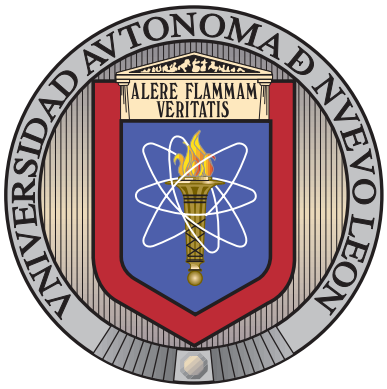
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado



Enfoque Paralelo Multiplataforma en Heuristicas para Problemas no Lineales de Alta Dimensión

por

José Adrián Rodríguez Aldape

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León noviembre 2011

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Enfoque Paralelo Multiplataforma en Heuristicas para Problemas no Lineales de Alta Dimensión

por

José Adrián Rodríguez Aldape

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León noviembre 2011

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Enfoque Paralelo Multiplataforma en Heuristicas para Problemas no Lineales de Alta Dimensión», realizada por el alumno José Adrián Rodríguez Aldape, con número de matrícula 1241024, sea aceptada para su defensa como opción al grado de Maestría en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

Dr. Arturo Berrones Santos
Director

Dr. Sergio Javier Mejía Rosales
Revisor

Dr. Hugo Jair Escalante
Revisor

Vo. Bo.

Dr. Moisés Hinojosa Rivera
División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, noviembre 2011

A mi familia por haberme motivado para lograr esta meta

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

A mi Director de Tesis el Dr. Arturo Berrones que con su experiencia y guía he sido capaz de crear esta tesis.

A mis Revisores de Tesis el Dr. Hugo Jair Escalante y el Dr. Sergio Javier Mejía Rosales que con su colaboración he logrado producir un trabajo de calidad.

Al personal Académico, Administrativo y Alumnado del Posgrado en Ingeniería de Sistemas que a lo largo de la maestría me han brindado su apoyo y consejo.

Al Consejo Nacional de Ciencia y Tecnología por haberme otorgado una beca ya que sin ésta hubiese sido muy difícil lograr este paso.

A la Universidad Autónoma de Nuevo León por haberme prestado sus instalaciones de primer nivel y apoyado en los momentos que fueron necesarios.

A mi familia quienes fueron los que me apoyaron a lo largo de toda esta gran experiencia.

Resumen

José Adrián Rodríguez Aldape.

Candidato para el grado de Maestría en Ciencias
con especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Enfoque Paralelo Multiplataforma en Heuristicas para Problemas no Lineales de Alta Dimensión

Número de páginas: 86.

Objetivos y método de estudio: En la presente tesis se ha paralelizado una heurística para optimización global desarrollada por el grupo de trabajo en que se ha estado involucrado. La heurística, denotada AGEDA por sus siglas en inglés [24] ha probado ser competitiva y robusta para problemas no restringidos con multiplicidad de óptimos locales, con no diferenciabilidad y con dimensionalidad creciente.

Un paso necesario para escalar los problemas de aplicación de AGEDA es su versión paralela. Por la naturaleza adaptativa de la heurística dicha paralelización no es directa y han debido desarrollarse y probarse distintas estrategias.

La contribución final de la tesis ha sido una versión paralela de esta heurística adaptativa novedosa. Mediante la versión paralela podrán estudiarse problemas antes intratables debido a su alta dimensión.

Como contribuciones adicionales se ha desarrollado una versión paralela del algoritmo PSO por propósitos comparativos. Además se han explorado otros aspectos de la computación paralela relativos a la simulación mediante agentes, los cuales se presentan a modo de apéndice, como lo son la generación de redes en base a las opiniones de los agentes y la dinámica de opiniones con agentes multicriterio.

Contribuciones y conclusiones: Gracias al desarrollo de esta Tesis se ha obtenido una heurística capaz de generar soluciones de calidad en un tiempo más corto en comparación con su versión serial, además de haberse explorado diversos métodos de paralelización.

En el caso de la simulación de opiniones mediante agentes se ha demostrado que un grupo de agentes por mas diversas que tengan las opiniones estas convergerán en un tiempo determinado, esto debido las características del modelo.

Firma del asesor: _____

Dr. Arturo Berrones Santos

Capítulo 1

Introducción

En esta tesis observaremos diversas implementaciones paralelas de algunas heurísticas aplicadas en diferentes problemas, comparando su versión en paralelo contra su versión serial.

1.1 Cómputo Paralelo

Normalmente un software se desarrolla con una serie de instrucciones seriales, esto es una línea de código o un conjunto de líneas de código que realizan algún procedimiento, cuando es utilizada una sola computadora con un solo procesador y usando instrucciones seriales se le llama cómputo serial.

En este tipo de cómputo un problema es dividido en una serie discreta de instrucciones, estas se ejecutan una después de otra y solo se ejecuta una instrucción a la vez. (Figura 1.1)

La programación en paralelo es el uso simultáneo de múltiples recursos para resolver un problema computacional que será ejecutado en múltiples unidades de procesamiento, es decir, se utilizan múltiples unidades de procesamiento para realizar diversos procesos pertenecientes al mismo problema simultáneamente.

En este tipo de cómputo un problema es dividido en partes discretas, cada parte se divide en una serie de instrucciones y las instrucciones de cada parte se ejecutan simultáneamente en diferentes unidades de procesamiento (Figura 1.2)

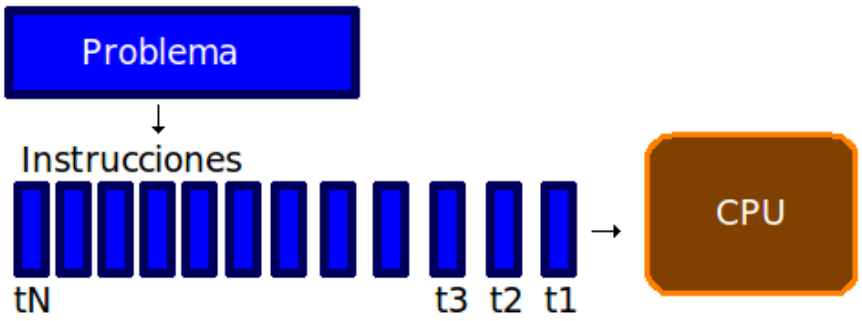


Figura 1.1: Problema Serial

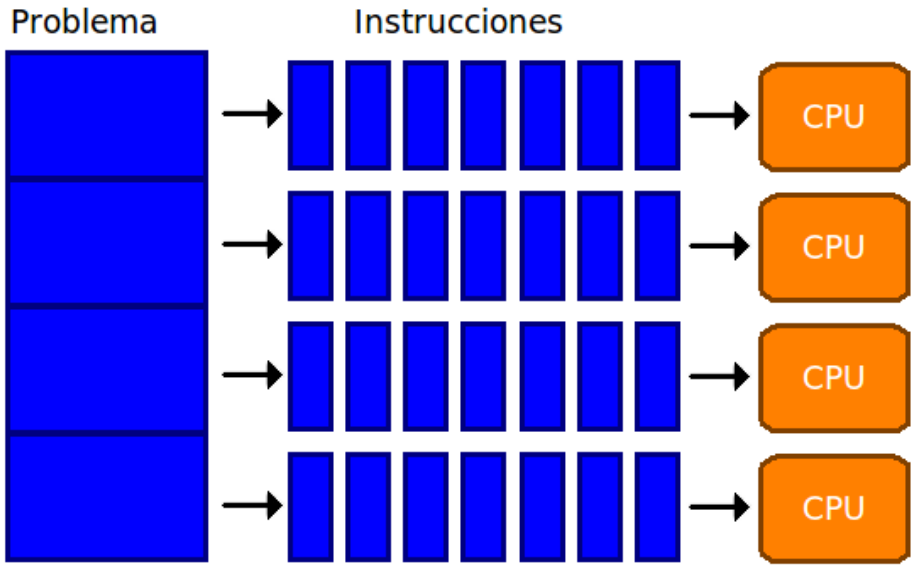


Figura 1.2: Problema en Paralelo

1.2 Plataformas

Existen diferentes tipos máquinas especializadas para el cómputo paralelo que son conocidas como plataformas, entre ellas están: Multinúcleo (Multicore), Grupo o Agrupamiento (Clúster) y Red (GRID).

1.2.1 Multicore

Todo tipo de computadora cuenta con una unidad central de procesamiento (CPU por sus siglas en inglés), en éstos se realizan todas las funciones de procesamiento y están presentes en cualquier computadora de cualquier tiempo.
Un procesador multinúcleo está compuesto por un arreglo de mozaicos.

Cada mosaico contiene un núcleo independiente y un puente de comunicación (switch) para conectarlo con sus vecinos. Debido a que los núcleos solo están conectados mediante ruteadores, este diseño tiene una gran facilidad para agregar mosaicos adicionales.[17]

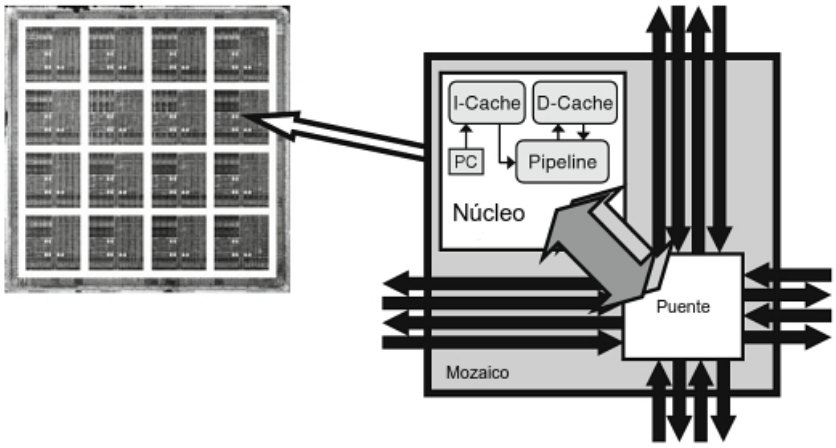


Figura 1.3: Procesador multicore [17]

1.2.2 Clúster

La arquitectura de un clúster está basada en una agrupación de varias computadoras cada una con uno o varios procesadores, en cada uno se ejecuta una copia de la misma aplicación o distinta pero con acceso a los mismos datos y se comparten recursos, especialmente el almacenamiento en disco. Los clústeres permiten el balanceo de carga, que es una técnica utilizada para distribuir las peticiones entre los equipos.[8]

1.2.3 GRID

GRID es un término que se refiere al cómputo distribuido que se extiende por varios lugares y/o múltiples organizaciones, arquitecturas de máquina y limites de software para proveer poder de cómputo, colaboración y acceso a la información.

Es una infraestructura que permite el uso integrado y colaborativo de computadoras, supercomputadoras, redes, bases de datos e instrumentos científicos administrados por múltiples organizaciones. [19]

1.3 Heurísticas

Una vez que se han definido las diversas arquitecturas computacionales y lo que es el cómputo paralelo, comenzamos con una breve introducción a las heurísticas para poder comprender de mejor manera las heurísticas en paralelo.

En optimización, existen diversos modelos de resolución de problemas: de forma exacta, que necesitan de un modelo matemático y de forma aproximada. A estos últimos se les conoce como heurísticas que son procedimientos simples que ofrecen una buena solución a problemas difíciles de modo fácil y rápido.

Las heurísticas son más comúnmente utilizadas cuando el tiempo para encontrar la solución óptima es muy elevado, para generar buenas soluciones iniciales o cuando no se dispone del software necesario para buscar una solución óptima.

Una metaheurística es una estrategia maestra que guía y modifica las heurísticas para producir soluciones más allá de aquellas que serian generadas por una heurística.

Las metaheurísticas son una clase de métodos aproximados que han sido diseñados para atacar problemas difíciles de optimización combinatoria donde las heurísticas clásicas no han logrado ser efectivas o eficientes. Las metaheurísticas proveen un marco general que permite generar nuevos híbridos combinando diferentes conceptos derivados de las heurísticas clásicas.[12]

Las metaheurísticas se han desarrollado ampliamente desde su introducción a principios de los 80s. Han tenido un amplio éxito atacando una gran variedad de problemas de optimización prácticos y de dificultad combinatoria. Las metaheurísticas incorporan conceptos basados en biología evolutiva, resolución inteligente de problemas, ciencias matemáticas y físicas y mecánica estadística.[12]

Definición 1.1 *Una metaheurística es un proceso de generación iterativo que guía una heurística subordinada combinando inteligentemente diferentes conceptos para explorar y explotar los espacios de búsqueda usando estrategias de aprendizaje para estructurar información y encontrar eficientemente soluciones cercanas al óptimo.[12]*

Las heurísticas en paralelo han sido ampliamente discutidas y estudiadas, ofrecen diferentes tipos de ventajas dependiendo de su implementación, la principal ventaja de las heurísticas en paralelo es la disminución de tiempo de cómputo, lo que permite abordar problemas más pesados computacionalmente, además pueden presentar otro tipo de ventajas donde no solo se reduce el tiempo de cómputo sino también pueden llegar a mejorar una solución en comparación a su implementación serial, como se puede ver en [16], [10] y [6].

1.3.1 Particle Swarm Optimization (PSO)

El PSO es un algoritmo basado en el movimiento coordinado de las aves (parvada) y peces (cardumen) que a pesar de ser múltiples organismos, reaccionan de una manera rápida y organizada ante cualquier estimulo, en 1987 Reynolds en [22] presentó un algoritmo basado en el comportamiento coordinado de estos animales, el cual consta de 3 reglas:

Separación: Cada agente trata de moverse lejos de sus vecinos si se encuentra muy cerca.

Alineación: Cada agente se dirige hacia la dirección promedio de sus vecinos.

Cohesión: Cada agente trata de ir hacia la posición de sus vecinos.

En 1995 Kennedy, J. y Eberhart, R. en [18] introdujeron un concepto de líder y su simulación consta de las siguientes reglas:

Cada partícula es atraída hacia la ubicación del líder.

Cada partícula recuerda su posición más cercana al líder.

Cada partícula comparte con sus vecinos (originales y nuevos) su posición más cercana al líder.

El pseudocódigo de un PSO simple se muestra en el pseudocódigo 1 en donde las partículas se inicializan aleatoriamente formando una matriz de dimensiones: numero de partículas $\times$ dimensionalidad, es decir si tenemos 10 partículas y cada partícula se encuentra en un espacio de 5 dimensiones entonces la matriz de enjambre es de 10×5. Cada celda en esta matriz es llenada con un número aleatorio de distribución uniforme entre un intervalo determinado, este intervalo es elegido dependiendo del espacio de búsqueda que se desea analizar en un problema determinado. Cada partícula se evalúa sustituyendo sus componentes en la función objetivo, esta función es el problema a analizar.

En el caso de la velocidad en el pseudocódigo 1 utilizamos un peso que determina la importancia de la inercia, es decir la velocidad en el paso anterior, este peso disminuye conforme avanzan las iteraciones hasta llegar a un valor mínimo, conocido como inercia final [21].

Pseudocódigo 1 Pseudocódigo de un PSO simple

```
1: para cada partícula hacer
2:     Inicializar partícula aleatoriamente.
3: fin para
4: repetir
5:     para cada partícula hacer
6:         Evaluar cada partícula.
7:         si su valor es el mejor que ha tenido entonces
8:             Actualizar su mejor valor por el actual.
9:         fin si
10:    fin para
11:    para cada partícula hacer
12:        Encontrar la mejor partícula del vecindario de la partícula actual.
13:        Calcular su velocidad de acuerdo a la ecuación 1.1.
14:        Aplicar el acotamiento de la velocidad.
15:        Actualizar la posición de la partícula de acuerdo a la ecuación 1.2.
16:    fin para
17: hasta que criterio de parada se cumpla
```

Además de utilizar dos constantes de aceleración $c1$ y $c2$ que multiplican unos números aleatorios uniformemente distribuidos en el intervalo [0,1] $R1$ $R2$. A el resultado de la resta de la mejor ubicación de la partícula (PBM) menos la ubicación actual de la partícula ($Swarm$) se le conoce como influencia personal y a la resta de la mejor ubicación de la mejor partícula (GMB) menos la ubicación actual de la partícula ($Swarm$) se le conoce como influencia global.

Cabe mencionar que ya que no hay un mecanismo para controlar la velocidad de una partícula se impuso un parámetro conocido como velocidad máxima V_{max}, ya que una velocidad máxima muy pequeña puede resultar en una exploración insuficiente del espacio de soluciones, mientras que una velocidad muy grande puede hacer que las partículas pasen por alto por alguna buena solución. Resultando la ecuación 1.1 en este caso se actualiza la velocidad de todo el enjambre en una sola instrucción.

$$Vel = W_{now} * Vel + (R1 * c1) * (PBM - Swarm) + (R2 * c2) * (GBM - Swarm) \quad (1.1)$$

Para actualizar la posición de una partícula es necesario agregar a su posición actual la velocidad calculada en el paso anterior, es decir se realiza una suma de vectores entre el vector que indica la posición actual con el vector de la velocidad, dando como resultado la ecuación 1.2 en donde se actualizan todas las partículas al mismo tiempo.

$$Swarm = Swarm + Vel \quad (1.2)$$

1.3.2 AGEDA: Adaptive Gibbs sampling Estimation Density Algorithm

Esta heurística pertenece al campo de los Algoritmos de Estimación de densidades (EDA) por sus siglas en ingles, fue desarrollada por el M.C. Jonás Velasco y el Dr. Arturo Berrones en el 2010 en [24], este tipo de algoritmos construyen una distribución de probabilidad y un conjunto de soluciones que muestrean esta distribución de probabilidad.

La principal diferencia entre AGEDA y los demás algoritmos de estimación de densidades es que AGEDA funciona utilizando un muestreo de Gibbs adaptativo para generar nuevas soluciones en combinación con una estrategia de búsqueda local.

Una ventaja de este algoritmo es que modifica las tasas de intensificación y exploración de manera adaptativa. Utilizando el muestreo de Gibbs como método de exploración y el método de Nelder-Mead [15] como estrategia de intensificación. El Pseudocódigo del AGEDA puede verse en el Pseudocódigo 2.

En el algoritmo de AGEDA M es el parámetro utilizado para elegir el tamaño de la población ya que el muestreo de Gibbs genera individuos a partir de un candidato inicial, los parámetros c_n's, siendo n la dimensionalidad del problema, representan el tamaño de la escala en el muestreo, el parámetro ϵ indica la tasa de aceptación en la componente n del individuo mientras que θ_n es el promedio de aceptación del muestreo para la variable n y β define la tasa de intensificación.

1.4 Motivación

Existe una gran cantidad de funciones en las que el número de variables es muy grande y los métodos utilizados necesitan de mucho tiempo computacional el cual no se dispone, es por esto que se ha investigado la programación paralela en donde es posible ahorrar tiempo computacional proporcional al número de procesadores utilizados.

Pseudocódigo 2 Algoritmo de AGEDA

1: Dada una población de tamaño M y parámetros iniciales c_n's.
2: Inicializar $t \Leftarrow 0$. Generar $M >> 0$ individuos aleatoriamente.
3: Evaluar los individuos usando la función de costo.
4: Seleccionar el mejor individuo inicial $x_t^{(best)}$ para el muestreo de Gibbs adaptativo.
5: **mientras** No se cumpla el criterio de Terminación **hacer**
6: Generar M nuevos individuos mediante el muestreo de Gibbs adaptativo (usando las c_n's y $x_t^{(best)}$).
7: **para** $n = 1$ to N **hacer**
8: **si** $\theta_n > \epsilon$ **entonces**
9: Actualizar c_n.
10: Reemplazar el valor de la n-ésima variable ($x_{n,t}^{(best)}$) aleatoriamente.
11: **fin si**
12: Actualizar c_n.
13: **fin para**
14: **si** $\langle \theta\text{'s} \rangle > \beta$ **entonces**
15: Mejorar la solución del mejor individuo x_t^{best} por medio de la estrategia de búsqueda local.
16: **fin si**
17: Actualizar $t \Leftarrow t + 1$
18: **fin mientras**

Además actualmente las industrias disponen de poco tiempo para obtener soluciones a sus problemas, por ejemplo de producción donde es necesario una solución rápida y casi diariamente para evitar pérdidas, de esta forma es cómo es posible contribuir con la programación paralela en diferentes casos.

1.5 Estado del arte

Actualmente el uso de algoritmos en paralelo crece, esto debido a la nueva variedad de herramientas de cómputo y las ventajas que la paralelización ofrece, entre estas ventajas esta el ahorro de tiempo principalmente ya que la programación en paralelo permite realizar procesos simultáneamente para después comparar resultados preliminares y seleccionar el mejor proceso para una siguiente iteración.

Con base en esta investigación podemos intuir que el paralelismo más allá de ser una herramienta en la que se logra mejorar el tiempo de cómputo, es también una gran oportunidad de implementar nuevos métodos en la búsqueda de mejores algoritmos.

Mark T. Jones y Paul E. Plassmann [16] desarrollaron un algoritmo paralelo para el coloreo de grafos, implementándolo mediante la creación de subgrafos en base a vértices no adyacentes para después colorear cada subgrafo de manera paralela, lograron obtener soluciones de calidad en un menor tiempo computacional para problemas con un mayor número de variables que los algoritmos conocidos.

De igual manera E.-G. Talbi et al. desarrollaron en [10] un algoritmo basado en el comportamiento de hormigas de forma paralela, utilizando un estilo de programación similar que en [16] en donde se creó un proceso máster que guía y organiza a otros procesos conocidos como trabajadores, éstos trabajadores se encargan de realizar las iteraciones reportando su solución final al proceso máster, el cual crea una matriz de frecuencias con las soluciones encontradas. Con este nuevo algoritmo paralelo lograron mejorar los resultados conocidos utilizando otros algoritmos además de lograrlo en tiempos competitivos.

Además en [6] realizaron un estudio utilizando heurísticas en paralelo en el que no solo encontraron que el paralelismo ayudo reduciendo los tiempos de cómputo sino que logró mejorar la calidad de las soluciones reportadas. Para la creación de sus heurísticas paralelas utilizaron un algoritmo evolutivo descentralizado, de manera que se crean subpoblaciones que evolucionan de manera independiente pero intercambian individuos con una determinada frecuencia. Gracias a su experimentación

lograron obtener resultados interesantes ya que el valor promedio obtenido fue mayor al de las heurísticas seriales, sin embargo una menor cantidad de veces lograron obtener una mejor solución, esto quiere decir que el paralelismo ayudo a incrementar el valor promedio de las soluciones pero pocas veces se logro obtener soluciones de calidad superior al promedio.

En [24] se propone un método nuevo e innovador utilizando un algoritmo basado en la estimación de densidades con un muestreo de Gibbs adaptativo, dicho método ha logrado demostrar un buen desempeño a diversos tipos de problemas con características como multiplicidad de óptimos locales, no diferenciabilidad además de una dimensionalidad creciente, por lo que vemos en este nuevo método una oportunidad de crear una versión paralela tratando de mejorar aún más su desempeño utilizando diversas técnicas de paralelización.

Además se realiza un algoritmo de optimización con enjambre de partículas (PSO por sus siglas en ingles) de forma paralela para lograr resultados comparativos preliminares.

1.6 Contribución

Se nos presentan dos diversas heurísticas PSO y AGEDA en donde el PSO trabaja con una población de soluciones que interactúan entre ellas mientras que el AGEDA funciona generando soluciones con base en una distribución de probabilidad y después de obtener una solución de calidad intensifica dicha búsqueda en la región donde se encuentra ubicada dicha solución.

Se desea paralelizar ambas heurísticas para reducir su tiempo de cómputo e intentar mejorar su desempeño aplicando diferentes formas de paralelización, entre las que se encuentran la paralelizacion de funciones y procesos paralelos con intercambio de soluciones. Dichas heurísticas fueron probadas en diferentes funciones para evaluar su comportamiento y su robustez.

De igual manera se presenta un problema de dinámica de opiniones en donde se pretende observar el comportamiento de las opiniones utilizando una simulación de agentes, los cuales interactúan entre sí, y cada agente posee una opinión propia para diversos temas de conversación de modo que la evolución de sus opiniones en los diversos temas no está totalmente definida, así como también la forma en que éstos se acercan para producir grupos que serán analizados mediante grafos.

1.7 Supuesto de investigación

Al inicio de un proyecto de paralelización se deben analizar las posibles alternativas a implementar para seleccionar la que resulte más adecuada y se adapte mejor a las diferentes propiedades de cada heurística.

Utilizando diferentes técnicas de procesamiento en paralelo ¿será posible desarrollar un nuevo método que obtenga resultados de buena calidad en tiempos menores para poder dar soluciones a problemas de mayor dimensionalidad?

Se realizan diversas implementaciones de heurísticas en paralelo para analizar su desempeño ante diversas funciones de prueba, de forma que se puede seleccionar la estrategia más adecuada para el tipo de función a optimizar. También se propone un análisis de cómputo no convencional utilizando una simulación mediante agentes, en las que se observa la dinámica de opiniones con respecto a la evolución del tiempo utilizando agentes vectoriales, es decir agentes con más de una opinión o tema de conversación.

1.8 Metodología de Tesis

La Metodología consiste en una parte teórica en donde se analizan las heurísticas y diversas funciones a utilizar y una parte practica en donde se comienza a implementar diversos tipos de procesamiento en paralelo a dichas heurísticas para evaluar su desempeño en comparación a la versión serial.

Nos enfocamos en dos heurísticas que son el Optimizador de partículas aleatorias (PSO) y el Algoritmo de estimación de densidades con un muestreo de Gibbs adaptativo (AGEDA), en el PSO comenzamos por implementarlo serialmente para después analizar las diferentes áreas de oportunidad para crear un PSO paralelo en el que se obtengan resultados de calidad en tiempos de cómputo menores, mientras en AGEDA implementamos diversas alternativas para evaluar de manera objetiva su mejor implementación.

Capítulo 2

Funciones de Prueba

2.1 Función de Rosenbrock

La función de Rosenbrock también conocida como la función de banana de Rosenbrock o la segunda función de De Jong [20] es un punto de referencia muy conocido para problemas de optimización no lineal, ya que es utilizado principalmente para probar el desempeño de los algoritmos evolutivos por tener un óptimo global conocido [25].

La versión clásica en dos dimensiones de esta función es unimodal pero en los últimos años se ha extendido a mas dimensiones, y en 2001 y 2002 Hansen [11] y Deb [9] encontraron que para mayores dimensiones la función de Rosenbrock no es unimodal más no mostraron un análisis teórico. En [25] se muestra el análisis para demostrar que para algunas dimensiones (4-30) la función de Rosenbrock no es unimodal.

En esta función el óptimo global se encuentra en un largo y estrecho valle en forma de parábola, en donde encontrar el valle es trivial pero encontrar el óptimo global es realmente difícil.

La definición de la función de Rosenbrock se muestra en la ecuación (2.1)

$$\sum_{i=1}^{n-1}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2] \tag{2.1}$$

Es un problema de minimización donde se puede apreciar su unimodularidad en dos dimensiones en la figura 2.1 tiene su mínimo en (1,1) con $f^* = 0$

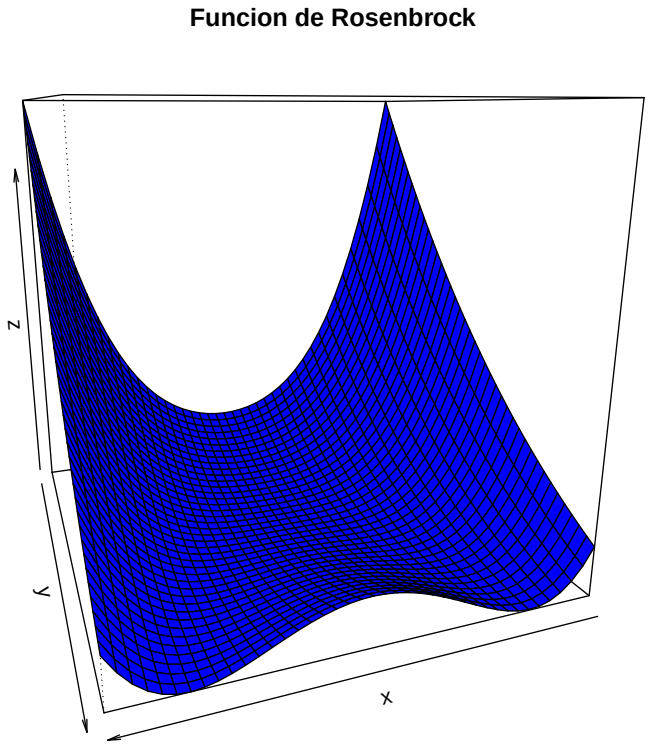


Figura 2.1: Función de Rosenbrock en 2D

2.2 Función Fractal

Uno de los intereses principales de las heurísticas es trabajar con problemas donde la solución óptima no se obtiene fácilmente, problemas con superficies rugosas son desafiantes tanto para métodos exactos y heurísticas, una función fractal tiene fuertes similitudes con un problema del mundo real por lo que hemos decidido incluir una función dentro de nuestros problemas de prueba, dicha función [3] se muestra en la ecuación (2.2).

$$\begin{aligned} minf(x) = &\sum_{i=1}^{n} C'(x_i) + x_i^2 - 1 \\ &-5 \leq x_i \geq 5 \end{aligned} \tag{2.2}$$

$$C'(x) = \begin{cases} \frac{C_{(x)}}{C(1)|x|^{2-D}}, & \text{if } x \neq 0 \\ 1, & \text{if } x = 0 \end{cases} \tag{2.3}$$

$$C(x) = \sum_{j=-\infty}^{\infty} \frac{1 - cos(b^j x)}{b^{(2-D)j}}, \tag{2.4}$$

Donde C(X) es una aproximación de la función fractal coseno Weierstrass-Mandelbrot, para esta función el parámetro D es la dimensión del fractal ($1 \leq D \geq 2$) [3] además de permitir aumentar o disminuir la complejidad de la función arbitrariamente, en esta función fractal es imposible indicar la posición exacta del optimo global, que se encuentra cerca del origen, sin embargo debido a que en esta región hay muchos picos zigzagueantes se han descubierto muchos óptimos locales con valores menores a cero en esta región.

En la figura 2.2 se puede apreciar la dificultad de aplicar un algoritmo de optimización a esta función dado a que tiene múltiples óptimos locales en cada región y la información que se obtiene del gradiente no puede ser utilizada para ver en qué dirección el valor de la función disminuye, para esta función se utilizaron los parámetros D = 1.85 y b = 1.5

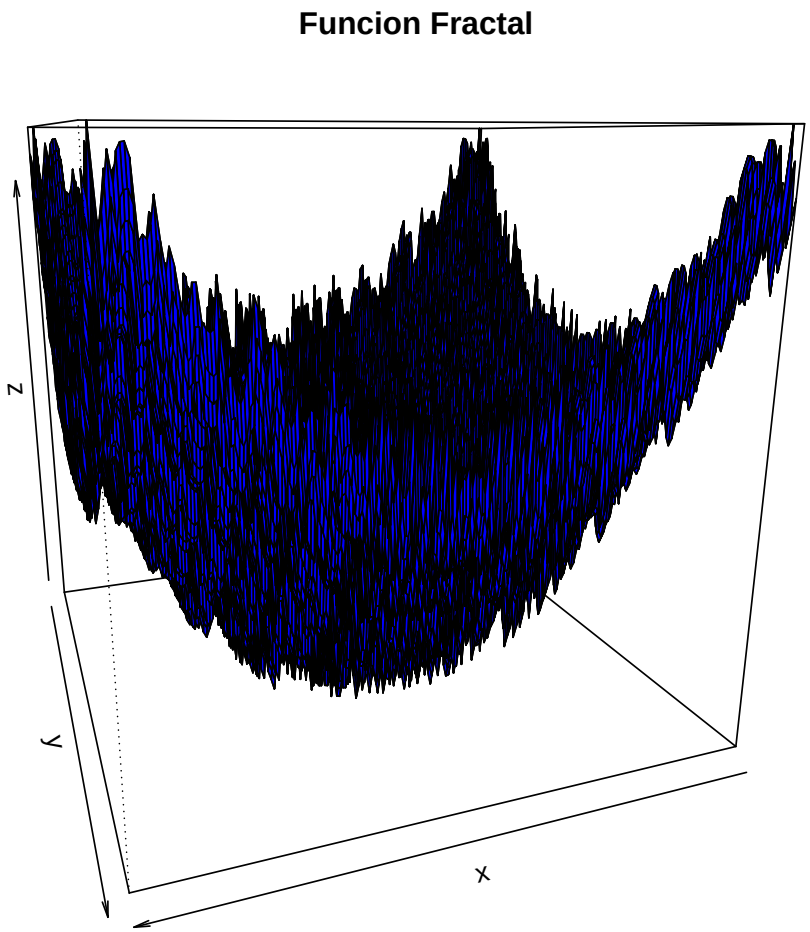


Figura 2.2: Función Fractal en 2D con parámetros D = 1.85 y b = 1.5

2.3 Clúster de Morse

Una aplicación importante para las técnicas de optimización global es la minimización en las estructuras de energía potencial, como lo es el caso de las proteínas y los nanomateriales. El Clúster de Morse es un modelo adecuado para representar diversos clústeres atómicos y es un retador punto de referencia para los algoritmos de optimización global.

El modelo consiste [14] en una expresión para las interacciones atómicas por parejas mostrada en la ecuación (2.5)

$$V_{ij} = e^{2p(1-r_{ij})} - 2e^{p(1-r_{ij})} \tag{2.5}$$

Donde r_{ij} es la distancia euclidiana entre átomos y p es el parámetro que representa la separación equilibrio de un par. El problema consiste en minimizar la energía potencial en un clúster con **N** átomos (2.6)

$$V = \sum_{i<j} V_{ij}. \tag{2.6}$$

Mediante el modelo de Morse, se pueden hacer predicciones realistas para clústeres como C_{60} (usando $p = 13.62$), Sodio (con $p = 3.15$) y Níquel ($p = 3.96$), solo por mencionar algunos. Las configuraciones de energías mínimas son importantes para direccionar las propiedades físicas y químicas de un sistema dado.

Desde hace tiempo, se ha reconocido tanto experimentalmente como teóricamente que las moléculas complejas y los Clústeres atómicos poseen propiedades únicas que los hacen objeto de investigación.

Además del entendimiento de problemas fundamentales como el funcionamiento de las leyes cuánticas y termodinámicas en sistemas de nano-escala o mecanismos para la formación de sistemas complejos multiatómicos además de su auto ensamblaje y funcionamiento.[2]

Capítulo 3

Heurísticas en Paralelo

3.1 PSO paralelo

Como ya se menciono el PSO consta de partículas que evolucionan con el tiempo siguiendo una partícula líder, lo que genera cierta dependencia de datos ya que el enjambre necesita tanto la ubicación como velocidad de la partícula líder para actualizar su posición, por lo que una paralelización dentro de un mismo PSO resultaría en un proceso más lento, de modo que una forma en la que se propone paralelizar el PSO es realizando múltiples PSO's con enjambre aleatoriamente inicializados de forma paralela, es decir, cada procesador realiza un algoritmo de PSO completo generando en cada proceso paralelo una nueva solución. Con las cuales se construye un nuevo enjambre para un nuevo proceso final del cual se obtiene la solución a mostrar al usuario.

Para este experimento se utilizó la función de Rosenbrock con 30 enjambres de 30 partículas con dimensión 30, por un máximo de 500 iteraciones cada uno en un rango de búsqueda de -10 a 10 para cada partícula, de modo que en cada proceso se realizaron 15 000 evaluaciones de la función objetivo, esto para los 30 procesos sumando un total de 450 000 evaluaciones de la función objetivo para terminar con un PSO final de 500 iteraciones con las 30 soluciones obtenidas de los procesos anteriores de modo que en total este nuevo algoritmo de PSO paralelo realiza 465 000 evaluaciones de la función objetivo.

283.90339	576.14161	142.71687	86.45816	264.74721	127.63341
165.57691	170.40439	442.41353	246.05781	254.14353	168.19607
103.23871	529.97114	404.20478	208.00080	300.31737	624.59884
106.38839	103.97389	222.67322	263.17914	1756.21821	827.53163
198.79307	113.75486	229.24805	242.50510	279.53279	142.11892

Tabla 3.1: Valores de la Función objetivo obtenidos con PSO paralelo

En el experimento se obtuvieron los resultados de la tabla 3.1 tomando estos resultados como parte de un nuevo enjambre inicial para un último PSO se obtuvo una solución final de 80.446 con 500 iteraciones en un tiempo total de 166.521s mientras que con una versión serial se obtuvo una solución similar en un tiempo total de 289.624, estos tiempos aquí mostrados incluyen el tiempo de procesamiento de los 30 procesos iniciales en ambos casos.

El caso del PSO serial se utilizó el mismo método anteriormente descrito, es decir, se realizaron 30 procesos preliminares con el fin de construir un enjambre para un proceso final.

3.2 PAGEDA: Parallel Adaptive Gibbs sampling Estimation Density Algorithm

Para este algoritmo se desarrollaron diversos métodos de paralelización para obtener el mejor desempeño posible, PAGEDA es una versión paralela de AGEDA en la que se paraleliza en una primera instancia el muestreo de Gibbs, ya que este consiste en la parte más pesada computacionalmente del algoritmo debido a que se realizan dentro de dicho ciclo la mayor cantidad de evaluaciones de la función objetivo esto además de funciones objetivo computacionalmente pesadas hacen que se consuma una mayor cantidad de tiempo en esta sección.

3.2.1 Muestreo de Gibbs

En el muestreo de Gibbs una Cadena de Markov que converge a una densidad de interés $p(\vec{x})$ es construida muestreando las condicionales $p(x_n|\{x_{j\neq n}\})$. Simular un valor para cada variable individual a partir de estas condicionales se le llama un muestreo de Gibbs [13]. Generalmente los resultados de esta simulación convergerán a la densidad objetivo con una progresión geométrica [5]. Si no es posible muestrear directamente desde las condicionales una solución es incorporar un algoritmo estilo metrópolis para simularlo desde cada una de ellas. Estos razonamientos son pasos esenciales en el muestreo de gibbs.

Los puntos candidatos se generan con la ecuación (3.1).

$$x_n^* = x_n^t + c_n Z, \tag{3.1}$$

Donde Z es una variable normal estándar y c_n es un parámetro escalar. El punto candidato será aceptado con la probabilidad obtenida en base a la ecuación (3.2)

$$P = min\left[1, \frac{p(x_1, x_2, ..., x_n^*, ..., x_N)}{p(x_1, x_2, ..., x_n^t, ..., x_N)}\right] \tag{3.2}$$

en otro caso $x_n^{t+1} = x_n^t$. Sin embargo con valores suficientemente grandes de las c_n's las tasas de aceptación serán pequeñas y con c_n's que tienden a cero las tasas de aceptación tienden a uno. Esta propiedad no solo permite definir intensificación en la búsqueda sino también nos da el criterio para la exploración de la superficie a nivel de cada variable.

En esta Heurística se seleccionan los periodos de intensificación con la ecuación (3.3)

$$c_n = c_n^o \tau^{-\alpha}, \alpha > 0, \tag{3.3}$$

Donde c_n^o es una constante elegida para que inicialmente las tasas de aceptación tiendan a cero. La variable τ representa el numero de iteración y en cada iteración un numero G de ciclos es realizado.

Pseudocódigo 3 Algoritmo del Muestreo de Gibbs

1: Dado un punto inicial s de longitud p, parámetros iniciales $/c_p$'s y numero de iteraciones m.
2: Generar un arreglo vacío vth de dimensiones $m \times p$
3: Evaluar la función utilizando el punto inicial.
4: Generar un vector $arate$ de longitud p.
5: Hacer $th0 \leftarrow s$.
6: **para** $i = 1$ to m **hacer**
7: **para** $j = 1$ to p **hacer**
8: Hacer $th1 \leftarrow th0$.
9: Modificar th1 en la posición [j] añadiendo $rn * c_j$.
10: Evaluar la función para el punto th1.
11: Aceptar o rechazar el punto con una probabilidad u.
12: Reemplazar el valor de la j-ésima variable si el punto es aceptado.
13: Actualizar el valor de la función si este es aceptado.
14: Actualizar el valor en el arreglo vth.
15: Actualizar el valor en el vector $arate$.
16: **fin para**
17: **fin para**
18: Obtener el promedio de Aceptación

En el algoritmo del muestreo de Gibbs mostrado en el Pseudocódigo 3, las c_p's son los parámetros que determinan la escala, es decir el tamaño del movimiento, el punto inicial s consta de p variables y se realizaran m iteraciones en el muestreo generando m nuevos puntos candidatos, en el arreglo vth se guardan los puntos candidatos variable por variable hasta completar un punto y después se prosigue con el siguiente punto candidato mientras que en el arreglo $arate$ se guardan las veces que una variable ha sido aceptada.

Al final del algoritmo se divide las veces que fue aceptada una variable entre el número total de iteraciones para obtener el promedio de aceptación, el término $rn * c_j$ hace referencia a un numero de distribución normal multiplicado por la escala correspondiente a la posición actual del punto en la componente j en el muestreo.

La probabilidad u de que un cambio sea aceptado varía dependiendo de los valores obtenidos en las evaluaciones, es decir se calcula como muestra la ecuación 3.4 donde $f1$ es el valor de la función con el punto modificado mientras $f0$ es el valor del punto antes del cambio y cuando un numero aleatorio de distribución uniforme es menor a dicha probabilidad, u toma un valor de 1 aceptando el movimiento de otra forma lo rechaza haciendo $u = 0$

$$u = U(0,1) < e^{(f1-f0)} \tag{3.4}$$

3.2.2 Muestreo de Gibbs Paralelo (Variable por Variable)

Como se puede apreciar en la Ecuación 3.1 un muestreo de Gibbs genera un punto candidato después de modificar el valor del punto inicial variable por variable para después aceptar o rechazar el punto con una determinada probabilidad, este fue el primer razonamiento para generar el muestreo de Gibbs en paralelo con el fin de aprovechar su paralelismo intrínseco es decir, si altera el punto inicial variable por variable se puede generar un ciclo en donde se altere el punto en cada variable al mismo tiempo reduciendo así el tiempo de cómputo dependiendo de la cantidad de procesadores, es decír, un procesador aplicará movimientos a una variable dejando fijo el resto, mientras que otro procesador mueve otra variable fijando la variable utilizada en el otro procesador.

Para evaluar la efectividad del muestreo de Gibbs se optó por reproducir el ejemplo utilizado en [1] obtenido del tutorial de la aplicación para Modelado flexible bayesiano y muestreo de cadenas de Markov. Cuya función es la ecuación (3.5).

$$V(\vec{x}) = -\left(\frac{1}{2}(x^2 + y^2 + z^2) + (x + y + z)^2 + \frac{10,000}{(1 + x^2 + y^2 + z^2)}\right) \tag{3.5}$$

El muestreo de esta función produce una densidad en forma de anillo en 3 dimensiones: Figura 3.1, sin embargo al implementar la paralelización por variables se obtuvo el siguiente resultado Figura:3.2.

De esta manera se puede apreciar como no cualquier estilo de paralelización

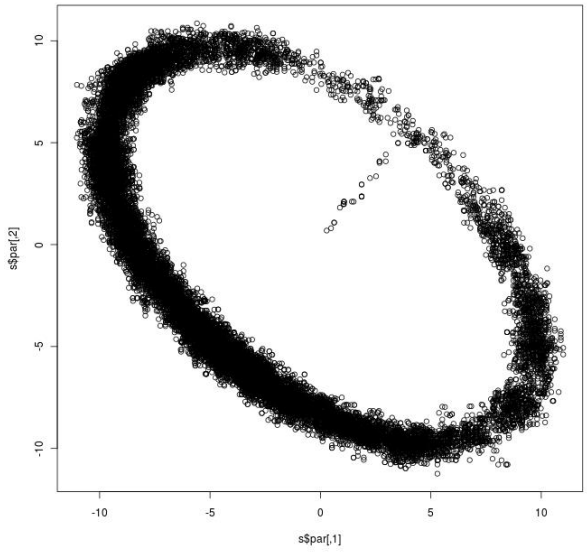


Figura 3.1: Muestreo de Gibbs Ordinario aplicado a la función 3.5

produce un buen resultado esto es debido a que al mover variables al mismo tiempo ocasiona que el muestreo se vaya siempre en la misma dirección con respecto a la variable, es decir, en una ecuación con más de una variable como en la ecuación (3.5), se observa que al fijar dos variables y mover solamente una la función varia de forma diferente que modificando las variables secuencialmente, de modo que no se produce un muestreo eficiente.

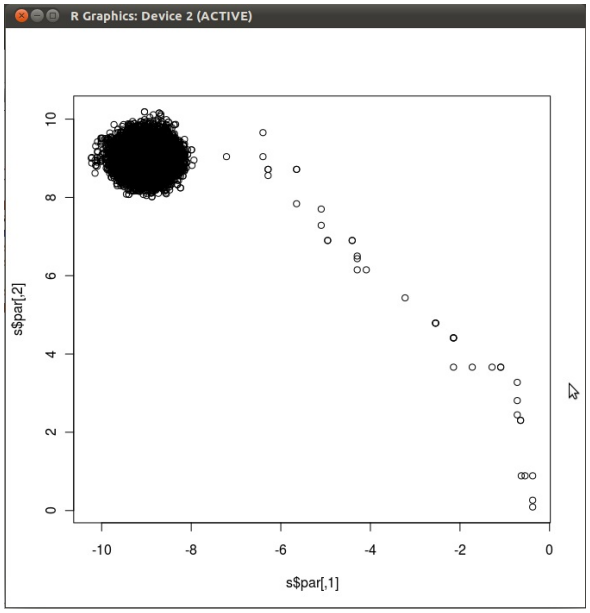


Figura 3.2: Muestreo de Gibbs paralelizando variables aplicado a la función 3.5

El pseudocódigo de este algoritmo se encuentra en el Pseudocódigo 4, en este algoritmo el vector *nump* se utiliza como un indicador de proceso, es decir en el proceso 1 tendrá un valor de 1 y en el proceso 2 un valor de 2 y así sucesivamente, lo que nos permite tener un criterio para ejecutar los procesos de diferente manera, en este caso es el indicador de la variable que va a evolucionar cada proceso en el muestreo de Gibbs.

El resultado de este algoritmo son p matrices de $m \times p$ de las cuales solo la columna *nump* tendrá un determinado valor y las demás serán ceros, de igual manera el vector con los promedios de aceptación solo tendrá un valor en la posición *nump* de modo que para fusionar los resultados de los procesos paralelos para que el resultado final sea equivalente al muestreo de Gibbs original, se suman los vectores *arate* y las matrices *vth* debido a que solo poseen valores en las respectivas posiciones al número de proceso y las posiciones restantes poseen un valor de cero.

Pseudocódigo 4 Algoritmo del Muestreo de Gibbs Paralelo por Variable

1: Dado un punto inicial s de longitud p, parámetros iniciales c_p's y numero de iteraciones m.
2: Generar un vector de procesos $nump$ de 1 hasta p.
3: Generar p procesos.
4: Inicio de Procesos paralelos.
5: Generar un arreglo vacío vth de dimensiones mxp
6: Evaluar la función utilizando el punto inicial.
7: Generar un vector $arate$ de longitud p.
8: Hacer $th0 \leftarrow s$.
9: **para** $i = 1$ to m **hacer**
10: Hacer $th1 \leftarrow th0$.
11: Modificar th1 en la posición $[nump]$ añadiendo $rn * C_j$.
12: Evaluar la función para el punto th1.
13: Aceptar o rechazar el punto con una probabilidad u.
14: Reemplazar el valor de la j-ésima variable si el punto es aceptado.
15: Actualizar el valor de la función si este es aceptado.
16: Actualizar el valor en el arreglo vth.
17: Actualizar el valor en el vector $arate$.
18: **fin para**
19: Obtener el promedio de Aceptación.
20: Fin de los Procesos paralelos.
21: Fusionar resultados

Es importante conservar el formato de salida de la versión serial del muestreo de Gibbs por cuestiones de compatibilidad con el PAGEDA de modo que sea más sencilla su implementación.

3.2.3 Muestreo de Gibbs Paralelo (Múltiple Inicio)

Como segunda alternativa para desarrollar un muestreo de Gibbs paralelo se desarrollo un algoritmo de modo que reciba múltiples puntos iniciales uno para cada procesador comenzando muestreos de Gibbs independientes paralelos para al final de dichos procesos fusionar sus resultados uniendo sus puntos candidatos y sus promedios de aceptación generando así finalmente un muestreo de Gibbs ordinario pero en un tiempo menor a un muestreo serial. Esto con el objetivo de aprovechar el transciente de la función, es decir, al realizar múltiples muestreos de Gibbs paralelos se obtiene una exploración mayor en las áreas cercanas al punto inicial del muestreo.

En una primera instancia se realizan muestreos paralelos con puntos iniciales aleatorios seleccionados de tal manera que los puntos iniciales estén lo suficientemente distanciados uno de otro además de que sean lo suficientemente diversos es decir estén en diferentes regiones de búsqueda, sin embargo esto origina una desventaja, ya que al no poderse introducir un punto inicial el muestreo de Gibbs no puede adaptarse al momento de integrarlo al PAGEDA esto quiere decir que siempre se muestrean regiones aleatorias y aunque el algoritmo pueda identificar regiones de interés en la función, el muestreo no puede enfocarse en ellas, por lo que esta versión fue modificada y diseñada de manera en que con un mismo punto inicial se realicen dos muestreos, esto tuvo interesantes repercusiones:

Por ejemplo se pueden aprovechar los transcientes de la función es decir los puntos cercanos al punto inicial en diferentes direcciones, ya que maneja probabilidades para hacer cambios en el punto, jamás se producirán dos muestreos iguales en igualdad de condiciones, esta propiedad logró mejoras en los resultados en comparación con la versión serial para problemas no unimodales de superficies rugosas, ya que se explora una mayor cantidad de puntos cercanos al punto inicial.

De igual manera para comprobar la funcionalidad de este estilo de paralelización se muestreo la densidad del anillo en 3 dimensiones y se obtuvo la Figura 3.3 donde se pueden apreciar claramente ambos transcientes de la función en este caso con dos procesadores, además de que se realiza un muestreo más parecido a la función original, sin embargo este método también trajo consigo una desventaja, el muestreo de Gibbs depende de realizar muchas evaluaciones a la función para alejarse del transciente y producir un muestreo de calidad, es decir hay funciones en las que es necesario obtener un muestreo más limpio y para ello debe eliminarse el transciente, sin embargo al realizar este muestreo paralelo el número total de iteraciones se divide entre los diferentes procesadores esto ocasiona que se ejecuten diversos muestreos más pequeños obteniendo en consecuencia más transciente, de modo que para algunos problemas este método puede ser más eficiente y para otros no lo es.

El Pseudocódigo 5 muestra el funcionamiento de este algoritmo.

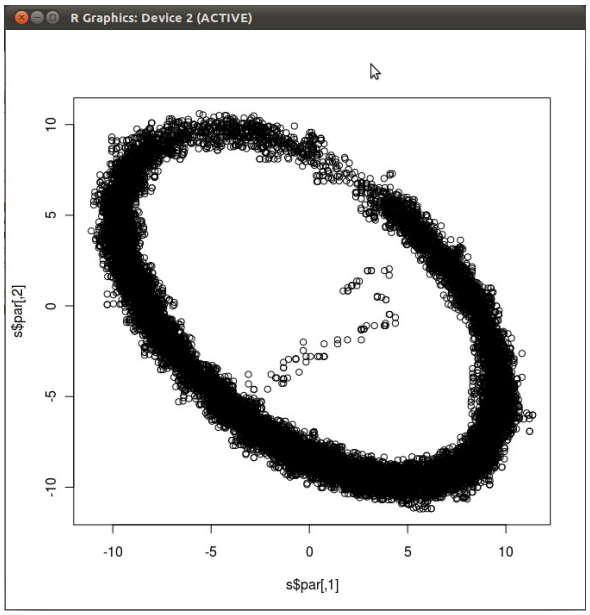


Figura 3.3: Muestreo de Gibbs Paralelo aplicado a la función 3.5

Pseudocódigo 5 Algoritmo del Muestreo de Gibbs Paralelo Múltiple Inicio

1: Dado un punto inicial s de longitud p, parámetros iniciales c_p's y numero de iteraciones m.
2: Generar un vector de procesos $nump$ de 1 hasta p.
3: Generar p procesos.
4: Hacer $num \leftarrow \lfloor m/p \rfloor$
5: Inicio de Procesos paralelos.
6: Generar un arreglo vacío vth de dimensiones $numxp$
7: Evaluar la función utilizando el punto inicial.
8: Generar un vector $arate$ de longitud p.
9: Hacer $th0 \leftarrow s$.
10: **para** $i = 1$ to m **hacer**
11: **para** $j = 1$ to p **hacer**
12: Hacer $th1 \leftarrow th0$.
13: Modificar th1 en la posición $[nump]$ añadiendo $rn * C_j$.
14: Evaluar la función para el punto th1.
15: Aceptar o rechazar el punto con una probabilidad u.
16: Reemplazar el valor de la j-ésima variable si el punto es aceptado.
17: Actualizar el valor de la función si este es aceptado.
18: Actualizar el valor en el arreglo vth.
19: Actualizar el valor en el vector $arate$.
20: **fin para**
21: **fin para**
22: Obtener el promedio de Aceptación.
23: Fin de los Procesos paralelos.
24: Fusionar resultados

En este algoritmo el vector *nump* se utiliza solo como un control y es necesario por la función para paralelizar, el resto de las variables y arreglos se utilizan de igual manera que en el pseudocódigo anterior.

Para fusionar los resultados en este algoritmo todas las soluciones candidato encontradas en cada proceso se guardan en la matriz *vth* modificada, es decir a la matriz vth del primer proceso se le añaden las soluciones encontradas en el resto de los procesos y en el caso de los promedios de aceptación de cada proceso, se calcula un promedio de cada tasa de aceptación, logrando así un resultado de formato similar al muestreo de Gibbs ordinario.

3.2.4 Muestreo de Gibbs con Temperatura

Para esta variante del muestreo de Gibbs se utiliza una escala más grande para producir un muestreo general y localizar de esa manera puntos de interés ya que al realizar muestreos a bajas temperaturas es probable que la función se quede atrapada en determinada región y produzca un muestreo no representativo [7].

Como se puede apreciar en la imagen 3.4 un muestreo de baja temperatura es probable que quede atrapado en algunas regiones como las sombreadas.

En el Pseudocódigo 6 se describe el muestreo de Gibbs con temperatura. Para este algoritmo fueron introducidos nuevos parámetros, como *bm* que es el número de iteraciones que realiza el muestreo de una escala más grande, es decir mayor temperatura, después de realizar el muestreo, se producen *bm* puntos de los cuales se seleccionaran *nump* puntos candidatos, uno para cada proceso, para seleccionar los puntos que serán utilizados como puntos iniciales en los muestreos de Gibbs paralelos utiliza el pseudocódigo 7, eliminando los puntos más cercanos a cada punto candidato *nump*, se garantiza que todos los puntos seleccionados estarán suficientemente dispersos en el espacio de búsqueda.

Como puede observarse en la imagen 3.4 se asegura que los puntos se encuentren en regiones dispersas eliminando los puntos más cercanos y también que se encuentran en zonas de interés al seleccionarlos por su valor en la función objetivo.

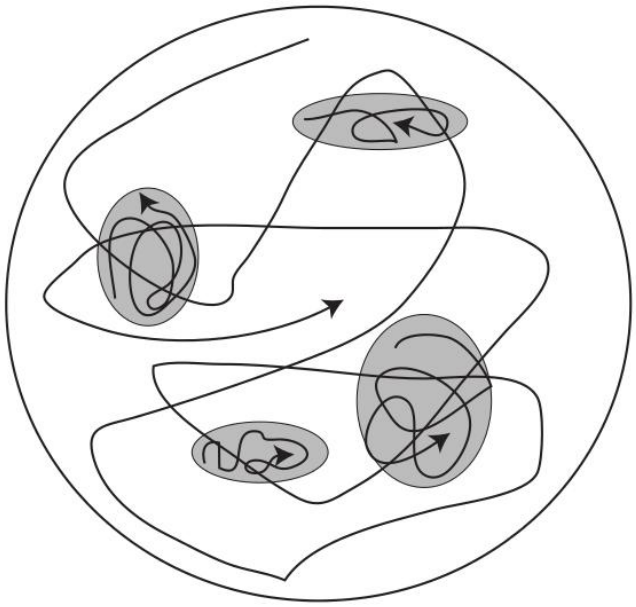


Figura 3.4: Muestreo con Temperatura (búsqueda exploratoria de alta temperatura y búsqueda de intensificación de baja temperatura)

Pseudocódigo 6 Algoritmo del Muestreo de Gibbs Paralelo con Temperatura

1: Dado un punto inicial s de longitud p, parámetros iniciales c_p's y numero de iteraciones m y numero de iteraciones de alta temperatura bm.
2: Realizar un muestreo de bm iteraciones con parámetros iniciales C_p's.
3: **para** $i = 1$ hasta p **hacer**
4: Seleccionar el mejor punto candidato encontrado.
5: Asignar el punto al proceso p.
6: Eliminar los $\lfloor bm/p \rfloor$ candidatos mas cercanos al seleccionado.
7: **fin para**
8: Generar $nump$ procesos paralelos.
9: Generar un arreglo vacío vth de dimensiones mxp
10: Evaluar la función utilizando el punto inicial.
11: Generar un vector $arate$ de longitud p.
12: Hacer $th0 \leftarrow s$.
13: **para** $i = 1$ to m **hacer**
14: **para** $j = 1$ to p **hacer**
15: Hacer $th1 \leftarrow th0$.
16: Modificar $th1$ en la posición $[j]$ añadiendo $rn * C_j$.
17: Evaluar la función para el punto $th1$.
18: Aceptar o rechazar el punto con una probabilidad u.
19: Reemplazar el valor de la j-ésima variable si el punto es aceptado.
20: Actualizar el valor de la función si este es aceptado.
21: Actualizar el valor en el arreglo vth.
22: Actualizar el valor en el vector $arate$.
23: **fin para**
24: **fin para**
25: Obtener el promedio de Aceptación.
26: Fin de los Procesos paralelos.
27: Fusionar resultados

Pseudocódigo 7 Selección de Candidatos

1: Seleccionar el mejor punto candidato.

2: Eliminar los $\lfloor bm/p \rfloor$ puntos más cercanos al punto seleccionado y pasar al paso 1.

Ya que PAGEDA usa una escala adaptativa para así producir el muestreo adaptativo, la escala de mayor temperatura es un múltiplo de la escala original para conservar estas propiedades.

3.2.5 PAGEDA

La versión final de PAGEDA se obtuvo implementando conocimientos del parallel tempering [7] que consta en el intercambio de soluciones por procesos equivalentes, es decir para esta versión final se paralelizo no solo el muestreo de Gibbs sino toda la heurística, de modo que al mismo tiempo se está ejecutando un PAGEDA en cada proceso y después con cierta probabilidad intercambian soluciones, logrando así búsquedas en diferentes regiones de interés para una mejor exploración de la función a optimizar.

Al templado en paralelo también se le conoce como replicas de intercambio, una técnica de simulación que puede ser rastreada hasta Swendsen, Robert H. and Wang, Jian-Sheng [23], estas replicas tienen diferentes criterios de búsqueda basados en temperatura e intercambian sus soluciones parciales con el proceso contiguo, sin embargo el proceso con temperatura más alta intercambia soluciones con el de temperatura más baja.

La idea general de este tipo de procesamiento es el tener M replicas de un proceso, cada replica generalmente con una diferente temperatura, de este modo las replicas con temperaturas más altas pueden muestrear una mayor cantidad de espacio mientras que las replicas de bajas temperaturas muestrean una región más pequeña y podrían quedar atrapadas en mínimos locales.

El templado paralelo produce buenos resultados ya que permite a las M replicas compartir configuraciones completas. De este modo las replicas de alta temperatura aseguran que las de baja temperatura pueden acceder a regiones representativas del espacio de búsqueda.

Estas simulaciones han logrado demostrar que son $1/M$ más efectivas que una simulación con una sola temperatura, este incremento en la eficiencia se logra al permitir a los sistemas de baja temperatura muestrear regiones en las que no tendrían acceso en una simulación simple a una sola temperatura. [7]

La probabilidad de intercambio recomendada para este tipo de simulaciones es de un 20 % como se muestra en [7].

El Pseudocódigo del Algoritmo PAGEDA se muestra en el Pseudocódigo 8. Donde PN es el numero de procesadores y el numero de procesos, los números aleatorios en el vector H tienen una distribución uniforme entre cero y uno, de modo que funcionan como vector de probabilidades, es decir cuando un numero sea menor a la probabilidad de intercambio estos procesos intercambiaran soluciones, de modo que es necesario que los procesos paralelos del algoritmo terminen para intercambiar soluciones, pero si este no ha cumplido con el criterio de terminación volverá a iterar después de haber intercambiado soluciones hasta cumplir el criterio de terminación.

Pseudocódigo 8 Algoritmo de PAGEDA

1: Se genera un vector de números aleatorios H.
2: Se generan las PN replicas.
3: **mientras** No se cumpla el criterio de Terminación **hacer**
4: Dada una población de tamaño M y parámetros iniciales c_n's.
5: Inicializar $t \Leftarrow 0$. Generar $M >> 0$ individuos aleatoriamente.
6: Evaluar los individuos usando la función de costo.
7: Seleccionar el mejor individuo inicial $x_t^{(best)}$ para el muestreo de Gibbs adaptativo.
8: **mientras** No se cumpla el criterio de Terminación y $H[cont] >$ probabilidad de intercambio **hacer**
9: Incrementar en uno el contador *cont*
10: Generar M nuevos individuos mediante el muestreo de Gibbs adaptativo (usando las c_n's y $x_t^{(best)}$.
11: **para** $n = 1$ to N **hacer**
12: **si** $\theta_n > \epsilon$ **entonces**
13: Actualizar c_n.
14: Reemplazar el valor de la n-ésima variable ($x_{n,t}^{(best)}$ aleatoriamente.
15: **fin si**
16: Actualizar c_n.
17: **fin para**
18: **si** $\langle \theta$'s$\rangle > \beta$ **entonces**
19: Mejorar la solución del mejor individuo x_t^{best} por medio de la estrategia de busqueda local.
20: **fin si**
21: Actualizar $t \Leftarrow t + 1$
22: **fin mientras**
23: Intercambio de soluciones
24: **fin mientras**

Capítulo 4

Experimentación

Para esta sección se utilizó una computadora de 4 procesadores con las especificaciones de la tabla 4.1.

En esta sección se realiza la experimentación correspondiente para cada método anteriormente mencionado a las diversas funciones de prueba ya definidas, haciendo mención a sus respectivos resultados.

En este experimento uno de los parámetros principales es el número de iteraciones del muestreo de Gibbs ya que esto le permite a PAGEDA muestrear una sección de la función y la cantidad de iteraciones afecta directamente en la calidad del muestreo realizado, pero de igual manera implica mayor tiempo de cómputo y consume una mayor cantidad de evaluaciones de la función objetivo lo que obliga a PAGEDA a realizar menos iteraciones.

model name	Intel(R) Xeon(R) CPU E5320 @ 1.86GHz
cpu MHz	1862.114
cache size	4096KB
MemTotal	3988684 kB

Tabla 4.1: Especificaciones de Sistema

El parámetro utilizado en esta experimentación es $m = 200$, cuando se menciona la mitad de iteraciones en el muestreo de Gibbs se hace referencia a $m = 100$, cuando se menciona el doble de iteraciones equivale a $m = 400$ y los tiempos de cómputo mencionados están expresados en segundos. Un análisis comparativo directo entre PAGEDA y AGEDA se muestra en la sección 4.5

4.1 Muestreo de Gibbs Múltiple Inicio

Este método fue desarrollando basado en la teoría de que una mayor exploración del transciente podría brindar resultados de mejor calidad, por lo que fue desarrollado y probado para comprobar dicha hipótesis.

4.1.1 Función de Rosenbrock

Para esta experimentación como criterio de parada para PAGEDA se eligió el número de evaluaciones de la función objetivo, ya que de este modo es posible comparar a PAGEDA con su versión serial y otros métodos como en [24] una vez que sea elegido el método de paralelización a comparar con la versión serial. Entre los primeros resultados se encuentran los que utilizaron el muestreo de Gibbs con múltiple inicio mencionado anteriormente, se evaluó su desempeño primeramente en la función de Rosenbrock, recordemos que esta función tiene un optimo global conocido $f(\vec{x}) = 0$ se probó el algoritmo para diversas dimensiones de la función de Rosenbrock los resultados para la función de Rosenbrock con 40 variables se encuentran en la tabla 4.2 en donde cada fila representa una corrida diferente con sus respectivos valores promedios y desviación estándar.

En la tabla 4.3 se muestran los resultados para la función de Rosenbrock de 55 variables de igual manera cada fila representa una corrida con sus valores promedio obtenidos, el tiempo promedio de cómputo y la desviación estándar.

Corrida	Valor Obtenido	Tiempo de Cómputo
1	24.3919	557.116
2	23.6619	591.704
3	21.69779	591.184
4	27.62489	584.0890000000002
5	29.58417	584.634
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
25.39213	581.7454	3.1702019068

Tabla 4.2: Resultados del Muestreo de Gibbs con múltiple inicio para la función de Rosenbrock con 40 Variables

Como se puede apreciar al aumentar el número de variables la calidad de solución disminuye, sin embargo para este problema en particular el tiempo de cómputo no parece tener muchos cambios, para la tabla 4.4 se utiliza el mismo criterio de parada solo con la diferencia que las iteraciones del muestreo de Gibbs fueron reducidas a la mitad en este caso también cada fila representa una corrida diferente, esto permitiéndole al algoritmo realizar más iteraciones mostrando los resultados promedio.

En la tabla 4.5 donde cada fila representa una corrida, se puede observar como el tiempo computacional es menor esto es debido que al incrementar el número de iteraciones permitidas por el muestreo de Gibbs se incrementa el número de evaluaciones realizadas en cada iteración del algoritmo, por lo que éste realiza una menor cantidad de iteraciones lo que reduce el tiempo.

En la tabla 4.6 se puede apreciar como la calidad de la solución incrementa en comparación con la tabla 4.5 así como también el tiempo de cómputo esto es ya que al permitirle el doble de evaluaciones el proceso toma el doble de tiempo en comparación al PAGEDA con solo el doble de iteraciones en el muestreo de Gibbs, más sin embargo como el número de evaluaciones permitidas es duplicado también esto le permite a PAGEDA realizar un mayor número de iteraciones lo que mejora la calidad de la solución obtenida.

Corrida	Valor Obtenido	Tiempo de Cómputo
1	45.46724	538.9699999999999
2	40.80482	581.66
3	42.43668	582.307
4	45.47016	582.4450000000002
5	45.49681	581.6379999999999
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
43.935142	573.404	2.1901321691

Tabla 4.3: Resultados de la Función de Rosenbrock con 55 Variables utilizando el muestreo de Gibbs con múltiple inicio

4.1.2 Función Fractal

De igual manera y con el mismo criterio de terminación se prueba PAGEDA con la función fractal para diversas dimensiones con los mismos cambios en los parámetros que en el experimento anterior, de manera que los resultados para la función fractal con 40 variables pueden observarse en la tabla 4.7 para esta función no hay un óptimo conocido, por lo que los valores negativos son bien aceptados, se muestran también los valores promedio así como la desviación estándar de las soluciones encontradas, en las tablas 4.7, 4.8, 4.9, 4.10, 4.11 las filas de cada tabla representan una corrida.

Como se puede apreciar en la tabla este problema posee una función más pesada computacionalmente por lo que PAGEDA requiere de más tiempo para cada evaluación de la función. Los resultados de la función fractal con 55 variables se encuentran en la tabla 4.8 como se puede apreciar el tiempo de cómputo se ha incrementado notablemente al introducir 15 variables mas.

Corrida	Valor Obtenido	Tiempo de Cómputo
1	47.4678	538.9699999999999
2	51.52415	581.66
3	47.50555	582.307
4	50.40901	582.4450000000002
5	47.79892	581.6379999999999
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
48.941086	573.404	1.8949207711

Tabla 4.4: Resultados de la función de Rosenbrock con 55 variables con la mitad de iteraciones en el muestreo de Gibbs de múltiple inicio

De igual manera al disminuir el número de iteraciones del muestreo de Gibbs el tiempo de cómputo se incrementa debido a que el PAGEDA realiza una cantidad mayor de iteraciones para este caso cabe destacar que otros procesos ocupaban recursos en el servidor por lo que los tiempos de cómputo fueron afectados, estos resultados se pueden apreciar en la tabla 4.9.

Los resultados para la función fractal con el doble de iteraciones en el muestreo de Gibbs pueden apreciarse en la tabla 4.10 de igual manera se puede apreciar que el tiempo de cómputo ha sido reducido ya que PAGEDA realiza una cantidad menor de iteraciones.

Los resultados para el doble de iteraciones y el doble de evaluaciones de la función objetivo se pueden observar en la tabla 4.11.

4.1.3 Clúster de Morse

Esta Función resulto ser la más retadora ya que tiene una alta complejidad y además cuenta con un optimo conocido dependiendo de la dimensión, para esta función no se realizaron experimentos con dimensionalidad 40, sin embargo si se realizaron experimentos para dimensión 55 como en las funciones anteriores, cuyos resultados pueden observarse en la tabla 4.12.

Corrida	Valor Obtenido	Tiempo de Cómputo
1	51.06905	303.413
2	48.57239	344.381
3	52.23756	341.972
4	50.38233	341.4540000000001
5	49.49183	342.0609999999999
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
50.350632	334.6562	1.4122067792

Tabla 4.5: Resultados de la función de Rosenbrock con 55 variables con el doble de iteraciones en el muestreo de Gibbs de múltiple inicio

Para dimensión 55 el optimo conocido es $f(\vec{x}) = -250.29$, En este caso en particular al momento de reducir el número de iteraciones en el muestreo los resultados fueron totalmente erróneos, sin embargo al momento de duplicar el número de iteraciones en el muestreo de Gibbs los resultados no fueron lo esperado como se puede apreciar en la tabla 4.13, de igual manera se puede apreciar como los valores son en promedio mejores mientras el tiempo de cómputo se incrementa al duplicar el número máximo de evaluaciones de la función objetivo permitidos, esto en la tabla 4.14.

4.2 Muestreo de Gibbs con Temperatura

Debido a la complejidad retadora de los Clúster de Morse esta versión de PAGEDA se concentro únicamente en esta función, logrando un mejor enfoque sobre su desempeño, en este experimento siempre se trabajo con dimensionalidad 55 y ahora hay otro parámetro a controlar, el tamaño de la escala de exploración así como también el numero de iteraciones para el muestreo exploratorio el parámetro.

Corrida	Valor Obtenido	Tiempo de Cómputo
1	47.3386	617.693
2	47.07303	681.056
3	46.73031	680.6030000000001
4	48.42704	681.173
5	46.4852	680.7399999999998
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
47.210836	668.253	0.7536443366

Tabla 4.6: Resultados de la función de Rosenbrock con 55 variables con el doble de iteraciones en el muestreo de Gibbs de múltiple inicio y el doble de Evaluaciones de la función objetivo

Inicialmente se fijo en $bm = 100$ y para el ultimo experimento se modifico a $bm = 160$, por lo que se comenzó por ajustar este primer parámetro, el tamaño de la escala, para comenzar a producir los primeros resultados en una primera instancia se comenzó con una escala mayor equivalente al triple de la escala normal (3x) y se produjeron los resultados mostrados en la tabla 4.15, en la tabla 4.16 se muestran los resultados con una escala de exploración de (2x) en la tabla 4.17 se muestran los resultados para una escala de exploración a (1x) se puede observar como los resultados mejoraron con la reducción de la escala de exploración.

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-2.911518	8719.014999999999
2	-3.095363	8615.141999999998
3	-2.823262	8572.132000000001
4	-3.652074	8361.849000000002
5	-3.504387	8528.796000000002
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-3.1973208	8559.3868	0.3650646324

Tabla 4.7: Resultados para la función Fractal con 40 Variables utilizando el muestreo de Gibbs de múltiple inicio

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-2.919411	12150.156
2	-1.727858	11827.769
3	-4.509923	12090.135
4	-2.243727	12016.08199999999
5	-4.318225	12356.174
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-3.1438288	12088.0632	1.2360157466

Tabla 4.8: Resultados de la función Fractal con 55 Variables utilizando el muestreo de Gibbs de múltiple inicio

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-0.9464984	20400.768
2	-2.53391	22276.728
3	-2.649514	23171.54
4	-1.174896	16868.27399999999
5	-4.387123	16668.643
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-2.33838828	19877.1906	1.3805264968

Tabla 4.9: Resultados para la función Fractal con 55 Variables y la mitad de iteraciones en el muestreo de Gibbs de múltiple inicio

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-3.404838	10762.65
2	-3.178315	10742.433
3	-2.810918	10771.294
4	-2.013788	10820.335
5	-2.960173	10774.582
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-2.8736064	10774.2588	0.5305105174

Tabla 4.10: Resultados para la función Fractal con 55 Variables y el doble de iteraciones en el muestreo de Gibbs de múltiple inicio

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-3.960083	21396.725
2	-4.130506	20529.308
3	-5.288282	23901.893
4	-4.611159	25231.04299999999
5	-4.906142	25164.42
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-4.5792344	23244.6778	0.5466584608

Tabla 4.11: Resultados de la función Fractal con 55 Variables, el doble de iteraciones en el muestreo de Gibbs y el doble de evaluaciones de la función objetivo

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-174.5787	5809.823
2	-174.3176	5783.618
3	-185.8439	5799.456999999999
4	-170.0027	5805.350000000002
5	-168.0673	5781.142
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-174.56204	5795.878	6.89763768313

Tabla 4.12: Resultados para el Clúster de Morse con dimensionalidad 55 usando el muestreo de Gibbs de múltiple inicio

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-140.2009	5317.218
2	-172.7034	5275.9
3	-154.3146	5303.735000000001
4	-171.9291	5287.334000000001
5	-139.5325	5342.097000000002
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-155.7361	5305.2568	16.2482481559

Tabla 4.13: Resultados para el Clúster de Morse con dimensionalidad 55 y el doble de iteraciones en el muestreo de Gibbs

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-182.8065	10596.203
2	-163.2549	10513.84
3	-168.4105	10585.21
4	-170.457	10555.465
5	-159.8805	10507.304
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-168.96188	10551.6044	8.79280942159

Tabla 4.14: Resultados para el Clúster de Morse con Dimensionalidad 55, el doble de iteraciones en el muestreo de Gibbs y el doble de evaluaciones de la Función Objetivo

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-146.3159	7150.808
2	-167.7334	6874.34
3	-167.625	6776.468999999999
4	-167.1082	6837.946
5	-166.7241	6787.573
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-163.10132	6885.4272	9.3921482195

Tabla 4.15: Resultados del Clúster de Morse con dimensionalidad 55 y Escala de exploración (3x)

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-130.7881	6805.5
2	-184.6746	6746.165999999999
3	-168.223	6777.145
4	-162.2651	7412.373
5	-168.6379	8030.194000000003
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-162.91774	7154.2756	19.7924649592

Tabla 4.16: Resultados del Clúster de Morse con Dimensionalidad 55 y Escala de exploración (2x)

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-156.2711	7479.705
2	-191.678	7078.839999999999
3	-189.1338	7702.306
4	-172.2712	6738.246999999999
5	-164.9702	6706.852999999999
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-174.86486	7141.1902	15.302316859292

Tabla 4.17: Resultados del Clúster de Morse con Dimensionalidad 55 y Escala de exploración (1x)

Debido al incremento de la calidad de las soluciones al reducir la escala de exploración se decidió seguirla reduciendo para la para observar la tendencia, los resultados con una escala de exploración de (0.3x) se muestran en la tabla 4.18 así mismo se utilizo una escala de exploración de (0.1x) cuyos resultados se muestran en la tabla 4.19. Otra variante que se exploro fue el incremento en las iteraciones del muestreo de exploración produciendo los resultados de la tabla 4.20. Dando como resultado que al momento de seguir reduciendo la escala de exploración a valores menores a 1x, los valores de las soluciones obtenidas comenzaban a incrementarse, por lo que decidio por utilizar la escala de exploración de 1x.

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-179.6878	7103.322999999999
2	-185.9923	6791.008000000001
3	-169.7731	6656.548000000001
4	-153.1796	6852.892
5	-177.1191	6595.437000000002
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-173.15038	6799.8416	12.5861191595

Tabla 4.18: Resultados del Clúster de Morse con Dimensionalidad 55 y Escala de exploración (0.3x)

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-168.4507	6584.168
2	-161.379	6596.702
3	-163.6877	6599.832
4	-151.7885	6616.346999999998
5	-169.1358	6601.192000000003
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-161.49775	6603.51825	7.0016808141

Tabla 4.19: Resultados del Clúster de Morse con Dimensionalidad 55 y Escala de exploración (0.1x)

Corrida	Valor Obtenido	Tiempo de Cómputo
1	-166.7715	8341.093000000001
2	-154.4885	8289.261
3	-168.103	8423.188999999998
4	-172.5624	8316.898999999998
5	-178.6523	8381.449999999997
Valor Promedio	Tiempo de Cómputo Promedio	Desviación Estándar
-168.11554	8350.3784	8.9204917899

Tabla 4.20: Resultados del Clúster de Morse con Dimensionalidad 55, Escala de exploración (0.1x) y 160 iteraciones de exploración

4.3 PAGEDA

La versión final de PAGEDA está compuesta por un grupo de AGEDAS que corren en paralelo con intercambio de soluciones con una determinada probabilidad como ya se había mencionado anteriormente, la diferencia entre procesos está dada por el criterio de intensificación, ya que PAGEDA utiliza la tasa de aceptación del muestreo de Gibbs para hacer una búsqueda local en una región utilizando el método de Nelder-Mead [15].

El objetivo de este nuevo método es la exploración en diversas regiones mediante el uso de temperaturas, ya que al tener diversos criterios para la intensificación el algoritmo por medio del muestreo de Gibbs, se obtienen puntos candidatos en diferentes regiones del espacio y cuando la exploración satisface la condición de generar una tasa de aceptación mayor a un ϵ este procede a intensificar, el punto clave es la tasa de aceptación ya que intensificando la búsqueda en diferentes zonas del espacio podemos encontrar soluciones más diversas y buenas obteniendo un mejor desempeño general.

Para este experimento se evaluó el comportamiento de PAGEDA en las tres funciones de prueba, Rosenbrock, Fractal y Clúster de Morse con diferentes parámetros, en este caso el parámetro importante es el rango para la tasa de aceptación, es decir los valores con los cuales cada AGEDA intensificara la búsqueda para el intercambio de soluciones, los valores están igualmente distribuidos a lo largo de un rango, es decir si el rango es de .1 a .3 en el criterio de aceptación y hay 4 procesos activos los valores para cada proceso son, .1, .17, .24, .31.

El criterio de parada para este algoritmo sigue siendo el numero de evaluaciones de la función objetivo ya que eso delimita el tiempo de cómputo de una manera aproximada y permite una comparación directa con el método serial, el número de evaluaciones limite es dividido equitativamente para cada proceso es decir cada proceso realiza las evaluaciones necesarias para completar su iteración y si esta llega a alcanzar el máximo permitido para su proceso este termina y su solución final es compara con la de los demás procesos mostrando al usuario la mejor de las soluciones obtenidas en caso de no alcanzar el máximo permitido para su proceso este puede terminar debido a la probabilidad de intercambio ya que para realizar un intercambio de soluciones es necesario terminar todos los procesos así intercambiar soluciones y continuar iterando hasta que la suma de todos las evaluaciones de todos los procesos sea mayor al número máximo de evaluaciones permitidas.

Para estos experimentos el servidor utilizado no estaba 100 % disponible por lo que el tiempo de cómputo de cada proceso no fue obtenido, por lo que solo servirán para evaluar el desempeño del algoritmo en cuestión de calidad de la solución para en la sección 4.5 proceder a la comparación en cuestión de tiempo de cómputo y calidad de la solución encontrada.

4.3.1 Rosenbrock

Utilizando un criterio de intensificación en la tasa de aceptación con valores de .60 .67 .74 y .81 para la función de Rosenbrock con 55 Variables realizando un total de 500 000 evaluaciones de la función objetivo produjo los resultados de la tabla 4.21.

Con un rango en las tasas de aceptación de .3 a .9 es decir: .3, .5, .7 y .9 se produjeron los resultados de la tabla 4.22.

Con un rango en las tasas de aceptación de .3 a .9 es decir: .3, .5, .7 y .9 además de incrementar el número de evaluaciones en la función objetivo a 1000000 se produjeron los resultados de la tabla 4.23 esto para comprobar si el promedio de soluciones encontradas puede mejorarse.

Corrida	Valor Obtenido
1	44.03678
2	50.24834
3	51.2877
4	50.90115
5	51.08494
Valor Promedio	Desviación Estandar
49.511782	3.0853370506

Tabla 4.21: Resultados de la Función de Rosenbrock con Dimensionalidad 55 Rango(6-8)

4.3.2 Función Fractal

En esta sección se prueba el desempeño de PAGEDA para la función fractal, con el objetivo de mostrar la robustez de este método paralelo ante un problema no diferenciable, de múltiples óptimos locales además de una alta dimensionalidad.

Utilizando un criterio de intensificación en la tasa de aceptación con valores de .60 .67 .74 y .81 para la función fractal con 55 Variables realizando un total de 500 000 evaluaciones de la función objetivo produjo los resultados de la tabla 4.24.

Con un rango en las tasas de aceptación de .3 a .9 es decir: .3, .5, .7 y .9 se produjeron los resultados de la tabla 4.25.

Con un rango en las tasas de aceptación de .3 a .9 es decir: .3, .5, .7 y .9 además de incrementar el número de evaluaciones en la función objetivo a 1000000 se produjeron los resultados de la tabla 4.26 esto para comprobar si el promedio de soluciones encontradas puede mejorarse.

Corrida	Valor Obtenido
1	52.0327
2	49.12865
3	47.5919
4	49.31232
5	41.53986
Valor Promedio	Desviación Estándar
47.921086	3.9092045619

Tabla 4.22: Resultados de la Función de Rosenbrock con Dimensionalidad 55 Rango (3-9)

4.3.3 Clúster de Morse

Utilizando un criterio de intensificación en la tasa de aceptación con valores de .60 .67 .74 y .81 para los Clúster de Morse con Dimensionalidad 55 realizando un total de 500 000 evaluaciones de la función objetivo produjeron los resultados de la tabla 4.27.

Con un rango en las tasas de aceptación de .3 a .9 es decir: .3, .5, .7 y .9 se produjeron los resultados de la tabla 4.28.

Con un rango en las tasas de aceptación de .3 a .9 es decir: .3, .5, .7 y .9 además de incrementar el número de evaluaciones en la función objetivo a 1000000 se produjeron los resultados de la tabla 4.29 esto para comprobar si el promedio de soluciones encontradas puede mejorarse.

Después de observar las diferentes tablas podemos elegir entre el método y los parámetros más indicados para la función a tratar así como los parámetros que elevan el tiempo de cómputo.

Corrida	Valor Obtenido
1	48.45395
2	49.61564
3	50.21032
4	45.4102
5	45.71202
Valor Promedio	Desviación Estándar
47.880426	2.2120298669

Tabla 4.23: Resultados de la Función de Rosenbrock con Dimensionalidad 55 Rango (3-9) y Doble de evaluaciones

Corrida	Valor Obtenido
1	3.672321
2	6.150828
3	5.431738
4	2.124426
5	6.712768
Valor Promedio	Desviación Estándar
4.8184162	1.8915054178

Tabla 4.24: Resultados de la Función Fractal con Dimensionalidad 55 Rango (6-8)

Corrida	Valor Obtenido
1	6.604872
2	5.056075
3	4.521162
4	5.120328
5	7.278917
Valor Promedio	Desviación Estándar
5.7162708	1.1673438437

Tabla 4.25: Resultados de la Función Fractal con Dimensionalidad 55 Rango (3-9)

Corrida	Valor Obtenido
1	7.752108
2	3.042723
3	3.297295
4	7.200509
5	7.889848
Valor Promedio	Desviación Estándar
5.8364966	2.4494420042

Tabla 4.26: Resultados para la Función Fractal con Dimensionalidad 55 Rango (3-9) y Doble de evaluaciones

Corrida	Valor Obtenido
1	-166.7407
2	-184.5758
3	-187.1934
4	-182.7935
5	-173.7309
Valor Promedio	Desviación Estándar
-179.00686	8.5244590364

Tabla 4.27: Resultados para el Clúster de Morse con Dimensionalidad 55 Rango (6-8)

Corrida	Valor Obtenido
1	-174.6303
2	-182.2968
3	-175.7143
4	-187.3457
5	-184.8398
Valor Promedio	Desviación Estándar
-180.96538	5.594626275

Tabla 4.28: Resultados para el Clúster de Morse con Dimensionalidad 55 Rango (3-9)

Corrida	Valor Obtenido
1	-188.0356
2	-192.6488
3	-200.0077
4	-196.0462
5	-183.0216
Valor Promedio	Desviación Estándar
-191.95198	6.6564355981

Tabla 4.29: Resultados para el Clúster de Morse con Dimensionalidad 55 Rango (3-9) y Doble de evaluaciones

4.4 Comparación de resultados promedio

4.4.1 PAGEDA

Ya que se han mostrado los resultados experimentales, en la figura 4.1 se muestran los valores promedio de la Función de Rosenbrock con Dimensionalidad 55 con PAGEDA en sus tres modalidades, rango 6-8, rango 3-9 y rango 3-9 con el doble de evaluaciones objetivo permitidas. El método 1 es utilizando un rango de 6 a 8 con 500 000 evaluaciones de la función objetivo, mientras que el método 2 es utilizando un rango de 3 a 9 y el método 3 es utilizando el mismo rango pero con 1 000 000 de evaluaciones de la función objetivo.

En la Figura 4.2 se muestran los valores promedio para la función Fractal. El

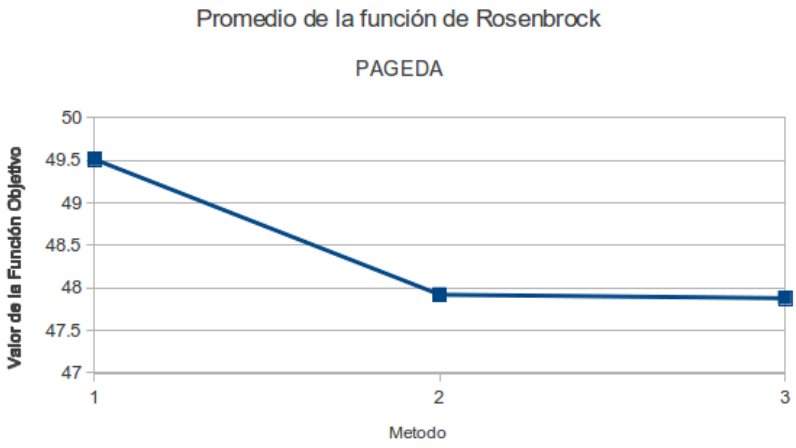


Figura 4.1: Promedio de PAGEDA en la Función de Rosenbrock

método 1 es utilizando un rango de 6 a 8 con 500 000 evaluaciones de la función objetivo, mientras que el método 2 es utilizando un rango de 3 a 9 y el método 3 es utilizando el mismo rango pero con 1 000 000 de evaluaciones de la función objetivo.

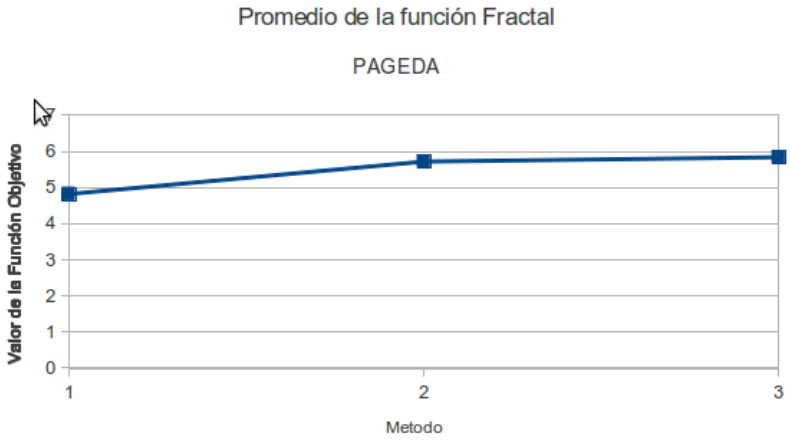


Figura 4.2: Promedio de PAGEDA en la Función Fractal

En la Figura 4.3 se muestran los valores promedio para la función de Clúster de Morse. El método 1 es utilizando un rango de 6 a 8 con 500 000 evaluaciones de la función objetivo, mientras que el método 2 es utilizando un rango de 3 a 9 y el método 3 es utilizando el mismo rango pero con 1 000 000 de evaluaciones de la función objetivo.

4.4.2 Muestreo de Gibbs con Temperatura

En esta sección se muestran los resultados en el análisis de la Función del Clúster de Morse ya que fue la función de prueba para este método, se muestran diversas figuras con los resultados promedio de diversas variantes.

En la Figura 4.4 se muestran los valores promedio de la función de Clúster de Morse en el método 1 se utiliza una escala de exploración de igual magnitud que la escala de intensificación con 100 iteraciones para el muestreo exploratorio, mientras en el método 2 se utiliza una escala de exploración de un tercio de la escala de intensificación y en el método 3 se utilizo una escala de igual magnitud que la escala de intensificación pero el numero de iteraciones exploratoria fue incrementado a 150.

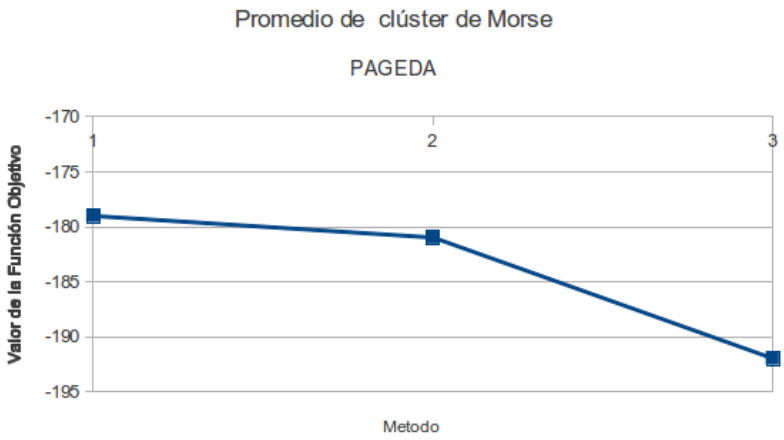


Figura 4.3: Promedio de PAGEDA con el Clúster de Morse

Ahora en base a los datos anteriores podemos concluir que PAGEDA ha demostrado ser una heurística de calidad proporcionando buenos resultados para diversas funciones de dificultad retadora y de diversas propiedades.

4.4.3 Resumen

En esta sección se muestra un resumen con todos los métodos anteriormente propuestos, los resultados de cada método propuesto están organizados por columnas, para todos los métodos se utilizaron las funciones de prueba con una dimensión de 55 variables, en las columnas del método de Gibbs de múltiple inicio la columna $2x55$ indica que el límite de evaluaciones de la función objetivo se ha incrementado a 1000000 de evaluaciones con la ayuda de la tabla 4.30 se puede observar y comparar fácilmente la calidad de los resultados de los diversos métodos propuestos.

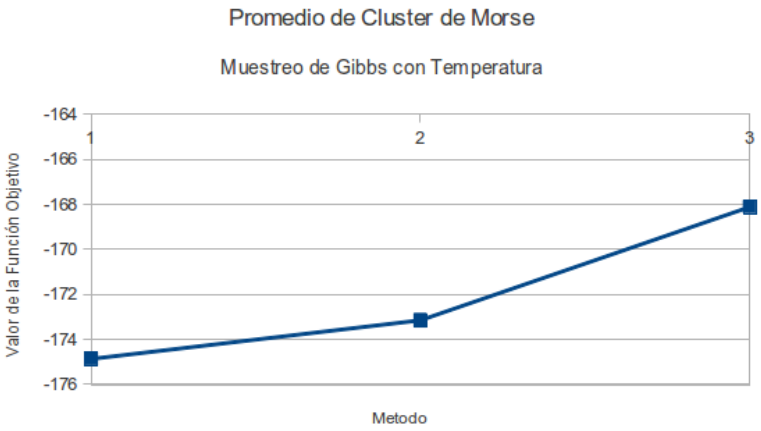


Figura 4.4: Promedios del muestreo de Gibbs

El método del muestreo de Gibbs con temperatura solo se aplico para los Clústeres de Morse debido a la dificultad retadora de esta función por lo tanto no hay resultados que mostrar en las funciones de Rosenbrock o Fractal, además en la columna donde se marca $+bm$ es el caso en donde se incrementó el número de iteraciones del muestreo de exploración de 100 a 160, PAGEDA con sus diversas variantes y el muestreo de Gibbs con múltiple inicio fueron aplicados a las 3 funciones.

En el caso de 2xRang3-9 el número de iteraciones permitidas fue elevado de 500000 a 1000000, se puede observar como para la función de Rosenbrock y la función Fractal PAGEDA con rango 6-8 ofreció mejores resultados, mientras que para los Clústeres de Morse el mejor resultado fue el obtenido con 2xRang3-9, sin embargo el muestreo de Gibbs con múltiple inicio con el doble de evaluaciones de la función objetivo permitidas ofreció un resultado de mejor calidad para la función de Rosenbrock y la función fractal esto es debido a las propiedades de cada función ya que este método ofrece una exploración más extensa de los puntos cercanos al punto candidato que se está explorando.

Sin embargo el método PAGEDA fue el que ofreció resultados de mejor calidad para la función de Clústeres de Morse, llegando al grado que incluso el menor de los resultados obtenidos es mayor que el mejor resultado de los otros métodos, en el caso de la función de Rosenbrock la diferencia de promedios no es muy grande como en el caso de la función Fractal. En base a la tabla 4.30 se puede concluir que el

Pageda			**Gibbs Multiple Inicio**		
Función de Rosenbrock					
Rango 6-8	Rango 3-9	2xRango 3-9	55	2x55	
44.03678	52.0327	48.4539	47.4678	47.3386	
50.24834	49.12865	49.61564	51.52415	47.07303	
51.2877	47.5919	50.21032	47.50555	46.73031	
50.90115	49.31232	45.4102	50.40901	48.42704	
51.08494	41.53986	45.71202	47.79892	46.4852	
49.511782	47.921086	47.880426	48.941086	47.210836	Promedio
Función Fractal					
3.672321	6.604872	7.752108	-2.919411	-3.960083	
6.150828	5.056075	3.042723	-1.727858	-4.130506	
5.431738	4.521162	3.297295	-4.059923	-5.288282	
2.124426	5.120328	7.200509	-2.243727	-4.611159	
6.712768	7.278917	7.889848	-4.318225	-4.906142	
4.8184162	5.7162708	5.8364966	-3.1438288	-4.5792344	Promedio
Clúster de Morse					
-166.7407	-174.6303	-188.0356	-174.5787	-182.8065	
-184.5758	-182.2968	-192.6488	-174.3176	-163.2549	
-187.1934	-175.7143	-200.0077	-185.8439	-168.4105	
-182.7935	-187.3457	-196.0462	-170.0027	-170.457	
-173.7309	-184.8398	-183.0216	-168.0673	-159.8805	
-179.00686	-180.96538	-191.95198	-174.56204	-168.96188	Promedio

Tabla 4.30: Comparación de Resultados para los métodos propuestos

método PAGEDA posee resultados equiparables al método del muestreo de Gibbs de múltiple inicio para la función de Rosenbrock, sin embargo muestra resultados de menor calidad para la función fractal, sin embargo para la función de clúster de Morse reporta mejores resultados.

Gibbs con temperatura			
Clúster de Morse			
Escala*1	Escala/3	Escala*0.01 + bm	
-156.2711	-179.6878	-166.7715	
-191.678	-185.9923	-154.4885	
-189.1338	-169.7731	-168.103	
-172.2712	-153.1796	-172.5624	
-164.9702	-177.1191	-178.6523	
-174.86486	-173.15038	-168.11554	Promedio

Tabla 4.31: Mejores Resultados del método del muestreo de Gibbs con temperatura

La experimentación anterior nos permite demostrar en que manera influyen los diferentes estilos de paralelización en el desempeño de una metaheurística, además de tomar una decisión sobre el método a desarrollar en base a los resultados obtenidos, en la tabla 4.31 se muestran los mejores resultados del método del muestreo de Gibbs con temperatura aplicado a la función clúster de Morse para poder compararlos con los métodos anteriormente propuestos.

4.5 Comparación entre AGEDA y PAGEDA

Esta experimentación se realizo en una Computadora en un sistema con las características mostradas en la imagen en la imagen 4.5.

El objetivo de este experimento es el evaluar el desempeño de PAGEDA en

Figura 4.5: Detalles del Sistema

comparación con la versión serial, se realizaron un total de 30 corridas, todas para el problema de Clústeres de Morse con dimensionalidad 55 ya que es el problema más retador de los antes mencionados, durante estas corridas se tomaron diversos aspectos en consideración: el tiempo total de cómputo del algoritmo, la calidad de la solución encontrada y el numero de evaluaciones a la función objetivo realizadas.

Para este experimento el método AGEDA utilizo un límite de 500 000 evaluaciones de la función objetivo, un valor de 0.7 para el promedio de aceptación mínimo requerido intensificar la búsqueda en dicha región, además de un total de 100 soluciones propuestas en el muestreo de Gibbs, mientras que en PAGEDA se utilizó una probabilidad de intercambio de solución del 35 % , un rango para los promedios de aceptación de 0.3 a 0.9 y un máximo de 500 000 evaluaciones, se puede observar que el numero de evaluaciones realizadas es mayor a este límite debido a que cada proceso lleva su conteo de evaluaciones independientemente, de igual manera se utilizo el mismo número de soluciones propuestas en el muestreo de Gibbs.

Se muestran en la tabla 4.32 los resultados se estas 15 corridas del algoritmo para ambas versiones.

AGEDA			**PAGEDA**		
Tiempo de Cómputo	Solución	Evaluaciones	Tiempo de Cómputo	Solución	Evaluaciones
8741.576	-196.3731	504138	1286.711	-186.6334	2485260
8665.623	-185.7014	504266	585.757	-187.4833	1758875
8850.27	-209.9768	504683	1220.932	-175.5632	1088320
8775.954	-202.9436	504094	1460.186	-193.3432	571934
8401.497	-197.2219	504445	1548.619	-181.3955	1462510
8416.093	-213.9735	504545	1322.194	-175.1754	1088595
8421.433	-210.0939	504760	849.878	-162.3055	714640
8419.789	-211.2763	504333	904.533	-181.3088	714640
8417.372	-199.8184	504379	1848.257	-183.766	1825585
8409.33	-181.505	504188	1592.636	-188.618	572053
8420.271	-212.073	504555	855.421	-180.8466	714555
8426.437	-186.0765	504379	609.7	-191.8149	1759075
8428.945	-189.6521	504397	2363.137	-187.7426	527000
8416.941	-209.1975	504576	1146.996	-187.0777	1088445
8428.443	-209.2421	504783	573.043	-184.121	1759150

Tabla 4.32: Comparación de Resultados

Obteniendo los valores promedio en cada columna se puede observas la tabla 4.33. En esta tabla se puede apreciar que se realizan muchas más evaluaciones con menos tiempo con la versión en paralelo, el tiempo de cómputo también es reducido considerablemente y la calidad de la solución solo disminuye en un 9 % en comparación al valor promedio obtenido por el método serial en este experimento.

En conclusión PAGEDA ofrece soluciones de buena calidad en mucho menos tiempo de cómputo, esto es una ventaja si se considera la escalabilidad del problema, es decir el número de variables, ya que a mayor cantidad de variables el tiempo de cómputo se verá reducido considerablemente.

AGEDA			PAGEDA		
Tiempo de Cómputo	Solución	Evaluaciones	Tiempo de Cómputo	Solución	Evaluaciones
8492.7427142857	-201.3394285714	504455.928571429	1205.8063571429	-182.8972642857	1117526.92857143

Tabla 4.33: Comparación de Promedios

Capítulo 5

Conclusiones

En esta tesis se abordaron diversos problemas de optimización no lineal en diversas dimensiones. Se utilizaron múltiples métodos y se determinaron los mejores parámetros para el mejor funcionamiento de PAGEDA.

Se utilizaron diversos métodos para la paralelización del muestreo de Gibbs, muestreo de Gibbs paralelo, muestreo de Gibbs múltiple inicio y el muestreo de Gibbs con temperatura, además de un método de paralelización temperado aplicado a todo AGEDA. Se probaron estas metodologías a diferentes problemas y se compararon los resultados logrando observar que diversos métodos fueron eficientes en diferentes tipos de problemas, esto es debido a que el PAGEDA con muestreo de Gibbs de múltiple inicio resulta ser más eficiente en la Función Fractal, sin embargo PAGEDA produce resultados equiparables en la función de Rosenbrock en comparación a los resultados obtenidos con el muestreo de Gibbs de múltiple inicio, además de que PAGEDA produce mejores resultados en la función de Clúster de Morse, esto es ya que el muestreo de Gibbs de múltiple inicio explora más puntos cercanos al punto inicial del muestreo y por la estructura del problema esto resulta ser más eficiente ya que tiene posee una superficie rugosa.

5.1 Trabajo Futuro

Entre las posibles vertientes para PAGEDA que se pueden desarrollar en un futuro se encuentra el generar y desarrollar un nuevo experimento en el que varíe el número de procesadores utilizados, esto para verificar si la paralelizacion del método es efectiva y que efecto tiene integrar un mayor número de procesadores a este proceso además de la reducción obvia de tiempo.

además de desarrollar un PAGEDA utilizando Clústeres computacionales, es decir una red de área local de computadoras interconectadas entre sí a modo de funcionar como una sola máquina, además de aplicar un balanceo de carga en el clúster, es decir, en caso de tener un clúster desequilibrado con computadoras más potentes que otras, tomar esto en consideración y a las computadoras de menor rendimiento asignarles una carga menor de trabajo, esto debido a que las computadoras de mayor rendimiento lograran terminar una carga de trabajo mayor en un tiempo parecido a las máquinas de menor rendimiento realizando una carga menor.

Otro posible camino para PAGEDA es utilizando la GRID, esto es, una red de computadoras de área amplia trabajando en conjunto como una sola computadora, esto permite trabajar con cientos o miles de equipos a lo largo de todo el mundo, logrando así un mayor alcance en la resolución de problemas de alta dimensión.

Apéndice A

Computo No Convencional

A.1 Formación de Redes en Base a Opiniones

Durante el Periodo de Maestría se realizó una estancia académica en Italia en colaboración con Laboratorio de Física y Sistemas Complejos del Departamento de Energía de la Università degli Studi di Firenze con el Dr. Franco Bagnoli, co-head of the Laboratory of Physics of Complex Systems (FiSiCo).

En la estancia se trabajo en otro tipo de cómputo, que se considera intrínsecamente paralelo como lo es la simulación mediante agentes, en el que agentes diferentes pueden interactuar unos con otros en un mismo instante de tiempo.

Se realizaron 2 proyectos utilizando la simulación con agentes con diferentes objetivos, la elaboración de una red basándose en la dinámica de opiniones y la dinámica de opiniones con múltiples criterios.

En el Primer proyecto se desarrolla un algoritmo capaz de crear una red utilizando un modelo de opiniones, para una mayor comprensión del modelo se definirán algunos conceptos:

Definición A.1 *Opinión: Es un escalar o vector cuyas componentes se encuentran todas en el intervalo (0,1)*

Definición A.2 *Afinidad: es el nivel de amistad entre dos agentes, también se encuentra en el intervalo (0,1)*

Definición A.3 *Distancia social: es la unidad de medida para la selección de pareja de conversación.*

Definición A.4 *Diferencia de opinión crítica: determina el máximo nivel de tolerancia tiene un agente para la opinión de los demás.*

Definición A.5 *Afinidad Crítica: determina el máximo nivel de tolerancia para producir un cambio en la opinión de un agente.*

Primeramente se inicializan los agentes utilizando números aleatorios de distribución uniforme en el intervalo (0,1) para las opiniones, estas son escalares para el proyecto uno y vectoriales en el proyecto dos.

En el primer proyecto, un agente es seleccionado aleatoriamente con probabilidad uniforme de entre todos los agentes, después se calcula la distancia social de este agente para con todos los agentes, a esta distancia se le añade una temperatura que es un número aleatorio de una distribución normal con una media de 0 y una desviación estándar de 0.07 esto para evitar que el modelo sea totalmente determinista.

Una vez ya obtenida la distancia social de el agente seleccionado con todos los demás agentes, este selecciona un numero k de parejas para conversar, este k es el grado del nodo en el grafo, de modo que se genera una lista de agentes y el agente seleccionado interactúa con los agentes en un orden secuencial y estos pueden o no cambiar su opinión en base a la afinidad y a la diferencia de opinión crítica, es decir si la diferencia de opinión excede a la diferencia de opinión critica el agente rechaza al agente con el que esta interactuando y su afinidad no cambia.

Sin embargo, si la diferencia de opinión es menor que la diferencia de opinión crítica su afinidad aumenta esto equivale a tener una conversación agradable con algún conocido, por lo que la amistad incrementa.

Un incremento en la afinidad provoca que un agente sea más fácilmente influenciado por otro agente, esta afinidad no es reciproca (de igual manera en la vida real), es decir, un agente puede considerar amigo a otro pero este último no.

La afinidad critica nos dice que tan amigo debe ser un agente de otro para tener una influencia en la opinión de este, ya que cuando la afinidad es mayor que la afinidad critica la opinión de el agente evoluciona sin embargo si la afinidad del segundo agente no es mayor a la afinidad critica su opinión no cambia emulando así la vida real.

Este proceso se repite para todos los agentes y los enlaces o conversaciones que mantienen unos con otros permanecen a lo largo de toda la iteración, es decir si un agente i forma un enlace con el agente j este agente j cuando sea su turno de seleccionar parejas para conversar iniciara con $k_j - 1$ agentes en su lista sin embargo la dinámica de opiniones no se aplicara en ese caso ya que solo se está manteniendo la conversación que ya fue iniciada en el turno del nodo i con esto concluye una iteración y al final de cada iteración se evalúa la probabilidad de que el enlace permanezca para la siguiente iteración o sea eliminado.

Al inicio de una nueva iteración si un agente ya tiene un determinado número de agentes en su lista de conversaciones este solo elegirá la cantidad disponible para formar sus enlaces restantes.

El resultado en la evolución de opiniones puede observarse en la Figura A.1 donde α_c es la afinidad critica, se puede apreciar como después de 50 iteraciones los agentes se dividen en 2 grupos y esto se ve reflejado en el grafo de la Figura A.2.

La dinámica de opiniones está conformada por la evolución de la afinidad y un parámetro μ, este parámetro se interpreta como la magnitud del cambio en la opinión y es un parámetro en el rango (0,1) esto quiere decir que solo las opiniones cambiaran en un máximo del tamaño de su diferencia de opinión, dicho parámetro se ha fijado en 0.5 [4] de modo que si ambos agentes se acercan en una magnitud equivalente a la mitad de la diferencia de opinión, ambos llegan a tener la misma opinión después del encuentro , sin embargo pueden darse otros casos.

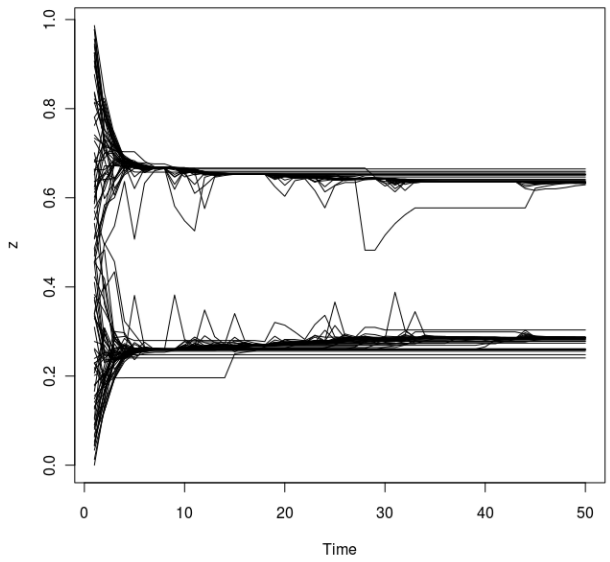


Figura A.1: Dinámica de Opiniones con 100 agentes, DO_c=0.5 y α_c=0.5

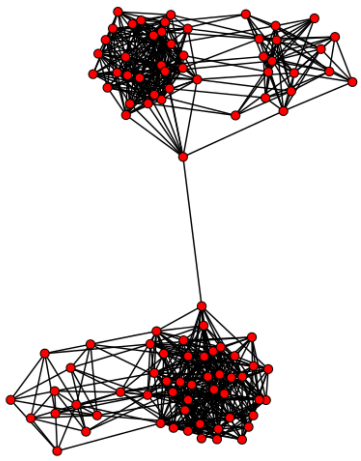


Figura A.2: Red obtenida con 100 agentes, DOc=0.5 y αc=0.5

La ecuación (A.1) muestra la forma en que las opiniones evolucionan con cada encuentro.

$$O_i^{t+1} = O_i^t - \mu \Delta O_{ij}^t \frac{1}{2}\{tanh[\beta(\alpha_{ij}^t - \alpha_c)] + 1\} \tag{A.1}$$

En donde μ es el parámetro que indica la proporción en el cambio de opiniones, ΔO_{ij} es la diferencia de opinión entre los agentes i y j, α_{ij}^t es la afinidad del agente i con el agente j la cual no es reciproca por lo que el resultado de la tangente puede ser: 1 si la afinidad es mayor a la afinidad critica (α_c) y 0 si la afinidad es menor ya que la función tangencial funciona como un salto gracias al parámetro β que es un escalar muy grande. De esta forma la opinión cambia para el agente i y después se aplica la misma ecuación para el agente j.

La evolución de la afinidad esta de igual manera basada en la diferencia de opiniones, se puede observar en la ecuación (A.2) el parámetro ΔO_c sirve para marcar la máxima diferencia de opinión que puede producir un cambio positivo en la afinidad del mismo modo que en la ecuación anterior el parámetro β convierte a la función tangente en una función de salto, es decir solo puede tomar valores de -1 y 1, de modo que la magnitud del cambio en la afinidad está dada por $(\alpha_{ij}^t[1 - \alpha_{ij}^t])$.

$$\alpha_i^{t+1} = \alpha_{ij}^t + \alpha_{ij}^t[1 - \alpha_{ij}^t](-\{tanh[\beta(|\Delta O_{ij}^t| - \Delta O_c)] + 1\}) \tag{A.2}$$

El algoritmo de este método se encuentra en el Pseudocódigo 9

Pseudocódigo 9 Formación de Redes en Base a Opiniones

```
Se declaran los parámetros.
para T = 1 to max_i t hacer
    Generar un vector aleatoriamente con el orden de selección de los agentes.
    mientras i < numag hacer
        Incrementar el valor de i.
        Seleccionar el agente i de la lista.
        Declarar el vector de parejas (chat) del tamaño igual al grado del agente
        numero i.
        para h = 1 to numag hacer
            Calcular la Distancia social del agente h al agente numero i.
        fin para
        Agregar Temperatura a todas las distancias sociales.
        Agregar al Vector chat los agentes con los que ya este hablando.
        Agregar al Vector chat los agentes con menor distancia social hasta llenarlo.
        para cada elemento del vector chat hacer
            Actualizar Matriz de Adyacencias.
            Elegir tema de conversación.
            Actualizar Afinidades.
            Calcular la probabilidad de que el enlace persista.
            si El enlace es rechazado entonces
                Actualizar Matriz de Adyacencias.
            fin si
            Actualizar Opiniones.
        fin para
        si Esta no es la última iteración entonces
            Calcular probabilidad de que el enlace permanezca a la siguiente iteración.
            si El enlace es eliminado entonces
                Actualizar Matriz de Adyacencias.
            fin si
        fin si
    fin mientras
fin para
```

A.2 Dinámica de opiniones con múltiples criterios

Este proyecto funciona muy parecido al anterior pero la diferencia principal es que ahora cada agente puede tener más de una opinión acerca de un tema, es decir, ahora un agente esta compuesto por un vector de opinión y no un escalar y las opiniones pueden ser variadas un tema puede ser agradable para un agente y para otro no serlo, lo que lleva a realizar diferentes cambios, la dinámica de los encuentros también cambio esta vez un agente selecciona uno y sólo un agente.

En este proyecto no se forman enlaces solo son eventos ,es decir, un agente interactúa con otro y al momento de seleccionar un tercer agente este puede platicar con cualquiera de los otros dos anteriores de modo que las opiniones evolucionan más lentamente ya que se produce un total de encuentros equivalente al número de agentes por iteración.

Para esta dinámica se utilizaron diferentes variantes como una nueva función del incremento en la afinidad, ya que con la función anterior al momento de la afinidad disminuir a cero o incrementar a uno se fija y ya no puede ser modificada, sin embargo con una función nueva de afinidad que permite disminuir cuando ya se ha alcanzado un valor de uno, es el equivalente a tener una conversación desagradable con un muy buen amigo lo que ocasiona que la amistad se vea afectada.

Esto es debido a que las opiniones son diversas y el tema de conversación es aleatorio y pueden escoger un tema de conversación en el que la diferencia de opiniones de estos agentes sea mayor a la diferencia de opinión crítica.

La nueva función para la dinámica de opiniones está en la ecuación (A.3) se utilizo una función sinodal modificada de modo que se comporte de una manera parecida a la función anterior, pero ahora permite movimiento en los límites del intervalo. La forma de asegurar que los valores de afinidad no salgan del intervalo (0,1) se encuentra en el código de la implementación.

$$\alpha_i^{t+1} = \alpha_{ij}^t + 0.25 * (sin(3 * (\alpha_{ij}^t + 0.02)))(-\{tanh[\beta(|\Delta O_{ij}^t| - \Delta O_c)] + 1\}) \quad \text{(A.3)}$$

Así mismo también se utilizo otra variante en la dinámica de opiniones, la opinión circular, esto se ilustra en la imagen A.3 de esta manera los agentes con opiniones cercanas a cero pueden interactuar con agentes con opiniones cercanas a uno.

Otro aspecto que fue sujeto de investigación es asignar cierta preferencia a cada agente por algún tema en particular, estas preferencias están en el intervalo (0,1) y la suma de todas es igual a uno ya que funcionan como distribución de probabilidad para escoger un tema particular, en vez de escoger aleatoriamente como en los casos anteriores.

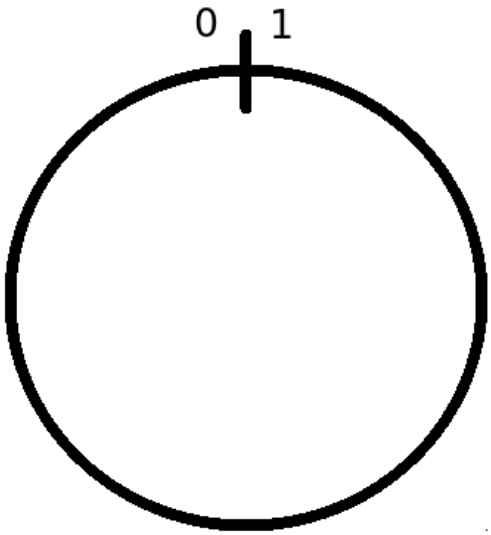


Figura A.3: Dinámica de Opiniones Circulares

El Algoritmo de este método se encuentra en el Pseudocódigo 10.

Una muestra del algoritmo funcionando con opiniones circulares está en la imagen A.4 este es el primer tema de conversación de un total de 2 temas, la evolución de opiniones en el segundo tema de conversación se encuentra en la imagen A.5, ambas imágenes fueron utilizando un vector de preferencias para cada opinión por lo que se aprecia convergen las opiniones en tiempos diferentes y se está utilizando la función de afinidad mostrada anteriormente, ya que permite que las afinidades se muevan en los límites del intervalo.

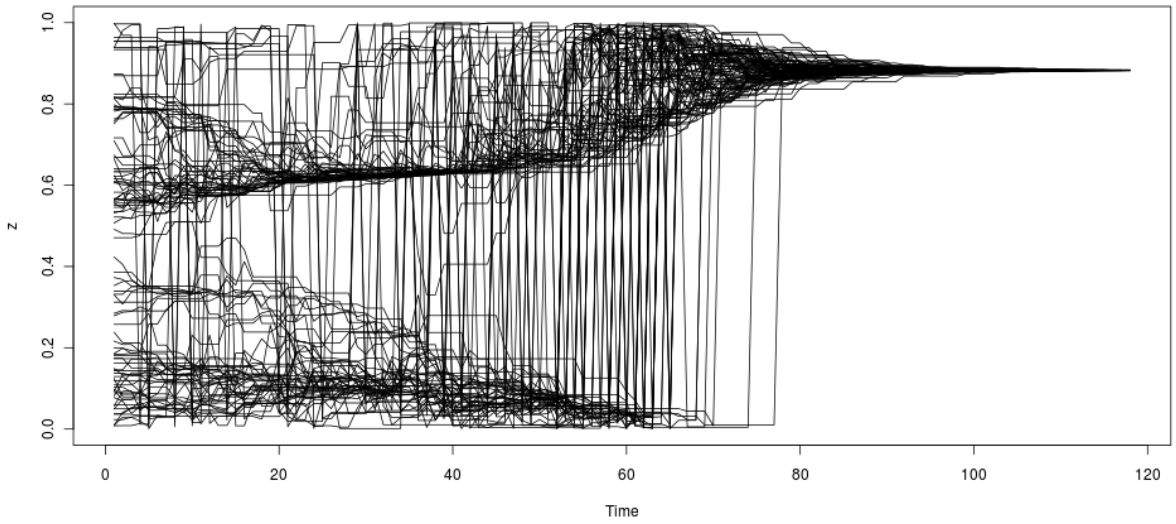


Figura A.4: Dinámica de Opiniones Circulares (Tema1)

Pseudocódigo 10 Formación de Redes en Base a Opiniones

```
Se declaran los parámetros.
para T = 1 to max_i t hacer
    Generar un vector aleatoriamente con el orden de selección de los agentes.
    mientras i < numag hacer
        Incrementar el valor de i.
        Seleccionar el agente i de la lista.
        Elegir tema de conversación basado en preferencias del agente numero i.
        para h = 1 to numag hacer
            Calcular la Distancia social del agente h al agente numero i.
        fin para
        Agregar Temperatura a todas las distancias sociales.
        Elegir el agente para conversar.
        Actualizar Matriz de Adyacencias.
        Actualizar Afinidades.
        Actualizar Opiniones.
        si Esta no es la última iteración entonces
            Calcular probabilidad de que el enlace permanezca a la siguiente iteración.
            si El enlace es eliminado entonces
                Actualizar Matriz de Adyacencias.
            fin si
        fin si
    fin mientras
fin para
```

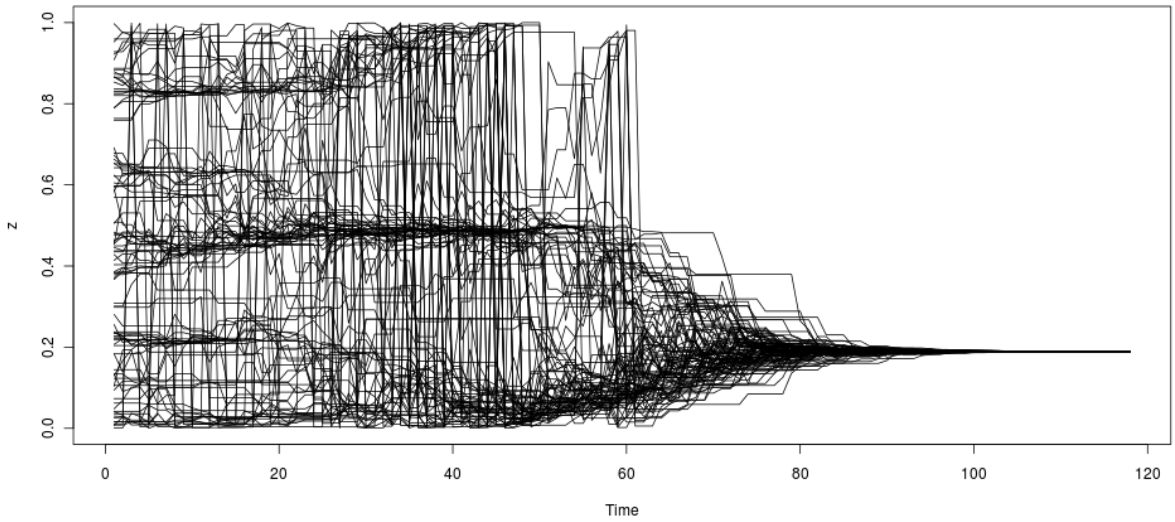


Figura A.5: Dinámica de Opiniones Circulares (Tema2)

Bibliografía

[1] A., B., «Bayesian Inference Based on Stationary Fokker-Planck Sampling», *Neural Computation*, **22**(6), págs. 1573–1596, 2010.

[2] Andrey V. Solov'yov, W. G., Jean-Patrick Connerade, «ATOMIC CLUSTER SCIENCE: INTRODUCTORY NOTES», *World Scientific*, 2003.

[3] Back, T., *Evolutionary algorithms in theory and practice: evolution strategies,evolutionary programming, genetic algorithms*, Oxford University Press Oxford, UK, UK, Oxford, 1996.

[4] Bagnoli, F., T. Carletti, D. Fanelli, A. Guarino y A. Guazzini, «Dynamical affinity in opinion dynamics modeling», *Phys. Rev. E*, **76**, pág. 4, 2007.

[5] Canty, A., «Hypothesis tests of convergence in markov chain monte carlo», *Journak of Computational and Graphical Statistics*, **8**, págs. 93–108, 1999.

[6] Coello Coello, C. A., E. Alba, G. Luque y A. H. Aguirre, «Comparing Different Serial and Parallel Heuristics to Design Combinational Logic Circuits», *IEEE Computer Society*, págs. 3–, 2003, URL `http://dl.acm.org/citation.cfm?id=938176.938197`.

[7] David J. Earl, M. W. D., «Parallel Tempering: Theory, Applications, and New Perspectives», *physics/0508111*, pág. 21, 2005.

[8] de Pablos Heredero, C., *ï»¿InformÃ¡tica y Comunicaciones en la Empresa*, ESIC Editorial, 2004.

[9] DEB, A., K. ; ANAND y D. JOSHI, «A computationally efficient evolutionary algorithm for real-parameter optimization», *Evolutionary Computation*, **10**(4), págs. 371–395, 2001.

[10] ET AL., E.-G. T., «Parallel ant colonies for the quadratic assignment problem», *Future Generation Computer Systems - Special issue on bio-impaired solutions to parallel processing*, **17**, págs. 441–449, 2001.

[11] HANSEN, N. y A. OSTERMEIER, «A computationally efficient evolutionary algorithm for real-parameter optimization», *Evolutionary Computation*, **9**(2), págs. 159–195, 2002.

[12] IBRAHIM H. OSMAN (EDITOR), J. P. K. E., *ï»¿Meta-heuristics: theory and applications*, Springer, http://www.springer.com, 1996.

[13] J., A., *Bayesian Computation with R.*, Springer, http://www.springer.com, 2006.

[14] J., P. D., *Global Optimization: Scientific and Engineering Case Studies*, Springer, http://www.springer.com, 2006.

[15] J. A., N. y M. R., «A Simplex Method for Function Minimization», *The Computer Journal*, **7**, págs. 308–313, 1965.

[16] JONES, M. T. y P. E. PLASSMANN, «A Parallel Graph Coloring Heuristic», *SIAM J. Sci. Comput*, **14**, págs. 654–669, 1992.

[17] KECKLER, S. W., *ï»¿Multicore Processors and Systems*, Springer, 2009.

[18] KENNEDY, J. y R. EBERHART, «Flocks, herds and schools: Adistributed behavioral model», *Neural Networks, 1995. Proceedings., IEEE Internacional Conference on*, **4**, págs. 1942–1948, 1995.

[19] MINOLI, D., *ï»¿A Networking Approach to Grid Computing*, Wiley-Interscience, 2004.

[20] MOLGA, M. y C. SMUTNICKI, «Test functions for optimization needs», *2007330 httpwww zsd ict pwr wroc plfilesdocsfunctions pdf*, **http://www.bioinformaticslaboratory.nl/twikidata/pub/Education/NBICResearchSchool/Optimization/VanKampen/BackgroundInformation/TestFunctions-Optimization.pdf**(c), págs. 1–43, 2005.

[21] PARSOPOULOS, K. E. y M. N. VRAHATIS, «Recent approaches to global optimization problems through Particle Swarm Optimization», *Natural Computing*, **1**, págs. 235–306, 2002.

[22] REYNOLDS, C. W., «Flocks, herds and schools: Adistributed behavioral model», *ACM SIGGRAPH Computer Graphics*, **21**(4), págs. 25–34, 1987.

[23] SWENDSEN, R. H. y J.-S. WANG, «Replica Monte Carlo Simulation of Spin-Glasses», *Phys. Rev. Lett.*, **57**, págs. 2607–2609, 1986.

[24] VELASCO, J. y A. BERRONES, «An estimation of distribution algorithm with adaptive Gibbs sampling for unconstrained global optimization», *Bajo Revision*, **0**, págs. 00–00, 2010.

[25] YUN-WEI, S. y Q. YU-HUANG, «A Note on the Extended Rosenbrock Function», *Evolutionary Computation*, **14**(1), págs. 119–126, 2006.

Ficha autobiográfica

José Adrián Rodríguez Aldape

Candidato para el grado de Mastería en Ciencias
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Enfoque Paralelo Multiplataforma en Heuristicas para Problemas no Lineales de Alta Dimensión

Nací en la ciudad de Monterrey, Nuevo León el día 31 de Enero de 1987. En Julio 2009 Me Gradue de la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León con el Título de Ingeniero Administrador de Sistemas y comence mis estudios de posgrado en Enero 2010 en la misma Facultad ahora cursando la Maestria en Ciencias en Ingenieria de Sistemas, periodo durante el cual realize una Estancia Academica en la UNIVERSITA DEGLI STUDI DI FIRENZE en Florencia, Italia bajo la supervision del Dr. Franco Bagnoli.