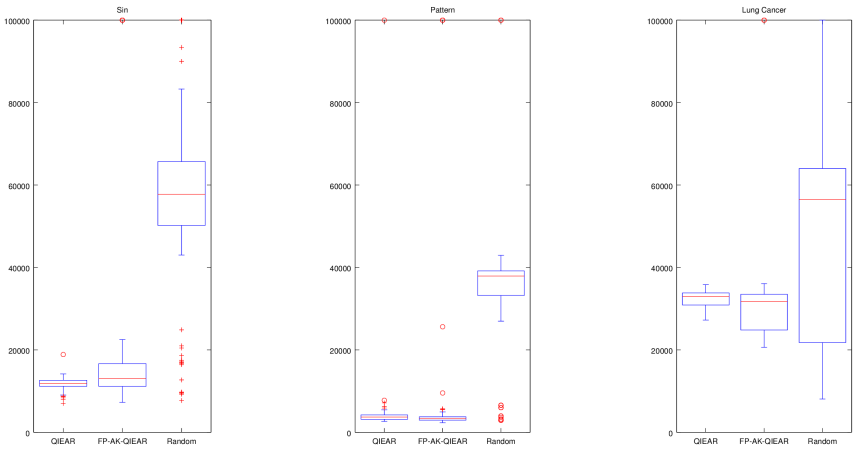


Figura 4.8: Diagramas de cajas de experimentos

En base a los resultados de los experimentos anteriores y la figura Fig. 4.8 podemos decir que el utilizar un modelo diferente al aleatorio en la inicialización de pesos impacta en el número de épocas total. Además el modelo FP-AK-QIEA-$\mathbb{R}$ y QIEAR ayudan a disminuir el número de épocas e impactan en la convergencia. Podemos decir que los modelos son similares en el caso de la función seno el modelo FP-AK-QIEAR tiene

un rango mayor que QIEAR y en el caso de cáncer pulmonar el rango es mayor pero incluye valores menores al rango de QIEAR.

4.4.4. *Protein folding*

Se realizaron 24 experimentos y se obtuvo el mejor resultado, la mediana y la desviación estándar para comparar con los resultados obtenidos en un estudio previo que compara PSO, *Gravitational Search Algorithm* (GSA), *Bat Algorithm* (BA), *Artificial Bee Colony* (ABC) (Parpinelli et al., 2014), en este trabajo el mejor resultado fue con PSO.

Cuadro 4.22: Comparación utilizando *Protein folding*

N	**PSO**		**FP-AK-QIEAR**	
	E_{avg}	E_{best}	E_{avg}	E_{best}
13	-23.102 ± 0.93	-24.888	-22.1814 ± 0.813222	-23.9409
21	-43.047 ± 2.34	-46.611	-41.9036 ± 2.74061	-46.0356
34	-70.866 ± 5.95	-80.409	-68.9528 ± 3.63282	-75.9811
55	-87.715 ± 19.27	-115.758	-107.276 ± 8.86613	-119.652

4.5. Discusiones

Como se observa en la tabla 4.12 y las figuras 4.1, 4.2 con los experimentos de dimensionalidad 30, el modelo FP-AK-QIEAR 2 superó a las demás propuestas aunque presenta resultados similares a FP-QIEAR pero este modelo basado en regresión multilineal presenta problema en Schwefel porque esta función presenta muchos óptimos globales y el modelo FP-SP-QIEAR está cercano a sus resultados pero presenta grandes tiempos de ejecución debido a la tabla que debe tener para la estimación de la función de distribución de probabilidad.

FP-AK-QIEAR 2 supera al modelo QIEAR pero presenta dificultades en la función Ackley porque la función presenta muchos óptimos locales y al recompensar los N mejores individuos es posible quedar atrapado en un óptimo local que impacta en la formación de la función de distribución de probabilidad y muestrear individuos cercanos al óptimo local. Además debemos mencionar que añadir los mejores individuos durante las iteraciones para actualizar la función de distribución tiene impacto en los resultado, por esta razón FP-AK-QIEAR 2 supera a su predecesor. El comportamiento se repite en los experimentos con dimensionalidad 100 mostrados en la tabla 4.13 y las figuras 4.3, 4.4. Debido a los resultados obtenidos en los experimentos con las funciones *benchmark* se escogió el modelo FP-AK-QIEAR 2 para los siguientes experimentos(renombrado FP-AK-QIEAR).

Por los experimentos realizados con las redes neuronales basados en las tablas 4.16, 4.18, 4.21 y la figuras 4.8, podemos afirmar que la inicialización aleatoria no es

la mejor opción y que el usar propuestas como QIEAR y FP-AK-QIEAR disminuye el número de épocas y acelera la convergencia. En el caso de la función seno, el modelo QIEAR tiene un menor diferencia entre el 1er cuartil y 3er cuartil a diferencia que FP-AK-QIEAR a pesar de eso las medianas están próximas. En el caso de los patrones binarios, los modelos QIEAR y FP-AK-QIEAR tienen valores del 1er y 3er cuartil similares así como el valor de la mediana. En el caso de detección, QIEAR y FP-AK-QIEAR tienes los valores del 3er cuartil similares pero FP-AK-QIEAR tiene un 1er cuartil significativamente menor por tanto los valores que obtiene FP-AK-QIEAR son menores y mejores que QIEAR. Basado en los resultados de los experimentos anteriores podemos decir que FP-AK-QIEAR puede tener un desempeño similar a QIEAR e incluso menor en algunos contextos.

Los experimentos realizados con *Protein Folding* muestran que el algoritmo FP-AK-QIEA$\mathbb{R}$ tiene un desempeño similar a PSO con valores próximos y menor variabilidad, logrando superar a PSO en el experimento con mayor dimensionalidad. La desviación estándar es menor a comparación de PSO porque está muestreando alrededor de un óptimo local pero la media del mejor cercana a PSO, esto nos hace recordar que el problema tiene muchos óptimos locales.

Capítulo 5

Conclusiones y Trabajos Futuros

En este trabajo se logró modelar un mecanismo inspirado en filtro de partículas y métodos de aproximación de funciones para una mejor generación de individuos clásicos de acuerdo a la naturaleza del problema.

- Se confirmaron las limitaciones del modelo QIEA-$\mathbb{R}$ a través de losp experimentos: utilizar una distribución uniforme para la generación de individuos clásicos.
- Los métodos para aproximar la función de distribución de probabilidad fueron usados para realizar un mejor muestreo.
- La interpolación de Akima supera a *splines* y realiza una mejor aproximación de funciones de distribución de probabilidad y no está limitada a la reparación. Por esta razón la propuesta FP-AK-QIEA-$\mathbb{R}$ fue superior a todos los modelos.
- Las funciones *benchmark* y la aplicación son útiles para poner a prueba los modelos y las medidas(valor mínimo, desviación estándar) son significativas para comparar los modelos.
- Los experimentos con funciones *benchmark* mostraron que la propuesta FP-AK-QIEA-$\mathbb{R}$ es mejor que el modelo inicial QIEA-$\mathbb{R}$ en las funciones benchmark pero presenta limitaciones con Ackley.
- Los experimentos realizados con la red neuronal perceptrón multicapa muestran que QIEA-$\mathbb{R}$ y FP-AK-QIEA-$\mathbb{R}$ son similares pero FP-AK-QIEA-$\mathbb{R}$ logra valores menores en el caso de detección de cáncer pulmonar.
- La propuesta logra superar a otros algoritmos evolutivos en problema de *Protein folding*, demostrando su buen desempeño.
- El camino está abierto para realizar modificaciones a la propuesta de acuerdo al contexto del problema abordado.

5.1. Limitaciones

Al realizar pruebas más exhaustivas, es decir con mayor cantidad de experimentos, mayor dimensionalidad y mayor muestreo, los experimentos se ven limitados por la memoria RAM disponible.

5.2. Recomendaciones

Se recomienda explorar métodos de búsqueda local para mejorar el desempeño de la propuesta.

Realizar experimentos exhaustivos (mayor dimensionalidad) para poder llegar a una mejor comparación entre los modelos existentes.

5.3. Trabajos futuros

Experimentar con otros métodos de aproximación unidimensional y multidimensional para una mejor aproximación de la función distribución de probabilidad y mejorar la generación de los individuos clásicos.

Realizar experimentos con problemas de dominio binario y mixto para analizar el desempeño del algoritmo en esos contextos.

Implementar los modelos en ambientes GP-GPU con programación de alto desempeño para poder realizar experimentos aumentando la dimensionalidad a decenas de miles logrando pruebas más exhaustivas.

Bibliografía

Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Norwell, MA, USA.

Akima, H. (1970). A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17(4):589–602.

Arnold, N. (2006). *Global optimization and constraint satisfac- tion*.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK.

Back, T., Fogel, D. B., et al., editors (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition.

Baeck, T., Fogel, D., et al. (1997). *Handbook of Evolutionary Computation*. Taylor & Francis.

Bianchi, L., Dorigo, M., et al. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287.

Bouaziz, A., Draa, A., et al. (2013). A quantum-inspired artificial bee colony algorithm for numerical optimisation. In *Programming and Systems (ISPS), 2013 11th International Symposium on*, pages 81–88.

Bremermann, H. J. (1962). Optimization through evolution and recombination. In Yovits, M. C., Jacobi, G. T., et al., editors, *Proceedings of the Conference on Self-Organizing Systems – 1962*, pages 93–106, Washington, DC. Spartan Books.

da Cruz, A. (2007). *Algoritmos Evolutivos com Inspiração Quântica para Problemas com Representação Numérica*. PhD thesis, Departamento de Engenharia Elétrica.

da Cruz, A., Vellasco, M., et al. (2010). Quantum-inspired evolutionary algorithms applied to numerical optimization problems. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–6.

Das, P. P. y Khan, M. H. (2015). Solving maximum clique problem using a novel quantum-inspired evolutionary algorithm. In *Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on*, pages 1–6.

Dixon, L. C. W. y Szego, G. P. (1978). The global optimization problem: an introduction. towards global optimization. pages 1–15.

Dortmund, T. (1995). *Evolutionary Algorithms in Theory and Practice : Evolution Strategies, Evolutionary Programming, Genetic Algorithms: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, USA.

Duda, R. O., Hart, P. E., et al. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience.

Escovedo, T., Abs da Cruz, A., et al. (2014). A neuro-evolutionary unlimited ensemble for adaptive learning. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 3331–3338.

Escovedo, T., Vargas Abs da Cruz, A., et al. (2013). Using ensembles for adaptive learning: A comparative approach. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7.

Fogel, L., Owens, A., et al. (1966). *Artificial Intelligence Through Simulated Evolution*. John Wiley &; Sons, Inc., New York, NY, USA.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Han, K.-H. y Kim, J.-H. (2000). Genetic quantum algorithm and its application to combinatorial optimization problem. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, pages 1354–1360 vol.2.

Hao, L. y Shiyong, L. (2012). Quantum particle swarm evolutionary algorithm with application to system identification. In *Measurement, Information and Control (MIC), 2012 International Conference on*, volume 2, pages 1032–1036.

Hinterding, R. (1999). Representation, constraint satisfaction and the knapsack problem. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, page 1292 Vol. 2.

Holland, J. (1975a). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA.

Holland, J. H. (1975b). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

Hongjian, Q., Fangzhao, Z., et al. (2009). An application of new quantum-inspired immune evolutionary algorithm. In *Database Technology and Applications, 2009 First International Workshop on*, pages 468–471.

Hossain, M., Hossain, M., et al. (2009). Hybrid real-coded quantum evolutionary algorithm based on particle swarm theory. In *Computers and Information Technology, 2009. ICCIT '09. 12th International Conference on*, pages 13–18.

Hunter, L. (1993). Artificial intelligence and molecular biology. chapter Molecular Biology for Computer Scientists, pages 1–46. American Association for Artificial Intelligence, Menlo Park, CA, USA.

Iriyama, S. y Ohya, M. (2014). On efficien quantum algorithm using classical amplificatio process. In *Cognitive Informatics Cognitive Computing (ICCI*CC), 2014 IEEE 13th International Conference on*, pages 88–93.

Judd, K. L. (1998). *Numerical Methods in Economics*. MIT Press.

Le Gall, F. (2014). Improved quantum algorithm for triangle finding via combinatorial arguments. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 216–225.

Liu, X. y Liu, X. (2013). Quantum-inspired genetic algorithm based on phase encoding. In *Natural Computation (ICNC), 2013 Ninth International Conference on*, pages 444–448.

Nunes, W., Vellasco, M., et al. (2013). A quantum-inspired evolutionary algorithm for fuzzy classification. In *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, pages 29–34.

Parpinelli, R. S., Benitiez, C. M. V., et al. (2014). Performance analysis of swarm intelligence algorithms for the 3d-ab off-lattice protein folding problem. *Multiple-Valued Logic and Soft Computing*, 22:267–286.

Rastrigin, L. A. (1974). Extremal control systems. *Theoretical Foundations of Engineering Cybernetics Series. Moscow: Nauka, Russian*.

Rechenberg, I. (1973). Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. *Frommann-Holzboog Verlag*.

Reeves, C. R., editor (1993). *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc., New York, NY, USA.

Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function. *Computer Journal*, pages 175–184.

Schwefel, H. P. (1981). Numerical optimization of computer models. *English translation of Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, 1977*.

Schwefel, H.-P. P. (1993). *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., New York, NY, USA.

Spector, L. (2006). *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Genetic Programming. Springer US.

Stillinger, F. H. y Head-Gordon, T. (1995). Collective aspects of protein folding illustrated by a toy model. *Phys. Rev. E*, 52:2872–2877.

Tsai, C.-W., Liao, Y.-H., et al. (2013). A quantum-inspired evolutionary clustering algorithm. In *Fuzzy Theory and Its Applications (iFUZZY), 2013 International Conference on*, pages 305–310.

van der Hauw, J. (1996). Evaluating and improving steady state evolutionary algorithms on constraint satisfaction problems. Master's thesis, Computer Science Department of Leiden University, Leiden, Netherlands.

Weise, T. (2008). Global optimization algorithms – theory and application.

Wu, X.-J., Xu, C., et al. (2009). Application of quantum evolutionary algorithm in blind source separation. In *Natural Computation, 2009. ICNC '09. Fifth International Conference on*, volume 6, pages 505–509.

Yanguang, C., Zhang, M., et al. (2010). A hybrid chaotic quantum evolutionary algorithm. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, volume 2, pages 771–776.

Zhang, R. y Gao, H. (2007). Improved quantum evolutionary algorithm for combinatorial optimization problem. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 6, pages 3501–3505.

Zhao, J., Li, M., et al. (2013). A novel binary quantum-behaved particle swarm optimization algorithm. In *Distributed Computing and Applications to Business, Engineering Science (DCABES), 2013 12th International Symposium on*, pages 119–123.