Universidad Michoacana De San Nicoás De Hidalgo

Facultad de Ciencias Físico Matemáticas

Búsqueda de parámetros de ecuaciones diferenciales usando algoritmos genéticos en paralelo

TESIS

QUE PARA OBTENER EL GRADO DE:

Licenciado En Ciencias Físico Matemáticas

PRESENTA:

Luis Raúl Torres Rojas

DIRECTOR DE TESIS:
Dr. José Antonio González Cervera

MORELIA,MICH. Agosto de 2018

Comité de evaluación

Dr. Fernando Iguazú Ramírez Zavaleta, Presidente
Dr. Eduardo Salvador Tututi Hernández, Sinodal titular
Dr. José Antonio González Cervera, Sinodal titular
Dr. Héctor Igor Pérez Aguilar, Suplente 1
Dr. Francisco Javier Domínguez Mota, Suplente 2

Índice general

Índice de figuras

Lista de algoritmos

Agradecimientos

Agradezco al Dr. González por su gran paciencia, dedicación y esfuerzo que tuvo conmigo en la elaboración de esta tesis de licenciatura.

Agraezco a los Drs. Fernando Iguazú Ramírez Zavaleta, Eduardo Salvador Tututi Hernández, Héctor Igor Pérez Aguilar y Francisco Javier Domínguez Mota en la culminación de esta tesis.

A mis padres, por haberme permitido estudiar la licenciatura que deseaba y por su gran apoyo durante mi formación, estaré eternamente agradecido con ustedes.

Finalmente, agradezco a todos mis compañeros estudiantes de Fismat que voluntariamente o involuntariamente, contribuyeron a la consolidación de esta tesis. A todo el personal académico y administrativo de la facultad de ciencias físico matemáticas. A la Universidad de San Nicolás de Hidalgo por presentarme todos los servicios para mi formación.

Resumen

En el estudio de los fenómenos naturales que pueden ser representados por medio de ecuaciones diferenciales y que los presenciamos en nuestra vida cotidiana, tales como las ondas sonoras o el movimiento de una partícula producido por la acción de una fuerza restauradora, usualmente es necesario estudiarlos desde una perspectiva computacional debido a su complejidad física, o a la dificultad de analizar toda la información que los representa. Dicho planteamiento nos lleva a la implementación de los métodos numéricos con el fin de obtener aproximaciones numéricas que proporcionen las propiedades relevantes que contiene dicho fenómeno y obtener una descripción física del mismo.

En este trabajo de tesis de licenciatura se propone el uso de los métodos numéricos como una herramienta de búsqueda de parámetros de algunas ecuaciones diferenciales, como el oscilador armónico forzado y la ecuación de onda y, de otros problemas basados en la búsqueda de un óptimo como el problema de Thomson. Se implementó un método heurístico llamado algoritmo genético que esta inspirado en el concepto de recombinación genética, la teoría de evolución de Darwin y la supervivencia del más apto. Con este método, se construyeron códigos que proporcionen un conjunto de soluciones de posibles parámetros para dichos problemas.

Los algoritmos genéticos tienen una gran variedad de parámetros y operadores que pueden afectar la eficiencia de los resultados, dichas variaciones se aplicaron al problema de Thomson para encontrar los que son más estables y eficientes. Así, se implementó un código con el algoritmo genético para la búsqueda de parámetros de las ecuaciones diferenciales ya mencionadas y en paralelo por medio de una interfaz de paso de mensajes, llamada MPI.

La investigación consistió en encontrar los mejores parámetros y operadores que presenta el algoritmo genético, aplicar las dos diferentes formas de paralelizar dicho algoritmo; usando una sola población, maestro y esclavo, o

un conjunto de poblaciones, método de islas. La siguiente etapa consistió en aplicar todo lo anterior a la búsqueda de parámetros, constantes relacionadas con la física del problema, de ecuaciones diferenciales, como las condiciones iniciales, condiciones de frontera y otros parámetros como la velocidad, constante de fricción, entre otros.

Con los resultados presentados se expone el potencial que proporcionan los algoritmos genéticos en la búsqueda de parámetros de ecuaciones diferenciales y la gran ventaja que tienen de ser programados en paralelo para obtener un conjunto de soluciones en un tiempo de computo más eficiente que otros métodos convencionales, como el método del gradiente o búsqueda aleatoria.

Palabras clave: *Algoritmo genético*, *métodos de Runge-Kutta*, *método de líneas*, *MPI*, *problema de Thomson*, *ecuación de oscilador forzado*, *ecuación de onda*.

Abstract

In the study of natural phenomena that can be represented by means of differential equations and that we witness them in our daily life, such as sound waves or the movement of a particle produced by the action of a restoring force, it is usually necessary to study them from a computational perspective due to their physical complexity, or the difficulty of analyzing all the information that represents them. This approach lead us to the implementation of numerical methods in order to obtain numercial approximations that provide the relevant properties contained in this pheomenon and obtain a physical description of it.

In this thesis work, we proporse the use of numerical methods as a search tool for parameters of some differential equations, such as the forced harmonic oscillator and the wave equation, and other problems based on the search for an optimal like Thomson's problem. We implemented a heuristic method called genetic algorithm that is inspired by the concept of genetic recombination, Darwin's theory of evolution and the survival of the fittest. With this method, codes were constructed that provide a set of possible parameter solutions for these problems.

Genetic algorithms have a wide variety of parameters and operators that can affect the effiency of the results, these variations were applied to the Thomson problem to find those that are more stable and efficient. Thus, a code was implemented with the genetic algorithm for the search of parameters of the aforementioned differential equations and in parallel by means of a message passing interface, called MPI.

The research consisted of finding the best parameters and operators presented by the genetic algorithm, applying the two different ways of parallelizing said algorithm; using a single population, master and slabe, or a ser of populations, islands method. The next stage consisted of applying all of the above to the search for parameters, constants related to the physics of the problem,

diferential equations, such as initial conditions, boundary conditions and other parameters such as speed, friction constant, among others.

With the presented results, the potential provided by the genetic algorithms in the search for parameters of differential equations and the great advantage they have of being prorammed in parallel to obtain a set of solutions in a more efficient computing time than other conventional methods is exposed, as the gradient or random search method.

Keywords: *genetic algorithm*, *Runge-Kutta methods*, *line method*, *MPI*, *Thomson's problem*, *forced oscillator equation*, *wave equation*.

Capítulo 1

Introducción

1.1. Motivación

La propagación de ondas puede encontrarse fácilmente en nuestra vida cotidiana, como las ondas sonoras producidas por nuestras cuerdas vocales para que puedan llegar a la oreja de nuestro amigo y realizar una comunicación entre emisor y receptor, la radiación electromagnética que puede ser percibida por nuestro ojo para tener una visión de nuestros alrededores, y las ondas producidas por el agua que vertimos en nuestra taza de café. Estos fenómenos pueden ser descritos por medio de la ecuación de onda que es una ecuación diferencial en derivadas parciales lineal de segundo orden. Así, como la propagación de onda, existen otros fenómenos físicos que pueden ser también descritos por medio de ecuaciones diferenciales, uno de ellos el movimiento que realizan los amortiguadores de un automóvil o para encontrar la diferencia de potencial de un circuito LC con pérdidas con oscilaciones forzadas, estos problemas pueden ser determinados al resolver la ecuación diferencial de un oscilador armónico amortiguado forzado. Por lo tanto, es importante conocer los parámetros y compararlos con las observaciones que conforman dichas ecuaciones para entender estos fenómenos.
Esto impulsa a buscar un método numérico estable y eficiente para encontrar los parámetros que satisfacen la solución de estos fenómenos. Aquí es donde entra en acción el algoritmo genético, ya que se destaca por ser una herramienta eficiente para proporcionar un conjunto de soluciones del problema a resolver.
Los algoritmos genéticos son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización inspirados en la evolución biológica y su base genética-molecular. Los principios básicos de los algoritmos genéticos fueron establecidos por Holland [1] a finales de los $60's$. Este

método se basa en los mecanismos de selección que utiliza la naturaleza, donde los individuos más aptos de una población sobreviven, para adaptarse a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos más deseables (es decir, los que permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente. Así, la nueva generación representa una población que puede adaptarse más fácilmente a su entorno a comparación de sus respectivos padres. En los algoritmos genéticos los mecanismos de selección son conocidos como operadores genéticos, estos son: el operador de selección, el operador de recombinación y el operador de mutación. Estos operadores dependen de parámetros que están relacionados con la evolución de la población durante cada generación, por ejemplo: la probabilidad que exista una recombinación de genes entre dos padres seleccionados, la probabilidad de que un hijo este sometido a una mutación de genes, el tamaño de la población, la cantidad de veces que se va a evolucionar una población y la cantidad de sobrevivientes de una generación a otra.
Existen una gran variedad de operadores genéticos y parámetros, lo que promueve a que se realice una búsqueda de los mejores operadores y parámetros de los algoritmos genéticos, esto se realiza al implementar los algoritmos genéticos al problema de Thomson [11] y, que nos funciona para calibrar nuestro algoritmo genético y conseguir la mejor configuración de operadores genéticos y parámetros.
Para enfrentar problemas más complejos como la dependencia del tiempo y la solución de problemas múltiples, se realizaron extensiones al algoritmo genético estándar realizado por Holland [3], tales como los algoritmos genéticos en paralelo. Así, la meta de estas nuevas extensiones es traer diversidad a la población y así incrementar la eficiencia de estos algoritmos.
De acuerdo al trabajo realizado por Cantú-Paz [6] existen dos maneras de implementar los algoritmos genéticos de forma paralela: usando una sola población y usando múltiples poblaciones. La primera consiste en que una computadora (el maestro) ejecuta las operaciones del algoritmo genético y distribuye individuos para que sean evaluadas por otras computadoras (los esclavos). En el segundo, el más popular, conocido también como el método de islas describe poblaciones naturales que son relativamente aisladas entre sí, como las islas, los individuos pueden migrar de vez en cuando a cualquier otra isla.
Este último modelo es muy llamativo e interesante debido a que se asemeja a un ecosistema más elaborado, es decir, más real, en donde hay comunicación relativa entre poblaciones de individuos. Pero, sufre de algunas dificultades debido a que tiene una gran cantidad de parámetros, estos son: determinar

el tamaño de individuos en una isla, la topología de intercambio entre islas, la probabilidad de migración que controla cuantos individuos pueden migrar y el criterio que determina que individuos migran y cuales son reemplazados.

1.2. Introducción a los algoritmos genéticos

Una definición bastante completa de un algoritmo genético es la propuesta por John Koza [7]:

> "Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse representado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud".

Como se ha mencionado antes, los algoritmos genéticos son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. Las poblaciones evolucionan en la naturaleza de acuerdo con los postulados de Darwin de la selección natural y la supervivencia del más apto [8] [1]. Los algoritmos genéticos imitan este proceso y son capaces de crear soluciones para problemas del mundo real. La evolución de dichas soluciones depende completamente de una medida adecuada de codificación del problema a tratar y una elección de los operadores y parámetros.
En la naturaleza los individuos de una población compiten entre sí para buscar los recursos necesarios que les aseguren su supervivencia en un ecosistema dado, incluso los miembros de una especie compiten en la búsqueda de un compañero. Aquellos individuos que tienen más éxito en sobrevivir y en atraer compañeros tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario, los individuos poco dotados producirán un menor número de descendientes. Esto significa que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos de manera creciente. La recombinación de buenas características

[1] Cabe destacar que tanto la supervivencia del más apto como la selección natural postulada por Darwin, se refiere únicamente a los animales.

puede llegar a producir "super-individuos", cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De este modo en cada generación se obtienen características cada vez mejor adaptadas a sus alrededores.
Los algoritmos genéticos tienen una analogía con este comportamiento natural. Se compone de una población de individuos, donde cada uno representa una solución factible al problema a tratar. A cada individuo se le asigna un valor que esta relacionado con la eficiencia de dicha solución. Esto equivaldría en la naturaleza al grado de efectividad de un organismo para competir en la obtención de los recursos necesarios para sobrevivir. Así, cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que sea seleccionado para reproducirse, cruzando su información genética con otro individuo que fue seleccionado de la misma manera y, sus descendientes contiene características de sus padres. Cuanto menor sea la adaptación de un individuo, este tendrá menos probabilidad de que sea seleccionado para reproducirse, y por lo tanto, su información genética no se propagará en las siguientes generaciones.
De este modo, se produce una nueva población de posibles soluciones y dicha población se reemplaza con la anterior y contiene un interesante rasgo de contener un conjunto de individuos con posibles mejores características en comparación de la población anterior. Así, para las siguientes generaciones, los individuos de la población tendrán mejores características en comparación de todos sus ancestros, en caso de que la recobinación haya sido progresiva. Sin embargo, no se garantiza que el algoritmo genético encuentre la solución óptima del problema, de hecho aún no existe una prueba de la convergencia de algoritmos genéticos, pero hay investigaciones y argumentos para convencer al lector de que los algoritmos genéticos son esencialmente destinados al éxito [9].
La eficiencia es un factor importante en la resolución de problemas. Cuando hablamos sobre algoritmos de computadora, la eficiencia usualmente recae en dos ejes principales: *tiempo* y *memoria* que se necesita para resolver un problema. En el contexto de la genética y cualquier otro algoritmo evolutivo, hay otro eje importante: *la calidad de la solución*. Este aspecto aparece debido a que muchos problemas que los algoritmos genéticos y algoritmos evolutivos tratan de resolver no pueden ser resueltos con un porcentaje de 100 % de confianza al menos que una enumeración completa del espacio de búsqueda, todo el espacio de las soluciones posibles, se realice. Por lo tanto, los algoritmos genéticos, como otros métodos, usan una búsqueda para tratar de dar una buena solución aproximada, bajo un criterio de confianza dado, sin realizar una completa exploración del espacio de búsqueda.
La idea de paralización aparece naturalmente como un método para mejorar la eficiencia de la tarea de resolver problemas. Usando múltiples computado-

ras en paralelo, existe una oportunidad de obtener mejores soluciones en periodos de tiempo más cortos.
Muchos algoritmos son difíciles de paralelizar, pero ese no es el caso con los algoritmos genéticos, porque los algoritmos genéticos trabajan con una población de soluciones que pueden ser evaluadas independientemente de cada uno. Sin embargo, en muchos problemas el tiempo se gasta en evaluar las soluciones en vez de los mecanismos internos de los operadores de los algoritmos genéticos. Como lo explicamos en la motivación, el trabajo realizado por Cantú-Paz [6] indica que existen dos maneras de implementar los algoritmos genéticos: usando una sola población ó múltiples poblaciones. En los códigos usamos una interfaz de pase de mensajes llamado MPI, modelando ambos métodos. Y los resultados que obtuvimos se encuentran en las siguientes secciones.

1.3. Ecuaciones diferenciales

Para poder describir el comportamiento de un fenómeno físico, ya sea la propagación de una onda, la propagación del calor, dinámica de moléculas, entre otras, usualmente recurrimos a la solución de alguna ecuación diferencial que las describa. Para comprender como puede aplicarse los algoritmos genéticos a una búsqueda de parámetros de alguna ecuación diferencial, como la ecuación de onda, necesitamos algunas definiciones clave sobre las ecuaciones diferenciales.

Definición 1.3.1 (Ecuación diferencial) *Una ecuación que contenga las derivadas de una o mas variables dependientes, con respecto a una o mas variables independientes, se dice que es una* ***ecuación diferencial****. [10]*

Estas ecuaciones diferenciales se pueden clasificar por tipo, orden y linealidad.
Si una ecuación contiene solo derivadas ordinarias de una o más variables dependientes con respecto a una sola variable independiente se dice que es una **ecuación diferencial ordinaria** (EDO). Por ejemplo, se denominará y a la distancia entre la posición de equilibrio y la masa, m. Se supondrá que la fuerza del resorte esta descrita por la Ley de Hooke.

$$F = -ky \tag{1.1}$$

donde k es la constante elástica de resorte. Y, la segunda ley de Newton nos dice:

$$F = ma = m\frac{dv}{dt} = m\frac{d^2y}{dt^2} \tag{1.2}$$

remplazando la ley de Hooke 1.1 la fuerza 1.2 obtenemos:

$$m = \frac{d^2y}{dt^2} = -ky \tag{1.3}$$

La ecuación 1.3 es una ecuación diferencial que describe el movimiento de una masa m unida al extremo de un muelle elástico, es el oscilador armónico simple.
Ahora una ecuación que involucra derivadas parciales de una o más variables dependientes de dos o más variables independientes se llama una **ecuación diferencial parcial**. Por ejemplo, una onda es una perturbación que se propaga a través de un determinado medio o en el vació, con transporte de energía pero sin transporte de materia descrita su movimiento a través de la ecuación de onda. En su forma más elemental, la ecuación de onda hace referencia a una función $u(x,t)$ que satisface

$$\frac{\partial^2 u}{\partial t^2} = v^2\nabla^2 u \tag{1.4}$$

donde $\Delta \equiv \nabla^2$ es el laplaciano [10] y donde v es una constante equivalente a la velocidad de propagación de la onda. Esta ecuación 1.4 es una ecuación diferencial en derivadas parciales.
El **orden** de una ecuación diferencial, ya sea ordinaria o parcial, es el orden de la derivada más alta. Por ejemplo, la ecuación onda 1.4 es una ecuación diferencial parcial de segundo orden y la del oscilador armónico simple 1.3 es también de dos.
Podemos expresar una ecuación diferencial ordinaria de $n-$ésimo orden en una variable dependiente por la forma general

$$F(x, y, y', \cdots, y^{(n)}) = 0 \tag{1.5}$$

donde F es una función real evaluada de $n+2$ variables $x, y, y', \cdots$. Una ecuación diferencial ordinaria 1.5 se dice que es **lineal** si F es lineal en $y, y', \cdots, y^{(n)}$. Esto significa que la ecuación diferencial ordinaria 1.5 de $n-$ésimo orden es lineal cuando

$$a_n(x)y^{(n)} + a_{n-1}(x)y^{(n-1)} + \cdots + a_1(x)y' + a_0(x)y - g(x) = 0 \tag{1.6}$$

Cuando resolvemos una ecuación diferencial ordinaria de primer orden $F(x, y, y') = 0$, usualmente obtenemos una solución que contiene una sola constante arbitraria o parámetro c. Una solución que contiene una constante arbitraria se representa un conjunto como $G(x, y, c) = 0$ de soluciones llamadas de **familia de soluciones de un parámetro**. Cuando resolvemos una ecuación diferencial de $n-$ésimo orden $F(x, y, \cdots, t^{(n)}) = 0$, buscamos **una familia de soluciones de n-parámetros** $G(x, y, c_1, c_2, c \cdots, c_n) = 0$. Esto significa que una sola ecuación diferencial puede poseer un número infinito de soluciones correspondientes a un número ilimitado de opciones para los parámetros. Una solución de una ecuación diferencial que es libre de los parámetros arbitrarios se llama una **solución particular**.
En la mayoría de las aplicaciones no estamos interesados en la solución general de una ecuación diferencial, sino en una solución particular que satisfaga ciertas condiciones dadas. Esto da origen a los problemas de valor inicial o de frontera.

Definición 1.3.2 (Problema del valor inicial o de Cauchy) *Un problema de valor inicial o de Cauchy consta de una ecuación diferencial de orden n y de n condiciones iniciales impuestas a la función desconocida y a sus $n-1$ primeras derivadas en un valor de la variable independiente. Es decir,*

$$\frac{d^n y}{dx^n} = f(x, y, y^{(1)}, y^{(2)}, \cdots, y^{(n-1)})$$

$$\begin{aligned} y(x_0) &= y_0 \\ y^{(1)}(x_0) &= y_0 \\ y^{(2)}(x_0) &= y_1 \\ \vdots &= \vdots \\ y^{(n)} &= y_{n-1} \end{aligned}$$

Definición 1.3.3 (Problema de valor frontera o de Dirichlet) *Un problema de valores en la frontera o de Dirichlet consta de una ecuación diferencial ordinaria de orden n y n condiciones de frontera impuestas sobre la función desconocida en n valores de la variable independiente. Es decir,*

$$\frac{d^n y}{dx^n} = f(x, y, y^{(1)}, y^{(2)}, \cdots, y^{(m-1)})$$

$$y(x_0) = y_0$$

$$
\begin{aligned}
y(x_0) &= y_1 \\
y(x_1) &= y_2 \\
\vdots &= \vdots \\
y(x_{n-1}) &= y_{n-1}
\end{aligned}
$$

Ahora ya contamos con el material suficiente para describir como se pueden aplicar los algoritmos genéticos a la búsqueda de parámetros y al compararlos con otros métodos de búsqueda, los parámetros serán los que están dados por los problemas de valor de frontera e iniciales, así como otras constantes que contenga la ecuación diferencial a resolver.

1.4. Estructura de la tesis de licenciatura

La tesis proporciona una descripción de los algoritmos genéticos con los operadores clásicos de selección, recombinación y mutación, variando el valor de sus parámetros como la probabilidad de mutación, probabilidad de recombinación, y número de sobrevivientes, las pruebas fueron realizadas para los códigos, estos fueron aplicadas al problema de Thomson para encontrar la mejor configuración de dichos parámetros y operadores y, a su vez aplicar dicha configuración al problema de búsqueda de parámetros, como la constante elástica, la frecuencia, la constante de amortiguación, la masa, la fuerza aplicada, para la ecuación del oscilador armónico forzado y a la ecuación de onda, como la velocidad de propagación. Esta organizada en un total de cinco capítulos que se exponen a continuación:

El Capítulo 1 presenta la motivación de esta tesis de licenciatura, y una breve explicación de lo que se verá durante los siguientes capítulos. A su vez se introduce los conceptos fundamentales necesarios de las ecuaciones diferenciales y una descripción general de los que son los algoritmos genéticos.

El Capítulo 2 proporciona una descripción completa de los algoritmos genéticos, sus operadores clásicos, su comparación con otros algoritmos de búsqueda, el modelo de maestro y esclavo para una población en paralelo y el modelo de islas para implementar un algoritmo genético en paralelo con muchas poblaciones.

El Capítulo 3 describe los datos que resultan al implementar los algoritmos genéticos al problema de Thomson [11], donde se obtienen los mejores

operadores genéticos clásicos y parámetros. También se aplican los dos modelos propuestos por Cantú Paz [6] para implementar un algoritmo genético en paralelo a dicho problema.

El Capítulo 4 describe la implementación de los algoritmos genéticos a la búsqueda de parámetros de la ecuación de oscilador armónico forzado y la ecuación de onda, donde los individuos son representados por las soluciones numéricas que presentan los métodos de Runge-Kutta y de línea, para resolver ecuaciones diferenciales ordinarias y parciales, respectivamente, con diferentes parámetros.

El Capítulo 5 presenta las conclusiones de esta tesis de licenciatura, así como la discusión de resultados de los capítulos 3 y 5, y los planes a futuro.

Capítulo 2

Algoritmos genéticos

En este capítulo se describe el algoritmo genético básico y, se responderán algunas preguntas como: ¿Como evolucionan los algoritmos genéticos entre una generación y otra? ¿En que áreas los algoritmos genéticos muestran superioridad?. Estas y otras preguntas serán contestadas durante el capítulo.

2.1. Visión general

Los algoritmos genéticos son un proceso discreto, aleatorio y no lineal, que no requiere una formulación matemática, donde el óptimo de la población evoluciona de una generación a otra. Es tan importante en la resolución de largos problemas complejos que requieren un tiempo largo de acuerdo a las técnicas tradicionales de programación y, tiene un conjunto de soluciones alternativas, donde la solución es igual o muy cerca a las observadas.
Los algoritmos genéticos han probado su fuerza y durabilidad en resolver muchos problemas, y por lo tanto son considerados como una herramienta de optimización [12].
Los algoritmos genéticos fueron inspirados en la teoría Darwiniana de "la supervivencia del más apto" [8] y la producción de nuevos cromosomas (individuos) por medio de los operadores de recombinación (crossover) y mutación, es decir, el individuo más apto es más probable que permanezca en la siguiente generación y se reproduzca. Así, los individuos de las siguientes generaciones serán mas aptos, debido a que son producidos de individuos aptos, es decir la solución evoluciona de una generación a otra.
Muchas extensiones se han realizado a los algoritmos genéticos estándar propuestos por Holland [1], como los trabajos de [13] , [14], con el operador de recombinación de múltiples puntos, y [15] en el método de selección por rueda ruleta y la selección por rangos.

También destacan los algoritmos genéticos en paralelo, donde, como lo hemos mencionado con anterioridad, existen dos maneras de programarlos: con una sola población, conocida como el método de maestro y esclavo, y el método de poblaciones múltiples, conocido como el método de islas. Estos dos últimos métodos se discutirán en secciones posteriores.

2.2. Un algoritmo genético simple

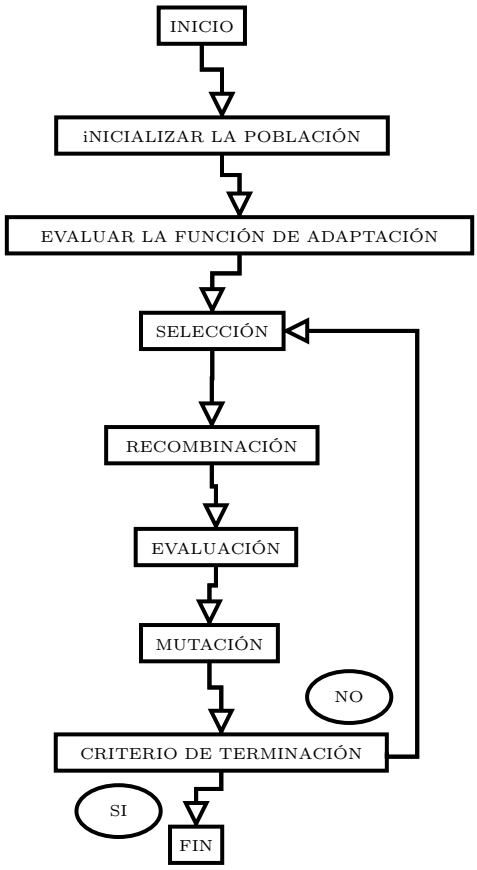


Figura 2.1: Esquema de un algoritmo genético simple

Un algoritmo genético simple o básico esta representado en la Figura 2.1. El primer paso en la implementación de un algoritmo genético es realizar la conexión entre el mundo real y el mundo de los algoritmos genéticos. A esto se le llama "la programación" o "representación" de las soluciones del problema (individuos) [2]. La representación depende principalmente de problema a

resolver, y por lo tanto se debe de seleccionar una técnica apropiada. Las técnicas más conocidas son las siguientes:

1. Representación en binario: Un cromosoma en esta técnica se representa usando cadenas binarias, como el problema de la mochila [55] (vea Figura 2.2).

Cromosoma A: 10101010111101010
Cromosoma B: 11101010101010001

Figura 2.2: Ejemplo de una representación en binario

2. Representación en permutaciones: Los cromosomas,en este tipo de técnica, representan una posición en una secuencia, donde esta técnica se usa en problemas de ordenamiento, es decir, como "el problema del vendedor ambulante" [16] [56] [2] (vea Figura 2.3).

Cromosoma A: 4381087504823
Cromosoma B: 3672938492394

Figura 2.3: Ejemplo de una representación en permutaciones

3. Representación en valores: cada cromosoma es representado usando una secuencia de algunos valores. Estos valores son posiblemente caracteres, números reales, etc. (vea figura 2.4) y pueden ser usados en una red neuronal [17] [2] ó, en la búsqueda de parámetros.

Cromosoma A: $arriba, abajo, izquierda, derecha$
Cromosoma B: $KHFPEJDLSJWEJFSDDS$
Cromosoma C: 0,1, 0,3, 0,8, 2,6, 5,7, 9,2, 3,5, 1,3

Figura 2.4: Ejemplo de una representación en valores

4. Representación de árboles: cada cromosoma es un árbol de algún objeto [18] [2]. Esta representación se usa en la programación genética (vea figura 2.5)

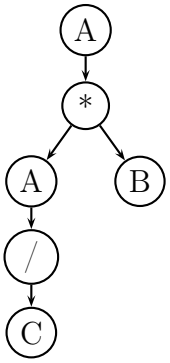


Figura 2.5: Representación de arboles. Árbol para la expresión de Lisp (A(*B(*A/C)))

Una vez que se haya identificado la técnica posible para enfrentar el problema dado, se recurre a la codificación. Ahora discutiremos las definiciones más importantes en el algoritmo genético.
Los *individuos* son las posibles soluciones del problema y pueden representarse como un conjunto de parámetros (que denominaremos *genes*), los cuales agrupadas forman un conjunto de valores (referidos como *cromosoma*). En términos biológicos, el conjunto de parámetros representando un cromosoma particular se denomina *fenotipo*. El fenotipo contiene la información requerida para construir un organismo, el cual se refiere como *genotipo*. Los mismos términos se utilizan en el campo de los algoritmos genéticos. La adaptación al problema de un individuo dependen de la evaluación del genotipo y, este último puede inferirse a partir del fenotipo.
La *función de adaptación* (función fitness) debe ser diseñada para cada problema de manera específica. Dado un cromosoma particular, la función de adaptación le asigna un número real, que refleja el nivel de adaptación del individuo al problema representado por el cromosoma.
Durante la *fase de reproducción* se seleccionan los individuos de la población para cruzarse y producir descendientes, que pertenecerán, una vez ya mutados, la siguiente generación de individuos. La *selección de padres* se efectúa por medio del *operador de selección*, donde se favorece a los individuos mejor adaptados, ya que a cada individuo se le asigna una probabilidad de ser seleccionado que es proporcional a su función de adaptación. Así, los individuos bien adaptados se escogerán con mayor frecuencia mientras que los pobremente adaptados al problema se escogerán de vez en cuando. En este operador destacan, la *selección por torneos*, *selección por rueda de ruleta*, *selección por rangos* y *selección estocástica universal de muestras* (vea sección 3 de este capítulo).

Cuando ya se hayan seleccionado los padres, sus cromosomas se combinan, usando los *operadores de recombinación* y *mutación*.
El *operador de recombinación* escoge un cierto número de genes del padre y otro de la madre y los combina y, el resultado son dos nuevos cromosomas completos (hijos). La manera en que se escoge el número de genes depende del tipo de operador de recombinación, donde, los operadores más destacados son: recombinación de un punto, recombinación de puntos múltiples y recombinación uniforme (vea sección 3 de este capítulo).
El *operador de mutación* se aplica a cada hijo de manera individual y consiste en la alteración aleatoria (normalmente con probabilidad pequeña, veremos en capítulos posteriores que es aproximadamente de 0.005 %) de cada gen componente del cromosoma. Existe varios operadores de mutación y, los que destacan son reajuste aleatorio, mutación de intercambio y mutación uniforme (vea sección 3 de este capítulo).
Así, tenemos los siguientes pasos principales que todo algoritmo genético simple se compone de manera resumida como:

1. **Inicio**: Crear una población de manera aleatoria (población inicial) que consiste de n individuos.

2. **Función de adaptación (fitness)**: Evalúe la función de adaptación $f(x)$ de cada individuo, x, en la población.

3. **Nueva población**: Repita los siguientes pasos para crear una nueva población hasta la finalización de la nueva población.

4. **Selección (reproducción)**: El proceso de elegir el padre más "apto" en la población para que se reproduzca, "apto" se define en base al problema a tratar, tiene un papel importante en resolver la convergencia prematura que resulta de una falta de diversidad en la población, por lo tanto, identificar un operador de selección apropiado es un paso crítico.

5. **Recombinación**: Este proceso toma dos padres (cromosomas) para crear un nuevos individuos por medio de intercambiar los segmentos de los genes de los padres. Es más probable que los nuevos individuos (hijos) contengan buenas características de los padres, y consecuentemente se desempeñen mejor a comparación de sus ancestros.

6. **Mutación**: Aquí es donde ocurre un cambio de genes específicos dentro de un solo cromosoma para crear cromosomas que puedan proporcionar nuevas soluciones para las siguientes generaciones y, de este modo, introduzcan un cierto nivel de diversidad a la población.

7. **Criterio de terminación**: Hay muchos condiciones de terminación que han sido aplicados a los algoritmos genéticos simples, pero nosotros nos limitaremos a que el criterio de terminación ocurra cuando se alcance el límite máximo de generaciones.

Los parámetros de los algoritmos genéticos

Existen 4 parámetros básicos de los algoritmos genéticos, estos son:

1. Probabilidad de recombinación: Este determina el número de veces que ocurre la recombinación en una generación, y se puede establecer desde 0 % hasta 100 %, y es también un parámetro sensible.

2. Probabilidad de mutación: Este determina la probabilidad de uno o un conjunto de genes muten en el cromosoma hijo, también es un parámetro sensible. Incrementar o disminuir las probabilidades de mutación y recombinación pueden tener un efecto positivo o negativo, como sugieren las investigaciones de [19].

3. Tamaño de población: Nuestra selección del tamaño de población es un tema delicado, si el tamaño de la población es pequeño, significa que el espacio de búsqueda es también pequeño, entonces es posible alcanzar el óptimo local. Sin embargo, si el tamaño de la población es muy grande, esto incrementará el área de búsqueda e incrementará la carga de operaciones y, por lo tanto, el proceso se convertirá muy lento. Por lo tanto, el tamaño de la población debe de ser razonable.

4. Número de generaciones: Un número de ciclos antes de la finalización del criterio de terminación. En algunos casos, cientos de íteraciones son suficientes y en otras no, esto depende en la complejidad del problema a resolver.

Todos estos parámetros son sumamente importantes porque ellos determinan la calidad de la solución . Estos parámetros típicamente interactúan entre sí de manera no lineal, así que no pueden ser optimizados una a la vez. Existe todavía una gran discusión sobre los ajustes de parámetros en la literatura de la computación evolutiva [2]. No existen resultados finales sobre cual es la mejor configuración de parámetros, Daremos revisión de las aproximaciones experimentales que los investigadores han tomado para encontrar la mejor configuración de parámetros.
De Jong (1975) [26] realizó un estudio sobre como los diferentes parámetros afectan el rendimiento de búsqueda "en línea" y "fuera de línea" de los algoritmos genéticos. Búsqueda "en línea" realizada a tiempo t es la función de

adaptación promedio de todos los individuos que han sido evaluados sobre t pasos de evolución y, la búsqueda "fuera de línea" a tiempo t es el valor promedio, sobre t pasos de evolución, de las mejores funciones de adaptación que han sido realizadas en cada paso de evolución. Los experimentos de De Jong indicaron que el mejor tamaño de población fue de $50-100$ individuos, la mejor probabilidad para el operador de recombinación de un solo punto fue de 0.6 por pareja de padres y la mejor probabilidad de mutación fue de 0.001 por gen.
Un tiempo después, Grefenstette (1986) [16], notó que como el algoritmo genético puede ser usado como un método de optimización, podría ser usado para optimizar los parámetros de otro algoritmo genético. En los experimentos de Grefenstette resultó que para el individuo más apto para una búsqueda "en línea" se establece una población de 30 individuos, la probabilidad de recombinación es de 0.95, la probabilidad de mutación es de 0.01, y se debe usar una selección de elitismo, es decir, los individuos más aptos de cada población sobreviven para la siguiente generación.
Schaffer,Caruana, Esherlman, y Das (1989) [28] pasaron más de un año probando un amplio rango de diferentes configuraciones de parámetros y, lo realizaron a una búsqueda "en línea" de un algoritmo genético. Schafer encontró que las mejores configuraciones para el tamaño de la población, probabilidad de recombinación y mutación eran independientes del problema a tratar. Estas configuraciones resultan similares a las que encontró Grefenstette: el tamaño de la población es de $20-30$ individuos, la probabilidad de recombinación resulta de $0.75-0.95$, y la probabilidad de mutación es de $0.005-0.01$. Otro estudio que llegó a configuraciones similares sugiere que se debe de trabajar con poblaciones más grandes [12].
Se cree que la aproximación más prometedora es que los valores de los parámetros se *adapten* en tiempo real a la búsqueda en curso. Donde sobresale las investigaciones de Lawrence Davis [29] para aproximar las probabilidades de operadores auto adaptables. Aunque se ha hecho muy poco trabajo de como medir las diferentes razones de adaptación y que tan bien armonizan en diferentes experimentos de adaptación de parámetros.

2.3. Operadores

operador de selección

Mencionaremos los tipos de selección más comúnes:

Selección por rueda de ruleta, también conocido como selección pro-

porcional a la función de adaptación. Este método es conceptualmente equivalente al juego de la ruleta, que es un juego de azar típico de los casinos, y consiste en dar a cada individuo un pedazo de un círculo de la ruleta igual en área a la función de adaptación del individuo. Para seleccionar al individuo se "gira" la rueda, se lanza la "pelota" cuando esté llegue al reposo en algún pedazo del círculo, el correspondiente individuo que le corresponda esa área será seleccionado (vea Figura 2.6).

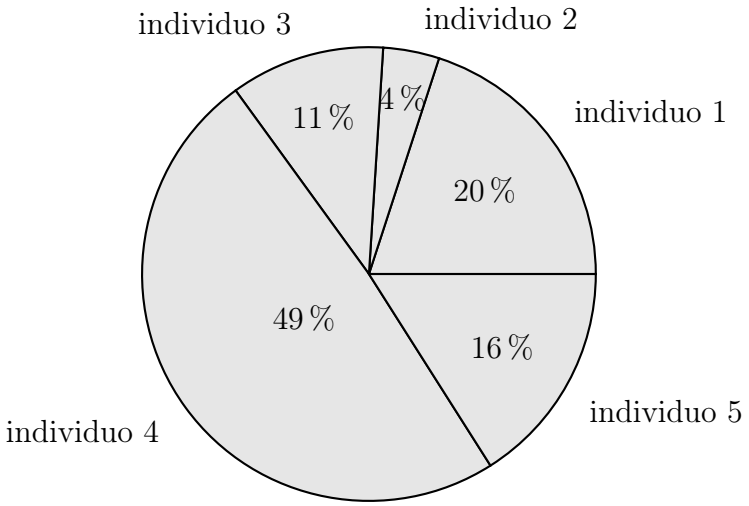


Figura 2.6: Método de selección por rueda de ruleta

De manera análoga, en el algoritmo genético lo que hace la "pelota" y el "giro" de la ruleta es seleccionar a un individuo de manera aleatoria, en el código esto se consigue por medio de un número aleatorio que se encuentra entre $(0, S)$, donde S es la suma de la función adaptación de todos los individuos. Se repite el mismo proceso para encontrar al otro individuo. Una vez seleccionado ambos padres, se someten a una recombinación. Este método se implementa de la siguiente manera:

1. Calcular la suma de todas las funciones de adaptación de los individuos en la población. Llamemos a esta suma S

$$S = \sum_{i=1}^{N} F_i \tag{2.1}$$

   Donde F_i es el valor de la función de adaptación del individuo i y N es el número de individuos en la población.

2. Repita lo siguiente hasta que N individuos sean creados

3. Genere un número aleatorio real, r, entre el intervalo $(0, S)$.

4. Sume el valor de la función de adaptación de los individuos hasta que la suma sea mayor o igual a r. El individuo cuyo valor esperado pone la suma sobre este límite es el individuo seleccionado.

Para tener una visualización de como implementar este método en la computadora, vease el algoritmo 2.1

algoritmo 2.1: Método de selección rueda de ruleta en Fortran 90

```
S=0.0d0
do i=1,N
        S=S+F(i)
end do

r=rand()*S

if(r<=F(i)) padre1=1

Suma=F(1)

do i=1,N-1
        Suma=Suma+F(i)
    if(r>Suma .and. r<=Suma+F(i+1)) padre1=i+1
end do
```

donde $padre1$ indica el número del individuo que es seleccionado.
Se observa que la probabilidad de que cada individuo sea seleccionado, P_i, esta dada por:

$$P_i = \frac{F_i}{S} \tag{2.2}$$

La **Selección estocástica universal por muestras** surgió con el fin de corregir el riesgo de la convergencia prematura que proporciona el método de selección por rueda de ruleta. Fue desarrollado por Baker [23],en 1987. Recordando el método de selección por rueda de ruleta, los individuos están dispersados en una área que es proporcional a su función de adaptación en la rueda de ruleta. Ahora, consideramos que esta ruleta puede estar dividida en k regiones o segmentos de igual tamaño (vea Figura 2.7), por lo tanto, en

cada región se encuentra al menos un individuo, así, tenemos una rueda de ruleta que esta distribuida por el área proporcional a la función de adaptación de cada individuo y por las regiones.

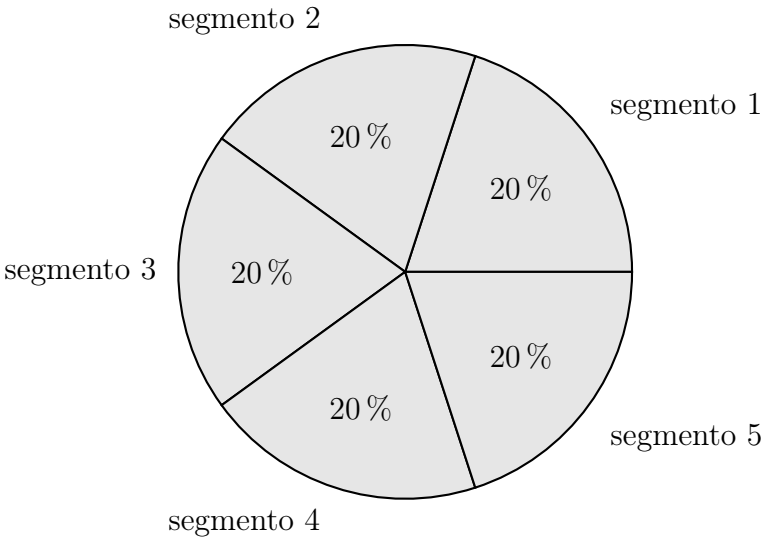


Figura 2.7: Método de selección estocástica universal por muestras. En este caso divididos la ruleta en 5 regiones iguales, la región se elige de manera aleatoria y con los individuos dentro de la región se procede con el método de selección por rueda de ruleta como se explica en la Figura 2.6

Ahora, este método consiste en lanzar la "pelota" y "girar" la rueda ruleta para elegir la región en que se va a trabajar, cuando la pelota llegue al reposo en algún pedazo de círculo la correspondiente región que le corresponda dicha área será seleccionado. Una vez seleccionada dicha región, se hace una nueva ruleta con los individuos que se encuentran en la región seleccionada. En esta ruleta ,de la misma manera, los individuos están dispersados en una área que es proporcional a su función de adaptación y se procede ahora como en el método de selección por rueda de ruleta para seleccionar a un individuo. Este método se implementa de la siguiente manera:

1. Calcular la suma de todas las funciones de adaptación de los individuos en la población. Llamemos a esta suma S

$$S = \sum_{i=1}^{N} F_i \tag{2.3}$$

Donde F_i es el valor de la función de adaptación del individuo i y N es el número de individuos en la población.

2. Repita los siguientes pasos hasta que N individuos sean seleccionados.

3. Genere un número aleatorio entero, q, en el intervalo de $(0, k-1)$, donde k es el número de regiones presentes en la ruleta. Donde el tamaño de la región k, k_size, está definida por:

$$k\_size = \frac{S}{k} \tag{2.4}$$

4. Genere un número aleatorio real, r, en el intervalo $(q * k\_size, (q+1) * k\_size)$

5. Sume el valor de la función de adaptación de los individuos de esta región hasta que la suma sea mayor o igual a r. El individuo cuyo valor esperado pone la suma sobre este límite es el individuo seleccionado.

Su implementación esta dada por :

algoritmo 2.2: Método de selección estocástica universal de muestras en Fortran 90

```
S=0
do i=1,N
        S=S+F(i)
end do

k_size=S/k

q=int(rand()*k)
r=dble(q)*k_size+rand()*k_size

if(r<=F(1)) padre1=1

Suma=F(1)
do i=1,N
        Suma=Suma+F(i)
    if(r>Suma .and. r<=Suma+F(i+1)) padre1=i+1
end do
```

Al momento de agregar el concepto de regiones se tienen mas oportunidades de elegir más individuos, lo que permite que exista una mayor diversidad.

Selección por rangos [2], este método es más usado cuando los individuos de la población tienen valores de su función de adaptación muy cercanos. Esta técnica clasifica a toda la población de acuerdo a su función de adaptación y entonces cada individuo recibe un rango, así, el valor esperado de cada individuo depende de su rango en vez de su función de adaptación. Para asignar el rango a cada individuo, en una población de N individuos, primero ordenamos a toda la población de acuerdo a su función de adaptación de menor a mayor, es decir, el individuo 1 tendrá el peor valor de la función de adaptación de la población mientras que el individuo N es el más apto de la generación actual. Ahora, les asignamos su correspondiente rango, de acuerdo al ordenamiento que hicimos, el individuo con peor valor de la función de adaptación será de rango 1, el segundo peor de rango 2, y así sucesivamente hasta llegar a N. Después se procede de la misma manera como en la selección por rueda de ruleta o por medio de la selección estocástica universal de muestras. La ventaja de este método es que da oportunidad a que individuos con una función de adaptación bajo sean seleccionados, ya que su selección solo depende de su rango. Para visualizar la distribución de rangos de cada individuo vea la Figura

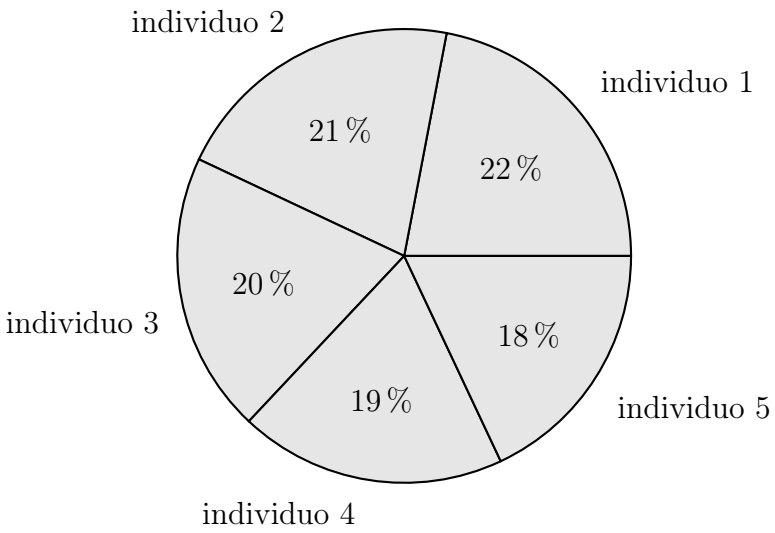


Figura 2.8: Método de selección por rangos.

Por lo tanto, este método se implementa de la siguiente manera:

1. Ordene a los individuos de menor a mayor de acuerdo a su función de

adaptación. En este paso podemos utilizar los métodos de ordenamiento más conocidos como: burbuja, quicksort,heapsort, insertionsort, bucketsort, radixsort, shellsort,etc. [22]. Por simplicidad damos el ejemplo del método ordenamiento burbuja en Fortran 90 (vea algoritmo 2.3), aunque se han utilizado los métodos dichos con anterioridad:

algoritmo 2.3: Método de ordenamiento burbuja aplicado a la función de adaptación de un algoritmo genético

```
do i=1,N
         do j=1,N-i
         if(F(j)<F(j+1))then
                  aux=F(j)
             F(j)=F(j+1)
             F(j+1)=aux
         end if
     end do
end do
```

donde N es el tamaño de la población y el $F(j)$ es el valor de la función de adaptación asociada a cada individuo.

2. Asigne a cada individuo de acuerdo a su función de adaptación. Así, el primer individuo tiene rango 1, el segundo individuo tiene rango 2, y así sucesivamente hasta llegar a N. Así, el de mayor rango tiene más oportunidad de ser seleccionado. Esto se implementa de manera sencilla en Fortran 90 como:

algoritmo 2.4: Clasificación de individuos

```
do i=1,N
         F(i)=i
end do
```

3. Implemente el método de selección por rueda ruleta o el método de la selección estocástica universal de muestras.

Ahora, recordando que la suma de los primeros N números naturales esta dada por la fórmula de Faulhaber [21], para el caso donde sus potencias son iguales a uno:

$$1 + 2 + 3 + \cdot + N = \frac{N^2 + N}{2} \tag{2.5}$$

Así, basados en su rango, cada individuo i tendrá una probabilidad P_i de ser seleccionado, definido como:

$$P_i = \frac{rango(i)}{\frac{N^2+N}{2}} \tag{2.6}$$

Es conveniente usar este método cuando se observe que el individuo más apto de cada generación se mantiene constante, ya que puede decir que los valores de la función de adaptación de cada individuo en la población son parecidos.

El **método de selección por torneos** [12] es probablemente el método de selección más popular en los algoritmos genéticos. En esta técnica, k individuos son seleccionados de manera aleatoria, y estos individuos compiten entre sí. El individuo con el valor mayor de la función de adaptación gana y es seleccionado para reproducirse (vea Figura 2.9)

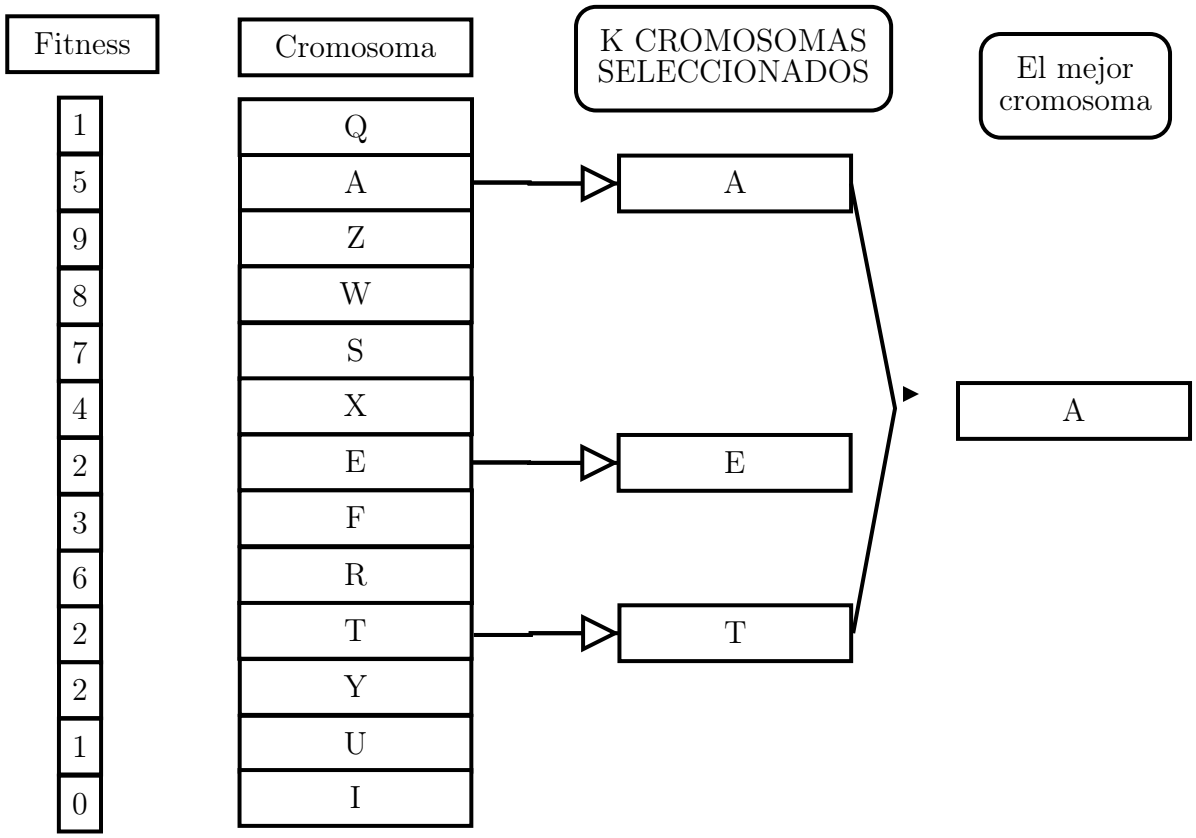


Figura 2.9: Método de selección por torneos

Se repite el procedimiento para encontrarle una pareja y para que se puedan reproducir posteriormente. Esta técnica es implementada de la siguiente forma:

1. Se escogen p participantes de una población de N individuos de manera aleatoria por un número entero q.
2. Se ordenan los individuos seleccionados de acuerdo a su función de adaptación de menor a mayor.
3. Se selecciona el individuo con la función de adaptación mayor
4. Se busca al padre que le corresponde dicha función de adaptación y es seleccionado.

Para ilustrar como se vería el código en Fortran 90, vea algoritmo 2.5

algoritmo 2.5: Implementación del método de selección por torneos

```
do i=1,p
        q=int(rand()*N)+1
    Fp(i)=F(q)
end do

call burbuja

do i=1,N
        if(F(i)==Fp(p)) padre1=i
end do
```

donde $F(i)$ es el valor de la función de adaptación del individuo i y $Fp(i)$ es el valor de la función de adaptación del individuo participante i.
Este método de selección da la oportunidad de que todos los individuos sean seleccionados y, por lo tanto, se preserva la diversidad.

operador de recombinación

Durante los años se han desarrollado varios operadores de recombinación [13, 14]. Nosotros nos concentraremos en los más conocidos, o que comúnmente se utilizan. En cada uno de los operadores de recombinación establecemos que el tama´ no de la población es de N individuos y suponemos que cada cromosoma consiste de k genes. Cabe destacar que consideramos

el caso más general, ya que el intercambio genético de los pares depende del problema a resolver.
El **operador de recombinación de un punto** [1,2]: Los algoritmos genéticos tradicionales usan recombinación de un punto. Se determina de manera aleatoria el punto p en los cromosomas y entre los padres seleccionados se intercambian genes a partir de ese punto para reproducir dos hijos (vea figura 2.10).

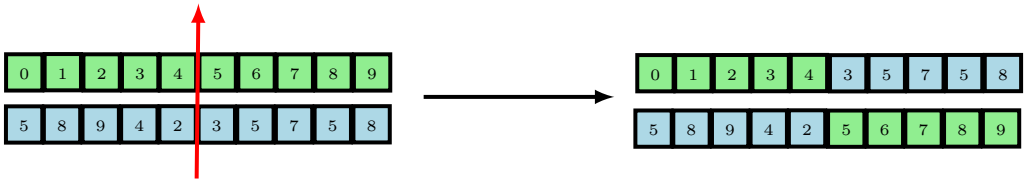


Figura 2.10: Operador de recombinación de un solo punto aplicado a una representanción en bits

Una idea de este intercambio es dado en Fortran 90 en el siguiente algoritmo

algoritmo 2.6: Implementación del operador de recombinación de un solo punto en Fortran 90

```
p=int(rand()*k)+1

do i=1,k
        if(i>=p)then
                hijo(contador,i)=padre(a,i)
        hijo(contador+1,i)=padre(b,i)
    else
        hijo(contador,i)=padre(b,i)
        hijo(contador+1,i)=padre(a,i)
    end if
end do
```

donde *contador* se refiere al número de hijo creado para la nueva generación, así, el $hijo(contador, i)$ representa el material genético almacenado en el gen i del hijo *contador*. De manera similar ocurre con $padre(a, i)$ y $padre(b, i)$, que representan el material genético del padre a y b en el gen i, respectivamente, dichos individuos fueron elegidos durante el proceso de selección de padres para la recombinación genética.

El **operador de recombinación de múltiples puntos** [14] : Tiene la misma característica que el operador anterior solo que ahora el intercambio genético se aplica en muchos puntos. En este operador p puntos son seleccionados en los cromosomas de los padres seleccionados para realizar el intercambio genético. El intercambio se realiza entre cada dos puntos en el cromosoma de los padres para producir dos hijos (vea Figura 2.11).

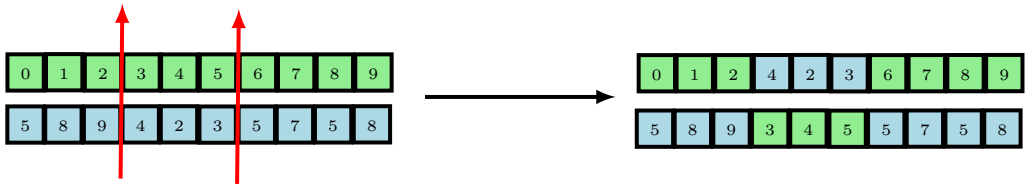


Figura 2.11: Operador de recombinación de múltiples puntos aplicado a una representanción en bits

algoritmo 2.7: Implementación del operador de recombinación de múltiples puntos

```
p=int(rand)*k)+1

call crear_multiP

hijo(counter,:)=padre(a,:)
hijo(counter+1,:)=padre(b,:)

do i=1,p-1,2
        do j=1,k
        if(j>multiP(i) .and. j<=multiP(i))then
                hijo(counter,j)=padre(b,j)
            hijo(counter+1,j)=padre(a,j)
        end if
    end do
end do
```

Estos puntos pueden ser seleccionados de manera aleatoria o implementados por el usuario. Por simplicidad consideremos el caso donde los puntos seleccionados están dados por el usuario, el usuario debe de escribir los pun-

tos en orden creciente. Una idea del intercambio en Fortran es presentado en la siguiente figura (vea Figura 2.7), donde $multiP$ es un arreglo unidimensional de los puntos para la recombinación, estos puntos son ingresados por el usuario por medio de la subrutina $crear\_multiP$. Las definiciones restantes están descritas en el método posterior.

El **operador de recombinación uniforme** [20]: Este método copia el gen correspondiente de un padre o del otro, elegido de acuerdo a una "mascara de recombinación" generado al azar para crear cada gen del descendiente. Donde, si es 1 en la "mascara de recombinación" el gen se copia del primer padre y donde es 0 en la "mascara de recombinación", el gen es copiado del segundo padre (vea figura 2.12).

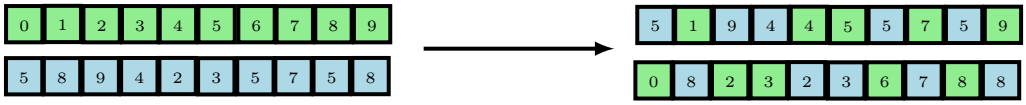


Figura 2.12: Operador de recombinación uniforme aplicado a una representanción en bits

El proceso se repite con el intercambio genético de los padres para producir el segundo descendiente. Una idea de este proceso puede ser programado en Fortran como en la siguiente figura (vea algoritmo 2.8), donde r y s son número aleatorios entre $(0, 1)$ y, por lo tanto, en nuestra "mascara de recombinación"si r o s son menores que 0.5 el gen es copiado del primer padre y, si son mayores a 0.5 el gen es copiado del segundo.

algoritmo 2.8: Implementación del operador de mutación uniforme en Fortran 90

```
do i=1,k
    !Para el primer hijo
        r=rand()
    if(r<0.50d0)then
        hijo(counter,i)=padre(a,i)
    else
        hijo(counter,i)=padre(b,i)
    end if

    !Para el segundo hijo
    s=rand
    if(s<0.50d0)then
        hijo(counter+1,i)=padre(a,i)
```

```
14      else
15          hijo(counter+1,i)=padre(b,i)
16      end if
17  end do
```

Operador de mutación

La mutación es una de las etapas más importantes de los algoritmos genéticos por su impacto en la exploración del óptimo, donde se hace un cambio entre genes específicos con un solo cromosoma para crear cromosomas que proporcionen nuevas soluciones para las siguientes generaciones.
Después de la recombinación los descendientes sufren mutación. Aunque, de manera general, el operador de recombinación es el principal responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, el operador de mutación se vuelve importante a medida que los individuos más aptos de cada generación convergen [24].
La mutación clásica (llamada mutación de cambio de bits) fue desarrollada por Holland [1] para tratar diferentes problemas que ya no se adaptan, es decir, el individuo más apto no cambia entre generaciones. Por lo tanto, muchos métodos de mutación fueron propuestos, incluyendo: Mutación uniforme, mutación de intercambio y mutación de reajuste al azar.
En cada uno de los siguientes operadores de mutación suponemos que cada individuo contiene un cromosoma con k genes.

El **operador de mutación uniforme** [25]: El operador de mutación cambia el valor del gen elegido con un valor aleatorio uniforme seleccionado en el intervalo superior e inferior especificado por el usuario (vea Figura 2.13). La manera en que se elige al gen para mutar es por medio de una analogía a la "mascara de recombinación" vista anteriormente en operadores de recombinación, llamada ahora "mascara de mutación", se aplica a cada gen del cromosoma, donde si es 0 el gen muta y no, si es 1 .Este operador puede ser usado solo para genes reales y enteros.

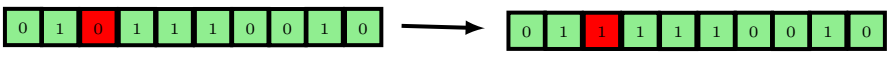


Figura 2.13: Operador de recombinación uniforme aplicado a una representanción en bits

Esta parte del código puede verse en Fortran de la siguiente forma (vea algoritmo 2.9), donde hemos elegido el gen mute si es menor que 0.5 y no

en caso contrario, ya que la función *rand*() obtiene números aleatorios en el rango de $(0, 1)$:

algoritmo 2.9: Implementación del operador de mutación uniforme

```
do i=1,k
         p=rand()
     if(p <=0.50d0)then
         q=rand()
         hijo(counter,i)=q*int_usuario
     end if

end do
```

donde p y q son números aleatorios entre $(0, 1)$, $hijo(counter, i)$ contiene el valor del gen i del hijo *counter* e $int\_usuario$ es el intervalo seleccionado por el usuario variando de acuerdo al problema a tratar.

El **operador mutación de intercambio**, también usado en la representación de permutaciones del algoritmo genético. Este método de mutación selecciona dos genes al azar, r y s, e intercambia sus posiciones (vea Figura 2.14), esto se hace una p cantidad de veces, p es un número aleatorio en el intervalo $(0, k)$. Preserva la mayoría de la información próxima de los individuos.

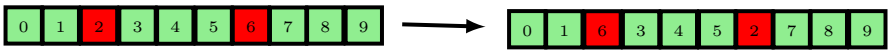


Figura 2.14: Operador de mutación de intercambio aplicado a una representanción en bits

Una forma de ver como funciona este método es la siguiente representación en Fortran (vea Figura 2.10):

algoritmo 2.10: Implementación del operador de mutación de intercambio

```

p=int(rand()*k)+1

do i=1,p
         r=int(rand()*k)+1
     s=int(rand()*k)+1


```

```
9       aux=hijo(counter,r)
10      hijo(counter,r)=hijo(counter,s)
11      hijo(counter,s)=aux
12
13  end do
```

donde $hijo(counter, r)$ contiene el gen r del individuo e $hijo(counter, s)$ contiene el gen s del mismo individuo y la variable *auxiliar* nos ayuda a guardar el valor del gen r.

El **operador de mutación de reajuste al azar** es una extensión de la mutación por cambios de bits de Holland [1] (vea Figura 2.13). En este método, un valor de manera aleatoria del conjunto de valores permitidos (de acuerdo al problema a tratar) es asignado a un gen, s, seleccionado al azar y, el número de genes que sufren mutación, r, son elegidos de manera aleatoria en un rango de $(1, k)$. Es decir, la diferencia entre la mutación uniforme y la de reajuste al azar radica en la cantidad de genes que se pueden someter a mutación, en la primera son k individuos y en la segunda son r, donde $k >= r$. Para visualizar la diferencia entre ambos métodos observamos su implementación en Fortran 90, vea los algoritmos 2.11 y 2.9

algoritmo 2.11: Implementación del operador de mutación de reajuste al azar

```
1
2   r=int(rand()*k)+1
3
4   do i=1,r
5           s=int(rand()*k)+1
6       p=rand()
7           hijo(counter,s)=p*int_us
8
9   end do
```

donde $hijo(counter, s)$ contiene el gen s del individuo $counter$ e int_us es el intervalo de valores permitidos establecido por el usuario.

2.4. Algoritmos genéticos en paralelo

Como hemos visto con anterioridad, se han hecho varios esfuerzos en el campo de la computación evolutiva para mejor la eficiencia de los algoritmos genéticos [2, 26–29]. Una de las técnicas para mejorar la eficiencia que ha sido investigada tanto teórica como práctica, es el tema de la paralelización.

Esta es una nueva clase de algoritmos genéticos cuyo mecanismo operacional difiere de los algoritmos genéticos simples [1]. La eficiencia en los algoritmos genéticos recae en 3 objetivos: (1) maximizar la calidad de la solución, (2) minimizar el tiempo de ejecución, y (3) minimizar las fuentes de memoria. El último usualmente no es algo preocupante debido a que en los algoritmos genéticos tradicionales, los requisitos de memoria se mantienen constante durante toda la ejecución.
Los algoritmos genéticos pueden ser programados en paralelo debido a que trabajan con una población de soluciones que puede ser evaluada independientemente de las otras. En la mayoría de los problemas el tiempo es gastado en evaluar las soluciones en lugar de los mecanismos internos de los operadores de los algoritmos genéticos. De hecho, el tiempo gastado en los operadores de los algoritmos genéticos es casi despreciable, excepto para poblaciones de miles de individuos, comparado con el tiempo usado en evaluar las soluciones de los individuos.
En el trabajo de Cantú-Paz [6], se han desarrollado modelos teóricos que dirigen a decisiones racionales para establecer los diferentes parámetros que involucran la paralelización de algoritmos genéticos. Existe dos maneras principales de implementar el algoritmo genético en paralelo:

1. Usando una sola población

2. Usando múltiples poblaciones

Cantú-Paz [6] concluyó que para el caso de la arquitectura de Maestro-Esclavo, los beneficios de la paralelización ocurren principalmente en problemas con función de adaptación grandes debido a que necesitan comunicación constante. En el método de islas se tienen menos costos de comunicación pero no evita completamente el problema de gasto de tiempo de computo en las comunicaciones. Así, en su investigación los costos de comunicación imponen un límite de que tan rápido pueden ser los algoritmos genéticos en paralelo.

Algoritmos genéticos en paralelo: "Maestro-Esclavo"

Este método se basa de un procesador que dirige toda la implementación del algoritmo, conocido como el procesador maestro, en este se concentran los operadores de los algoritmos genéticos y en otros procesadores, conocidos como esclavos, se encargan de evaluar la función de adaptación, como se ve en la Figura 2.15

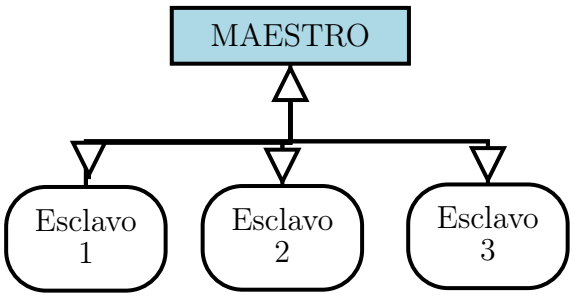


Figura 2.15: Esquema de la implementación del método Esclavo-Maestro. El maestro manda a que los esclavos evalúen la funciónde adaptación de un conjunto de individuos distribuidos del mismo tamaño entre los esclavos.

Así, un algoritmo genético de una sola población en paralelo usualmente tiene la siguiente arquitectura, recordando que el maestro es el que aplica los operadores

1. Distribuya una cantidad de N/k individuos entre cada procesador, donde suponemos que N es el tamaño de la población y k el número de procesadores.Cabe destacar que N/k debe de ser divisible y convenientemente par.

2. Calcule su función de adaptación de cada individuo en el procesador.

3. Regrese la función de adaptación de cada individuo al procesador maestro.

4. Aplique el operador de selección hasta que N padres sean seleccionados.

5. Distribuya los padres seleccionados a cada esclavo.

6. En cada procesador aplique los operadores de recombinación y mutación

7. Regrese los individuos al maestro.

8. Vuelva al punto 4 hasta que se cumpla el criterio de terminación.

La mayoría de las publicaciones disponibles son de practicantes que necesitan encontrar soluciones a sus aplicaciones de manera rápida, pero hay muy pocos estudios teóricos. A continuación mencionamos algunos trabajos relevantes en esta área.
Bethke (1976) [30] fue el primero en describir la implementación en paralelo

de un algoritmo genético convencional. Concluyó que la eficiencia del uso de la capacidad de procesadores puede ser cercana al 100 % si las evaluaciones de la función de adaptación son costosas en tiempo de computo a comparación de los operadores de los algoritmos genéticos. Sin embargo, dicho análisis ignora el costo de las comunicaciones entre procesadores.
Fogarty y Huang [31] intentaron evolucionar un conjunto de reglas para un equilibrio de polos que toma un tiempo considerable para simular. Sus investigaciones los llevaron a la conclusión de que la configuración física de la redes de comunicación entre nodos no hacen una diferencia significativa en el tiempo de ejecución.
En general, las implementaciones de maestro-esclavo son más eficientes cuando las evaluaciones son más costosas en tiempo de cómputo. Por ejemplo, Grefenstette [32] obtuvo cerca de un 80 % de eficiencia en un trabajo de aprendizaje de robots, pero solo el 50 % con una tarea más fácil.
Además de la evaluación de la función de adaptación, podemos también paralelizar los operadores. Así, los operadores de recombinación y mutación pueden realizarse en paralelo usando la misma idea de partir a la población en nodos y distribuir el trabajo entre los procesadores. Sin embargo, estos operadores son tan simples que cualquier ganancia en el rendimiento será compensada con el tiempo requerido de la comunicación entre procesadores. El costo de la comunicación también complica la paralelización de la selección, principalmente porque existen diferentes métodos de selección que necesitan información de la población completa y requerirán intercambios excesivos de comunicación. Así, el método de selección tampoco contribuye de manera significante en el tiempo de ejecución del algoritmo genético.
De acuerdo al análisis teórico realizado por Cantú-Paz [6], el tiempo transcurrido para una generación de algoritmos genéticos en paralelo puede ser estimado como:

$$T_p = \mathcal{P}T_c + \frac{nT_f}{\mathcal{P}} \tag{2.7}$$

Donde T_c es el tiempo usado para comunicar al maestro con cada esclavo, T_f es el tiempo requerido para evaluar un individuo, $\mathcal{P}$ es el número de procesadores usados, n es el tamaño de la población. Se observa de 2.7 que mientras más esclavos se usen, el tiempo de computo disminuye, pero el tiempo de las comunicaciones incrementa. La eficiencia del algoritmo genético se define como la razón de la aceleración dividida por el número de procesadores [6]

$$E_p = \frac{T_s}{T_p \mathcal{P}} \tag{2.8}$$

donde T_s es el tiempo que un simple algoritmo genético usa en una generación y se define como $T_s = nT_f$.

El tiempo que un simple algoritmo genético usa en una generación es $T_s = nT_f$. Esto indica que la eficiencia sería del 100 % si la aceleración paralela es igual al número de procesadores pero, debido al costo de las comunicaciones esto no sucede.

Método de islas

Ahora nuestra atención esta centrada en los algoritmos genéticos con múltiples poblaciones, que son los más populares. También son conocidas como el *método de islas* de algoritmos genéticos porque se asemejan al modelo que es usado para describir poblaciones naturales que son relativamente aisladas entre sí, como islas. En este método los individuos pueden migrar de una población a otra, como se muestra en la Figura 2.16.

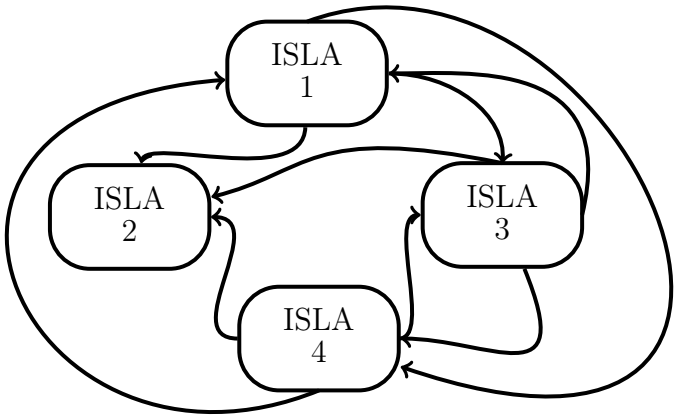


Figura 2.16: Esquema de la implementación del método de islas con migraciones de individuos de una isla hacia las demás.

Como ya se mencionó en el capítulo uno, el diseño del método de islas involucra dificultades relacionadas con la elección de sus cuatro principales parámetros: (1) el tamaño y número de islas, (2) la topología que comunica las islas, (3) la razón de migración que determina la cantidad de individuos que migra, (4) la frecuencia de la migración que determina que tan seguido las migraciones ocurren, y (5) la regla de migración que determina cuales individuos deben migrar y cuales son reemplazados en la isla que los recibe. Damos una reseña literaria de los trabajos más sobresalientes en esta área.
Bossert (1967) [33] probablemente fue el primero en proponer un algoritmo evolutivo con múltiples poblaciones para mejorar la calidad de las soluciones en un problema de optimización. Las poblaciones de Bossert compiten para sobrevivir entre sí.
Grefenstette(1981) [34] describe un algoritmo genético en paralelo con poblaciones múltiples. En su algoritmo, los mejores individuos de cada isla son transmitidos en cada generación a todas las demás.
Grosso (1985) [35] fue el primero que observó que la mejora de la razón de la función de adaptación fue rápida en una población dividida en islas que en una sola población. En sus experimentos las migraciones ocurren en cada generación y, los individuos que migran y sus destinos son seleccionados de manera aleatoria.
Braun (1990) [36] usa una idea similar a la de Grosso pero presentando un algoritmo donde la migración ocurre después que las islas convergen completamente (el usa el término "degenerado"). El propósito de la migración fue restaurar la diversidad en las islas para prevenir la convergencia prematura a una solución inferior.
Munetomo, takai, y Sato (1993) [37] realizaron una estrategia similar donde la migración se activa cuando la diversidad de la población cae por debajo de cierto nivel. En este caso, la topología de las comunicaciones no es fija: los individuos que migran son enviados a poblaciones con poca diversidad. Las topologías dinámicas siguen siendo una idea interesante y que se sigue estudiando [38].
Cantú Paz y Goldberg (1996) [39] asumen que el tiempo de computo es dominado por el tiempo usado para evaluar la función de adaptación e ignoran el tiempo usado en los operadores de los algoritmos genéticos, selección, recombinación y mutación. Llegan a la conclusión de que existe una ventaja moderada en usar el modelo de islas aisladas.
Tanese inició un estudio sobre la migración y sus efectos en la eficiencia y calidad de los algoritmos genéticos en paralelo [40] [41]. Sus experimentos consistieron en dividir un número fijo de individuos en islas de igual tamaño, y variar las razones de migración y los intervalos de migración. Ella encontró que la "aceleración" casi lineal puede ser obtenida cuando la migración no es

frecuente y la razón de migración es bajo.
Resumiendo, el método de islas en general puede implementarse de la siguiente manera:

1. Cada procesador tendrá una población de N individuos.
2. Se calcula el valor de adaptación de los individuos de cada procesador, sin considerar lo que se realice en los demás procesadores.
3. Se realizan los operadores de selección, recombinación y mutación en cada procesador.
4. Se revisa si el criterio de migración indica que la generación actual exista migración de individuos
5. Si 4 se cumple, entonces, se migran una cantidad de k individuos a las islas de manera aleatoria. Estos individuos son seleccionados de manera aleatoria.
6. Cada isla tiene su propio criterio de supervivencia para la siguiente generación, incluyendo si existe migración de individuos o no.
7. Se vuelve al punto 3 hasta que el criterio de terminación se cumpla.

en algunos textos los métodos de islas con diferentes cantidades de poblaciones tiene diferentes nombres, si la isla tiene una población grande, ese modelo es conocido como algoritmo de **grano grueso** y las islas con poblaciones pequeñas son conocidas como algoritmos de **grano fino**.

MPI

Para paralelizar el algoritmo genético utilizamos MPI, que es una interfaz de paso de mensajes, y nos basamos en el libro *Parallel programming with MPI* [42] para entender las funciones que MPI nos puede proporcionar.
Para usar MPI, cada programa debe de contener la directiva de preprocesador, *use mpi*. Antes de llamar a cualquier función de MPI, se debe de llamar a la función **call** $MPI\_INIT(ierr)$, y solo una vez, donde el parámetro $ierr$ es un argumento adicional en todas las subrutinas de Fortran 90. Después de que un programa haya terminado de usar la librería de MPI, se debe de llamar a la siguiente subrutine, *call* $MPI\_FINALIZE(ierr)$.
Como el control del programa depende del rango y número de procesadores debido a que cada intercambio de mensajes consiste de dos partes: los datos ha transmitir y el sobre del mensaje. El sobre de un mensaje contiene

1. El rango del receptor
2. El rango del transmisor
3. La etiqueta del mensaje
4. Un comunicador

MPI proporciona la siguiente función para obtener el rango del procesador con la llamada a la subrutina

```
call MPI_COMM_RANK(MPI_COMM_WOLD,rango,ierr)
```

donde el primer parámetro es el *comunicador*, que es una colección de procesadores que pueden interactuar con el programa y, por ahora, nuestro *comunicador* es MPI_COMM_WORLD que esta predefinido en todos los sistemas de MPI y consiste de todos los procesadores cuando la ejecución del programa empieza y el segundo es el rango del procesador.
Para encontrar cuantos procesadores están involucrados en un programa, llamamos la siguiente subrutina:

```
call MPI_COMM_SIZE(MPI_COMM_WOLD,np,ierr)
```

donde el segundo parámetro *np* indica el número de procesadores.
Los métodos más comunes de programación al momento de paralelizar un algoritmo es el intercambio de mensajes, o alguna variante de ellos. En el intercambio básico de mensajes, los procesadores coordinan sus actividades explícitamente por medio de enviar y recibir mensajes. Para esto, MPI proporciona una función de envío de mensajes:

```
int MPI_SEND(void* mensaje, int cantidad,
MPI_Datatype tipo de dato,int destino,
int etiqueta, MPI_COMM WORLD,ierr)
```

y una función para recibir mensajes:

```
int MPI_RECV(void* mensaje, int cantidad,
MPI_Datatype tipo de dato, int fuente,
int etiqueta, MPI_COMM_WORLD,
MPI_STATUS* estado,ierr)
```

donde los contenidos del mensaje son guardados en un bloque de memoria por el parámetro *mensaje*. Los siguientes dos parámetros, *cantidad*

y *tipo de dato*, permiten al sistema determina cuanta memoria se necesita para el mensaje: el mensaje contiene una secuencia de *cantidad* de valores, cada uno teniendo un tipo de dato de MPI llamado *tipo de dato*, para los cálculos los que más usamos son de tipo flotante por lo tanto se debe de usar $MPI\_DOUBLE\_PRECISION$ o MPI_FLOAT y para enteros MPI_INT. Los parámetros *destino* y *fuente* son, repectivamente, los rangos de los procesadores que envían y reciben. La *etiqueta* es solo un entero especificado por el programador que el sistema agrega al desarrollo del mensaje con el propósito de distinguir el mensaje de otros que se pueden enviar o recibir en las comunicaciones realizadas. El penúltimo parámetro de MPI_RECV, *estado*, regresa la información de los datos que son recibidos.

Capítulo 3

Problema de Thomson

3.1. Introducción

Como hemos observado a lo largo del capítulo anterior, existen varios parámetros en los algoritmos genéticos a determinar, como el tamaño de población, probabilidad de mutación y recombinación. Debido a las investigaciones realizadas por Schaffer [28] la búsqueda para determinar el valor de dichos parámetros es independiente del problema a tratar, por esta razón elegimos un problema simple e interesante para poder determinar estos valores, este problema se llama "*el problema de Thomson*" [11], también conocido como "el modelo del pudín de pasas".
Dicho problema consiste en determinar la localización de N electrones confinados en una esfera de manera que la configuración minimice la energía potencial electrostática.
En el algoritmo genético el gen esta dada por la posición de una carga en la esfera, por lo tanto, cada individuo tiene N genes (N cargas) y su función de adaptación esta dada por la energía potencial electrostática que resulta de la configuración de las N cargas. De este modo el individuo más apto de la población será el individuo que minimice la energía potencial electrostática a comparación del resto de los individuos en toda la población. Por consiguiente, se aplican los diferentes operadores de selección, recombinación y mutación con diferentes parámetros de los algoritmos genéticos
Los resultados obtenidos para determinar los parámetros son muy similares a los encontrados por [28] y [12].
Ya que hemos obtenido los parámetros de los algoritmos genéticos procedimos a paralelizar el problema de Thomson para observar las diferencias entre el modelo de Maestro-Esclavo y el modelo de islas. En estos parámetros utilizamos los recomendados por Tanese [40] [41], donde hacemos poco

frecuente la migraciones de individuos seleccionados de manera aleatoria y una probabilidad de migración baja.

3.2. Problema de Thomson

Como ya se menciono en la sección anterior, el problema de Thomson consiste en determinar la localización de N electrones confinados en una esfera de manera que la configuración minimice la energía potencial electrostática. Este problema fue propuesto por el físico J.J. Thomson (1904) [11] después de proponer un modelo atómico, que después fue conocido como el modelo del pudín de pasas.
La fuerza electrostática $\mathbf{F}$ que actúa en una carga q puede ser descrita en términos del campo eléctrico $\mathbf{E}$ como en la ecuación 3.1.

$$\mathbf{F} = q\mathbf{E}. \tag{3.1}$$

El cambio en la energía potencial electrostática, U_E, de una carga puntual q que se mueve de la posición de referencia $\mathbf{r}_{ref}$ a la posición $\mathbf{r}$ en la presencia de un campo eléctrico $\mathbf{E}$ es el negativo del trabajo realizado por la fuerza electrostática, ecuación 3.1, para traerla de la posición de referencia $\mathbf{r}_{ref}$ a esa posición $\mathbf{r}$. Es decir

$$U_E(r) - U_e(r_{ref}) = -W_{r_{ref}\to r} = -\int_{r_{ref}}^{r} q\mathbf{E}\cdot d\mathbf{s}. \tag{3.2}$$

donde $\mathbf{r}(x, y, z)$ es la posición en el espacio tridimensional de la carga q y tomando la posición de la carga Q en $\mathbf{r} = (0, 0, 0)$, el escalar $r = |\mathbf{r}|$ es la norma del vector de posición, $d\mathbf{s}$ es el vector diferencial de desplazamiento a lo largo del camino C que va de $\mathbf{r}_{ref}$ a $\mathbf{r}$, $W_{r_{ref}\to r}$ en la ecuación 3.2 es el trabajo realizado por la fuerza electrostática para traer la carga de la posición de referencia $\mathbf{r}_{ref}$ a $\mathbf{r}$. Mantenemos que U_E es igual a cero en el infinito, debido a que esa interacción es despreciable

$$U_E(r_{ref} = \infty) = 0 \tag{3.3}$$

Así, obtenemos

$$U_E(r) = -\int_{\infty}^{r} q\mathbf{E}\cdot d\mathbf{s} \tag{3.4}$$

Si ocurre que $\nabla \times \mathbf{E} = 0$, la integral de línea en la ecuación 3.4 no depende del camino C y solo depende de los puntos finales. Esto sucede en campos eléctricos estáticos. Así, en este caso el campo eléctrico es conservativo y podemos usar la ley de Coulomb.
Usando la ley de Coulomb, se sabe que la fuerza electrostática $\mathbf{E}$ y el campo eléctrico $\mathbf{E}$ creado por una carga puntual Q se dirigen radialmente desde Q, se sigue que $\mathbf{r}$ y $\mathbf{s}$ también se dirigen radialmente de Q, Entonces, $\mathbf{E}$ y $d\mathbf{s}$ deben de ser paralelos, esto es expresado como

$$\mathbf{E} \cdot d\mathbf{s} = |\mathbf{E}| \cdot |d\mathbf{s}| \cos(0) = Eds. \tag{3.5}$$

Usando la ley de Coulomb, el campo eléctrico esta dado por

$$|\mathbf{E}| = E = k_e \frac{Q}{s^2}, \tag{3.6}$$

donde k_e es la constante de Coulomb y la integral puede ser fácilmente evaluada, sustituyendo 3.6 en 3.5 y esta última sustituyendo en 3.4 se obtiene

$$U_E(r) = -\int_{\infty}^{r} q\mathbf{E} \cdot d\mathbf{s} = -\int_{\infty}^{r} k_e \frac{qQ}{s^2} ds = k_e \frac{qQ}{r}. \tag{3.7}$$

Entonces, tenemos que la energía potencial electrostática, U_E, de una carga puntual Q en la presencia de n cargas puntuales q_i tomando una separación infinita entre las cargas como el punto de referencia 3.3 esta dada por

$$U_E(r) = k_e Q \sum_{i=1}^{n} \frac{q_i}{r_i}. \tag{3.8}$$

Ahora, la energía potencial electrostática U_E almacenada en un sistema de N cargas $q_1, q_2, \cdots, q_N$ en posiciones $\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N$ respectivamente, es representada por la ecuación 3.9.

$$U_E = k_e \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j>i}}^{N} \frac{q_i q_j}{r_{ij}}, \tag{3.9}$$

donde $r_{ij} = |\mathbf{r}_{ij}|$. La ecuación 3.10 se puede intepretar como el trabajo que realiza el traer un conjunto de N cargas una por una desde muy lejos,

supongamos desde infinito.
Para el problema de Thomson, las cargas son iguales ya que son solamente electrones, es decir $q_i = q_j = e$, y sin pérdida de generalidad y para no obtener errores de redondeo o truncamiento en los cálculos, simplificamos las unidades como $e = 1$ y $k_e = 1$, y consideramos que las cargas están confinadas en una esfera unitaria. Así,la energía potencial electrostática de una configuración de N electrones confinados en una esfera se expresa por

$$U_E = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j>i}}^{N} \frac{1}{r_{ij}} \tag{3.10}$$

donde ahora $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ es la distancia entre cada par de electrones en la esfera definidos por los vectores $\mathbf{r}_i$ y $\mathbf{r}_j$, respectivamente en coordenadas esféricas. Recordando el cambio de coordenadas cartesianas a coordenadas esféricas esta descrita por las ecuaciones 3.11.

$$x = r\sin\theta\cos\phi \qquad y = r\sin\theta\sin\phi \qquad z = r\cos\theta \tag{3.11}$$

donde el radio r va de $r \in [0, \infty)$, el ángulo polar θ de $\theta \in [0, \pi]$ y el azimut ϕ de $\phi \in [0, 2\pi)$, estos puntos se utilizan para determinar la posición espacial de un punto mediante una distancia y dos ángulos.
Ahora el problema solo consiste en encontrar la localización de N cargas de manera que minimice la energía potencial electrostática dada por la ecuación 3.10.

Proyección estereográfica

Para obtener una visualización de la localización de las N cargas al final de la ejecución del programa, utilizamos la proyección estereográfica para localizar las cargas en un plano.
La proyección estereográfica representa una aplicación continua definida desde la esfera de Riemann hacia el plano (ya sea $\mathbb{R}^2$ o $\mathbb{C}$).

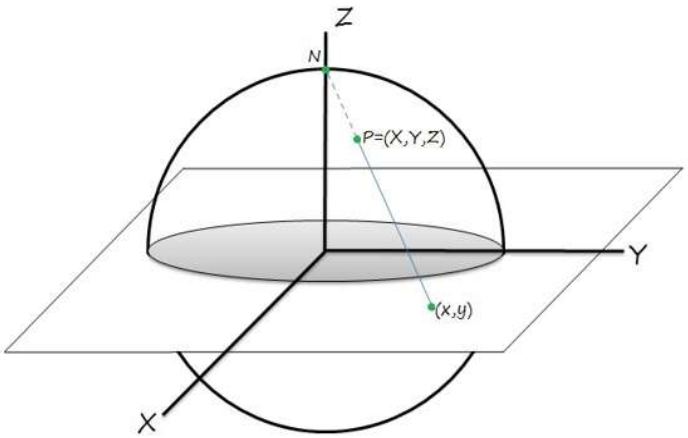


Figura 3.1: Proyección estereográfica

Como se observa en la Figura 3.1, se traza una recta desde el polo norte,N, de la esfera unitaria hasta un punto en el plano complejo (x, y). Esta recta intersecta a la esfera en un punto distinto del polo, y además del plano, este punto es $P = (X, Y, Z)$.
La esfera unitaria en el espacio tridimensional $\mathbb{R}^3$ es el conjunto de puntos (X, Y, Z) tal que $X^2 + Y^2 + Z^2 = 1$. Sea $N = (0, 0, 1)$ el *polo norte*, y establezcamos como M el resto de la esfera. El ecuador es la intersección de la esfera con este plano.
Para cualquier punto $\mathbf{P}$ en M, hay una única línea a través de N y $\mathbf{P}$. En coordenadas cartesianas (X, Y, Z) en la esfera y (x, y) en el plano, la proyección y su inversa están dadas por las fórmulas

$$(x, y) = \left(\frac{X}{1 - Z}, \frac{Y}{1 - Z}\right) \tag{3.12}$$

y

$$(X, Y, Z) = \left(\frac{2x}{1 + x^2 + y^2}, \frac{2y}{1 + x^2 + y^2}, \frac{-1 + x^2 + y^2}{1 + x^2 + y^2}\right). \tag{3.13}$$

La proyección no esta definida en el punto de proyección de $N = (0, 0, 1)$. Vecindades cercanas a este punto son mandadas a subconjuntos del plano muy alejadas del $(0, 0)$. Mientras más cerca este $\mathbf{P}$ de $(0, 0, 1)$, su imagen $(0, 0)$ del plano será muy distante. Por esta razón es común hablar de $(0, 0, 1)$ como el mapeo hacia el *infinito* en el plano.

3.3. Aplicación de algoritmos genéticos resolviendo el problema de Thomson

Para resolver el problema de Thomson por medio de algoritmos genéticos, el primer paso a realizar es implementar la conexión entre el problema de Thomson y los algoritmos genéticos, esto se logra por medio de una representación en valores visto en el capítulo 2.2. Ahora, como los individuos son las posibles soluciones al problema y son representados por un conjunto de parámetros llamados genes, los cuales agrupados forman un conjunto de valores (referidos como cromosomas), en este problema consideramos que cada individuo contiene solo un cromosoma. Por lo tanto, en nuestra representación en valores del algoritmo genético, un individuo es una esfera de radio 1 con N cargas colocadas sobre su superficie, los genes son las posiciones de cada carga y se caracterizan por dos números (θ, ϕ), cuyos rangos son $0 \leq \theta \leq \pi$ y $0 \leq \phi \leq 2\pi$, respectivamente. De este modo, el material genético de cada individuo consiste en $2N$ números $(\theta_1, \phi_1, \theta_2, \phi_2, \cdots, \theta_N, \phi_N)$.
Lo que prosigue es diseñar una función de adaptación que refleje el nivel de adaptación del individuo al problema de Thomson. Para hacer esto, utilizamos el error relativo dado por la ecuación 3.14.

$$\epsilon_r = \frac{|X_{real} - X_{calculo}|}{X_{real}} \tag{3.14}$$

donde X_{real} es el valor que minimiza la energía potencial electrostática en presencia de N cargas más preciso que se haya hecho hasta ahora, en caso de no saber dicho valor se procede a tomar el valor de la energía más pequeña y sustituirlo como el X_{real} en la ecuación 3.14, consulte [43, 44], y $X_{calculo}$ es el valor de la energía que presenta la configuración de cargas de cada individuo en el algoritmo genético.
En la fase de reproducción implementamos los cuatro operadores de selección vistos en el capítulo 2.3, que son: la selección por rueda de ruleta, la selección estocástica universal de muestras, la selección por rangos y la selección por torneos. El siguiente paso es la recombinación genética entre padres, donde el intercambio es realizado entre posiciones de electrones de los dos individuos seleccionados para crear el material genético del descendiente. Ya que se haya creado al hijo, este es sometido a un criterio de mutación y, en caso de que mute, los afectados serán las posiciones de las cargas en la esfera.
Ahora que ya tenemos nuestro algoritmo genético aplicado al problema de Thomson, debemos de encontrar la combinación más eficiente entre operadores y parámetros, es decir, la combinación que dé como resultado el individuo más apto a comparación de las configuraciones.

Por último paralelizamos el algoritmo genético con los dos modelos, modelo de Maestro-esclavo y el modelo de islas. Los resultados son presentados en la siguiente sección

3.4. Resultados y conclusiones

Implementamos cada uno de los operadores descritos en el capítulo 2.3 al problema de Thomson con 100 porque para calcular su energía electrostática potencial, ecuación 3.10, requiere un tiempo de cómputo considerable de 9.63×10^{-4} segundos y en cada uno de los cálculos realizados se tomaron los siguientes parámetros: la probabilidad de recombinación se mantuvo de 0.9 muy cercana a lo dicho por Grefenstette [27] y Schaffer [28], el número de individuos en la población se mantuvo de 1000 individuos debido a que el espacio de búsqueda de soluciones se encuentra en los reales, se necesita tomar más individuos para tener una mejor aproximación a una solución eficiente, el número de generaciones que establecimos es de 1000, para que de este modo se pueda observar como converge en las primeras etapas de la ejecución del programa y cual es su resultado al final de las iteraciones. El parámetro que hacemos variar es la probabilidad de mutación en un rango de $0.005 - 0.2$ tomando solo los valores de $0.005, 0.01, 0.05, 0.1, 0.2$ y, el error relativo que indicamos en las gráficas se refiere al error relativo dada por la ecuación 3.14.
Para cada configuración de operadores y parámetros nos limitamos a tomar el conjunto de valores que este más cerca de un promedio de cinco implementaciones del mismo programa, debido a la falta de tiempo consideramos que este número de ejecuciones es suficiente para obtener nuestro objetivo y se realizaron al menos 10000 cálculos.
La Figura 3.2 representa como evoluciona una población durante cada generación. En las primeras 100 generaciones se tiene una convergencia rápida y, depués de la iteración 600 aproximadamente el error deja de disminuir notablemente.

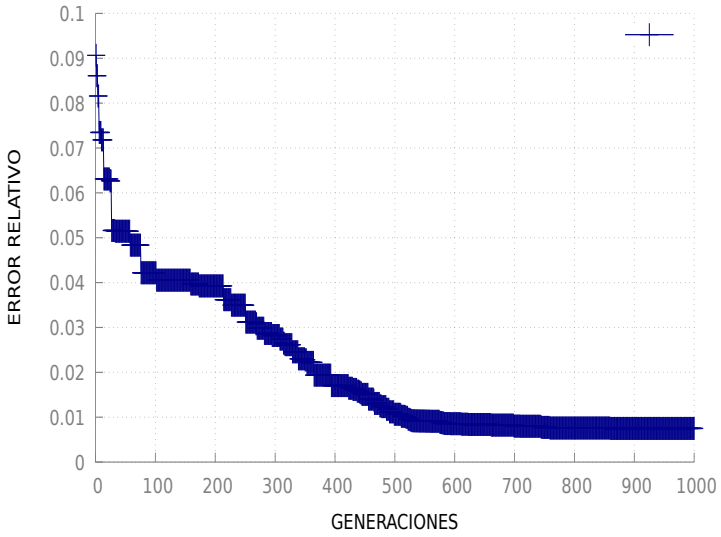


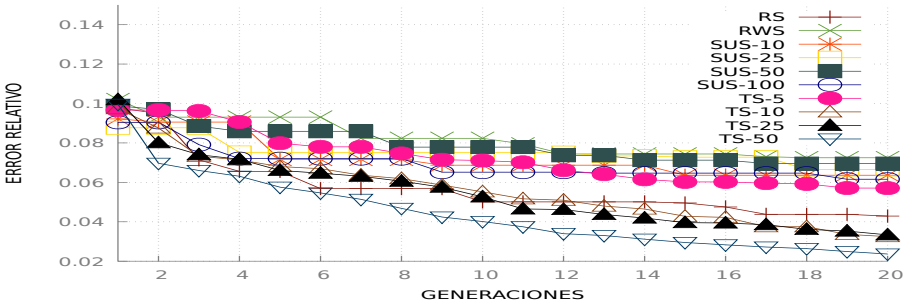
Figura 3.2: Mejor individuo de todas las posibles configuraciones previamente dichas en el capítulo 2

El mejor individuo resultó con la siguiente configuración:

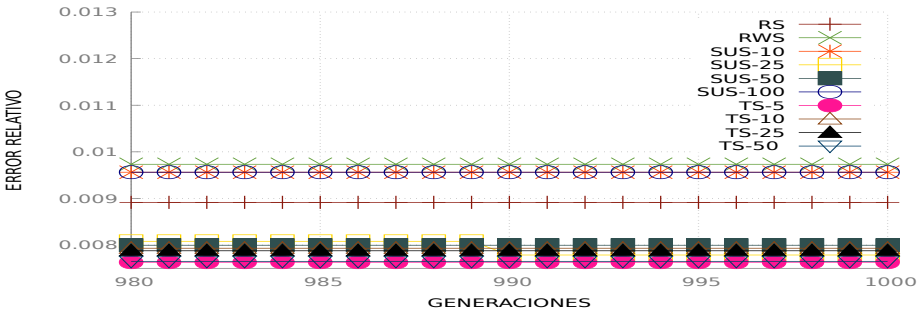
1. *Operador de selección*: selección por torneos con 5 participantes.
2. *Operador de recombinación*: recombinación de múltiples puntos, utilizando 100 puntos.
3. *Operador de mutación*: mutación de reajuste al azar.
4. *probabilidad de mutación*: 0.005

En nuestro análisis esta configuración es debido a que el número de participantes es pequeño y puede que algún individuo con un valor de su función de adaptación baja, pueda ser seleccionado como padre e integre diversidad a la población en la siguiente generación, la recombinación indica que el material genético estará compuesto de la siguiente forma, el primer gen del padre, el segundo de la madre, el tercero del padre, el cuarto de la madre, y así sucesivamente hasta completar la cadena de genes del cromosoma. La mutación en cada población es muy baja pero cuando ocurre solo se hace un reajuste a la cadena de genes del individuo, es decir, unos cuantos de sus genes mutan, dando la oportunidad de que si el individuo que muta es un gran candidato a sobrevivir, no se pierda por completo su material genético y de la oportunidad de poder mejor su adaptabilidad en generaciones posteriores.

Debido a que es difícil observar el cambio del error con respecto a las generaciones en cada iteración, se gráfica las primeras y últimas iteraciones.



(a) Primeras 20 generaciones

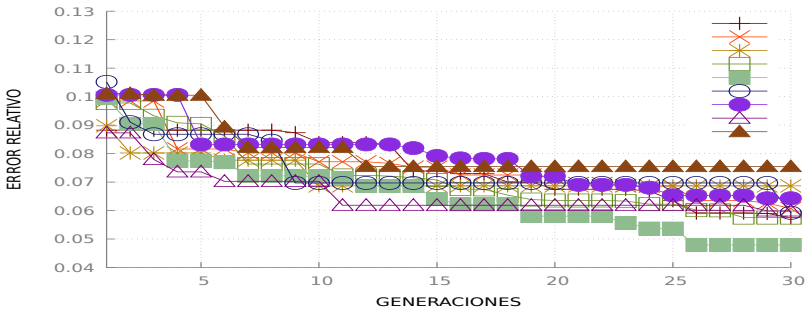


(b) Ultimas 20 generaciones

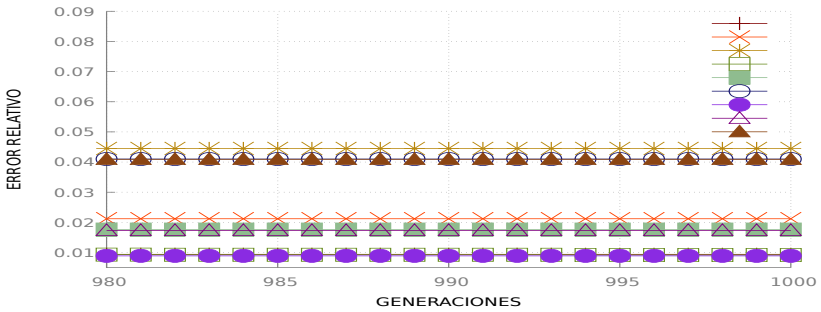
Figura 3.3: Mejores combinaciones de cada método de selección. $TS - k$: selección por torneos con k participantes, $SUS - K$: selección estocástica universal con k segmentos, RWS: selección por rueda de ruleta, RS: selección por rangos

En la Figura 3.3, representa las mejores configuraciones que se encontraron para cada método de selección. De la misma gráfica podemos observar que el error relativo entre las primeras generaciones y las últimas no es muy diferente, esto es debido a que a lo largo de las generaciones se va perdiendo la diversidad en las soluciones y, por lo tanto, existe una convergencia prematura para algunos métodos, como el método de rueda por ruleta y una ligera mejora en otros, como el método de selección por torneos. Esta falta de diversidad es común en los algoritmos genéticos ya que si aumentamos más la población en realidad no sería muy diferente al método de búsqueda aleatoria en el espacio de soluciones, por lo tanto, el algoritmo genético te aproxima a esta solución en un tiempo de computación más corto.
Ahora observemos como se comporta el método de recombinación de múl-

tiples puntos, para esto utilizamos el operador de selección por rangos, de hecho podría haber sido cualquiera, y se gráfica la mejor configuración de probabilidad de mutación a cada operador , nosotros solo observamos el cambio que sucede en $25, 50$ y 100 puntos. El resultado es presentado en la Figura 3.4



(a) Primeras 20 generaciones

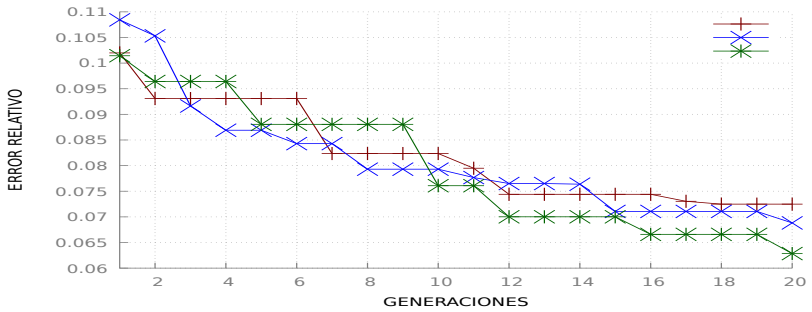


(b) Ultimas 20 generaciones

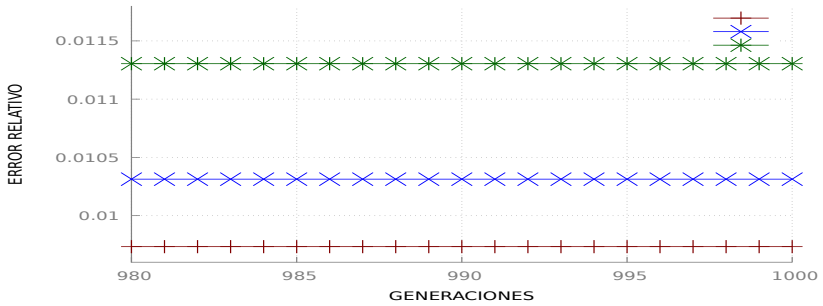
Figura 3.4: Las primeras tres líneas, de arriba hacia abajo, representan las gráficas de la recombinación de 25 puntos, las tres segundas de 50 puntos y las tres últimas a 100 puntos. En cada tercia se aplicaron los operados operadores de mutación de reajuste al azar, intercambio y uniforme, en ese orden.

En la parte (b) de la Figura 3.4, se puede observar que convergen en tercias. En la primera tercia, de arriba hacia abajo, corresponden a la mutación de reajuste al azar, en la segunda a la mutación por intercambio y a la tercera por mutación uniforme. En cada una, se puede observar un ligero cambio en la convergencia a la solución para cada elección de puntos en el operador de recombinación, donde el que sobresale siempre de los demás la implementación del operador con 100 puntos. En esta gráfica podemos también concluir que el mejor operador de mutación es el de reajuste al azar debido a que se aplica los tres operadores a cada método de recombinación y este resulta el

más eficiente.
Para convencernos de que este operador de mutación es el más eficiente utilizamos el operador de selección de rueda por ruleta y el operador de recombinación uniforme y grásficamos cada método de mutación con su mejor probabilidad de que un individuo mute. Así, obtenemos la siguientes gráficas mostradas en la Figura 3.5.



(a) Primeras 20 generaciones

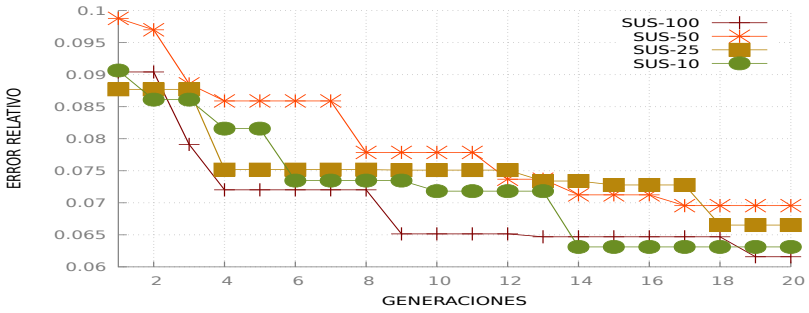


(b) Ultimas 20 generaciones

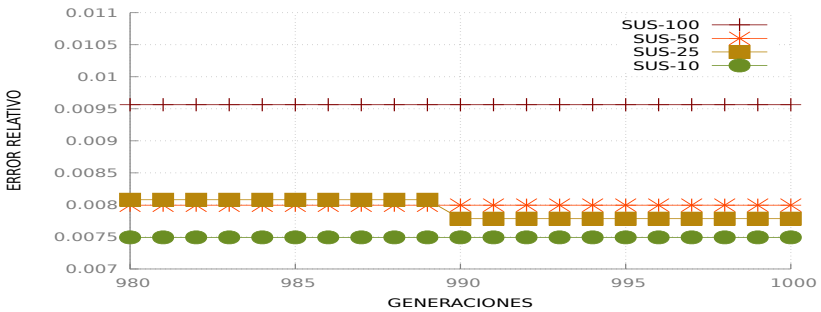
Figura 3.5: Mejor operador de mutación. La linea café muestra la mutación de reajuste al azar, la azul de intercambio y la verde uniforme.

En estas gráficas observamos que la mutación uniforme en las primeras generaciones converge más rápido que las demás pero en las últimas generaciones observamos que el reajuste al azar se vuelve el más eficiente. Por lo tanto, nos quedamos con el operador de mutación de reajuste al azar.
Sabemos que el método estocástico universal de muestras tiene un parámetro que varía, el cual es el número de segmentos o regiones. Para observar el cambio que produce la elección de dicha variación gráficamos la mejor configuración de cada caso, vea Figura 3.6.

Los resultados presentados en la Figura 3.6 sugieren que la mejor implementación para este método es con pocas regiones. Esto es debido a que mientras el número de regiones crezca este operador se parece más al método



(a) Primeras 20 generaciones

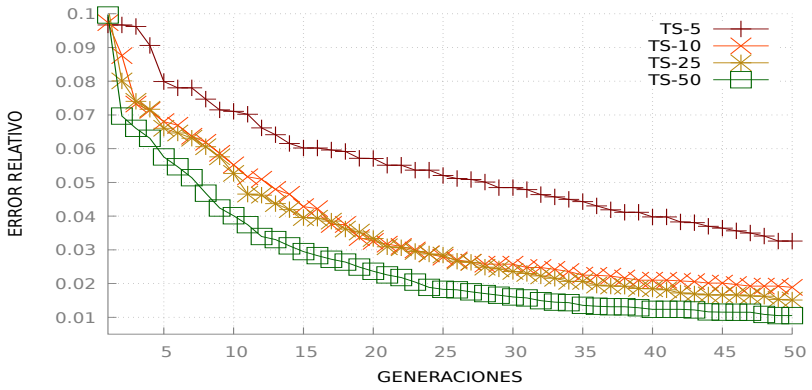


(b) Ultimas 20 generaciones

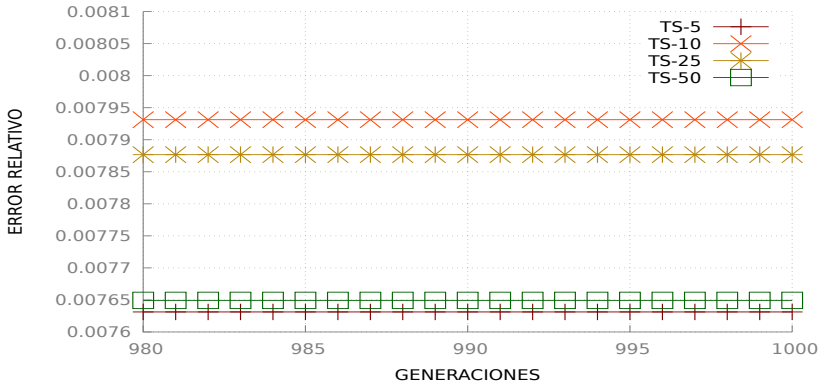
Figura 3.6: Selección estocástica universal de muestras.

por rangos, lo cual sabemos que solo es eficiente al final de las iteraciones, no al principio y, por lo tanto, no permite que los individuos más aptos sobresalgan de los demás y se pierda un buen material genético para la búsqeda de soluciones

Ahora veamos que cambio produce el cambiar el número de participantes en el operador de selección por torneos. Para visualizar esto, consideramos $5, 10, 25, 50$ participantes en dicho método, cada uno con su mejor configuración del resto de operadores y parámetros. El resultado esta presentado en las siguientes gráficas, vea Figura 3.7.



(a) Primeras 20 generaciones



(b) Ultimas 20 generaciones

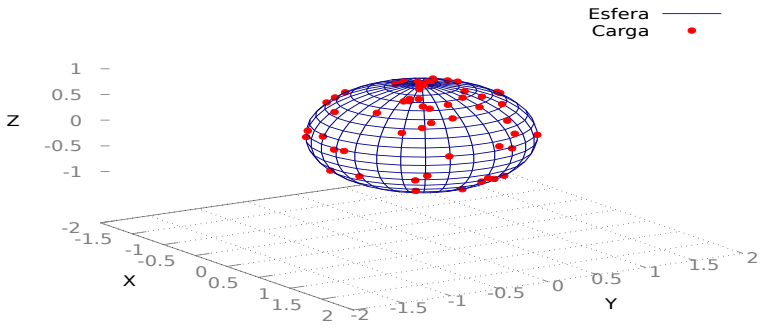
Figura 3.7: Selección por torneos.

Observamos que en las primeras generaciones para tener un resultado rápido y eficiente se debe de utilizar más participantes, esto es debido a que solo sobreviven los más aptos, sacrificando la diversidad de la población, por lo tanto, al final de la ejecución no converge. De este modo, la gráfica sugiere la implementación de 5 participantes, guardando así la diversidad

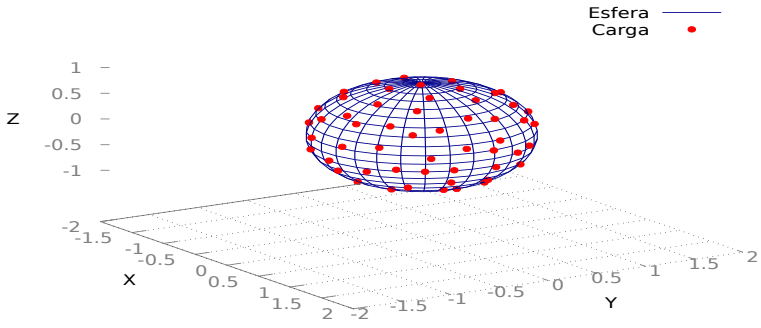
de la población y por consiguiente, se tiene una mejor convergencia en las últimas generaciones.

Visualización de la esfera con 100 cargas

Ahora que hemos obtenido la mejor configuración de operadores y parámetros que ofrece el algoritmo genético clásico, veamos la posición final de las cargas que minimizan la energía potencial electrostática para 100 carga en comparación con la posición del peor individuo de la primera generación, vea Figura 3.8



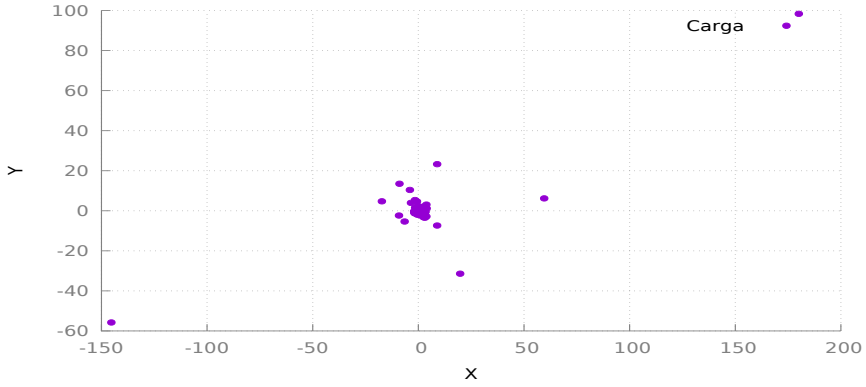
(a) Peor distribución de cargas de la primera generación



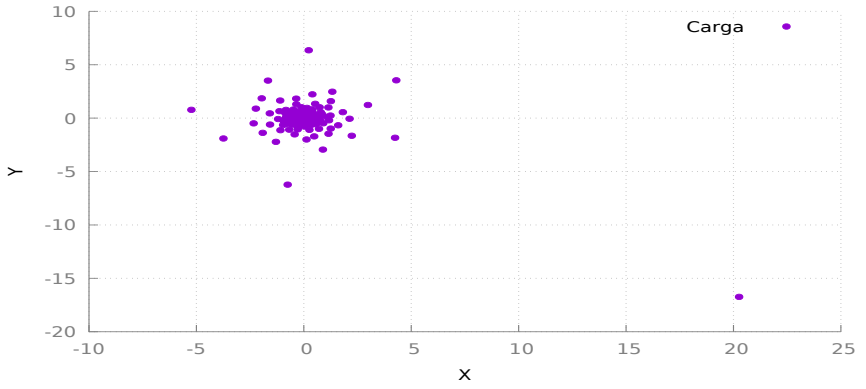
(b) Mejor distribución de cargas de la última generación

Figura 3.8: Representación de la localización de las 100 cargas del mejor individuo y del peor individuo que minimizan la energía potencial electrostática

Estos puntos en la representación de una proyección estereográfica dada por 3.12, se localizan en el plano de la siguiente forma, vea Figura 3.9



(a) Peor distribución de cargas de la primera generación



(b) Mejor distribución de cargas de la última generación

Figura 3.9: Representación de la localización de las 100 cargas del mejor y peor individuo que minimizan la energía potencial electrostática

Las configuraciones mínimas de energía han sido rigurosamente identificadas en un solo puñado de casos [43], [44]:

Para $N = 1$, la solución es trivial ya que el electrón puede residir en cualquier punto de la superficie de la esfera unitaria. La energía total de la configuración se define como cero, ya que el electrón no está sujeto al campo eléctrico debido a otras fuentes de carga.
Para $N = 2$, al configuración óptima consiste en electrones en puntos antipodales.
Para $N = 3$, los electrones residen en los vértices de un triángulo equilátero.
Para $N = 4$, los electrones residen en los vértices de un tetraedro regular.
Para $N = 5$, los electrones residen en los vértices de una dipirámide triangular.
Para $N = 6$, los electrones residen en los vértices de un octaedro regular.
Para $N = 12$, los electrones residen en los vértices de un icosaedro regular.

En particular $N = 4, 6$ Y 12 se conocen como sólidos platónicos cuyas caras son todos triángulos equiláteros congruentes. Las soluciones numéricas para $N = 8$ y 20 no son las configuraciones poliédricas convexas regulas de los restantes dos sólidos platónicos, cuyas caras son cuadradas y pentagonales, respectivamente.

Implementación en paralelo

Grefenstette [32] sugiere que la función de adaptación, debe de ser costosa en tiempo de cómputo en el modelo de Maestro Esclavo, esto es, aproximadamente, mayor que 1 segundo.
Para comprobar este resultado usamos la ecuaciones que indican la eficiencia de la implementación en paralelo del modelo Maestro-Esclavo, ecuaciones 2.7 y 2.8, después realizamos evaluaciones de computo pequeñas, es decir, menores a un segundo, esto se logra aplicando el problema de Thomson con 100 cargas donde el tiempo que tarda en evaluar la función de adaptación de un solo individuo es de alrededor de 9.63×10^{-4} segundos. Usamos $5, 10$ y 15 procesadores con tiempos de computo para realizar una comunicación entre Maestro-Esclavo de 2.323×10^{-2}, 5.226×10^{-2} y 8.492×10^{-2}, respectivamente.
Para el caso mayor que 1 segundo utilizamos 2572 cargas [54], y resulta que el tiempo de computo para evaluar la función de adaptación de un individuo es de 1.20779 segundos. En este caso, como debemos utilizar procesadores que dividan de manera entera a 2572, usamos $2, 4, 8$ y 12 esclavos que obtuvieron tiempos de computo para implementar una comunicación de 1.276×10^{-3},

4.832×10^{-3}, 1.013×10^{-2} y 1.667×10^{-2}, respectivamente. Comparamos en cada uno de los casos su eficiencia y el resultado, a partir de las ecuaciones de Cantú 2.7, 2.8, esta descrito en la Figura 3.10.

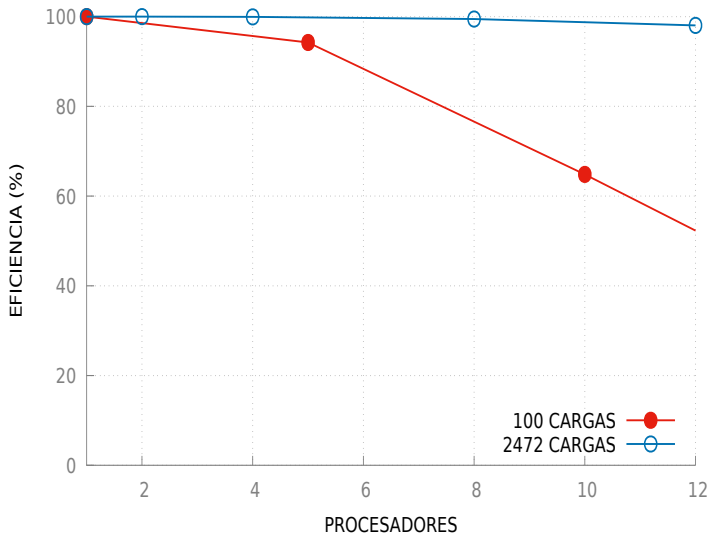
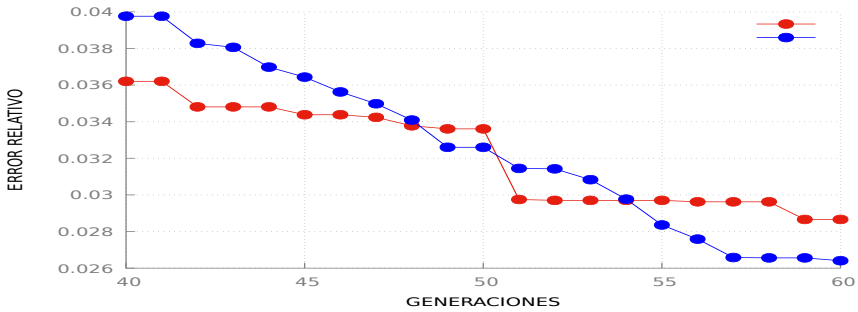


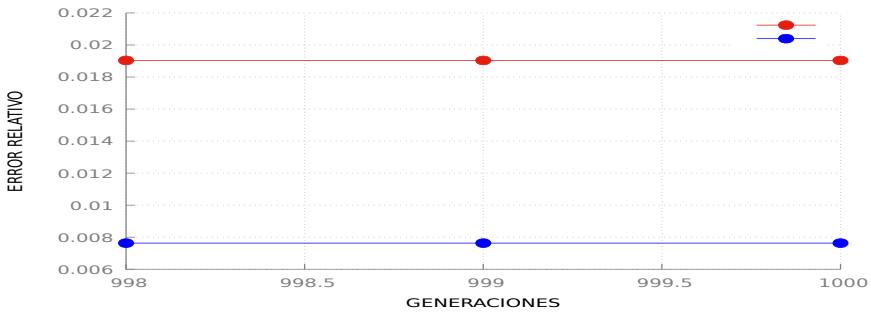
Figura 3.10: Comparación de la eficiencia entre valores de evaluación menores y mayores a 1.

La Figura 3.10, nos indica que la aceleración paralela es semejante al número de procesadores, es decir, si el tiempo que tarda en ejecutar el programa en un procesador es de 100 segundos, en dos procesadores el tiempo sería de 50 segundos, en cuatro de 25, y así sucesivamente. Esto casí no se logra debido a los costos entre las comunicaciones de Maestro-Esclavo. Nosotros obtuvimos una eficiencia por arriba del 98 % hasta 12 procesadores, esto es debido a que la comunicación entre Maestro-Esclavo es muy sencilla, ya que solo enviamos las configuraciones de los individuos y los esclavos calculan su función de adaptación, en otros programas se intercambia más información debido al problema a tratar. Cabe destacar que para cada procesador se implemento la configuración de los mejores operadores y parámetros que ofrece el algoritmo genético clásico.
Al implementar el modelo de islas basados en los parámetros dichos por [40], donde dejamos que el tamaño de la isla sea el número total de individuos dividido por el número de procesadores, con una frecuencia de migración cada 50 generaciones y con una cantidad de individuos que migran de 25. En la fase de migración cada isla manda sus 25 individuos a las demás islas y el criterio de supervivencia consiste en elegir a los mejores individuos de la isla más los individuos que migran y mantenerlos en la siguiente generación.

El caso que examinamos consistió de 1000 individuos en la población total, donde solo hemos utilizados 10 islas, donde en cada isla esta constituida con la mejor configuración de parámetros y operadores del algoritmo genético y lo comparamos con su versión sin paralelizar, vea Figura 3.11.



(a) En la generación 50 ocurre migración



(b) Últimas generaciones

Figura 3.11: Comparación de la mejor configuración de operadores y parámetros del algoritmo genético en su versión normal y utilizando el modelo de islas.

Cabe destacar que la gráfica (b) presentada en la Figura 3.11 es el error relativo de una sola isla, en nuestro caso consideramos la isla interpretada por el procesador 0. Como podemos observar en el punto donde ocurre la migración existe un "salto" a una mejor solución, esto es debido a que existe un individuo más apto para sobrevivir proveniente de otra isla, en la convergencia se muestra una gran diferencia, debido a que hace falta diversidad en las islas y a que puede existir que la configuración de un individuo pueda procrear uno más apto pero están en diferentes islas y estos no necesariamente contienen una función de fitness alto.
Consideramos que hace falta investigar más sobre el modelo de islas debido a que solo implementamos los datos dados por Tanese [41], empezando con la variación de sus parámetros y poder crear un híbrido entre los dos modelos

para observar su comportamiento tanto en eficiencia de tiempo de computo como en la solución.
En el siguiente capítulo utilizaremos los resultados obtenidos en este para la implementación de búsqueda de parámetros para la ecuación del oscilador forzado y la ecuación de onda.

Capítulo 4

Búsqueda de parámetros de ecuaciones diferenciales

4.1. Introducción

Las leyes del universo están escritas en el lenguaje de matemáticas. El álgebra es suficiente para resolver muchos problemas estáticos, pero la mayoría de los fenómenos naturales más interesantes, como la conducción de calor, la propagación de ondas, la dinámica molecular, el crecimiento de poblaciones biológicas, entre otros, involucran cambios descritos por las ecuaciones que relacionan las cantidades que los cambian, esto son las ecuaciones diferenciales en esencia.
En el estudio de las ecuaciones diferenciales se tienen tres metas principales:

1. Descubrir la ecuación diferencial que describe un fenómeno físico específico.

2. Encontrar de manera exacta o aproximada la solución de dicha ecuación.

3. Interpretar la solución encontrada.

Por ejemplo, el movimiento de una masa colgada en un resorte en presencia de fricción y que se le aplica una fuerza que varíe de manera sinusoidal esta descrito por el oscilador forzado. Otro ejemplo, para entender el comportamiento que hacen las cuerdas al momento de tocar una canción en nuestra guitarra es necesario resolver la ecuación de onda 1.4. Debido a la complejidad de obtener soluciones exactas de las ecuaciones mencionadas y principalmente en ecuaciones diferenciales no lineales [45] [10], se promueve el uso de métodos numéricos para resolver estas ecuaciones, entre los que

destacan los métodos de Runge-Kutta para la aproximación de ecuaciones diferenciales ordinarias y el método de líneas para ecuaciones diferenciales parciales.
Una de las preguntas que nos podemos hacer es, si te dan la solución de una ecuación diferencial, es decir, una serie de puntos que describen un fenómeno físico y varían con el tiempo, ¿Cuáles son los parámetros que de los depende dicha solución?. Los parámetros pueden ser las constantes que se encuentran en la ecuación que los describe junto con las condiciones iniciales y de frontera. En esta etapa es donde podemos implementar nuestro algoritmo genético para la búsqueda de estos parámetros. Donde ahora el individuo estará constituido de una configuración de parámetros que da la solución de la ecuación diferencial a tratar.

4.2. Oscilador forzado

El movimiento de una masa m unida a un resorte,con constante k, constante elástica, y a un amortiguador, es decir, la fuerza de frenado que proviene de la viscosidad o de las pérdidas aerodinámicas y se trata del caso más simple cuando la fuerza es proporcional a la velocidad con constante b, sobre la que además actúa una fuerza externa $F(t)$, esta dada por la ecuación diferencial 4.1.

$$mx'' + bx' + kx = F(t) \tag{4.1}$$

Las máquinas con componentes giratorios comúnmente involucran sistemas masa-resorte (o sus equivalentes), en los cuales la fuerza externa es armónica y de la forma

$$F(t) = F_0 \cos \omega t \qquad o \qquad F(t) = F_0 \sin \omega t \tag{4.2}$$

donde la constante F_0 es la amplitud de la fuerza periódica y ω es su frecuencia angular.

4.3. Ecuación de onda

El problema de la cuerda vibrante, como las que se encuentran en las guitarras, fue estudiado por Jean le Rond dÁlembert (1746), Leonhard (1748),

Daniel Bernoulli (1753) y Joseph-Louis Lagrange (1759) y la ecuación que describe este fenómeno estada dada por 1.4.

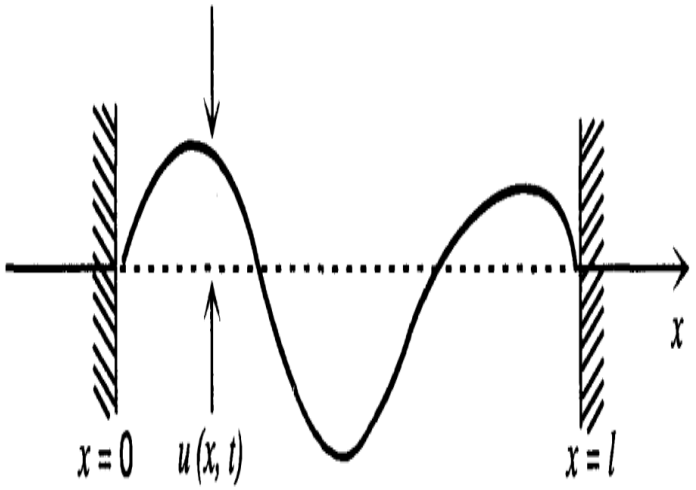


Figura 4.1: Cuerda vibrante [50]

Esta ecuación surge en casi cualquier análisis matemático de los fenómenos que comprenden la propagación de ondas en un medio continuo, como los estudios de ondas acústicas, ondas en el agua y ondas electromagnéticas todos se basan en esta ecuación.
La forma más fácil de visualizarla es en las vibraciones mecánicas. Suponemos que una cuerda elástica, digamos la cuerda de un violín, de longitud l se tensa con firmeza entre dos soportes al mismo nivel horizontal, de modo que el eje x quede a lo largo de la cuerda (vea Figura 4.1) y la cuerda se ponen en movimiento, es decir, jalamos la cuerda al momento de empezar a tocar el instrumento, de modo que vibre en un plano vertical y denotamos por $u(x,t)$ el desplazamiento vertical experimentado por la cuerda en el punto x, en el instante t. Despreciando los efectos de amortiguamiento, entonces $u(x,t)$ satisface la ecuación diferencial parcial dada por

$$v^2 u_{xx} = u_{tt} \tag{4.3}$$

en el dominio $0 < x < l$ $t > 0$, donde $u_{xx} = \partial^2 u/\partial x^2$ y $u_{tt} = \partial^2 u/\partial t^2$.
Cabe destacar que 1.4 es de manera general, y la ecuación 4.3 es una representación unidimensional que sirve para visualizar su significado.
Para describir por completo el movimiento de la cuerda, también es necesario

especificar condiciones iniciales y en la frontera. Se supone que los extremos permanecen fijos, y por lo tanto, las condiciones de frontera están establecidas por

$$u(0,t) = 0, \qquad u(l,t) = 0, \qquad t \geq 0 \tag{4.4}$$

En virtud de que la ecuación diferencial parcial 4.3 es de segundo orden con respecto a t, es necesario escribir dos condiciones iniciales; la posición inicial de la cuerda dada por

$$u(x,0) = f(x) \qquad 0 \leq x \leq l \tag{4.5}$$

y su velocidad inicial

$$u_t(x,0) = g(x) \qquad 0 \leq x \leq l \tag{4.6}$$

Así, el problema matemático es determinar la solución de la ecuación de onda 4.3 que satisfaga las condiciones de frontera 4.4 y las condiciones iniciales 4.5 y 4.6.

4.4. Método de Runge-Kutta

Como ya hemos mencionado anteriormente, los métodos de Runge-Kutta son un conjunto de métodos genéricos iterativos de resolución numérica de ecuaciones diferenciales ordinarias, concretamente, del problema de valor inicial, y toman su nombre de los matemáticos alemanes Carl Runge [48] y Wilhelm Kutta [47].
Los siguientes razonamientos están descritos de manera más detallada en [45]. Se presenta ahora un método para aproximar la solución $y = y(x)$ del problema de valor inicial, representado por la ecuación 4.7.

$$\frac{dy}{dx} = f(x,y), \qquad y(x_0) = y_0 \tag{4.7}$$

ahora suponemos que se han calculado las aproximaciones $y_1, y_2, \cdots, y_n$ para los valores reales $y(x_1), y(x_2), \cdots, y(x_n)$ y ahora se quiere calcular $y_{n+1} \approx y(x_{n+1})$ y, se escoge un *tamaño de paso* h fijo (horizontal) para utilizarlo en cada paso que se haga de un punto al siguiente. Suponga que se ha iniciado

en el punto (x_0, y_0) y después de n pasos se ha alcanzado el punto (x_n, y_n). Entonces, tenemos la ecuación 4.8.

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} y'(x)dx = \int_{x_n}^{x_n+h} y'(x)dx \tag{4.8}$$

por el teorema fundamental del calculo se llega a lo siguiente

$$y_{n+1} \approx y_n + \frac{h}{6}\left[y'(x_n) + 2y'\left(x_n + \frac{h}{2}\right) + 2y'\left(x_n + \frac{h}{2}\right) + y'(x_n + 1)\right] \tag{4.9}$$

las dos sumas parecidas dentro de los paréntesis en realidad son calculados con diferentes pendientes 4.11,4.12. En el lado derecho de 4.9 se sustituyen los valores de la pendiente (real) $y'(x_n)$, $y'(x_n + \frac{1}{2}h)$, $y'(x_{n+1} + \frac{1}{2}h)$ y $y'(x_{n+1})$, respectivamente, con las siguientes estimaciones, dadas por

$$k_1 = f(x_n, y_n) \tag{4.10}$$

Ésta es la pendiente del método de Euler [49] en x_n

$$k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \tag{4.11}$$

Esto es una estimación de la pendiente en el punto medio del intervalo $[x_n, x_{n+1}]$ utilizando el método de Euler para predecir la ordenada en ese punto.

$$k_3 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2) \tag{4.12}$$

Éste valor del método de Euler mejorado para la pendiente en el punto medio

$$k_4 = f(x_{n+1}, y_n + hk_3) \tag{4.13}$$

Ésta es la pendiente en el método de Euler en el punto x_{n+1} utilizando la pendiente mejorada k_3 en el punto medio para pasas a x_{n+1}
Cuando estas sustituciones se realizan en 4.9, el resultado es la fórmula iterativa representada por

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.14}$$

para la pendiente aproximada en el punto $[x_n, x_{n+1}]$. Se puede rescribir la ecuación 4.14 de la siguiente forma, ecuación4.15:

$$y_{n+1} = y_n + h \cdot k \tag{4.15}$$

si se escribe k como lo expresa la ecuación 4.16

$$k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.16}$$

El método 4.14 es el método de Runge-Kutta de cuarto orden, ya que su error acumulado es del orden de h^4

4.5. Método de líneas

Existen diversos métodos numéricos para la solución de ecuaciones diferenciales parciales, desde diferencias finitas, colocación ortogonal y elementos finitos [46].
Consideremos el problema de ecuaciones diferenciales parciales, representado por la ecuación 4.17.

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}) \qquad \mathbf{x}_L < \mathbf{x} < \mathbf{x}_R, \qquad t > 0 \tag{4.17}$$

donde $\mathbf{u}_t = \partial\mathbf{u}/\partial t$, $\mathbf{u}$ es el vector de las variables dependientes, t es el valor inicial de la variable independiente, $\mathbf{x}$ son los valores de frontera de las variables independientes, $\mathbf{f}$ es el operador diferencial espacial $\mathbf{f} = \mathbf{f}(\mathbf{x}, t, \mathbf{u}, \mathbf{u_x}, \mathbf{u_{xx}}, \cdots)$.
El *método de líneas* es una aproximación computacional para resolver problemas de ecuaciones diferenciales parciales de la forma de 4.17 y consiste de dos pasos:

1. las derivadas parciales, $\mathbf{u_x}, \mathbf{u_{xx}}, \cdots$, son aproximadas usando el método de *diferencias finitas* o *elementos finitos*

2. El sistema resultante de ecuaciones diferenciales semi-discretas en la variable de valor inicial se integra en el tiempo t.

Para ilustrar el método de líneas, consideremos la ecuación de aducción aproximada a una malla espacial en x de N puntos de malla separados uniformemente por una distancia Δx. que esta dada por:

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} \tag{4.18}$$

con la condición inicial

$$u(x, 0) = f(x) \tag{4.19}$$

Si la derivada espacial $\frac{\partial u}{\partial x}$ de 4.18 es reemplazada con una de segundo orden, centramos las diferencias finitas en el punto malla i

$$\frac{\partial u}{\partial x} = -v\frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2), \qquad i = 1, 2, \cdots, N \tag{4.20}$$

entonces la substitución de esta aproximación 4.20 en 4.18 con $v = 1$ da un sistema de N ecuaciones diferenciales ordinarias

$$\frac{du_i}{dt} = -\frac{u_{i+1} - u_{i-1}}{2\Delta x}, \qquad i = 1, 2, \cdots, N \tag{4.21}$$

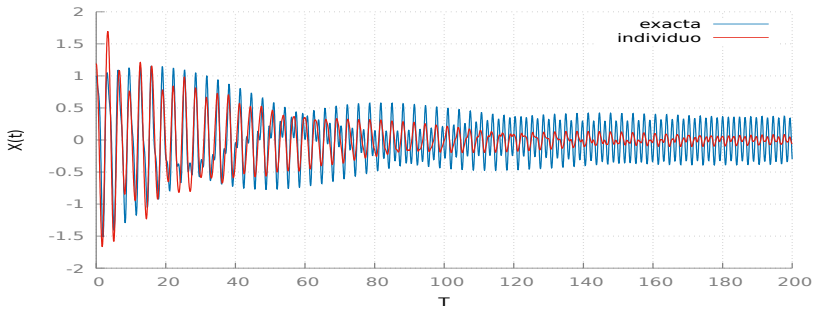
En el proceso la condición inicial 4.19 se debe de satisfacer en los N puntos de malla; también, u_0 y u_{N+1} son *puntos ficticios* (fuera del dominio espacial) que deben de ser incluidas en las ecuaciones diferenciales ordinarias para $i = 1$ e $i = N$.

4.6. Resultados y conclusiones

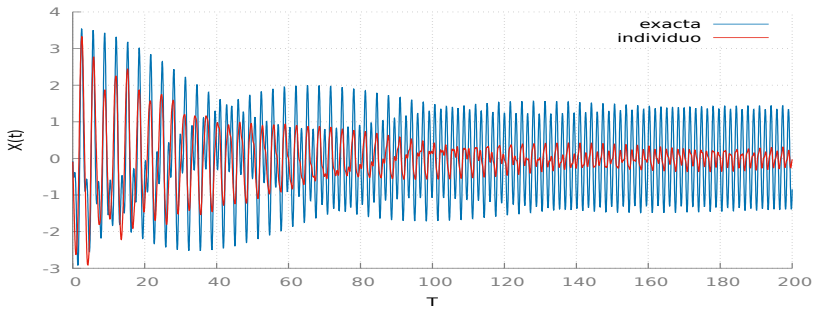
Como lo hicimos con el problema de Thomson en el capítulo anterior tenemos que encontrar la conexión entre el problema de búsqueda de parámetros con los algoritmos genéticos. En este caso, cada individuo será representado como una solución de la ecuación diferencial correspondiente con una configuración de parámetros, para el caso del oscilador forzado, el individuo es la solución de dicha ecuación usando el método de Runge-kutta de cuarto orden y para la ecuación de onda es la solución de esta ecuación usando el método de líneas, en cada uno se utilizaron 10000 puntos. En esta representación en valores, los genes del algoritmo genético son los parámetros de cada ecuación, ya que son los que hacen variar las soluciones de las ecuaciones diferenciales.

Para ambas ecuaciones los parámetros son las condiciones iniciales y las de frontera, más unas constantes que constituyen a cada ecuación, respectivamente y la función de adaptación esta dada por el error entre cada punto de la solución dada por un individuo con respecto al dado por el usuario, es decir, la solución que resulta con los parámetros escritos por el usuario. Como ambas ecuaciones son de segundo orden, primero calculamos el error con respecto a x y después con respecto a y y sumamos ambos errores y ese será el valor de adaptación del individuo, por lo tanto, el problema consiste en encontrar el individuo con la función de adaptación mínima.
Para el caso del oscilador forzado, los parámetros son la masa del resorte, la constante elástica k, la constante de amortiguación b y además de la fuerza externa dada por 4.2 con las constantes F_0 y ω. Para la ecuación de onda, es la velocidad con la que se propaga, v.
Como el problema fundamental es buscar los parámetros que satisfagan el conjunto de puntos espaciales-temporales, que son soluciones de las ecuaciones diferenciales correspondientes. Así, la función de adaptación será que tan bien se aproxima las soluciones que otorga un individuo a comparación de los puntos dados, esto se logra con el error relativo entre ambas. En el intercambio genético entre individuos los genes que se intercambian son los parámetros. Debido a que la función de adaptación toma un tiempo de computo cercano a 1 segundo, nos motiva a paralelizar dicho problema, por gusto, utilizamos el modelo de Maestro-Esclavo.
Para el oscilador forzado los puntos dados son la solución de la ecuación 4.1 con 4.2 por medio de un Runge-Kutta de cuarto orden. Donde los parámetros implementados son $m = 0.5$, $b = 0.02$, $k = 2.0$, $F_0 = 2.0$ y $\omega = 3.9$, con las condiciones iniciales de $x_0 = 1.0$, $y = 0.0$ y $t = 0.0$. En ese caso, no consideramos las condiciones de frontera. El rango de selección para cada uno de los parámetros es el siguiente, denotaremos por subíndice u los parámetros puestos por el usuario: $m \in (0, 2m_u)$, $b \in (0, 2b_u)$, $k \in (0, 2k_u)$, $F0 \in (0, 2f0_u)$, $\omega \in (0, \omega_u)$, $x_0 \in (0, 5(x_0)_u)$, $y_0 \in (0, 2(y_0)_u)$ y $t_0 \in (0, 2(t_0)_u)$. En la Figura 4.2 se muestra la gráfica del mejor individuo al final de 100 generaciones con 1000 individuos junto con la solución del oscilador forzado.

Donde la gráfica azul representa la solución del oscilador forzado con los parámetros dados por el usuario.
Los parámetros del mejor usuario son: $m = 0.18$, $b = 0{,}032$, $k = 1{,}73$, $F_0 = 3{,}30$, $\omega = 6{,}52$, $x_0 = 4{,}13$, $y_0 = 0$ y $t_0 = 0$. Como podemos observar existen unos parámetros que se aproximan demasiado a las dadas por el usuario como es el caso de y_0, t_0, b y k, aunque la gráfica de dicho individuo que se presenta en la Figura 4.2 parece que no se acerca a la gráfica dada por el usuario, esto se debe existen parámetros como ω y x que afectan a la solución de la ecuación debido a que se encuentran lejos de los parámetros



(a) Diagrama de posición $y(t)$ con respecto al tiempo t.

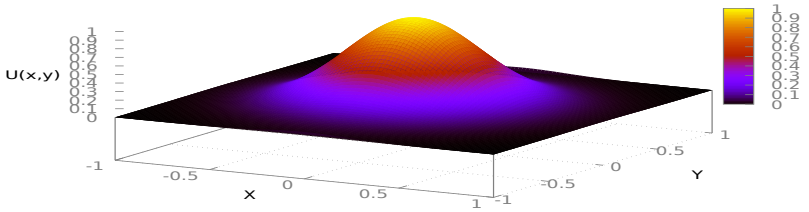


(b) Diagrama de posición $y(t)$ con respecto al tiempo t

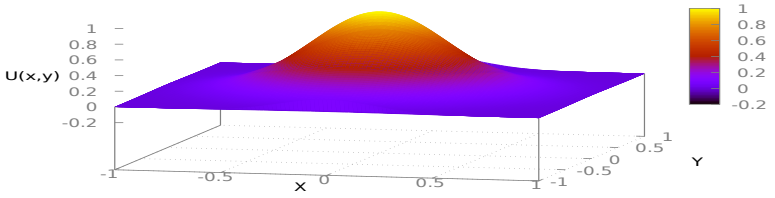
Figura 4.2: Comparación de la solución con los parámetros dados por el usuario, en azul, con respecto al mejor individuo, en rojo.

del usuario.

En nuestra ecuación de onda suponemos que $U(x, y, t) = e^{-\frac{-(x^2+y^2)}{\gamma}}$, donde $\gamma = 0.2$, $x \in (-1, 1)$, $y \in (-1, 1)$, por parte del usuario, vea Figura 4.3. Ahora el rango de parámetros para la ecuación de onda están dados de este modo: $x \in (-2, 2)$, $y \in (-2, 2)$ y $\gamma \in (0, 1)$, es decir, menos parámetros que los dados por la ecuación del oscilador forzado. En la Figura 4.4 se muestra un poco la evolución con respecto al tiempo de la solución de la ecuación de onda dada por el usuario



(a) En t=0

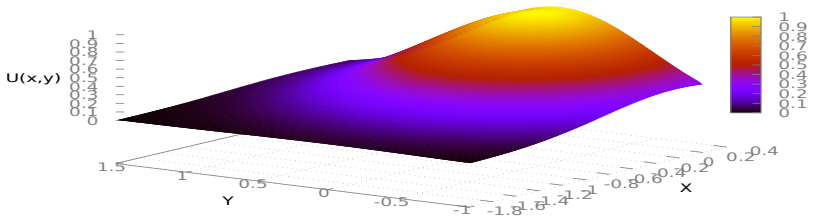


(b) En t=0.1seg

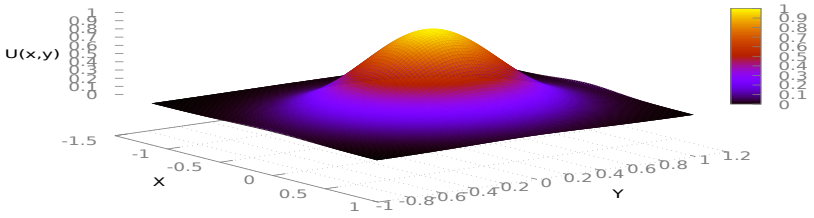
Figura 4.3: Evolución de la solución de la ecuación de onda con parámetros dados por el usuario

Para la búsqueda de parámetros de la solución de la ecuación de onda, realizamos 1000 generaciones con 1000 individuos. En la Figura 4.4 (a) representa el individuo menos apto de la primera generación mientras que en la parte (b) el mejor individuo de la última generación.

Los parámetros del mejor individuo son $x \in (-1{,}183, 0{,}959)$, $y \in (-0{,}992, 1{,}003)$ y $\gamma = 0{,}278$, los parámetros son demasiado parecidos a los dados por el usuario. Como podemos observar en la Figura 4.4 las gráficas están muy parecidas ya que hemos implementado pocos parámetros para los algoritmos genéticos



(a) Individuo con la función de adaptación más baja de la primera generación



(b) El individuo más apto de la última generación

Figura 4.4: Implementación de la búsquea de parámetros a la ecuación de onda

y realizado muchas iteraciones. Nuestros resultados para la búsqueda de parámetros indican que se debe de utilizar un número grande de generaciones e individuos debido a que estamos trabajando con número reales y por consecuente más tiempo de cómput, la ventaja es que podemos usar un método de paralelización de los ya vistos, si no sabemos de manera a priori el intervalo donde se puede encontrar uno o varios parámetros, se debe de realizar varias ejecuciones del algoritmo con diferentes intervalo y, generaciones e individuos pequeños, aproximadamente de 100 generaciones e individuos, si se observa que el error relativo se mantiene arriba del 10 % con el individuo más apto, se descarta ese intervalo y se procede a calcular otro hasta que dicho individuo este por debajo de ese porcentaje. Una vez localizado el intervalo, se procede a ejecutar más generaciones con más individuos. En el siguiente capítulo discutiremos los resultados obtenidos en el capítulo 3 y 4 de la tesis de licenciatura y futuros trabajos.

Capítulo 5

Conclusiones y futuros trabajos

Futuro trabajos

En este trabajo de tesis de licenciatura hemos afirmado que los algoritmos geéticos pueden ser una herramienta poderosa para resolver problemas de optimización, búsqueda y para simular sistemas naturales, como lo hemos observado en el problema de Thomson y la búsqueda de parámetros de ecuaciones diferenciales realizados en paralelo.
Los resultados obtenidos en el problema de Thomson muestran como varía el individuo más apto de cada generación con respecto a diferentes configuraciones de operadores y paámetros que constituyen al algoritmo genético, de este modo, se obtuvo la mejor configuración que de el individuo más apto de todas las generaciones con respecto a las demás posibles configuraciones. La ventaja más sobresaliente del algoritmo genético es que se puede paralelizar y esto hace que se ahorre demasiado tiempo de cómputo y, de este modo, no frustre al programador en la espera de sus resultados, la desventaja es que solo es recomendable paralelizar cuando la función de adaptación es mayor a 1 segundo, esto quiere decir que para problemas fáciles no se debe de paralelizar el algoritmo.
Los resultados presentados para la búsqueda de parámetros de ecuaciones diferenciales indican que se debe de implementar el algoritmo genético con una gran cantidad de generaciones e individuos, debido a la cantidad de parámetros que puede presentar una ecuación diferencial. En caso de no saber de manera a priori la posible ubicación de los parámetros se procede a realizar una búsqueda de intervalos donde se encuentre el parámetro buscado. Esto puede ser tedioso ya que estamos trabajando con números reales y el programador puede llegar a desesperarse por no encontrar una solución más eficiente, por lo tanto, se debe de implementar un algoritmo genético que pueda distinguir cuales intervalos son los más adecuados y reducir el tamaño

de dicho intervalo, ya que si trabajamos con intervalos grandes el tiempo de cómputo que necesita el algoritmo para encontrar una solución aceptable sería muy grande sin paralelizar.
En general se ha observado que el algoritmo genético solo se aproxima al valor observado debido a que el espacio de búsqueda de soluciones es bastante grande y para tener una mayor precisión al resultado esperado se debe de incrementar la diversidad de los individuos mediante un incremento en la población, pero esto ocasiona a que se realice más tiempo de cómputo, por lo tanto, es importante mantener un criterio para decidir hasta que punto la solución presentada por el algoritmo genético es aceptable.
Hemos observado que existe mucho trabajo que realizar en esta área, por lo que un trabajo a futuro, y no tan distante, es crear un algoritmo genético que cambie sus parámetros y operadores con el tiempo para crear soluciones más eficientes, de este modo, se tendría un algoritmo que tenga una gran semejanza con la naturaleza, uno de los pioneros para esta implementación es el operador de selección de Boltzmann [51]. También realizar una búsqueda intensiva de los mejores parámetros que constituyen al modelo de islas para después poder crear un híbrido entre los dos métodos de paralelización para obtener un modelo de paralización que pueda tener una eficiencia más grande para valores de la función de adaptación que duren menos de 1 segundo. Un caso interesante que se puede implementar en el modelo de islas sería, ¿ Que pasa si alguna de las islas esta sometida a una fuerte radiación?, es decir, £‘Que pasa si en una sola isla la probabilidad de mutación es cerca al 100 %. Otra de las ideas para poder mejor el algoritmo genético es buscar características en la naturaleza que proporcionen información evolutiva, como el tamaño de la población no se mantenga física, la exposición a radiación en una población que haga que muten los individuos en una isla por ejemplo, etc. Una vez hecho esto, podemos aplicarlo a problemas muchos más complejos y de nuestro interés como los trabajos realizados en modelos de energía oscura [52] y en la búsqueda de parámetros de galaxias interactuantes [53]. Como hemos observado los algoritmos genéticos pueden aparecer en una infinidad de áreas, como la economía en la optimización de gastos de las empresas, en biología para buscar los parámetros necesarios para que un ecosistema se mantenga estable, en la inteligencia artificial para la evolución de computadoras, entre otras. Es una área interesante y que vale la pena investigar sobre las posibles aplicaciones que podemos realizar.

Bibliografía

[1] Holland, John Henry. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[2] Mitchell, Melanie. *An introduction to genetic algorithms*. MIT press, 1998.

[3] Bäck, Thomas, and Hans-Paul-Schwefel. "*An overview of evolutionary algorithms for parameter optimization*". Evolutionary computation 1.1 (1993):1-23.

[4] Lobo, Fernando G., CláudioF. Lima, and Hugo Mártires. "*An architecture for massive parallelization of the compact genetic algorith.*"Gentic and Evolutionary Computation Conference, Springer, Berlín, Heidelberg, 2004.

[5] Addoun, Otman, and Jaafar Abouchabaka. "*A comparative study of adaptive crossover operators for genetic algorithms to resolver the traveling salesman problem*". arXiv preprint arXiv:1203.3097 (2012)

[6] Cantú-Paz, Erick. Efficient and accurate parallel genetic algorithms. Vol. 1. Springer Science & Business Media, 2000.

[7] Koza,John R. "*Genetic programming as a menas for programming computers by natural selection*". Statics and computing 4,2 (1994): 87-122.

[8] Darwin, Charles. On the origin of species, 1859. Routledge, 2004.

[9] Gutowski, Marek W. "*Amazing geometry of genetic space or are genetic algorithms convergent?*". asXiv preprint cs/0512019 (2005).

[10] Zill, Dennis G., and Warren S. Wright. *Diferential equations with boundary value problems*. Cengage learning, 2012.

[11] Thomson, Joseph John. XXIV. *On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure*. The London, Edinburgh, and DUblin Philosophical Magazine and Journal of Science, 1904, vol 7. no 39, p. 237-265.

[12] Golberg, David E. "*Genetic algorithms in search, optimization, and machine learning.*.Addion Wesley. Reading (1989)

[13] Abdoun, Otman, and Jaafar abouchabaka, "*A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem*". arXiv preprint arXiv:1203.2097 (2012)

[14] Spears, W.M., & De Jong, K.A. (1991). "*An analysis of multipoint crossover. In Foundations of genetic algorithms*". Elsevier, 1991, p, 301 − 315.

[15] Kumar, Rakesh. "**Blending roulette wheel selection** & *rank selection in genetic algorithms*". INternational Journal of Machine Learning and Computing 2.4 (2012): 365.

[16] Grefenstette, John, et al. "*Genetic algorithms for the traveling salesman problem.*"Proceedings of the first International Conference on Genetic Algorithms and their Applications. 1985

[17] McCulloch, Warren S., and Walter Pitts. "*A logical calculus of the ideas immanent in nervous activity.*"The bulletin of mathematical biophysics 5.4 (1943): 115-133.

[18] Kumar, R. (2012). "*Novel encoding scheme in genetic algoriths for better fitness*". International Journal of Engineering and Advanced Technology, 1(6), 214 − 219.

[19] Tuson, Andrew L., and P. Ross "*Adapting operator probabilities in genetic algorithms.*"Master's thesis, Department of Artificial Intelligence, University of Edinburgh (1995)

[20] Syswerda, Gilbert. "*Uniform crossover in genetic algorithms*.En Proceedings of the third international conference on Genetic algorithms. Morgan Kaufmann Publishers, 1989.

[21] Knuth, Donald E. "*JOhan Faulhaber and sums of powers.*"Mathematics of Computation 61,203 (1993) : 277 − 294.

[22] Cormen, Thomas H., et al. *Introduction to algorithms*. MIT press, 2009.

[23] Baker, James E. "*Reducing bias and inefficiency in the selection algorithm.*"Proceedings of the second international conference on genetic algorithms. Vol. 206,1987.

[24] Davis, Lawrence "*Applying adaptive algorithms to epistatic domains.*.[En] IJCAI. 1985. p. 162 – 164.

[25] Soni, Nitasha, and Tapas Kumar "*Study of various mutation operators in genetic algorithms.*Ïnternational Journal of COmputer Science and information Technologies 5,3 (2014): 4519 – 4521.

[26] De Jong KA. "*Analysis of the behavior of a class of genetic adaptive systems.*"1975

[27] Grefenstette J.J., 1986 "*Optimization of control parameters for genetic algorithms*"$IEEE$ Transactions on systems, man, and cyernetics, 16(1), 122-128.

[28] Schaffer, J. "*A study of control parameters affecting online performance of genetic algorithms for function optimization.*"San Meteo, California (1989).

[29] Davis. L. "*Adapting operator probabilities in genetic algorithms.*ÏnProc. Erd INternational Conference on Genetic ALgorithms 1989.

[30] Bethke AD. "*Comparison of genetic algorithms and gradient-based optimizers on parallel processors: Efficiency of use of processing capacity.*"(1976).

[31] Huang R, Fogarty TC. "*Adaptive classifcation and control-rule optimisation via a learning algorithm for controlling a dynamic system.*ÏnDecision and Control, 1991., Proceedings of the 30th IEEE Conference on 1991 Dec 11 (pp. 867 – 868). IEEE.

[32] Grefenstette, J.J., 1995. "*Robot learning with parallel genetic algorithms on netweorked computers.*În SUMMER COMPUTER SIMLUATION CONFERENCE, pp. 352 – 360. SOCIETY FOR COMPUTER SIMULATION, ETC.

[33] Bossert, W. "*Mathematical optimization: Are there abstract limits on natural selection?.*"The Wistar Institute symposium monograph. Vol. 5. 1967

[34] Grefenstette, J.J., 1981. "*Parallel Adaptive Algorithms for function optimization*:(preliminary Report).Çomputer Sciencie Department, Vanderbilt University.

[35] Grosso, P., 1985. "*Computer simulations of genetic adaptation: Parallel subcomponent interaction in multilocus model.*"Ph. D. Dissertation, University of Michigan.

[36] Braun, H., 1990, October. "*On solving travelling salesman problems by genetic algorithms.*Ïn International Conference on Parallel Problem Solving from Nature (pp. 129 − 133). Springer, Berlin, Heidelberg.

[37] Munetomo, M., Takai, Y. and Sato, Y., 1993, June. "*An efficient migration scheme for subpopulation base asynchronously parallel genetic algorithms.*Ïn Proceedings of the 5th International COnference on Genetic Algorith,s (p. (649)). Morgan Kaufmann Publishers Inc.

[38] Nowostawski M. Poli R. "*Dynamic demes parallel genetic algorithm. InKnowledge-Based Intelligent Information Engineering Systems*", 1999. Third International Conference 1999 Dec (pp. 93 − 98). *IEEE*.

[39] Cantu-Paz, E. and Goldber, D.E., 1996. "*Predicting speedups of idealized bounding cases of parallel genetic algorithms.*Ïn *In Back*, T.(Ed.), Proceedings of the Seventh INternational Conference on Genetic ALgorithms (pp. 113 − 121).

[40] Tanese, R., 1989. "*Distributed genetic algorithms for function optimization*"

[41] Tanese, R., 1987. "*Parallel genetic algorithm for a hypercube.*Ïn Genetic algorithms and theri applications: proceedings of the second international Conference on Genetic Algorithms: July 28 − 31, 1987 at the Massachussets Institute of Technology, Cambridge, MA. Hillsdale, NJ: L. Erlhaum Associates, 1987.

[42] Pacheco PS. "*Parellel proamming with MPI.*"Morgan Kaufmann

[43] Yudin, V.A., 1993. "*The minimun of potential energy of a system of point charges*". Discrete Mathematics and Applications, 3(1), pp. 75 − 82.

[44] Schwartz, R.E., 2013. "*The five electron case of Thomson problem.*.Experimental Mathematics, 22(2). PP. 157 − 186.

[45] Edwards. C.H., Penny, D.E. and Calvis, D.T., 2016 "*Differential equations and boundary value problems.*"Pearson Education Limited; 2016.

[46] Villadsen, J. and Michelsen, M.L., 1978. "*Solution of differential equation models by polynomial approximation.*"Prentice-Hall.

[47] Kutta W. "Beitrag zur näheringweisen Integration totaler Differentialgleichungen".

[48] Runge, C., 1895 "Über die numerische Ausflösung von Differentialgleichungen". Mathematische Annalen 46, no. 2 (1895) : 167 − 178.

[49] Euler, Leonhard. "*Institutionum calculi integralis*, 1768."

[50] Boyce, W.E., DiPrima,R.C. and Meadde, D.B., 1992. "*Elementary differential equations and boundary value problems* (Vo. 9)."New York: Wiley.

[51] Lee, C.Y., 2003. "*Entropy-Boltzmann selection in the genetic algorithms*". IEEE Transactions on SYstems,Man, and Cybernetics, Part B (Cybernetics), 33(1), *pp*,138 − 149.

[52] Nesseris. S. and García-Bellido, J., 2012. "*A new perspective on Dark Energy modeling via Genetic Algorithms*". Journal of Cosmology and Astroparicle Physics 2012, no. 11 (2012) : 033.

[53] Wahde, m., 1998. "*Determination of orbital parameters of interacting galaxies using genetic algorithm-Description of the method and application to artificial data*". Astronomy and Astrophysics SUpplement Series, 132(3), pp. 417 − 429.

[54] Altschuler, E.L., Williams, T.J., Ratner, E.R., Tipton, R., Stong, R., Dowla, F. and Wooten, F. 1997. "*Possible global minimum lattice configurations for Thomson's problem of charges on a sphere.*"Physical Review Letters, 78(14), p.2681.

[55] Karp, R.M., 1972. "*Reducibility among combinatorial problems.*ÏnComplexity of computer computations 1972 (pp. 85 − 103). Springer, Boston, MA.

[56] Flood, M.M., 1956. "*The traveling salesman problem..*°perations Research, 4(1), pp. 61 − 75