Universidad Nacional Autónoma de México.
Facultad de Estudios Superiores Acatlán.

Algoritmos genéticos para la optimización global de problemas petroleros.

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

Sócrates Ordaz Castillo
Asesor: Dra. Susana Gómez G.
FES-Acatlán.

4 de junio de 2006

Agradecimientos:

A mis padres **Juan Ordaz R. y Francisca Ortiz C.**; porque siempre han sido y seran mis mejores e incomparables maestros de toda la vida.

A mis **hermanos** porque siempre confiaron en mí; gracias por su ayuda y comprensión.

A la **Facultad de Estudios Superiores Acatlán**, por que sin los conocimientos ahí adquiridos no fuera posible la realización de este trabajo de investigación.

Al **Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas**, por brindarme los recursos necesarios para la realización de este trabajo.

A la **Dra. Susana Gómez** por su incanzable asesoría y dedicación en la realización de este trabajo.

A los profesores **Fís. Esperanza Georgina Valdés Medina, Mtra. María del Carmen Villar Patiño, Ing. Gabriel Díaz Mirón Mac Donough, Dr. Angel López Gómez**, gracias por su paciencia y dedicación en la revisión de este trabajo.

Al **ing. Carlos Paez** de la Secretaría de Hacienda y Crédito Público, porque siempre me motivó en la realización de este trabajo.

A los **ings. Rafael Barrera M. y Evaristo Pineda F.** de la Universidad Atónoma Chapingo.

A **Dios** por mantenerme con vida y estar siempre a mi lado.

Dedicatoria:

A **Rosalba** y nuestra familia, gracias por todos los momentos tan hermosos que hemos pasado juntos.

A **Fulvia**, porque las circunstancias no te permitieron realizar lo que algún día soñaste.

A ***Isabel Ruiz*** y ***Teresa Ortiz***.
IN MEMORIAM.

"Si lo puedes soñar, lo puedes hacer".

Walt Disney.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Fué en 1859 cuando Charles Darwin publica su libro, ***El origen de las especies***, causando una gran polémica en el mundo científico por sus revolucionarias teorías. La idea principal que Darwin daba a conocer en su obra era de que las especies evolucionan de acuerdo a su adaptación al medio en que viven. Con estas ideas las especies son un conjunto de individuos en constante competición y evolución para poder mantener su especie al paso del tiempo.

Segun Darwin la existencia de las especies tiene un comportamiento dinámico; las especies se crean, evolucionan con el tiempo y mueren si no se adaptan. Los individuos que mejor se adaptan al medio, llegan a la madurez y encuentran una pareja para perpetuar sus aptitudes que los hacen cada vez más aptos.

John Holland[34], inspirándose en las teorías desarrolladas por Charles Darwin, desarrolla junto con sus colaboradores en la universidad de Michigan los llamados ***algoritmos genéticos***. Una de las preguntas que Holland y sus colaboradroes se hacían en ese entonces era, ¿cómo logra la naturaleza crear individuos cada vez más perfectos?. El grupo de trabajo de Holland en ese entonces no tenía una respuesta, pero contaban con una cierta idea de cómo hallarla. Haciendo pequeños modelos de la naturaleza que tuvieran algunas de sus características y de su funcionamiento, trataban de simular el comportamiento evolutivo de la naturaleza y de esa manera obtenían cada vez más y mejores conclusiones.

De las ideas de Darwin, Holland retoma que la evolución es una forma de adaptación más potente que el simple aprendizaje, y decidió aplicar éstas ideas para desarrollar programas bien adaptados para un fin determinado. De esta manera cuando Holland y sus colaborades desarrollan los algoritmos genéticos el objetivo que se plantearon fué: imitar los procesos adaptativos tomando en cuenta los sistemas naturales y diseñar sistemas artificiales (programas) que retuvieran los mecanismos importantes de los sistemas naturales.

Dada que la evolución opera en los cromosomas en lugar de en los indivíduos a los que representa, el proceso de selección natural es el proceso por el que los cromosomas con buenas estructuras se reproducen más a menudo que los demás. En este proceso la evolución ocurre mediante la combinación de los cromosomas de los progenitores. Así pues, la evolución tiene lugar en los cromosomas en donde está codificada toda la información genética de cada ser vivo.

La información almacenada en el cromosoma varía de unas generaciones a otras, de esta manera la evolución biológica no tiene memoria en el sentido de que en la formación de los cromosomas únicamente se considera la información del período anterior.

Los algoritmos genéticos desarrollados por Holland, establecen una analogía entre el conjunto de soluciones de un problema y el conjunto de individuos de una población natural, codificando o transformando la información de cada solución en una cadena (vector binario de ceros y unos) a modo de representar un cromosoma, para poder hallar soluciones a problemas muy complejos. En palabras del propio Holland: ***"Se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de evolución simulada".***

La idea de Holland era de crear programas autoadaptables y que evolucionaran por sí sólas

tal como se hace en un sistema natural y que las computadoras en un futuro pudieran ser capaces de aprender por sí mismas sin la intervención del hombre para poder resolver un problema en general.

El siguiente trabajo de tesis se organiza de la siguiente manera. En el primer capítulo se describe el algoritmo genético simple sus diferentes operadores y parámetros. En el segundo capítulo se describe el diseño del algoritmo genético simple y la experimentación numérica con el fin de determinar los distintos operadores y parámetros con mejores resultados. El tercer capítulo describe el algoritmo genético con subpoblaciones, el diseño y la experimentación numérica y se hace una comparación de resultados con el algoritmo genético simple. En el cuarto capítulo se aplica el algoritmo genético con subpoblaciones a un problema de *caracterización de yacimientos petroleros naturalmente fracturados*, ya que los problemas tipo Newton no funcionan debido a que la función objetivo contiene valles muy profundos.[1] Es por eso que se resuelve un problema de optimización de mínimos cuadrados que caracteriza al yacimiento. Por último se dan las conclusiones del trabajo.

[1] Quiere decir que cuando un problema en general contiene valles muy profundos o estrechos o muy planos y se resuelven con un método tipo Newton que utiliza información de gradientes o derivadas, no funcionan debido a que a mayor profundidad en el valle mayor tamaño de la derivada y por lo consiguiente tamaños de paso muy cortos que trae como consecuencia el estancamiento o paro del método y no hallar la solución adecuada con la precisión requerida.

1. DESCRIPCIÓN DE LOS ALGORITMOS GENÉTICOS.

1.1. Historia.

Las primeras aportaciones sobre las ideas de los algoritmos genéticos aparecieron en artículos publicados por John Holland[33] a principios de 1960. En estas publicaciones Holland presenta un amplio y ambicioso tema para entender los principios de los *sistemas adaptativos*, que son capaces de automodificarse en respuesta a las interacciones del entorno en que funcionan.

Con las publicaciones de Holland se presenta la clave principal para el desarrollo de *sistemas de adaptación natural robustos* que hacen uso de la competición y la innovación y tener la habilidad de responder dinámicamente a eventos no esperados y cambiar de entornos.

Holland hizo modelos simples de la evolución biológica para capturar sus ideas de evolución, simplemente observando el proceso de la sobrevivencia del más apto y la continua producción de nuevos hijos. A mediados de 1960 las ideas de Holland comienzan a tomar varias formas computacionales con la ayuda de sus estudiantes de doctorado como se resume a continuación, Bäck[2].

Bagley (1967) realizó algunos de los primeros trabajos experimentales, hizo uso de representaciones binarias, el rol de la inversión y los mecanismos de la selección. En contraste Rosenberg (1967) realizó una clase distinta de simulación de la evolución utilizando un simple sistema biomédico en el cual un simple organismo célula es capaz de producir enzimas, hizo uso de representaciones binarias y sus experimentos fueron para producir concentraciones químicas apropiadas. Los resultados de las investigaciones de Rosenberg es que fueron los primeros experimentos que utilizaron operadores de cruza adaptativos.

Cavicchio (1970) vió estas ideas como una forma de búsqueda adaptativa y los probó experimentalmente sobre problemas de búsqueda difíciles, desarrollando subrutinas de selección y reconocimiento de patrones. Los resultados de Cavicchio muestran los primeros estudios sobre elitismo, formas de selección e ideas para adaptar el radio de la cruza y la mutación.

Hollstein (1971) trabajó sobre alternativas de selección y esquemas de cruzamiento. Experimentó con una variedad de estrategias de reproducción semejantes a las técnicas usadas por los animales reproductores. Los estudios de Hollstein muestran el uso de las cadenas binarias y las primeras observaciones sobre las virtudes del código Grey.

A la par con estos estudios experimentales, Holland continuó trabajando sobre la teoría general de sistemas adaptativos. Durante este período desarrolló su ahora famoso análisis de esquemas de sistemas adaptativos, haciendo uso del modelo estocástico llamado *la banda de k-brazos*. Holland usó estas ideas para desarrollar un análisis más teórico de sus planes reproductivos y publicar posteriormente su famoso libro ***adaptación en sistemas naturales y artificiales*** (1975), Holland[34].

A principios de 1970 hubo un considerable interés por entender mejor el desarrollo de la implantación de algoritmos genéticos. En particular era claro que escoger el tamaño de la población, representación, los operadores y sus parámetros, tiene un efecto significativo en el desarrollo de los algoritmos genéticos.

Frantz (1972) estudió en detalle los roles de la cruza y la inversión en una población de

tamaño 100. Fue el primer trabajo en utilizar la cruza multipunto.

De Jong (1975) amplió la línea de estudio para analizar teórica y experimentalmente los efectos interactuantes del tamaño de la población, cruza y mutación sobre un conjunto de funciones de optimización. Al término de los estudios de De Jong los algoritmos genéticos comienzan a hacerse fuertes en el sentido de que tienen un significado potencial para resolver problemas de optimización difíciles.

A mediados de 1970 grupos de diversas universidades incluyendo la universidad de Michigan, la universidad de Pittsburgh y la universidad de Alberta, organizaron un taller de sistemas adaptativos en el verano de 1976 en Ann Arbor, Michigan, el cual se repetía año tras año. Holland, De Jong y Sampson obtuvieron de la fundación NSF (Fundación Selección Natural) apoyo para formar un taller interdisciplinario sobre sistemas adaptativos, el cual fue patrocinado por la universidad de Michigan en el verano de 1981.

En esos tiempos se formaron diversos grupos de investigadores para trabajar sobre algoritmos genéticos. En la universidad de Michigan, Bethke, Goldberg y Booker continuaron desarrollando algoritmos genéticos para explorar los sistemas adaptativos de Holland como parte de sus trabajos doctorales. En la universidad de Pittsburgh, Smith y Wetzel trabajaron junto con De Jong sobre varios algoritmos genéticos modernos incluyendo la propuesta de *Piit* a roles de aprendizaje. En la universidad de Alberta, Brindle continuó trabajando en problemas de optimización bajo la dirección de Sampson.

El contínuo interés por el desarrollo de algoritmos genéticos provocó una serie de discusiones y planes para apoyar a la *Primera Conferencia Internacional Sobre Algoritmos Genéticos* (ICGA) en Pittsburgh Pensylvania, en 1985. Participaron alrededor de 75 personas presentando y discutiendo un amplio avance de nuevos desarrollos teóricos y prácticos de los algoritmos genéticos, el cual se convirtió en una conferencia bianual y la aparición de diversos libros que hablaban sobre el tema. El libro de Goldberg en particular sirvió como una catálisis significativa para representar la actual teoría de los algoritmos genético y sus aplicaciones en una forma clara y concisa fácil de entender por una amplia audiencia de científicos e ingenieros.

En 1989 la conferencia ICGA y otras actividades, tuvieron como resultado la formación de la *Sociedad Internacional de Algoritmos Genéticos* (ISGA) que tenía como propósito la organización de las posteriores conferencias.

El periodo de 1990 se caracterizó por un tremendo desarrollo sobre algoritmos genéticos como consecuencia de las diferentes actividades realizadas en las conferencias (ICGA). Nuevos paradigmas tales como *algoritmos genéticos desordenados* (messy) (Goldberg et al 1991) y *programación genética* (Koza 1992). La interacción con otras comunidades de computación evolutiva ha resultado un considerable intercambio de ideas generando muchos algoritmos evolutivos híbridos. Muchas aplicaciones de algoritmos genéticos continúan desarrollándose aplicados a problemas de ingeniería, investigación de operaciones y programación automática.

1.2. Descripción de un algoritmo genético simple.

Los algoritmos genéticos son algoritmos de búsqueda basados en los mecanismos de la selecćon natural y genética natural. Combinan la supervivencia del más apto entre estructuras de cadenas e intercambian información aleatoriamente para formar algoritmos de búsquedas parecidos a la reproducción humana. En cada generación, se crea un nuevo conjunto de criaturas (cadenas) tomando partes del más adaptado y sustituyéndolas por otras. Los algoritmos genéticos no producen una caminata aleatoria, sino que exploran eficientemente la información

anterior para poder especular nuevos puntos de búsqueda, Goldberg[22].

Según el Dr. Coello Coello Carlos[18], un algoritmo genético 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 presentado 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.

Dada éstas definiciones un algoritmo genético no es mas que un algoritmo de búsqueda basado en los roles de la evolución natural de los seres vivos. A modo de pseudocódigo la tabla 1.1, muestra un ejemplo de un algoritmo genético simple, Chipperfield [17].

```
Procedure Algoritmo_Genético
comenzar
    t ← 0
    inicializar P(t)
    evaluar P(t)
    mientras no se ha alcanzado la condición de paro hacer
        comenzar
            t = t + 1
            seleccionar P(t) de P(t-1)
            cruzar P(t)
            mutar P(t)
            evaluar P(t)
        fin comenzar
    fin mientras
fin comenzar
```

Tab. 1.1: Algoritmo genético simple.

A continuación se describen las diferentes fases del algoritmo genético como se muestra en la tabla 1.1.

1.3. Representación; ¿binaria o real?

El primer paso decisivo en la creación de la población inicial, es el tipo de representación que se decida utilizar para representar las estructuras o individuos con las que opera un algoritmo genético. Como en muchos problemas de ingeniería, un individuo puede ser un vector de valores reales que especifica el número de parámetros del problema o en problemas de sistemas de control un individuo es un tiempo o una variación funcional de frecuencias que representan los parámetros de control, o en juegos y algunos problemas de inteligencia artificial, un individuo es una estrategia o un algoritmo para resolver una tarea en particular.

Como la estructura de un individuo varía de problema a problema, la solución de un problema en particular puede ser representado de varias maneras. Usualmente un método de búsqueda es más eficiente con una representación particular y no es eficiente con otras representaciones. Así,

escoger un esquema de representación eficiente no depende solamente del problema a resolver sino también del método de búsqueda escogido. La eficiencia y complejidad de un algoritmo de búsqueda depende grandemente de cómo son representadas las soluciones y cómo la representación apropiada está en el contexto de los operadores de búsqueda subordinados.

En un método clásico de optimización y búsqueda, todas las variables de decisión son usualmente representadas como vectores de números reales y el algoritmo trabaja sobre un vector de soluciones para crear un nuevo vector solución (Deb 1995, Reklaitis et al 1983), Bäck[2]. En muchas aplicaciones de algoritmos genéticos las variables de decisión son codificadas en cadenas binarias de 0 y 1. Sin embargo pueden ser enteras o valores reales, y las cadenas binarias son representadas con una longitud específica dependiendo de la precisión requerida en la solución. Por ejemplo, una variable real x_i acotada en el rango (a, b) puede ser codificada en una cadena de cinco bits con las cadenas (00000) y (11111) representando los valores reales a y b respectivamente. Nótese que con cinco bits la máxima precisión alcanzada es de $(b\text{-}a)/(2^5\text{-}1)$, Bäck[2].

Desde los inicios de la investigación de los algoritmos genéticos el uso de la representación binaria no mapea adecuadamente el espacio de búsqueda con el espacio de representación. Por ejemplo, codificando en binario los enteros 5 (101) y 6 (110), los cuales son adyacentes en el espacio de búsqueda y que difieren en 2 bits (el primero y el segundo de derecha a izquierda) en el espacio de representación. A este fenómeno se le conoce como *risco de Hamming* y ha conducido a muchos investigadores a proponer una representación alternativa en la que la propiedad de adyacencia existente en el espacio de búsqueda pueda preservarse en el espacio de representación. A este tipo de codificación usada por muchos investigadores se le llama código Gray, parecido a la codificación binaria, que consiste de 0s y 1s. Pero la codificación y decodificación de cadenas para obtener variables en código Gray y viceversa son diferentes. La codificación de variables a cadenas de código Gray se desarrolla en dos pasos: de la variable codificada en binario por ejemplo $(b1, b2, ..., bl)$ donde $b_i \in \{0, 1\}$ se convierte a código Gray $(a1, a2, ..., al)$ usando un mapeo $\gamma^{-1} : \mathbb{B}^l \to \mathbb{B}^l$, Bäck [3]:

$$a_i = \begin{cases} b_i & \text{Si } i\text{=}1 \\ b_{i-1} \oplus b_i & \text{Otro caso} \end{cases}$$

donde $\oplus$ denota adición modulo 2. Como muchos investigadores indican, la principal ventaja del código Gray es que solamente cambia un bit conforme avanza de un número al siguiente; en otras palabras, el cambio de cualquier número al siguiente en una secuencia se reconoce mediante un cambio de solamente un bit de 0 a 1 ó de 1 a 0. La decodificación de una cadena Gray en la variable correspondiente se hace de la siguiente manera: primero la cadena Gray se convierte a una cadena binaria simple como sigue:

$$b_i = \bigoplus_{j=1}^{i} a_j \qquad \text{para} \qquad i = 1, \ldots, l$$

Donde l es la longitud de la cadena y se codifica a un número real en el rango $[u_i, v_i]$. En la tabla 1.2 se muestra un ejemplo de codificación binaria y Gray, Bäck[3].

Entero	Binario	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Tab. 1.2: Ejemplo de codificación binaria y Gray

Mientras la codificación binaria es la forma más común, hubo un gran interés en alternativas de estrategias de codificación, tales como representaciones enteras y reales. Para algunos dominios de problemas, es claro que la representación binaria es en efecto engañosa en el sentido de que oscurece la naturaleza de la búsqueda.

Alden H. Wright[1], realizó experimentos con problemas de optimización utilizando representación binaria y real. Sus resultados mostraron que el algoritmo genético en código real dió mejores resultados que el de código binario en la mayoría de las funciones de prueba experimentadas. Además también concluyó que el algoritmo genético en código real tiene las siguientes ventajas:

1. **Incremento en la eficiencia**: los pobladores ya no necesitan convertirse a números reales para cada evaluación de la función.
2. **Incremento en la precisión**: dado que no es necesario hacer una conversión a números reales, no hay pérdida de precisión como en la representación binaria.
3. Existe gran libertad para usar diferentes técnicas de cruzamiento y mutación.

Janikow Z. Cesary y Michalewicz Zbigniew [45], también realizaron estudios comparativos con representaciones binaria y real en un problema de control dinámico y los resultados que obtuvieron fué que la representación real es más rápida, más consistente de corrida en corrida y provee una alta precisión especialmente en dominios grandes en donde la codificación binaria requiere representaciones grandes.

Debido al análisis anterior, en este trabajo se usa la representación real.

1.4. Población inicial.

Una vez que se tiene decidido el tipo de representación a usar, para comenzar el ciclo evolutivo de un algoritmo genético es necesario crear una población inicial. Escoger el tamaño de la población para un algoritmo genético es una decisión fundamental para cualquier usuario que desee implantarlo. Por una parte si se selecciona un tamaño de población muy pequeña, el algoritmo genético puede converger rápidamente con insuficiente procesamiento de esquemas o cadenas, lo que traería como consecuencia una convergencia prematura o la obtención de soluciones subóptimas como pudieran ser óptimos locales.

Por otro lado una población con muchos miembros o pobladores requiere mucho consumo de tiempo computacional sobre todo cuando evaluamos la función objetivo y dicha función es muy costosa. Para estimar el tamaño adecuado de la población se basa en la teoría desarrollada por Holland, el cual dice que el número de individuos procesados efectivamente es proporcional al cubo del tamaño de la población $O(n^3)$[1].

Schaffer J. David et al [44], hicieron estudios empíricos con diferentes parámetros y determinaron que una población muy grande puede alcanzar a muestrear el espacio de búsqueda al menos en la primera generación. Sin embargo determinaron que poblaciones grandes imponen un gran costo computacional por generación y la exploración para individuos no presentes en la población inicial puede ser alcanzado por los operadores durante las generaciones posteriores.

Goldberg [25], basándose en la teoría desarrollada por Holland calculó el número único esperado de pobladores en una población aleatoria y usó esa cantidad junto con una estimación del tiempo de convergencia para encontrar el tamaño óptimo de la población que maximiza el radio de procesamiento de esquemas. Goldberg realizó pruebas en serial y en paralelo y sus resultados muestran que hay un alto procesamiento de esquemas con pequeñas poblaciones en serial y con grandes poblaciones en paralelo.

Mühlenbein y Schlierkamp-Voosen [41], derivaron una expresión para poblaciones pequeñas con el cual se logra converger rápidamente al óptimo con una probabilidad alta. Ellos concluyeron que el tamaño óptimo de la población depende del suministro inicial deseado de **alelos**, sobre el tamaño del problema, y sobre un parámetro llamado *intensidad de selección*.

A pesar de los diferentes estudios que se han hecho y se siguen haciendo para determinar el número óptimo de pobladores en un algoritmo genético, aun quedan muchos aspectos por discernir. Es por eso que en el proceso de creación de la población inicial el tamaño de la población se selecciona empíricamente, generalmente entre 30 y 100 pobladores. La creación de dicha población se hace aleatoriamente haciendo uso de un generador de números pseudoaleatorios y un intervalo de búsqueda (cotas) en el cual pensamos que puede estar la solución óptima.

Otro aspecto muy importante y crítico para la creación de la población inicial, además del tamaño de la población, es el generador de números pseudoaleatorios que se decida utilizar. Dado que la convergencia no depende solamente del generador de números pseudoaleatorios escogido sino de varios parámetros que garanticen la convergencia del algoritmo, es deseable entonces pensar en generadores de números pseudoaleatorios que generen diversidad. Es decir que nos aseguren una mayor equidistribución normal, tomando en cuenta las propiedades de un buen generador de números pseudoaleatorios como son:

[1] Los procesos que emplean ciclos triplemente anidados comúnmente emplean un tiempo cúbico en su ejecución.

1. Rapidez

2. Poco consumo de memoria

3. Portabilidad

4. Sencillez en la implantación

5. Reproducibilidad y mutabilidad

6. Periodo suficientemente largo.

Tomando en cuenta estas propiedades la quinta propiedad es esencial en simulación y caracteriza a los generadores algorítmicos de números pseudoaleatorios frente a generadores físicos. Probablemente, la última propiedad sea la más relevante e importante de las seis anteriores, puesto que a mayor periodo se obtiene mayor diversidad, Ríos[42].

El generador de números pseudoaleatorios utilizado en este trabajo fué el desarrollado por Makoto Matsumoto y Takuji Nishimura [38], el MT19937 (Mersenne Twister), el cual tiene un período tremendamente largo $2^{19937} - 1$ consumiendo un área de trabajo de 624 palabras y una propiedad de equidistribución de 623-dimensiones a 32 bits de precisión. Los autores compararon la velocidad del MT19937 con otros generadores de números pseudoaleatorios en una Sun Workstation y obtuvieron los resultados que se muestran en la tabla 1.3, Matsumoto[38]:

ID	COMBO	KISS	rand_array	rand	taus88	TT800	MT19937
Tiempo-CPU (segs.)	11.14	9.24	23.23	9.64	7.95	9.97	10.18
Area de Trabajo (palabras)	4	5	1000	1	3	25	624
Período	$\sim 2^{61}$	$\sim 2^{127}$	$\sim 2^{129}$	$\sim 2^{31}$	$\sim 2^{88}$	$\sim 2^{800}$-1	$\sim 2^{19937}$-1

Tab. 1.3: Comparación entre generadores de números pseudoaleatorios

Cantú-Paz [16], tomando en cuenta que uno de los aspectos importantes en el proceso de la selección estocástica, la recombinación y las operaciones de mutación, es el generador de números pseudoaleatorios, estudió los efectos que representan en un algoritmo genético simple. Utilizó dos generadores de números pseudoaleatorios: el MT19937 (Mersenne Twister) y otro tradicional de la gran variedad que se encuentra en la literatura. Sus estudios demostraron que utilizar uno u otro generador de números pseudoaleatorios para crear la población inicial es crítico en el proceso de ejecución de un algoritmo genético. Los generadores utilzados para los

otros procesos (selección, cruza y mutación) no afecta la ejecución significativamente. Los estudios de Cantú-Paz reportaron que el impacto de los generadores de números pseudoaleatorios puede ser mucho más dramático que los estudios reportados anteriormente. Y sugiere hacer uso de un buen generador de números pseudoaleatorios al menos para crear la población inicial, para tener una inicialización cada vez mas uniforme o diversa y de esta manera poder conservar por más tiempo la diversidad en la población y garantizar la convergencia del algoritmo.

Una vez creada la población inicial se procede a calcular los valores de la función objetivo (fenotipos) de cada individuo de la población, para obtener una estimación de cuáles son los individuos con mejor aptitud y poder proceder con los demás procesos del algoritmo genético.

1.5. Asignación de aptitud.

Después de que se ha evaluado cada individuo de la población en la función objetivo, se asigna un valor de aptitud (fitness) a cada individuo de la población. Dado que el valor de la función objetivo (fenotipo) proporciona una medida de cómo los individuos están evolucionando en el dominio del problema, entonces para los casos de problemas de minimización el individuo más apto es el que tiene el valor numérico de la función objetivo más bajo y para un problema de maximización es lo contrario.

Evaluar los individuos de la población es un paso intermedio y sirve básicamente para determinar el progreso relativo de los individuos. Ahora para transformar los valores de la función objetivo a una cantidad relativa de aptitud (fitness) se hace uso de una *función de aptitud* como se muestra en la ecuación 1.1, Bäck[2].

$$F(x) = g(f(x)) \tag{1.1}$$

donde f es el valor de la función objetivo y g transforma los valores de la función objetivo a un valor numérico no negativo dando como resultado la aptitud relativa F. Hacer este mapeo de la función objetivo es necesario en la mayoría de los algoritmos genéticos, ya que proporciona una medidad de sobrevivencia de los individuos para poder pasar a la siguiente generación. Muchos investigadores Chipperfield [17] consideran el valor de la función de aptitud como el número de hijos que un individuo espera producir en la siguiente generación.

Existen comúnmente dos métodos de asignación de aptitud: proporcionales y basados en el rango (ranking).

1.5.1. Asignación de aptitud proporcional.

Uno de los primeros métodos de asigación de aptitud fué en proporción a la aptitud de los individuos. Aquí la aptitud $F(x_i)$ de cada individuo se calcula como la aptitud del individuo i (f_i) relativo a la población entera como se muestra en la ecuacion 1.2, Chipperfield [17].

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{Nind} f(x_i)} \tag{1.2}$$

Donde N_{ind} es el tamaño de la población y x_i es el valor fenotípico del individuo i. Esta asignación de aptitud asegura que cada individuo tiene una probabilidad de reproducción de acuerdo a su aptitud relativa, pero falla para valores negativos de la función objetivo.

Escalación lineal.

Como su nombre lo indica, simplemente calculamos la aptitud $f^{'}$ como se muestra en la ecuación 1.3, Goldberg[22].

$$f^{'} = af + b \tag{1.3}$$

Donde a es +1 para problemas de maximización y -1 para problemas de minimización y b asegura que el resultado del valor de aptitud no sea negativo y toma el valor del peor individuo observado en la última generación.

Escalación truncamiento sigma(σ)

La escalación lineal trabaja bien excepto cuando el cálculo de la aptitud negativa impide su uso, Goldberg [22]. La aparición de valores negativos es un problema que se presenta en las últimas generaciones ya que van apareciendo individuos con aptitud cada vez más grande. Para evitar este problema de escalación, Forrest(1985) sugirió usar la varianza de la población para procesar los valores de aptitud antes de hacer la escalación. En este procedimiento llamado truncamiento sigma(σ) se hace uso de la desviación estándar para estimar el valor de aptitud como se muestra en la ecuación 1.4, Goldberg[22].

$$f^{'} = f - (\bar{f} - c \cdot \sigma) \tag{1.4}$$

En la ecuación 1.4 la constante c se escoge como un múltiplo de la desviación estándar de la población en el rango de [1,3] y resultados negativos ($f^{'} < 0$) son arbitrariamente puestos en ceros. Siguiendo estos procedimientos el truncamiento sigma(σ) puede asignar valores de aptitud sin peligro de valores negativos.

Escalación ley de poder

Gillies(1985) sugirió otra forma de escalación donde la aptitud se toma como una potencia especificada de la función objetivo f, como se muestra en la ecuación 1.5, Goldberg[22].

$$f^{'} = f^{k} \tag{1.5}$$

Gillies realizó estudios en una máquina de aplicación visual y tomó $k = 1{,}005$ donde $k \in \mathbb{R}$. Sin embargo, los valores de k dependen del problema y es necesario cambiarlo para extraerlo o contraerlo al rango apropiado.

1.5.2. Asignación de aptitud basados en el rango (ranking)

Los mecanismos de selección que se mencionan más adelante, seleccionan individuos para reproducirse basándose en el valor de su aptitud relativa. Cuando se hace uso de los métodos de asiganción de aptitud proporcionales, el número que se espera tener de padres es proporcional a la aptitud de todos los individuos de la población. Dado a que no existe una restricción en la reproducción de individuos en una generación dada, puede darse el caso de la aparición de superindividuos o individuos con un valor de aptitud muy alto y dominar en los procesos reproductivos, que traen como consecuencia una rápida convergencia a posiblemente soluciones subóptimas.

De lo contrario, si hay una desviación de aptitud de la población muy pequeña, entonces la escalación provee solamente una pequeña predisposición hacia los individuos más aptos. Y para eliminar estos tipos de problemas existen básicamente dos tipos de asignación de aptitud basados en el rango de los individuos.

Ranking lineal

Baker[6] tomando en cuenta las serias desventajas de los métodos de asignación de aptitud proporcionales, sugirió limitar el rango reproductivo de los individuos, de modo que los individuos no generen un excesivo número de hijos y de esta manera poder prevenir la convergencia prematura.

El método propuesto por Baker primero hace una ordenación de los individuos y los valores de aptitud se asignan segun el rango y a la posición que un individuo ocupe en la población. Para hacer esto Baker hace uso de una variable llamada *presión de selección* (SP) la cual proporciona la predesposición de los individuos a ser seleccionados o una medida de elitismo en el proceso de selección y calcula los valores de aptitud como se muestra en la ecuación 1.6, Bäck[2].

$$F(X_i) = 2 - SP + 2(SP - 1)\frac{X_i - 1}{N_{ind} - 1}. \tag{1.6}$$

donde X_i es la posición del individuo i en la población previamente ordenada, N_{ind} es el número de individuos de la población y SP es la presión de selección que toma valores en el rango de $[1, 2]$.

Ranking no lineal

Existe también un nuevo método basado en el rango de los individuos el cual usa una distribución no lineal. Este método usualmente se usa en poblaciones grandes ya que permite presión de selección muy altos. La asignación de aptitud en este método se hace como en la ecuación 1.7, Bäck[2].

$$F(X_i) = N_{ind} \cdot \frac{X^{X_i - 1}}{\sum_{i=1}^{N_{ind}} X^{(i-1)}} \tag{1.7}$$

X se calcula como la raíz del polinomio dado en la ecuación 1.8,

$$0 = (SP - 1) \cdot X^{(N_{ind}-1)} + SP \cdot X^{(N_{ind}-2)} + ... + SP \cdot X + SP \tag{1.8}$$

donde SP toma valores en el rango de $[1, N_{ind} - 2]$.

La figura 1.1 muestra un ejemplo de asignación de aptitud para los métodos basados en el rango, Bäck[2].

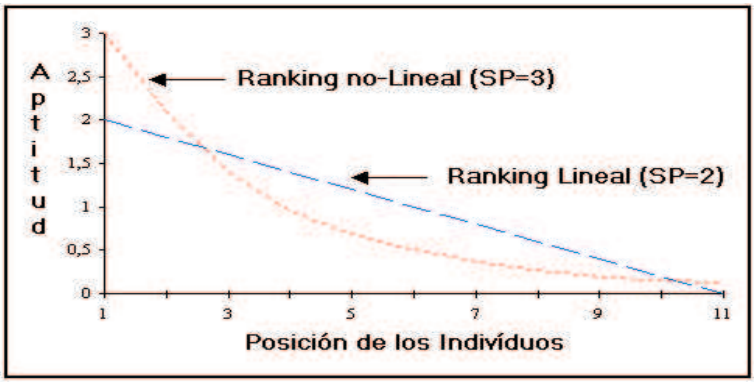

Fig. 1.1: Asignación de aptitud basados en el rango.

1.6. Selección.

La selección es uno de los principales operadores de los algoritmos evolutivos y su objetivo primario es escoger mejores soluciones en una población. Este operador no crea una solución nueva, sino selecciona relativamente buenas soluciones de una población y borra las soluciones no tan buenas.

Así, el operador de selección es una mezcla de dos diferentes conceptos: reproducción y selección. Cuando una o más copias de una buena solución son reproducidas, a esta operación se le llama reproducción. La identificación de buenas o malas soluciones en una población se realiza usualmente de acuerdo al valor asociado de su función objetivo. La idea esencial es que una solución que tiene una mayor aptitud, debe tener una alta probabilidad de selección. Sin embargo los operadores de selección difieren en la manera en que asignan el número de copias de los individuos seleccionados, como posibles padres. Algunos operadores ordenan la población de acuerdo a su valor de aptitud y determinísticamente escogen las mejores soluciones, mientras que otros operadores asignan una probabilidad de selección a cada solución de acuerdo a su aptitud y hacen una copia usando esta distribución de probabilidad.

En los operadores de selección probabilísticas hay una probabilidad finita de rechazar una buena solución y escoger una mala solución debido a la aleatoriedad del método. Sin embargo, un operador de selección usualmente se diseña para que esto tenga una mínima probabilidad de ocurrir. Por ejemplo en un problema no lineal complejo los mejores individuos en la población puede a veces representar una región subóptima y si se usa un operador de selección determinista estas aparentemente buenas soluciones pueden ser enfatizadas y la población puede finalmente converger a una mala solución. En cambio si usamos un método de selección estocástico se puede mantener la diversidad de la población escogiendo soluciones buenas y no tan buenas.

Dado que los operadores de selección varían en los diferentes estudios de algoritmos evolutivos, es difícil presentar un código común para todos los operadores de selección. Sin embargo en la tabla 1.4 Bäck[2], se presenta un algoritmo genérico para muchos de los operadores de selección utilizados en algoritmos genéticos y computación evolutiva.

Entrada: $\mu, \lambda, q, P(t) \in I^{\mu}, P^{'}(t) \in I^{\lambda}, F(t)$
Salida: $P^{''}(t) = a_1^{''}, a_2^{''}, ..., a_{\mu}^{''} \in I^{\mu}$
1 para $i \leftarrow 1$ hasta μ hacer
$a_i^{''}(t) \leftarrow S_{seleccion}(P(t), P^{'}(t), F(t), q)$
fin para
2 return $(a_1^{''}(t), ..., a_{\mu}^{''}(t))$

Tab. 1.4: Algoritmo general para la selección.

Los parámetros μ y λ de la tabla 1.4 representan el número de soluciones padres y soluciones hijos después de los operadores de recombinación y mutación respectivamente. El parámetro q es la presión de selección. La población en la iteración t se denota por $P(t) = a_1, a_2, ..., a_{\mu}$ y la población obtenida después de los operadores de recombinación y mutación se denota por $P^{'}(t) = a_1, a_2, ..., a_{\mu}$. Dado que las técnicas de los algoritmos genéticos y la programación genética usan primero el operador de selección, la población $P^{'}(t)$ antes de la operación de selección es un conjunto vacío, sin soluciones. La función de aptitud está representada por $F(t)$. Existen principalmente tres tipos de selección: proporcionales, basados en el rango y la selección por torneo.

1.6.1. Métodos de selección proporcionales.

Como su nombre lo indica los métodos de selección proporcionales crean padres en proporción a la aptitud de los individuos. Este método fué propuesto y analizado por Holland (1975) y se usa ampliamente en muchas implantaciones de algoritmos evolutivos. En el proceso de selección se lleva acabo básicamente los siguientes pasos, Bäck [2]:

1. Mapear la función objetivo a un valor de aptitud (fitness)

2. Crear una probabilidad de distribución proporcional a la aptitud

3. Obtener las muestras (padres) de esta distribución.

El número uno ya fué explicado en la sección 1.5, los números 2 y 3 dependen del método de selección que se use y a continuación se describen algunas de ellas.

Ruleta.

Una vez que los valores de aptitud ya han sido asignado a cada individuo de la población, el siguiente paso es crear una distribución de probabilidad y depende mucho del método de selección que se use. El primer método de selección fué creado tambiém por Holland (1975) llamado *rueda de ruleta*, el cual crea una distribución de probabilidad tal que la probabilidad de seleccionar a un individuo dado para reproducirse es proporcional a la aptitud de los individuos como se muestra en la ecuación 1.9, Bäck[2].

$$Pr(i) = \frac{fi}{\sum_{i=1}^{\mu} fi} \tag{1.9}$$

La ecuación 1.9 indica que la probabilidad de seleccionar al individuo i es igual a su aptitud (fi) entre la suma de todas las aptitudes de la población.

Entender el funcionamiento de la ruleta es bastante fácil. Una vez que ya se han calculado la probabilidad de selección de los individuos lo que se hace es distribuír esta probabilidad en una rueda o ruleta de tal menera que el individuo con mayor probabilidad de selección ocupará una porción más grande de la ruleta como se muestra en la figura 1.2.

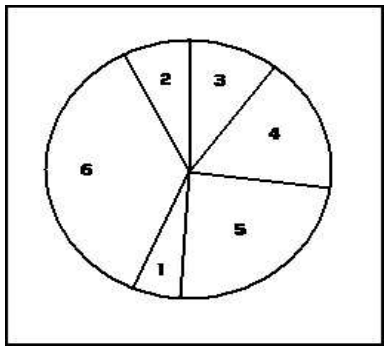

Fig. 1.2: Selección por ruleta.

Para seleccionar a los individuos giramos la ruleta y lanzamos un dardo y en la porción donde cae el dardo ese individuo es seleccionado. Hacer girar la ruleta y lanzar el dardo significa generar un número aleatorio entre $[0, 1]$ y se determina en que porción de la ruleta está el número aleatorio generado para saber el individuo a seleccionar.

Específicamente, para efectos de programación la ruleta consiste en crear un intervalo de números reales (Sum) que representa la suma de las probabilidades de selección de los individuos en la población actual. Entonces se mapea a los individuos uno a uno en un intervalo contínuo en el rango $[0, Sum]$. El tamaño de cada intervalo corresponde a los valores de probabilidad de selección del individuo asociado. Por ejemplo en la figura 1.2, se representa la suma de las probabilidades de selección de seis individuos. Se observa que el individuo 6 es el más apto y ocupa la porción más grande, los individuos 1 y 2 son los menos aptos ya que ocupan la porción más pequeña en la ruleta. Entonces para seleccionar un individuo, se genera un número aleatorio entre $[0, 1]$ y el individuo que ocupe el intervalo dado por el número aleatorio será seleccionado. Este proceso se repite hasta que se complete el número de individuos requerido. A modo de pseudocódigo la tabla 1.5 Bäck[2], muestra un algoritmo para la selección por ruleta básica.

Entrada: La probabilidad de distribución Pr
Salida: n el padre seleccionado.
1 Rueda_Ruleta (Pr):
2 $n \leftarrow 1$;
3 $sum \leftarrow Pr(n)$;
4 random $u \sim U(0,1)$
5 mientras $sum < u$ hacer
6 $n \leftarrow (n+1)$;
7 $sum \leftarrow sum + Pr(n)$
8 fin mientras
9 retornar (n);

Tab. 1.5: Selección vía ruleta básica.

El método descrito hasta aquí se le conoce como *muestreo estocástico con reemplazo* (SSR) el cual tiene una complejidad en tiempo de ejecución de $O(NlogN)$[2]. Existen otros métodos Michalewicz[39], derivados del método básico (SSR), el *muestreo estocástico con reemplazo parcial* (SSPR) el cual consiste en redimensionar la porción que ocupa un individuo si es seleccionado. Cada vez que un individuo es seleccionado el tamaño de su segmento se reduce a 1, si el tamaño alcanza ser negativo, entonces se pone en 0. Otro método es el *muestreo estocastico del resto con reemplazo* (RSSR). En este método los individuos son seleccionados determinísticamente de acuerdo a la parte entera de su número esperado de hijos. El resto de los individuos son seleccionados probabilísticamente de la parte fraccional de los valores esperados de los individuos. El último método derivado de la ruleta es el *muestreo estocástico del resto sin reemplazo* (RSSWR) pone la parte fraccional de los valores esperados de un individuo en cero.

Muestreo estocástico universal.

En los algoritmos genéticos, generalmente la población se reemplaza durante cada generación, por lo tanto, la distribución de probabilidad se muestrea μ veces. En la ruleta esto se hace por μ independientes llamadas, pero hacer esto puede ocasionar una alta varianza en el número de hijos asignados a cada individuo. Esto quiere decir que como es un procedimiento completamente estocástico, en una generación dada un individuo con alta probabilidad de selección no sea seleccionado y puede seleccionarse un mal individuo, o también puede darse el caso de que asigne demasiadas copias a un buen o mal individuo.

Baker [6] tomando en cuenta las desventajas que presentaba la selección por ruleta, desarrolló un algoritmo llamado *muestreo estocástico universal* (SUS) el cual tiene una complejidad en tiempo de ejecución de $O(N)$[3] y exhibe menos varianza que las repetidas llamadas al procedimiento de la ruleta. La idea del SUS es hacer un sólo giro de la ruleta y no μ, para tener una distribución uniforme y determinar cuántas copias se asignan a todos los padres según su valor de aptitud asociado. Para efectos de programación el SUS funciona de la siguiente manera; usa n (n = número de individuos) punteros igualmente espaciados; entonces la distancia entre

[2] El tiempo se eleva en los programas que resuelven un problema por medio de puequeños subproblemas resolviendo estos independientemente y combinando la solución

[3] Generalmente un programa tiene una función de tiempo lineal cuando una pequeña cantidad del proceso es dada por cada elemento de entrada

los punteros está dada por $1/n$ y la posición del primer puntero se genera creando un número aleatorio en el rango $[0, 1/n]$. Las probabilidades de selección de los individuos que ocupen el espacio de los punteros son seleccionados. A modo de pseudocódigo la tabla 1.6 Bäck[2], presenta un algoritmo para el muestreo estocástico universal.

Entrada: La probabilidad de distribución Pr
Salida: $c = (c_1, ..., c_\mu)$, donde c_i es el número de hijos . asignados al individuo a_i, y $\sum c_i = \lambda$.

```
1 SUS (Pr, λ):
2     random u ~ U(0, 1/λ);
3     sum ← 0,0;
4     para i = 1 hasta μ hacer
5         c_i ← 0;
6         sum ← sum + Pr(i);
7         mientras u < sum hacer;
8             c_i ← c_i+1;
9             u ← u + 1/λ;
10         fin mientras
11    fin para
12 retornar c;
```

Tab. 1.6: Selección vía muestreo estocástico universal.

Se observa en la tabla 1.6 el número de hijos que SUS asigna a un individuo i es $\lambda Pr(i)$. Finalmente SUS es óptimamente eficiente, ya que hace un simple paso sobre todos los individuos para asignarles las copias necesarias segun su probabilidad de selección.

1.6.2. Métodos de selección basados en el rango (rank).

Como se describió en la sección 1.6, la selección es el proceso de escoger individuos para la reproducción o la sobrevivencia en los algoritmos evolutivos. El proceso de selección basado en el rango o ranking significa que solamente la ordenación por rango de la aptitud de los individuos en la población actual determina la probabilidad de selección.

Los métodos de ranking eliminan la necesidad de escalar la función dado que se mantiene la presión de selección (SP), incluso si los valores de la función objetivo dentro de la población convergen a un rango muy estrecho, lo que sucede a menudo cuando las generaciones aumentan. Dado que los métodos de selección por ruleta y SUS son métodos de selección universales, los métodos de selección basados en el rango o ranking hacen uso de cualquiera de ellas y se diferencían por la forma en que asignan probabilidades de selección.

Además de su simplicidad, otros motivos para usar métodos de selección basados en el rango o ranking Bäck [2], son:

1. Bajo la selección proporcional, un superindividuo puede tomar el control de la población en una simple generación, a menos que se ponga un límite artificial para el número máximo de descendientes de un individuo. Ranking ayuda a prevenir la convergencia prematura causado por superindividuos, ya que a los mejores individuos se les asigna una probabilidad de selección de acuerdo a su rango y la presión de seleción (SP), valga lo que valga el

valor de su función objetivo asociado.

2. La selección por ranking puede ser una opción natural para problemas en los cuales es difícil precisar la función objetivo.

Existen principalmente dos métodos de selección basados en el rango o ranking: el *lineal* y el *exponencial*.

Ranking lineal.

En la selección por *ranking lineal* primero se ordena la población de acuerdo a su valor de aptitud y se asigna una probabilidad de selección a cada individuo proporcional a su rango (donde el rango del peor individuo es cero, y el rango del mejor individuo es $\mu - 1$ dada una población de tamaño μ). El ranking lineal se implanta especificando el parámetro $\beta rank$ que es el número esperado de hijos asignado al mejor individuo en cada generación. La probabilidad de selección para el individuo i se define entonces como en la ecuación 1.10, Bäck [2].

$$Pr_{rank}(i) = \frac{\alpha_{rank} + [\frac{rank(i)}{\mu-1}](\beta_{rank} - \alpha_{rank})}{\mu} \tag{1.10}$$

En la ecuación 1.10 α_{rank} es el número de hijos asignados al peor individuo y $\alpha_{rank} = 2 - \beta_{rank}$, y $1 \leq \beta_{rank} \leq 2$. Esto es, el número esperado de hijos del mejor individuo no es mas de dos veces el del promedio de la población. Esto presenta cómo el ranking puede evitar la convergencia prematura causada por superindividuos. A modo de algoritmo la tabla 1.7 Blickle[11], muestra un pseudocódigo para la selección via ranking lineal.

Entrada: La población $P(\tau)$ y β_{rank}
Salida: La población $P(\tau)^{'}$ seleccionada.
Ranking_Lineal($\beta_{rank}, J_1, ..., J_n$):
    $\bar{J} \leftarrow$ ordenar la población J de acuerdo a su aptitud con el peor individuo en la primera posición.
  $s_0 \leftarrow 0$
  para $i \leftarrow 1$ hasta N hacer
    $s_i \leftarrow s_{i-1} + p_i$ *(ecuación 1.10)*
  fin para
  para $i \leftarrow 1$ hasta N hacer
    $r \leftarrow$random[0,s_N]
    $J_i^{'} \leftarrow \bar{J}_l$ tal que $s_{l-1} \leq r < s_l$
  fin para
retornar $J_1^{'}, ..., J_N^{'}$

Tab. 1.7: Selección vía ranking lineal.

La tabla 1.7 muestra el algoritmo para la selección vía ranking lineal el cuál requiere de la ordenación de la población, de aquí que la complejidad del algoritmo está dominada por la complejidad del algoritmo de ordenación que puede ser por ejemplo $O(NlogN)$.

Ranking exponencial.

El método de selección ranking exponencial difiere de la selección ranking lineal en que las probabilidades de selección de los individuos rankeados se determinan exponencialmente. La base del exponente es el parámetro $0 < c < 1$ del método, que funciona más que nada como una presión de selección para controlar el elitismo en el proceso de selección. Nuevamente los individuos son ordenados según sus valores de aptitud con el peor individuo en la primera posición y el mejor individuo en la última posición. Las probabilidades de selección se determinan como en la ecuación 1.11, Blickle [11].

$$p_i = \frac{c-1}{c^N - 1} c^{N-i}; \qquad i \in \{1, \ldots, N\} \tag{1.11}$$

El algoritmo de este método de selección es similar al del ranking lineal y la única diferencia es la forma en que asignan las probabilidades de selección. La tabla 1.8 Blickle[11], muestra un pseudocódigo para este método de selección.

Entrada: La población $P(\tau)$ y $c \in [0, 1]$
Salida: La población $P(\tau)^{'}$ seleccionada.
Ranking_Exponencial$(c, J_1, ..., J_n)$:
    $\bar{J} \leftarrow$ ordenar la población J de acuerdo a su aptitud con el peor individuo en la primera posición.
    $s_0 \leftarrow 0$
    para $i \leftarrow 1$ hasta N hacer
        $s_i \leftarrow s_{i-1} + p_i$ *(ecuación 1.11)*
        fin para
    para $i \leftarrow 1$ hsta N hacer
        $r \leftarrow$random$[0, s_N]$
        $J_i^{'} \leftarrow \bar{J}_l$ tal que $s_{l-1} \leq r < s_l$
    fin para
retornar $\{J_1^{'}, ..., J_N^{'}\}$

Tab. 1.8: Selección vía ranking exponencial.

Se nota que los métodos de selección basados en el rango o ranking difieren en la manera de asignar probabilidades de selección a los individuos y para realizar la selección hacen uso de uno de los métodos proporcionales (ruleta o SUS). Blickle y Thiele [10], tomando en cuenta que uno de los métodos de selección que exhibe menos varianza en el proceso de obtener el número de copias de un individuo es el SUS, diseñaron un algoritmo que consideraron como universal y que se muestra en la tabla 1.9.

Entrada: La población $P(\tau)$
Salida: La población $P(\tau)^{'}$ seleccionada.
Selección_Universal$(J_1, ..., J_N)$:
    $s \leftarrow distribucion\_de\_aptitud(J_1, ..., J_N)$
    $r \leftarrow ditribucion\_de\_probabilidad(s)$
    $J^{'} \leftarrow SUS(r, J)$
retornar $J^{'}$

Tab. 1.9: Método de selección universal.

1.6.3. Método de selección por torneo

En la selección por torneo un grupo q de individuos es aleatoriamente seleccionado de la población. La selección puede ser con reemplazo o sin reemplazo, y este grupo de individuos participa en un *torneo*; esto es, el individuo ganador se determina dependiendo de su valor de aptitud. El mejor individuo que tiene el valor más alto de aptitud usualmente se escoge determinísticamente, aunque en algunos casos se puede usar un método estocástico. En ambos casos solamente el ganador pasa a la siguiente generación y el proceso se repite λ veces hasta obtener la nueva población. A menudo, el torneo se desarrolla entre dos individuos (torneo binario). Sin embargo, puede ser generalizado para un grupo arbitrario de tamaño q llamado *tamaño del torneo*.
El pseudocódigo mostrado en la tabla 1.10 Blickle[11], asume que los individuos son seleccionados con reemplazo y el individuo ganador se selecciona determinísticamente.

Entrada: La población $P(\tau)$ y el tamaño del torneo $q \in \{1, 2, ..., N\}$
Salida: La población $P(\tau)$ después de la selección.
Torneo$(q, J_1, ..., J_N)$
    para $i \leftarrow$1 hasta N hacer
        $J_i^{'} \leftarrow$ el mejor individuo de aleatoriamente
            q individuos seleccionados de $\{J_1, ..., J_N\}$;
    fin para
retornar $\{J_1^{'}, ..., J_N^{'}\}$

Tab. 1.10: Selección vía torneo.

La selección por torneo puede ser implantada muy eficientemente y tiene un tiempo de complejidad de $O(N)$ cuando no se requiere ordenar la población y es altamente paralelizable.

1.6.4. Análisis de la presión de selección (SP)

A continuación se presenta gráficamente el comportamiento de los diferentes operadores de selección cuando utilizan métodos de asignación de aptitud basados en el ranking de los individuos. Los operadores de selección proporcionales y los basados en el rango hacen uso del parámetro SP (presión de selección) el cual analizaremos gráficamente con diferentes valores, digamos bajo, medio y alto. Comenzaremos con la ruleta, y su proceso de selección se muestra

en la figura 1.3:

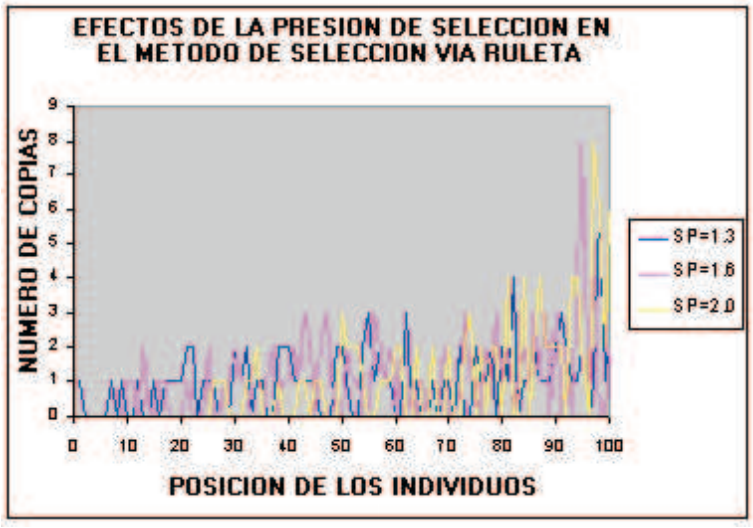

Fig. 1.3: Análisis ruleta-SP.

Observando la figura 1.3, en el eje X se representa a los individuos ordenados con el peor individuo en la primera posición y el mejor individuo en la última posición. En el eje Y se muestra el número de copias que el método asiga a cada individuo seleccionado. La curva de color rojo muestra el efecto de la selección tomando una presión de selección de SP=1.3 (baja), la curva azul una SP=1.6 (media) y la curva amarilla una SP=2.0 (alta). Como podemos ver con presión de selección baja el método se comporta completamente estocástico asignando aproximadamente 5 copias a los mejores individuos. Con SP media (curva azul) vemos que el método ya no selecciona los individuos del 0 al 10 aproximadamente, a pesar de que la ruleta es un proceso completamente estocástico. Con SP=2.0 alta (curva amarilla) vemos que ya no selecciona a los individuos del 0 al 25 aproximadamente y la selección se recarga cada vez mas a los mejores individuos asignandole 8 copias aproximadamente a los mejores individuos y volviendose cada vez más elitista.

Veamos el comportamiento gráfico del muestreo estocástico universal, que fue analizado con los mismos datos de SP (baja, media, alta). En la figura 1.4 se representa al método.

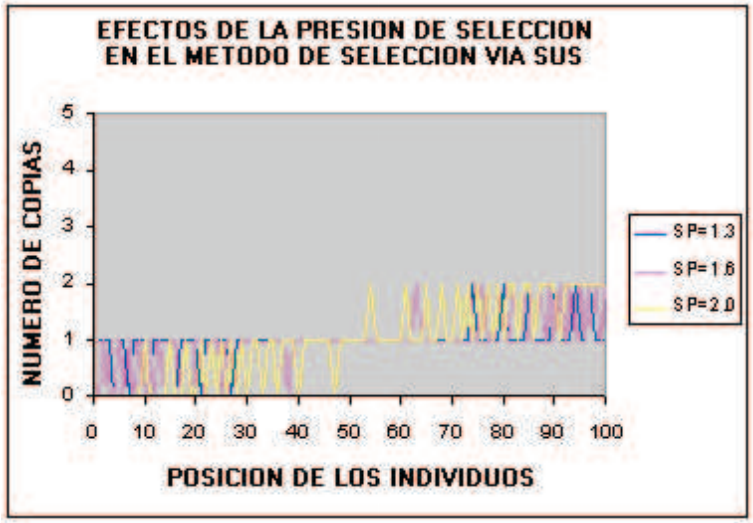

Fig. 1.4: Análisis SUS-SP.

Se observa que el SUS a diferencia de la ruleta, asigna a lo mas 2 copias a los mejores individuos con los diferentes parámetros de SP y nunca descarta la posibilidad de seleccionar a los malos individuos. Esto, en un momento dado puede ser bueno para conservar la diversidad de la población si conocemos de antemano la forma gráfica de nuestra función objetivo.

Analizando el ranking lineal que a diferencia de los dos métodos anteriores, utiliza un pseudónimo de la presión de selección SP, el parámetro β_{rank} o selección de *Bias* (tendencia) y cuya gráfica se muestra en la figura 1.5.

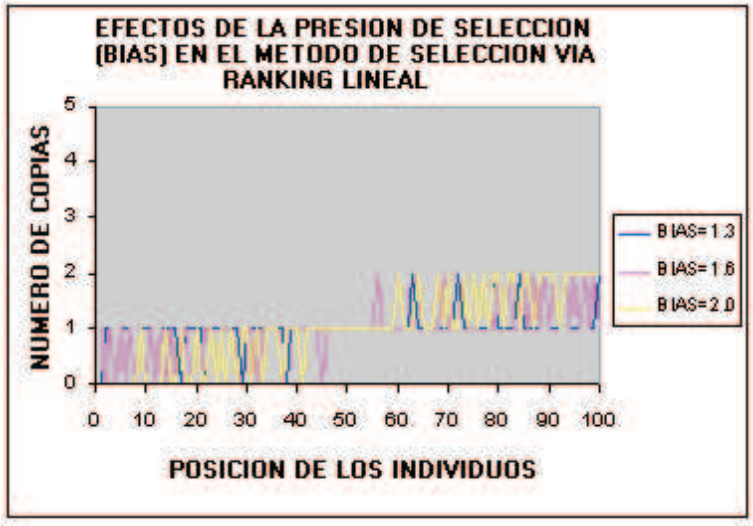

Fig. 1.5: Análisis ranking lineal-SP.

Como podemos ver el ranking lineal tiene un comportamiento parecido al muestreo estocástico universal (SUS) ya que asigna a lo más 2 copias a los mejores individuos y no descarta la posibilidad de seleccionar a malos individuos. Por último analizando el ranking exponencial que

hace uso también de un parámetro c selección de *Bias*, que funciona como presión de selección y le da la exponencialidad al método. Gráficamente se representa en la figura 1.6.

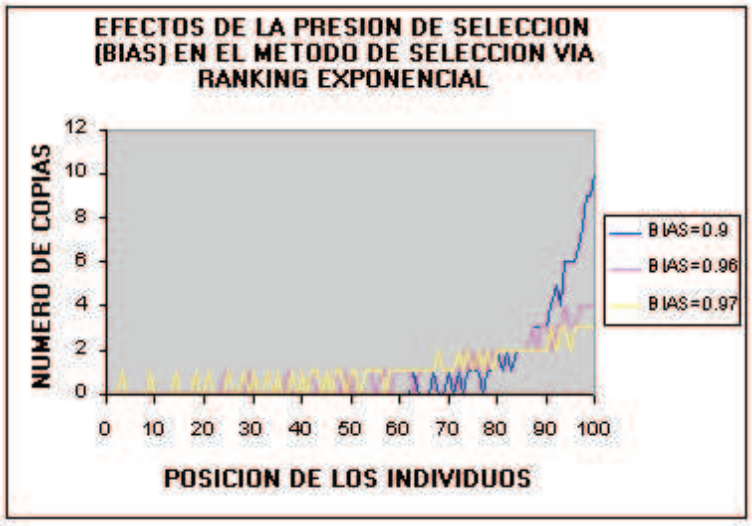

Fig. 1.6: Análisis ranking exponencial-SP.

Definitivamente se observa en la figura 1.6 que el método de selección por ranking exponencial le da preferencia a los mejores individuos. Mientras mas cercano sea el valor de *Bias* a 0, el método se vuelve cada vez más elitista.

A modo de resumen de esta sección y segun los datos analizados, la presión de selección no es más que un parámetro que controla el concepto de *elitismo* en el proceso de selección, sea cual sea el método. Mientras más alto sea este parámetro el método se vuelve cada vez más elitista. Si conocemos de antemano la forma gráfica de nuestra función objetivo, o sea si tiene muchos mínimos locales y un global o si tiene muchos mínimos globales y muchos locales o el valle es muy plano o estrecho, esta información puede ayudar a escoger el tamaño adecuado de este parámetro. Entonces es claro que el tamaño de este parámetro depende del problema que se esté resolviendo.

1.7. Cruza.

En sistemas biológicos la cruza es un proceso complejo que ocurre entre pares de cromosomas. Dos cromosomas son físicamente alineados y ocurre un rompimiento en una o más posiciones sobre cada cromosoma, y los fragmentos de cromosomas homólogos son intercambiados antes de que las rupturas sean reparadas. Esto resulta una recombinación de material genético que contribuye a la variabilidad en la población. En algoritmos evolutivos, este proceso fué abstraído en operadores de cruzamiento que intercambian subcadenas entre cromosomas representados como cadenas lineales de símbolos.

La operación básica de cruza, introducido por Holland (1975), es un procedimiento de tres pasos. Primero, dos individuos son escogidos aleatoriamente del grupo de padres seleccionados por el operador de selección. Segundo, se escogen uno o más localizaciones en las cadenas como puntos de rupturas (o puntos de cruza) delineando los segmentos de cadenas para intercambiarse.

Finalmente, los segmentos de cadenas padres son intercambiados y combinados para producir como resultado dos individuos hijos.

El grupo seleccionado de padres que pasan por el proceso de cruza durante una generación se controla por la *probabilidad de cruza* $p_c \in [0, 1]$, que determina el porcentaje de individuos a cruzar.

1.7.1. Métodos de cruza para representación binaria.

Dado que la representación binaria proporciona una gran facilidad para manejar los diferentes operadores en un algoritmo genético, a continuación se describen los principales operadores de cruza normalmente utilizados.

Cruza simple.

El operador de cruza más básico para representación binaria fúe desarrollado por Holland(1975) el cual le llamó *cruza simple*, que consiste en crear un punto de ruptura de los cromosomas padres para posteriormente intercambiar las subcadenas apartir del punto de ruptura y de esta manera obtener los nuevos hijos. Por ejemplo:
Dado los padres:

Padre1= 1 0 0 1 0

Padre2= 1 1 0 1 1

Se genera aleatoriamente un número en el intervalo $[1, l-1]$ para determinar un punto de ruptura el cual se identifica como | y l la longitud de los cromosomas padres. Suponiendo que el número generado para determinar el punto de ruptura es 3. Entonces a partir de esta posición se hace el intercambio de subcadenas como se muestra a continuación.

Padre1= 1 0 0 | 1 0

Padre2= 1 1 0 | 1 1

Intercambiando las subcadenas subrayadas se obtienen los siguientes hijos:

Hijo1= 1 0 0 1 1

Hijo2= 1 1 0 1 0.

Cruza multipunto.

La cruza multipunto fué propuesta por De Jong (1975) generalizando a la cruza simple con un parámetro llamado puntos de cruza. Con $n = 2$ se designan dos puntos de cruza, el cual es una buena selección que minimiza los efectos de rompimiento y es frecuentemente usado en muchas aplicaciones. No hay consensos acerca de las ventajas y desventajas de usar valores ≥ 3,

y los estudios empíricos sobre esto están aún inconclusos. La idea principal de la cruza multipunto es crear varios puntos de cruza e intercambiar segmentos de cadenas como se muestra en el siguiente ejemplo.

Padre1= 1 1 0 0 1 0 1 1 1 0

Padre2= 0 1 1 1 0 1 0 0 0 1

Generando aleatoriamente 3 puntos de cruza, digamos (2,5,7) de la siguiente manera:

Padre1= 1 1 |0 0 1 |0 1 |1 1 0

Padre2= 0 1 |1 1 0 |1 0 |0 0 1

Intercambiando las subcadenas subrayadas para generar los siguientes hijos:

Hijo1= 1 1 1 1 0 0 1 0 0 1

Hijo2= 0 1 0 0 1 1 0 1 1 0

La idea de la cruza multipunto es hacer una mayor exploración del espacio de búsqueda para favorecer la convergencia y hacerla más rápida.

Cruza uniforme.

Este operador fué introducido por Ackley (1987) y distribuído por Syswerda (1989). La cruza simple y multipunto definen puntos de cruza como lugares entre los 0s y 1s donde los fragmentos de cromosomas pueden intercambiarse. La cruza uniforme generaliza este esquema para hacer cada dígito un punto de cruza potencial. Se crea una máscara de cruza aleatoriamente de la misma longitud que los cromosomas padres, y la paridad de los bits de la máscara es el que indica que padres pueden crear hijos con estos bits. Por ejemplo:

Padre1 = 1 0 1 1 0 0 0 1 1 1

Padre2 = 0 0 0 1 1 1 1 0 0 0

Máscara=0 0 1 1 0 0 1 1 0 0

Hijo1 = 0 0 1 1 1 1 0 1 0 0

Hijo2 = 1 0 0 1 0 0 1 0 1 1

Aquí, el Hijo1, se produce tomando los bits del Padre1 si los bits de la máscara es 1 o del Padre2 si los bits de la máscara es 0. El Hijo2 se crea usando la inversa de la máscara, o de manera equivalente, intercambiando el Padre1 y el Padre2.

1.7.2. Métodos de cruza para representación real.

La recombinación actua sobre dos o más cromosomas en una población para generar al menos un hijo. Cuando los cromosomas son vectores de valores reales, la recombinación puede ser implantada por una gran variedad de formas. Muchas de estas formas se derivaron con el esfuerzo de comunidades de estrategias evolutivas envueltos en problemas de optimización contínuos. Las versiones simples fueron popularizados en investigaciones con algoritmos genéticos, los cuales se describen a continuación.

Cruza discreta.

El método más simple para representación real es la cruza discreta ya que realiza el intercambio de variables entre los individuos padres seleccionados por el operador de selección. Por ejemplo sean los padres $X = (X1, ..., Xn)$ y $Y = (Y1, ..., Yn)$ entonces los hijos $Z = (Z1, ..., Zn)$ se calculan de la siguiente manera como en la ecuación 1.12, Mühlenbein[40].

$$Z_i = \{X_i\} \quad o \quad \{Y_i\} \tag{1.12}$$

De la ecuación 1.12 X_i ó Y_i se escoge con probabilidades de 0.5 para cada variable, aplicado a ambas variables para los dos hijos. Por ejemplo.
Dado los padres:

Padre1 = 10.8736 -5.3848 17.1294 11.2824 5.2825

Padre2 = -0.1814 3.0451 68.1078 -3.7333 24.2838

Se genera aleatoriamente las probabilidades para intercambiar las variables (<0.5 para la variable uno y >0.5 para la variable 2) por ejemplo (0.2, 0.6, 0.8, 0.03, 0.4). Entonces los nuevos hijos son:

Hijo1 = -0.1814, -5.3848, 17.1294, -3.7333, 24.2838

Hijo2 = -0.1814, -5.3848, 68.1078, -3.7333, 24.2838

Se nota que la cruza discreta es parecida a la cruza uniforme en representación binaria. La figura 1.7 hace una representación gráfica de la cruza discreta.

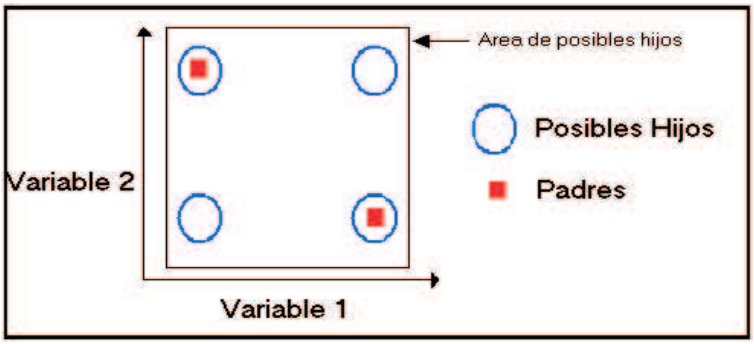

Fig. 1.7: Cruza discreta.

Cruza intermedia.

La cruza intermedia es un método que sólo se aplica a variables de tipo real y no a variables binarias. En este método los valores de las variables o cromosomas se escogen entre o alrededor de los valores de las variables o cromosomas de los padres como se muestra en la ecuación 1.13, Mühlenbein[40].

$$Z_i = X_i + \alpha(Y_i - X_i) \qquad \text{donde} \qquad i = 1, \ldots, Nind \tag{1.13}$$

Donde α es un factor de escala escogido aleatoriamente uniforme en el intervalo de [-0.25, 1.25]. Cada variable en los hijos es el resultado de la combinación de las variables de acuerdo a la expresión de la ecuación 1.13 con una alfa diferente para cada variable. A modo de ejemplo. Dado los Padres:

Padre1 = -7.3845, 6.5623, 0.6798, 3.1461, 7.3951

Padre2 = -3.0302, 0.2307, 0.1718, 5.3008, 17.1437

Aleatoriamente se generan los valores de las α^s para el Hijo1 (-0.20, 1.02, 1.23, 0.50, -0.17) y para el Hijo2 (0.58, -0.23, -0.16, 1.24, 1.09) y aplicando la ecuación 1.13 se obtienen los siguientes hijos:

Hijo1 = -8.2553, 0.1033, 0.0549, 1.2234, 18.2951

Hijo2 = 2.5254, 1.6869, 0.2530, 7.9726, 27.7696

La figura 1.8 muestra una representación gráfica de la cruza intermedia.

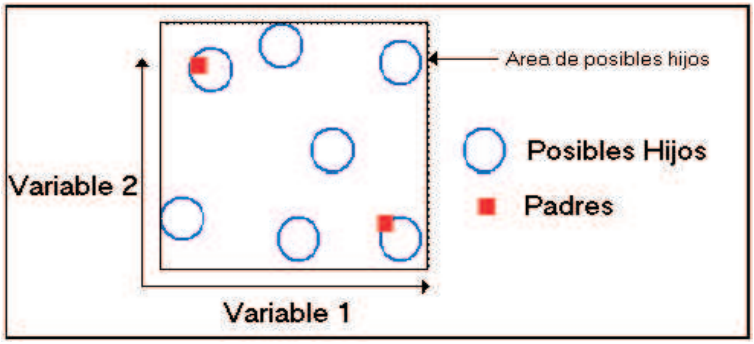

Fig. 1.8: Cruza intermedia.

Observando la figura 1.8, la cruza intermedia realiza una mayor exploración del espacio de búsqueda comparada con la cruza discreta, el cual le da la ventaja de poder converger más rápidamente y conservar una mayor diversidad de la población.

Cruza lineal.

La cruza lineal es similar al de la cruza intermedia excepto a que un sólo valor de alfa se utiliza para todas las variables. Las variables de los hijos se calculan a partir de las variables de los padres como en la ecuación 1.13, Mühlenbein[40]. Donde alfa es nuevamente un factor de escala escogido aleatoriamente uniforme en el intervalo de [-0.25, 1.25]. Por ejemplo.
Dado los Padres:

Padre1 = 6.0256, 3.7823, 0.4466, 5.5566, 3.2111

Padre2 = 2.6699, 207.1235, 38.7777, 66.1441, 5.5511

Generando aleatoriamente las $\alpha^{'}$s, para el Hijo1 (α=0.53) y para el Hijo2 (α= -0.21) y usando la ecuación 1.13 se obtienen los siguientes hijos:

Hijo1 = 4.2470, 111.5531, 20.7620, 37.6679, 4.4513

Hijo2 = 3.3745, 164.4218, 30.7281, 53.4207, 5.0597

La figura 1.9 representa gráficamente la cruza lineal.

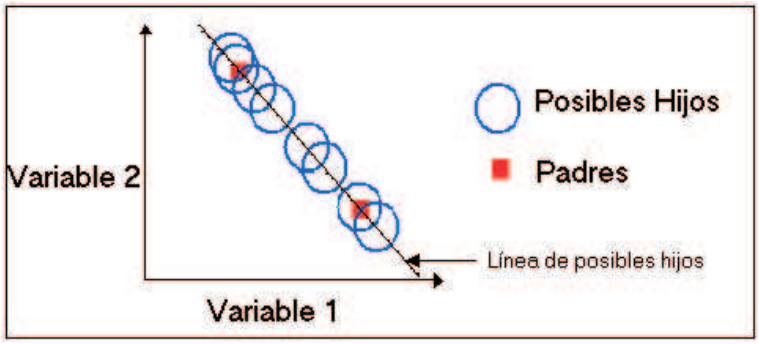

Fig. 1.9: Cruza lineal.

Como vemos en la cruza lineal el área de posibles hijos está determinado por una línea recta entre los cromosomas padres.

Hay otras variaciones de operadores de cruza para representación real Michalewicz[39], como son: *cruza heurística* de Wright (1994) que toma la siguiente forma $x^{'} = u(x_2 - x_1) + x_2$ donde u es una variable aleatoria uniforme en el intervalo $[0, 1]$ y x_1 y x_2 son los vectores padres sujeto a la condición de que x_2 no es peor que x_1. La *cruza simplex* de Renders y Bersini (1994) selecciona $k > 2$ padres, determina el mejor padre y el peor del grupo seleccionado (digamos x_1 y x_2), calcula el centroide del grupo sin x_2 (digamos c) y calcula el vector reflejado $x^{'}$ (el hijo) obtenido del vector x_2 como $x^{'} = c + (c - x_2)$. La *cruza geométrica* de Michalewicz et al (1996) toma dos padres x_1 y x_2 y produce un hijo simple $x^{'}$ como $x^{'} = [(x_{11}x_{21})^{0,5}, ..., (x_{1n}x_{2n})^{0,5}]$.

1.8. Mutación

Uno de los principales problemas de la mayoría de los algoritmos genéticos es que después de un cierto número de generaciones comienzan a perder diversidad en la población. Que quiere decir esto, que en un instante dado los individuos de la población entera comienzan a parecerse todos entre sí, o en un sentido más común comenzamos a ver individuos cada vez mas *bonitos* o *mejores*, el cual podría ser una indicación de *convergencia prematura* o que ya se está atrapado en un óptimo local.

Como sucede en la naturaleza, la *mutación* es un proceso aleatorio donde los alelos de los cromosomas son alterados físicamente para generar nuevos cromosomas. Pero a diferencia de lo que sucede en la naturaleza en la mayoría de los algoritmos genéticos la mutación es un proceso provocado dada una *probabilidad de mutación* p_{mut}, que proporciona una cantidad de individuos que pasarán por el proceso de mutación y que generalmente varía en el rango de $p_{mut} \in [0{,}001, 0{,}01]$.

El objetivo principal del operador de mutación en algoritmos genéticos es el de cambiar a los individuos que pasen por el proceso de mutación para poder conservar por mas tiempo la diversidad de la población y tener una mayor exploración del espacio de búsqueda y poder ir remontando los valles que en determinados momentos nos atrapan en óptimos locales si lo vemos desde un punto de vista de la optimización global de funciones.

Existen principalmente dos tipos de mutacón: *binaria* y *real* que se muestran a continuación.

1.8.1. Mutación binaria.

El operador de mutación para representación binaria es relativamente fácil de implantar como sucede en los operadores de cruza. Consiste en seleccionar aleatoriamente un punto en el cromosoma y cambiar el valor de 0 a 1 dependiendo de la paridad que tenga. Existen varios métodos de mutación como sucede con los operadores de cruza. Veamos un ejemplo de mutación en punto simple.
Suponiendo que se selecciona el siguiente individuo:

$$Individuo_1 = 0001100010$$

Aleatoriamente se genera el punto que se quiere cambiar, digamos el punto 3 que sería el alelo 0 de izquierda a derecha. Convirtiendo el alelo 0 a 1 se tiene el siguiente individuo mutado:

$$Individuo_1^{'} = 00\mathbf{1}1100010$$

Como vemos el proceso de mutación es relativamente fácil y puede ser aplicado también en forma multipunto.

1.8.2. Mutación real.

Dada una representación de valores reales donde cada elemento de la población es un vector de n dimensiones $x \in \mathbb{R}^n$, entonces la forma general de la mutación puede ser escrita como en la ecuación 1.14, Bäck[2].

$$x^{'} = m(x) \tag{1.14}$$

En la ecuación 1.14 x es el vector padre, m es la función de mutación, y $x^{'}$ es el vector mutado resultante. Existen varios tipos de operadores de mutación para representación real, a continuación describiremos la *mutación real* que fué utilizado en este trabajo.
En la mutación real una variable X_i es seleccionada con probabilidad P_{mutv} para ser mutada. No se debe confundir P_{mut} y P_{mutv} ya que la primera indica la proporción de individuos que seran mutados y la segunda indica qué variables del individuo serán mutados. En el algoritmo genético simple normalmente se usa $P_{mutv} = 1/n$ donde n es el número de variables de nuestra función objetivo y con esto se asegura que al menos una variable será mutada. Un valor fuera del intervalo $[-rango, rango]$ se agrega a la variable seleccionada. Rango define el radio de mutación. Esto es, normalmente se calcula como $0{,}5 \cdot intervalo\_de\_busqueda$, que es la definición del dominio de la variable X_i.

Entonces el nuevo individuo Z_i se obtiene como en la ecuación 1.15, Mühlenbein[40].

$$Z_i = X_i \pm rango_i \cdot \delta \tag{1.15}$$

El signo + o - se escoge con probabilidad de 0.5. El valor de δ se calcula de la siguiente manera:

$$\delta = \sum_{i=0}^{15} \alpha_i 2^{-i}; \qquad \alpha_i \in 0, 1$$

Donde α inicialmente toma el valor de 0 y se cambia a 1 con probabilidad de 1/16. Veamos un ejemplo en representación real. Dado el individuo en el intervalo de búsqueda $[-5, 5]$:

$Individuo_1$ = 5.0391, 10.3121, 0.7686, -0.1551, -0.3322

Suponiendo que las variables a mutar son la 2 y la 4, entonces para la variable 2 se tiene que aleatoriamente se selecciona el signo (-), $rango = 0{,}5 * 10 = 5$. Calculando delta con valores de α aleatoriamente generados $(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0)$, y realizando la sumatoria se obtiene un valor de $\alpha = 0.0815$.
Entonces usando la ecuación 1.15 se tiene:

Z_2=10.3121-(5·0.0815)= 9.9046

Para la variable 4, aleatoriamente se selecciona el signo +, y $rango = 5$. Calculando delta con valores de α aleatoriamente generados (0,1,1,1,0,0,0,0,1,1,0,1,0,0,0,1) y realizando la sumatoria se obtiene un valor de $\alpha = 0{,}8812$. Entonces usando la ecuación 1.15 se tiene:

Z_4 = -0.1551+(5·0.8812) = 4.2509

Finalmente se obtiene el individuo mutado:

$Individuo_1^{'}$ = 5.0391, 9.9046, 0.7686, 4.2509, -0.3322.

La figura 1.10 muestra una representación gráfica de la mutación.

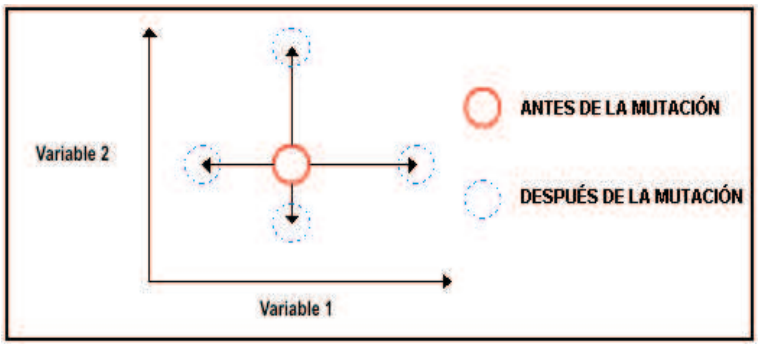

Fig. 1.10: Mutación.

1.9. Elitismo.

Los operadores de cruza y mutación generan nuevos individuos que se crean a partir de cromosomas padres. Dado que son operadores completamente aleatorios, es decir, que para cruzar o mutar individuos la selección de cromosomas padres se hace al azar. En la evolución natural y en algoritmos genéticos suele suceder que cuando cruzamos o mutamos dos individuos, los hijos resultantes suelen ser superiores o inferiores a los cromosomas padres, o sea que tienen un valor objetivo peor o mejor que el de los padres.

En otras palabras puede suceder que si cruzamos un cromosoma bueno o bonito con uno malo o feo, el hijo resultante puede ser también feo o también puede darse el caso de que resulte bueno o bonito como sucede en la naturaleza. Debido a este fenómeno nuestro algoritmo genético en cierto momento puede permanecer estancado o divergiendo en vez de ir convergiendo al resultado deseado.

Para evitar este problema se hace uso de un concepto llamado *elitismo*. En cada generación se determina un cierto porcentaje de los mejores individuos que pasan automáticamente a la siguiente generación sin pasar por los diferentes operadores del algoritmo genético. Esto se hace con el fin de acelerar la convergencia a la solución deseada. Sin embargo, es importante no escoger un porcentaje de elitismo demasiado grande, pues traería como consecuencia una pérdida de la diversidad en la población o una convergencia prematura.

1.10. Fundamento matemático de los algoritmos genéticos.

Los fundamentos teóricos de los algoritmos genéticos fuerón desarrollados por John Holland (1975) y todo su análisis fué desarrollado para representación binaria y sobre el concepto de *esquemas*. En la teoría de los algoritmos genéticos un *esquema* es una "plantilla similar"de 0s y 1s que describe un subconjunto del espacio de cromosomas. En el caso de representación binaria un esquema se describe con el alfabeto: $\{0, 1, *\}$. Donde el * se le conoce como símbolo *no importa*.

Por ejemplo el esquema (* 1 1 1 1 0 0 1 0 0) deriva las siguientes cadenas:

$\{(0111100100), (1111100100)\}$.

y el esquema (*1*1100100) deriva cuatro cadenas:

$\{(0101100100), (0111100100), (1101100100), (1111100100)\}$.

Ahora el esquema (1001110001) es claro que representa únicamente una cadena (1001110001), y el esquema (**********) todas las cadenas de longitud 10. Entonces cada esquema deriva exactamente 2^r cadenas, donde r es el número de símbolos * *no importa*. Por otro lado cada cadena de longitud m puede derivar 2^m esquemas. Por ejemplo en una cadena de longitud 10 (1001110001) puede derivar 2^{10} esquemas siguientes:

(1001110001)
(*001110001)
(1*01110001)
(10*1110001)
⋮
(100111000*)
(**01110001)
(*0*1110001)
⋮
(10011100**)
(***1110001)
⋮

(**********).

Considerando cadenas de longitud m, hay en total 3^m posibles esquemas. En una población de tamaño n se pueden representar entre 2^m y $n \cdot 2^m$ esquemas diferentes.

Los esquemas tienen diferentes características, se nota que el número de símbolos *no importa* * en un esquema determina el número de cadenas que derivan de ese esquema. Entonces dos propiedades importantes de los esquemas son:

Orden: El *Orden* del esquema S (denotado por o(S)) es el número de posiciones 0 y 1 presentes en el esquema, en otras palabras es la longitud de la plantilla menos el número de simbolos *no importa* *. El orden define la especialidad de un esquema. Por ejemplo en los esquemas:

S_1=(***001*110),
S_2=(****00**0*),
S_3=(11101**001),

tienen el suguiente orden: $o(S_1) = 6$, $o(S_2 = 3)$ y $o(S_3 = 8)$. La noción de orden de un esquema sirve para calcular la probabilidad de sobrevivencia de un esquema al operador de mutación.

Longitud: La longitud del esquema S (denotado por $\delta(S)$) es la distancia entre la primera posición 1 y la última y define la conpacidad de información contenida en un esquema. Por ejemplo en los esquemas anteriores tienen las siguientes longitudes:

$\delta(S_1) = 10 - 4 = 6$, $\delta(S_2) = 9 - 5 = 4$, y $\delta(S_3) = 10 - 1 = 9$.

La noción de longitud de un esquema es útil para calcular la probabilidad de sobrevivencia de un esquema al operador de cruza.

El principal fenómeno del proceso evolutivo en un algoritmo genético ocurre principalmente en dos pasos: La selección y la cruza. Veamos el efecto de estos dos pasos sobre el número esperado de esquemas representados en una población. Suponiendo que se tiene una población de tamaño Tpob = 20, y cadenas de longitud $m = 33$ y asumiendo que en un instante dado en la generación t la población es como la que sigue:

v_1 = (10011010000000111111110100110111111)
v_2 = (11100010010011011100101010001101O)
v_3 = (00001000001100100000101011101110 1)
v_4 = (10001100010110100111100000111001O)
v_5 = (00011101100101001101011111100010 1)
v_6 = (00010100001001010100101011111101 1)
v_7 = (00100010000011010111101101111101 1)
v_8 = (10000110000111010001011010110011 1)
v_9 = (01000000010110001011000000111110 0)
v_{10} = (00000111100011000001101000011101 1)
v_{11} = (01100111111011010110000110111100 0)
v_{12} = (11010001011110110100010101000000 0)
v_{13} = (11101111101000100011000000100011 0)

$v_{14} = (01001001100000101010011110010 1001)$
$v_{15} = (11101110110111000010001111101 1110)$
$v_{16} = (11001111000001111110000110100 1011)$
$v_{17} = (01010111111100111101000110111 1101)$
$v_{18} = (01110100000000111010011111010 1101)$
$v_{19} = (00010101001111111111000011000 1100)$
$v_{20} = (10111001011001111001100010111 1110)$

Ahora denotando el número de cadenas en el tiempo t derivados del esquema S como $\xi(S,t)$. Por ejemplo dado el esquema:

$$S_0 = (****111**********************)$$

tiene $\xi(S_0,t) = 3$, v_{13}, v_{15}, y $v16$ que derivan del esquema S_0. El orden del esquema S_0, $o(S_0) = 3$ y su longitud $\delta(S_0) = 7 - 5 = 2$.

Otra propiedad de los esquemas es su aptitud en el tiempo o generación t, $Apt(S,t)$, que define la aptitud promedio de todas las cadenas en la población derivados del esquema S. Asumiendo que hay p cadenas $\{v_{i1}, ..., v_{ip}\}$ que derivan del esquema S en el tiempo o generación t. Entonces se tiene:

$$Apt(S,t) = \sum_{j=1}^{p} Apt(v_{i_j})/p.$$

Como se analizó en el proceso de selección, la probabilidad de seleccionar una cadena simple v_i es $p_i = Apt(v_i)/F(t)$, $F(t)$ es la aptitud total de la población entera en el tiempo o generación t. Después del proceso de selección, esperamos tener $\xi(S,t+1)$ cadenas derivados del esquema S. Dado que, para un promedio de cadenas derivados del esquema S, la probabilidad de selección es igual a $Apt(S,t)/F(t)$ y el número de cadenas derivados del esquema S es $\xi(S,t)$, y el número de selecciones simples de cadenas es $Tpop$, entonces es claro que:

$$\xi(S,t+1) = \xi(S,t) \cdot Tpop \cdot Apt(S,t)/F(t),$$

Tomando en cuenta el promedio de aptitud de la población $\overline{F(t)} = F(t)/Tpop$ se obtiene la ecuación 1.16

$$\xi(S,t+1) = \xi(S,t) \cdot Apt(S,t)/\overline{F(t)}. \tag{1.16}$$

En otras palabras, el número de cadenas en la población crece a razón del número de cadenas derivados del esquema S en el tiempo o generación t por el promedio de aptitud de las cadenas derivados del esquema S en el tiempo o generación t . Esto significa que esquemas por encima del promedio reciben un incremento en el número de cadenas en la siguiente generación y esquemas por abajo del promedio reciben un decremento en el número de cadenas, y un esquema promedio permanece en el mismo nivel.

Si se asume que el esquema S permanece por encima del promedio por un $\epsilon\,\%$, por ejemplo $Apt(S,t) = \overline{F(t)} + \epsilon.\overline{F(t)}$, entonces:

$$\xi(S,t) = \xi(S,0)(1+\epsilon)^t,$$

y $\epsilon = (Apt(S,t) - \overline{F(t)})/\overline{F(t)}$ ($\epsilon > 0$ para esquemas por encima del promedio $\epsilon < 0$ para esquemas por debajo del promedio).

Esta es una ecuación de progresión geométrica y se puede decir que los esquemas por encima del promedio reciben un incremento en el número de cadenas en la siguiente generación, sino que reciben un incremento *exponencial* en el número de cadenas en las siguientes generaciones.

La ecuación 1.16 se le conoce como *Ecuación de Desarrollo Reproductivo de Esquemas*. Volviendo al ejemplo del esquema S_0 del cual se derivan las cadenas v_{13}, v_{15}, v_{16}, y la aptitud promedio es:

$$Apt(S_0,t) = (27{,}316702 + 30{,}060205 + 23{,}867227)/3 = 27{,}081378.$$

Al mismo tiempo, el promedio de aptitud de la población entera es:

$$\overline{F(t)} = \sum_{i=1}^{20} Apt(v_i)/Tpob = 387{,}776822/20 = 19{,}388841,$$

y el cociente de la aptitud del esquema S_0 sobre el promedio de aptitud de la población es:

$$Apt(S_0,t)/\overline{F(t)} = 1{,}396751.$$

Esto significa que el esquema S_0 permanece por encima del promedio, el cual implica que recibirá un incremento exponencial en el número de cadenas en las siguientes generaciones. Si el esquema S_o permanece constante por encima del promedio por el factor 1.396751 en el tiempo $t+1$, entonces se espera tener $3 \times 1\texttt{.}396751 = 4\texttt{.}190253$ cadenas derivados de S_0. En el tiempo $t+2$, se tendrían $3 \times 1\texttt{.}396751^2 = 5{,}85$, etc.

Se puede intuír que el esquema S_0 define una parte promisoria del espacio de búsqueda y se puede muestrear de una manera exponencial. Veamos estas predicciones para el esquema S_0 después del proceso de selección con la misma población anterior tenemos las siguientes cadenas:

$v_1' = (01100111111011010110000110111100 0)$ (v_{11})
$v_2' = (10001100010110100111100000111001 0)$ (v_4)
$v_3' = (00100010000011010111101101111101 1)$ (v_7)
$v_4' = (01100111111011010110000110111100 0)$ (v_{11})
$v_5' = (00010101001111111111100001100011 00)$ (v_{19})
$v_6' = (10001100010110100111100000111001 0)$ (v_4)
$v_7' = (11101110110111000010001111101111 0)$ (v_{15})
$v_8' = (00011101100101001101011111100010 1)$ (v_5)
$v_9' = (01100111111011010110000110111100 0)$ (v_{11})
$v_{10}' = (00001000001100100000010101110111 01)$ (v_3)
$v_{11}' = (11101110110111000010001111101111 0)$ (v_{15})
$v_{12}' = (01000000010110001011000000011111 00)$ (v_9)
$v_{13}' = (00010100001001010100101011111101 1)$ (v_6)
$v_{14}' = (10000110000111010001011010110011 1)$ (v_8)
$v_{15}' = (10111001011001111001100010111111 0)$ (v_{20})
$v_{16}' = (11100110011000010100010001010000 1)$ (v_1)
$v_{17}' = (11100110011000010000010101011101 1)$ (v_{10})
$v_{18}' = (11101111101000100011000000100011 0)$ (v_{13})
$v_{19}' = (11101110110111000010001111101111 0)$ (v_{15})

$v_{20}^{'} = (11001111000001111110000 1101001011)\ (v_{16})$

Ahora el esquema S_0 en el tiempo $(t+1)$ derivó 5 cadenas: $v_7^{'}$, $v_{11}^{'}$, $v_{18}^{'}$, $v_{19}^{'}$ y $v_{20}^{'}$. Sin embargo, la selección no introduce nuevos puntos para consideración del espacio de búsqueda, de lo contrario asigna algunas copias para formar una población intermedia. Analizaremos el segundo paso del ciclo evolutivo: la recombinación. Consideremos la cadena $v_{18}^{'}$:

(11101111101000100011000000 1000110),

deriva 2^{30} esquemas, en particular, la cadena anterior se deriva de los dos esquemas siguientes:

$S_0 = (****111**********************)$
$S_1 = (111**********************10)$

Cruzando las siguientes cadenas en el punto de cruza 20 de la siguiente manera:

$v_{18}^{'} = (1110\mathbf{111}1101000100011|0000001000110)$,
$v_{13}^{'} = (0001010000100101010 0|1010111111011)$,

produciendo los siguientes hijos:

$v_{18}^{''} = (1110\mathbf{111}1101000100011101011111 1011)$,
$v_{13}^{''} = (0001010000100101010 0000000001000110)$.

Es claro que el esquema S_0 sobrevive a la cruza ya que uno de los hijos deriva de él. La razón es que el punto de cruza preserva la cadena **111** en la posición 15, 16 y 17. De lo contrario, el esquema S_1 se destruye porque ninguno de los hijos deriva de él. La definición de longitud de un esquema juega un rol significativo en la probabilidad de destrucción y sobrevivencia. Se nota que la definición de longitud del esquema S_0 es de $\delta(S_0) = 2$ y la definición de longitud del esquema S_1 es de $\delta(S_1) = 32$.

En general un punto de cruza se selecciona uniformemente entre $m-1$ posibles sitios. Esto implica que la probabilidad de destrucción de un esquema S es:

$$p_d(S) = \frac{\delta(S)}{m-1},$$

y la probabilidad de que el esquema sobreviva es:

$$p_s(S) = 1 - \frac{\delta(S)}{m-1}.$$

Por lo tanto las probabilidades de sobrevivencia y destrucción del esquema S_0 y S_1 son:

$$p_d(S_0) = 2/32,\ p_s(S_0) = 30/32,\ p_d(S_1) = 32/32 = 1,\ P_d(S_1) = 0.$$

Es importante notar que solamente algunos cromosomas son afectados por el operador de cruza ya que se utiliza una probabilidad selectiva de cruzamiento p_c. Entonces esto significa que la probabilidad de que un esquema sobreviva es en efecto:

$$p_s(S) = 1 - p_c \cdot \frac{\delta(S)}{m-1}.$$

Tomando en cuenta que tenemos una probabilidad de cruza de $p_c = 0{,}25$ la nueva probabilidad de sobrevivencia del esquema S_0 es:

$$p_s(S_0) = 1 - 0{,}25 \cdot \frac{2}{32} = 63/64 = 0{,}984375.$$

Si seleccionamos cada sitio de cruza entre posiciones fijas en un esquema, hay una pequeña posibilidad de que un esquema sobreviva, por ejemplo si las cadenas $v_{18}^{'}$ y $v_{13}^{'}$ ambas comenzaran con 111 y terminaran con 10 entonces el esquema S_1 podría sobrevivir a la cruza. Debido a esto se puede modificar la fórmula para la probabilidad de que un esquema sobreviva:

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(S)}{m-1}.$$

Así el efecto combinado de la selección y la cruza dan una nueva forma de la *Ecuación de Desarrollo Reproductivo de Esquemas* 1.17

$$\xi(S, t+1) \geq \xi(S, t) \cdot Apt(S, t)/\overline{F(t)} \left[1 - p_c \cdot \frac{\delta(S)}{m-1}\right]. \tag{1.17}$$

La ecuación 1.17 dice que el número esperado de cadenas derivados del esquema S en la siguiente generación está en función del número actual de cadenas derivados del esquema, la aptitud relativa del esquema y su definición de longitud. Es claro que esquemas por encima del promedio con definición de longitud corta pueden ser muestreados en forma exponencial. Por ejemplo para el esquema S_0 se tiene:

$$Apt(S_0, t)/\overline{F(t)} \left[1 - p_c \cdot \tfrac{\delta(S)}{m-1}\right] = 1{,}396751 \cdot 0{,}984375 = 1{,}374927.$$

Esto significa que el esquema S_0 puede recibir un incremento exponencial en el número de cadenas en las siguientes generaciones.

El siguiente operador considerado es la mutación, y recordemos que este operador cambia una simple posición de un cromosoma dada una probabilidad de mutación p_m. Es claro que todas las posiciones fijas de un esquema pueden permanecer sin cambios si el esquema sobrevive a la mutación. Por ejemplo, consideremos nuevamente una cadena simple de la población anterior $v_{19}^{'}$:

(11101110110111000010001111111011110)

y el esquema S_0:

$S_0 = (****111**********************)$.

Asumiendo que la cadena $v_{19}^{'}$ sufre la mutación en al menos un bit en la octava posición de la siguiente manera:

$v_{19}^{'} = (11101110\mathbf{0}10111000010001111111011110)$

El quinto, el sexto y el séptimo bits son importantes; pues la mutación en cualquiera de estos bits destruiría el esquema S_0. Es claro que el número de bits importantes en el esquema S_0 es igual al orden del esquema, o sea el número de posiciones fijas.

Dado que la probabilidad de alteración de un bit es p_m, entonces la probabilidad de que un simple bit sobreviva es $1-p_m$. Así la probabilidad de que un esquema S sobreviva a la mutación es:

$$p_s(S) = (1-p_m)^{o(S)}.$$

Dado que la $p_m \ll 1$, esta probabilidad puede ser aproximada por:

$$p_s(S) \approx 1 - o(S) \cdot p_m.$$

Volviendo al ejemplo del esquema S_0 y suponiendo que se tiene una $p_m = 0{,}01$, entonces la probabilidad de que el esquema S_0 sobreviva es:

$$p_s(S_0) \approx 1 - 3 \cdot 0{,}01 = 0{,}97.$$

Combinando el efecto de la selección, la cruza y la mutación se tiene una nueva forma de la *Ecuacion de Desarrollo Reproductivo de Esquemas* 1.18

$$\xi(S, t+1) \geq \xi(S,t) \cdot Apt(S,t)/\overline{F(t)} \left[1 - p_c \cdot \frac{\delta(S)}{m-1} - o(S) \cdot p_m\right]. \tag{1.18}$$

Como ya se mencionó en la ecuación 1.16, 1.17 y la ecuación 1.18 dicen que el número esperado de cadenas derivadas del esquema S en la siguiente generación está en función del número actual de cadenas, la aptitud relativa del esquema y su definición de longitud y orden. Nuevamente esquemas por encima del promedio con corta definición de longitud y bajo orden se muestrean con incrementos exponenciales. Se ilustra de nueva cuenta el ejemplo del esquema S_0:

$$Apt(S_0,t)/\overline{F(t)} \left[1 - p_c \cdot \tfrac{\delta(S)}{m-1} - o(S_0) \cdot p_m\right] = 1{,}396751 \cdot 0{,}954375 = 1{,}333024.$$

Esto significa que esquemas de bajo orden, por encima del promedio del esquema S_0 recibirán incrementos exponenciales en el número de cadenas en las siguientes generaciones. Por ejemplo en el tiempo $(t+1)$ esperamos tener $3{\times}1.333024 \approx 4.00$ esquemas derivados del esquema S_0 y en el tiempo $(t+2)$ $3{\times}1{,}333024^2 = 5{,}33$ cadenas.

El resultado final de la ecuación 1.18 que también se le conoce como *Toerema Fundamental de los Algoritmos Genéticos* Michalewicz[39], puede ser escrita de la siguiente manera:

Teorema del Esquema. *Esquemas cortos, de bajo orden, con aptitud por encima del promedio reciben un incremento exponencial de cadenas descendientes en subsecuentes generaciones de un algoritmo genético.*

Un resultado inmediato de este teorema es que los algoritmos genéticos exploran el espacio de búsqueda con esquemas cortos de bajo orden y esta información se usa en el proceso de

cruzamiento para guiar el proceso de búsqueda.

2. DISEÑO Y EXPERIMENTACIÓN NUMÉRICA.

2.1. Diseño y programación del algoritmo genético simple.

En este capítulo se hace uso de los diferentes operadores del algoritmo genético analizados en el capítulo 1 para describir la estructura general del programa computacional. Dicho programa se programó en Digital Visual Fortran 6.5 con los siguientes parámetros: (tipo de representación : $\mathbb{R}$, tamaño de población 100; asignación de aptitud: ranking lineal; métodos de selección: ruleta, SUS, ranking lineal, ranking exponencial y torneo; métodos de cruza: discreta, intermedia y lineal; método de mutación: $\mathbb{R}$) y su estructura general se muestra en la tabla 2.1.

```
Program Algoritmo_Genético;
    declaracion_de_variables;
    llamar lectura_de_datos();
    llamar poblacion_inicial();
    hacer i=1, gen
        llamar ordena();
        llamar met_de_seleccion();
        llamar met_de_cruza();
        llamar mutacion();
    fin hacer
fin.
```

Tab. 2.1: Algoritmo genético simple para optimización global.

A continuación se describe línea por línea el pseudocódigo mostrado en la tabla 2.1.

La primera línea representa el encabezado de cualquier programa en Fortran en donde se describe el nombre del programa. La siguiente línea corresponde al área de declaración de variables que se usan en el programa, sus tipos, dimensiones, etc. La siguiente línea corresponde a la subrutina que lee los datos de entrada del programa como son: dimensión del problema, intervalo de búsqueda, tamaño de población, número de generaciones, presión de selección, tipo de selección, tipo de cruza, probabilidad de cruza, probabilidad de mutación, etc. La siguiente línea corresponde a la subrutina que crea la población aleatoria inicial. Posteriormente comienza un ciclo que inicia de 1 hasta el número de generaciones (gen) el cual corresponde al ciclo evolutivo del algoritmo genético. La siguiente línea corresponde a la subrutina de ordenación de la población, la cual sirve para realizar la asignación de aptitud de la población y saber la convergencia de nuestro algoritmo. La siguiente línea corresponde a la subrutina de la selección la cual llama al método de selección previamente determinado en la lectura de datos. Básicamente en la subrutina de método de selección se realiza el proceso de asignación de aptitud, asignación de la probabilidad de selección y selecciona a los individuos padres. La siguiente línea corresponde al método de cruza el cual llama al método determinado en la lectura de datos; básicamente en

la subrutina de método de cruza se obtienen a los hijos segun los parámetros del método. La siguiente línea corresponde a la subrutina para ejecutar el operador de mutación, con sus diferentes parámetros, a los individuos que sean seleccionados. En esta misma subrutina se agregan los individuos que pasan a la siguiente generación por medio de elitismo. La penúltima línea incrementa el ciclo evolutivo del algoritmo hasta alcanzar el número de generaciones. No existe un criterio teórico para detener el algoritmo Goldbert[22], pero generalmente se hace dado un determinado número de generaciones o después de que el valor de nuestra función objetivo no ha mejorado en un cierto número de generaciones.

2.2. Descripción de los problemas de pruebas académicas.

En esta sección se describen las diferentes funciones de prueba, las cuales son consideradas como funciones de optimización global típicas para los algoritmos de optimización que utilizan información de derivadas (gradiente) y que se describen en la tabla 2.2.

Nombre	Función	Cotas	# de Variables
Rosenbrock	$f(x_i) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	$-2{,}048 \leq x_i \leq 2{,}048$	$i = 1, 10$
Schaffer	$f(x_i) = 0{,}5 - \frac{(sin\sqrt{x^2+y^2})^2 - 0{,}5}{(1{,}0+0{,}001(x^2+y^2))^2}$	$-30 \leq x_i \leq 30$	$i = 1, 10$
Rastrigin	$f(x_i) = 10n * \sum_{i=1}^{n} x_i^2 - 10cos(2\pi x_i)$	$-5{,}12 \leq x_i \leq 5{,}12$	$i = 1, 10$
Griewangk	$f(x_i) = \frac{\sum_{i=1}^{n} x_i^2}{4000} - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-100 \leq x_i \leq 100$	$i = 1, 10$
Schubert	$f(x_i) = \{\sum_{i=1}^{5} icos[(i+1)x_1 + i]\}\{\sum_{i=1}^{5} icos[(i+1)x_2 + i]\}$	$-10 \leq x_i \leq 10$	$i = 1, 2$
Six-Hump-Camelback	$f(x_i) = [4 - 2{,}1x_1^2 + \frac{1}{3}x_1^4]x_1^2 + x_1x_2 + [-4 + 4x_2^2]x_2^2$	$-3 \leq x_i \leq 2$	$i = 1, 2$

Tab. 2.2: Funciones de prueba.

2.3. Clasificación de los problemas de pruebas académicas.

En esta sección se clasifican los problemas de prueba de la sección 2.2. Según la forma de los mínimos locales o globales, o si son planos o muy estrechos como se muestra en la tabla 2.3.

Varios Mínimos Globales	Varios Mínimos Locales	Mínimos muy Estrechos
Schubert	Rastrigin	Rosenbrock
Six_Hump_Camelback	Griewangk	Schaffer

Tab. 2.3: Clasificación de funciones de prueba.

Clasificar las funciones de la tabla 2.3 es para tomar en cuenta los diferentes tipos de valles que se pueden presentar en un problema de optimización y de esta manera tomar decisiones sobre

los parámetros adecuados a usar en el momento de implantar el algoritmo genético.

2.4. Clasificación de parámetros del algoritmo genético simple.

Para analizar los distintos tipos de problemas que se describen en la sección 2.3 también se necesita hacer una clasificación de los distintos parámetros del algoritmo genético. Como son presión de selección, probabilidad de cruza, probabilidad de mutación. Dichos parámetros se clasifican en *baja*, *media* y *alta*. Estos parámetros se prueban con los diferentes operadores de selección y cruza y se determinan los parámetros y sus operadores con mejores resultados. Para que a la hora de resolver el problema de aplicación el valor ha tomar de estos no sea tan empírica. La tabla 2.4 muestra la clasificación de parámetros.

Parámetro	Baja	Media	Alta
SP	1.3	1.6	2.0
PC	0.4	0.7	0.9
PM	0.01	0.05	0.1

Tab. 2.4: Clasificación de parámetros.

En la tabla 2.4 SP corresponde a la presión de selección, PC a probabilidad de cruza y PM probabilidad de mutación.

2.5. Pruebas del algoritmo genético simple.

Se ejecutaron corridas de 1000 generaciones en los diferentes casos de funciones de prueba para evaluar los operadores y sus parámetros, obteniéndose los siguientes resultados que se muestran en la tabla 2.5:

VARIOS MINIMOS GLOBALES						
Tipo_Selección	Tipo_Cruza	PC	PM	SP	Resultado	Función
TODOS	CRUZAI y CRUZAL	T	T	T	-186.730898	Schubert
TODOS	CRUZAI y CRUZAL	T	T	T	-1.03162876	Six_Hump_Camelback
VARIOS MINIMOS LOCALES						
RL	CRUZAI	M	M	M	0.082646845	Rastrigin
SUS	CRUZAI	M	M	M	0.04002313	Griewangk
MINIMOS MUY ESTRECHOS						
RE	CRUZAI	M	M	M	0.009715909	Schaffer
RL	CRUZAI	M	M	M	0.239111541	Rosenbrock

Tab. 2.5: Resultados del algoritmo genético simple.

Para interpretar la tabla 2.5 es necesario definir las variables como sigue: (Tipo_Selección= tipo de selección, Tipo_Cruza= tipo de cruza, PC= probabilidad de cruza, PM= probabilidad de mutación, SP= presión de selección, RL= ranking lineal, SUS= muestreo estocástico universal, CRUZAI= cruza intermedia, CRUZAL= cruza lineal, M= media, B= baja, A= alta). La columna de Resultado se refiere al valor obtenido de la función objetivo después de 1000 generaciones y los valores óptimos globales exactos para los diferentes funciones de pruebas son: Schubert

= -186.730908, Six_Hump_Camelback= -1.0316285, Rastrigin= 0.00000, Griewangk= 0.00000, Schaffer= 0.00000, Rosenbrock= 0.00000.

Se puede resumir la tabla 2.5 en la tabla 2.6:

VARIOS MINIMOS GLOBALES				
Tipo_Selección	Tipo_Cruza	PC	PM	SP
TODOS	CRUZAI y CRUZAL	T	T	T
TODOS	CRUZAI y CRUZAL	T	T	T
VARIOS MINIMOS LOCALES				
RL	CRUZAI	M	M	M
SUS	CRUZAI	M	M	M
MINIMOS MUY ESTRECHOS				
RE	CRUZAI	M	M	M
RL	CRUZAI	M	M	M

Tab. 2.6: Resumen de resultados del algoritmo genético simple.

Observando la tabla 2.6 se nota que para funciones con varios mínimos globales es indiferente usar uno u otro método y sus parámetros.

Para funciones con varios mínimos locales los mejores métodos de selección fuerón el ranking lineal y el SUS con método de cruza intermedia, probabilidades de cruza y mutación media y presión de selección de media.

Para valles muy estrechos los mejores métodos de selección fueron el ranking exponencial y el ranking lineal con método de cruza intermedia y probabilidades de cruza y mutación de media y presión de selección de media.

Haciendo un conteo y resumiendo aun más la tabla 2.6 en la tabla 2.7,

Tipo_Selección	Tipo_Cruza	PC	PM	SP
RL	CRUZAI	M	M	M

Tab. 2.7: Resumen del mejor operador y sus parámetros.

nuestros resultados muestran que el mejor método de selección para los diferentes casos de prueba fué el ranking lineal, y el mejor método de cruza es la intermedia, probabilidad de cruza de media, probabilidad de mutación de media, y SP de media.

3. ALGORITMO GENÉTICO CON SUBPOBLACIONES.

3.1. Descripción del algoritmo genético con subpoblaciones.

Uno de los principales motivos para inplantar algoritmos genéticos con subpoblaciones es el de explorar al máximo el espacio de búsqueda, conservar la diversidad y acelerar la convergencia del algoritmo, principalmente en funciones donde se tiene que remontar muchos valles. El modelo más básico de algoritmo genético con subpoblaciones es el modelo de *islas* o *demes* Bäck[3] que es considerado en este trabajo. En el modelo de islas la población se dividide en n subpoblaciones aisladas que evolucionan independientemente realizando su propio algoritmo genético. En este trabajo cada subpoblación tiene un genético con diferentes operadores. Después de un número de generaciones llamado *época* las distintas subpoblaciones realizan la migración de individuos entre las subpoblaciones, que consiste en intercambiar individuos entre las distintas subpoblaciones. La cuestión en esta fase es: qué individuos migrar, hacia qué subpoblaciones migrar, qué porcentaje de individuos serán migrados y a cuáles individuos reemplazar en las distintas subpoblaciones. A pesar de que ya existen estudios realizados Cantú-Paz[15], para determinar estos distintos parámetros óptimos, aún quedan muchos aspectos que actualmente están siendo discutidos. Un aspecto muy importante en el proceso de migración es la topología que se describe en la siguiente sección.

3.2. Topologías de subpoblaciones.

Las topologías de algoritmos genéticos con subpoblaciones dicen hacia donde se migran los individuos. Básicamente existen tres tipos, Chipperfield[17].

Anillo: En este tipo de topología las subpoblaciones migran a los individuos a su vecino más próximo por la derecha como se muestra en la figura 3.1.

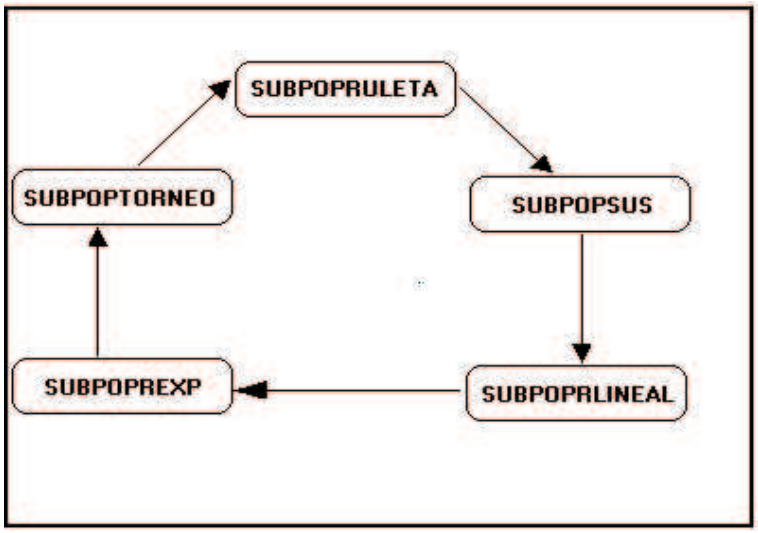

Fig. 3.1: Topología en anillo.

En la figura 3.1 el óvalo que dice SUBPOPRULETA corresponde a la subpoblación ruleta el cual migra un cierto porcentaje de individuos a la subpoblación SUS (SUBPOPSUS) que migra un cierto porcentaje de individuos a la subpoblación ranking lineal (SUBPOPRL), el cual a su vez migra un cierto porcentaje de individuos a la subpoblación ranking exponencial (SUBPOPREXP) que también migra un cierto porcentaje de individuos a la subpoblación torneo (SUBPOPTORNEO), que por último migra un cierto porcentaje de individuos a la subpoblación ruleta para terminar el proceso de migración en anillo.

Vecindad: En este tipo de topología cada subpoblación migra un cierto porcentaje de individuos a sus vecinos más cercanos tanto a la izquierda como a la derecha como se muestra en la figura 3.2.

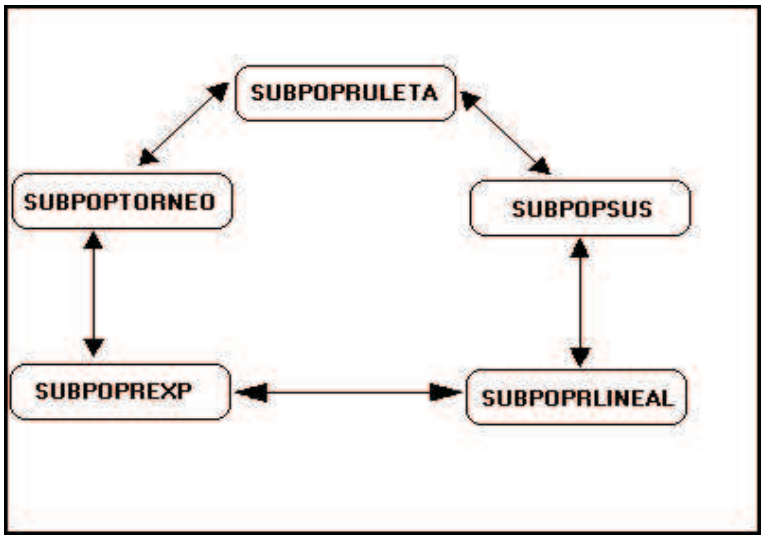

Fig. 3.2: Topología en vecindad.

En la figura 3.2 vemos como las distintas subpoblaciones migran a los individuos en dos direcciones al mismo tiempo que reciben individuos de sus diferentes vecinos.

No restringida: En la topología no restringida las subpoblaciones migran un cierto porcentaje de individuos hacia todas las subpoblacioes sin restricción alguna, como se muestra en la figura 3.3.

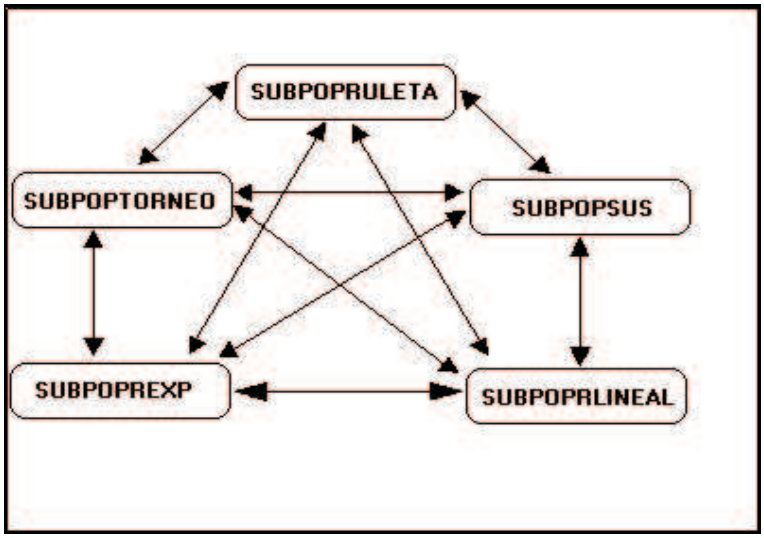

Fig. 3.3: Topología no restringida.

Como podemos observar en la figura 3.3 las subpoblaciones intercambian migrantes entre todas las subpoblaciones sin restricción.

En la siguiente sección se analiza el efecto de usar uno u otro tipo de topología al realizar la experimentación numérica con el fin de determinar la topología con mejores resultados.

3.3. Diseño del algoritmo genético con subpoblaciones.

A continuación se presenta el diseño del algoritmo genético con subpoblaciones que simula el proceso paralelo, ya que realmente el programa se ejecuta en un microprocesador. La tabla 3.1 muestra la estructura básica del algoritmo genético con subpoblaciones.

```
Program Algoritmo_Genético_con_Subpoblaciones;
    declaracion_de_variables;
    llamar lectura_de_datos();
    llamar poblacion_inicial();
    hacer epoca=1, n
        llamar subpoblacion_ruleta();
        llamar subpoblacion_SUS();
        llamar subpoblacion_rl();
        llamar subpoblacion_re();
        llamar subpoblacion_torneo();
        llamar migracion();
    fin hacer
fin.
```

Tab. 3.1: Algoritmo genético con subpoblaciones.

Explicaremos brevemente la tabla 3.1 que es muy parecida a la del capítulo 2. Las cuatro primeras líneas son principalmente para declaración de variables, lectura de datos y creación de la población inicial. En la quinta línea de la tabla comienza el ciclo evolutivo del algoritmo genético con subpoblaciones. Dado que el algoritmo genético con subpoblaciones opera por medio de migraciones y la migración se realiza cada cierto número de generaciones, entonces se introduce el concepto de época para llevar acabo la migración de individuos. Las cinco líneas siguientes corresponden a las subpoblaciones con un operador de selección diferente. Dado que la cruza intermedia y la lineal dieron mejores resultados en el capítulo 2 se decidió utilizar este operador de cruza para todas las subpoblaciones. Cada subpoblación realiza un algoritmo genético como se describió en el capítulo 1 y la migración se realiza cada época de 20 generaciones.

3.4. Experimentación numérica del algoritmo genético con subpoblaciones.

Para analizar el algoritmo genético con subpoblaciones se utilizaron las mismas funciones de prueba del capítulo 2 con el mismo tipo de clasificación y la misma clasificación de parámetros. Se utilizarón 5 subpoblaciones diferentes con un método de selección diferente, cruza intermedia y lineal, mutación real. El programa se ejecutó en 50 épocas y cada subpoblación realizó 1000 generaciones obteniéndose los siguientes resultados que se muestran en la tabla 3.2.

VARIOS MINIMOS GLOBALES			
Topología	Subpoblación	Resultado	Función
TODAS	TODAS	-186.730908	Schubert
TODAS	TODAS	-1.0316284	Six_Hump_Camelback
VARIOS MINIMOS LOCALES			
TODAS	TORNEO y RULETA	0.0085457211	Rastrigin
TODAS	REXP y RULETA	0.0013619244	Griewangk
MINIMOS MUY ESTRECHOS			
TODAS	REXP	0.0003312995	Rosenbrock
TODAS	TODAS	0.0097159098	Schaffer

Tab. 3.2: Resultados del algoritmo genético con subpoblaciones.

Observando la tabla 3.2 vemos que para funciones con varios mínimos globales todas las topologías en las diferentes subpoblaciones obtuvieron el mismo resultado. En las funciones con varios mínimos locales nuevamente todas las topologías obtuvieron el mismo resultado en casi todas las subpoblacines. Con valles muy estrechos nuevamente todas las topologías obtuvieron el mismo resultado en casi todas las subpoblaciones.

Como podemos observar en los resultados no hay diferencia de acuerdo a los datos utilizados en usar uno u otro tipo de topología. En cuanto a los métodos de selección usados en las diferentes subpoblaciones segun los resultados de la tabla 3.2 son la ruleta y el ranking exponencial.

3.5. Comparación entre el algoritmo genético simple con el de subpoblaciones.

Observando la columna de resultado en la tabla 2.5 del capítulo 2 y la columna de Resultados en la tabla 3.2 de este capítulo, se nota que para funciones con varios mínimos globales; tanto el genético simple como el de subpoblaciones obtuvieron el mismo resultado en los dos casos de prueba.

Para funciones con varios mínimos locales nuevamente observando las dos tablas mencionadas en el parrafo anterior, la precisión en el resultado obtenido por el genético con subpoblaciones fué de al menos una décima mejor contra el resultado obtenido por el genético simple en las dos funciones de prueba.

Para funciones con valles muy estrechos la precisión obtenida por el genético con subpoblaciones fué de una milésima con respecto al resultado obtenido por el genético simple en al menos una de las funciones de prueba.

4. APLICACION DEL AG CON SUBPOBLACIONES A PROBLEMAS PETROLEROS.

4.1. Descripción del problema de caracterización de yacimientos petroleros naturalmente fracturados.

Es bien sabido Fuentes[21] que el mayor porcentaje de la producción de hidrocarburos a nivel mundial, proviene de yacimientos naturalmente fracturados (YNF) que tienen un complejo sistema de poros. A menudo, el comportamiento de estos sistemas se representan con modelos de doble porosidad Fuentes[21]. Sin embargo, cuando un caso en particular exhibe además de fracturas naturales la presencia de vúgulos, los modelos tradicionales no son suficientes para captar algunas características asociadas a la presencia de la porosidad vugular. Así, estudios recientes han demostrado la influencia definitiva que la presencia de los vúgulos tiene sobre el comportamiento de presión y las curvas de descenso en YNF. En nuestro país existen muchos pozos naturalmente fracturados vugulares y el efecto de la porosidad de la matriz, las fracturas y los vúgulos, originan que en estos yacimientos se presenten características particulares en las curvas de descenso de presión registrados.

Para poder predecir la producción en un pozo es necesario conocer las curvas de caida de presión en el tiempo y el espacio del pozo. Esto se puede obtener a través de un modelo matemático de difusión del flujo de petróleo Velázquez[13]. Sin embargo, para poder resolver el modelo es necesario conocer un conjunto de coeficientes (parámetros) que caracterizan el tipo de medio poroso alrededor del pozo. Dichos coeficientes varían de un pozo a otro y de un yacimiento a otro, y son en general desconocidos.

En este trabajo, se describe una metodología para encontrar estos coeficientes resolviendo un problema de optimización. Los métodos clásicos del tipo Newton, que necesitan información de las derivadas, no funcionan correctamente debido a la presencia de valles muy profundos en la función objetivo y de valles muy planos debido al mal condicionamiento de las matrices hessianas (Gradientes muy grandes y por tanto tamaños de paso muy pequeños que trae como consecuencia el paro del método).

Existen principalmente tres tipos de porosidad en YNF: de la matríz, de las fracturas y de los vúgulos.

La porosidad en la **matríz** es una mezcla de microfracturas intercristalinas adyacentes a la zona de vúgulos.

Las **fracturas** se definen como superficies planas de discontinuidad, en donde la roca ha perdido cohesión y los procesos de deformación y alteración de la misma pueden ser ocupadas por fluidos. Desde una perspectiva general se manejan diversos modelos para representar los medios fracturados. En el más simple se consideran bloques de roca, separados por planos de ancho variable, representando fracturas. En este modelo se considera que las fracturas tienen poca influencia sobre la porosidad de las formaciones y alta repercusión en la permeabilidad del sistema, así la capacidad de desplazamiento de fluidos está controlada por las fracturas, mientras

los bloques de la matriz se relacionan con la capacidad de almacenamiento. El modelo explica, de manera simple, los yacimientos naturalmente fracturados; sin embargo, debe ser readaptado para las formaciones que se encuentran en territorio mexicano, pues adicionalmente se presenta un sistema de cavernas de disolución que altera de manera notable al modelo anterior.

Los **vúgulos** son el resultado de la disolución de carbonatos y sulfatos. La porosidad vugular es común en muchos yacimientos de carbono dada su importancia petrofísica y han sido reconocidos en trabajos recientes debido a la gran influencia que tienen en la producción petrolífera. La modelación de un YNF vugular que permite flujo primario a través de los vúgulos involucra desarrollar un método para identificarlos usando pruebas de pozo y curvas de descenso de presión, evaluación de porosidad asociado con los vúgulos y las fracturas y la determinación de la conectividad entre ellas.

Los simuladores de porosidad dual consideran solamente la matriz y el sistema de fracturas, donde el petróleo se produce a través del sistema de fracturas. El modelo de triple-porosidad permeabilidad simple fue propuesto por Abdassah y Ershaghi [13]. Estos autores consideraron un modelo de flujo interporoso pseudoestacionario entre el sistema de fracturas, dos tipos de bloques de matriz y flujo primario solamente a través del sistema de fracturas. Dado que los vúgulos interactuan con la matriz y el sistema de fracturas, entonces la porosidad y la permeabilidad de los vúgulos y el sistema de fracturas pueden ser muy diferentes y es muy importante reconocer que la porosidad de los vúgulos cambian las propiedades del yacimiento. Por estas razones, es necesario distinguir entre estos dos sistemas y modelar su comportamiento como dos sistemas separados pero interactuantes. Por estos comentarios, es importante establecer un modelo de triple porosidad, que permita una interacción entre la matriz, las fracturas y los vúgulos, donde el flujo primario ocurre solamente a través de la red de fracturas y el sistema de vúgulos. Asi el modelo puede ser un modelo de permeabilidad simple de triple porosidad.

Considerando una simetría cilindrica, la ecuación diferencial para el coeficiente de almacenamiento de las fracturas, usando variables adimensionales, para el modelo de permeabilidad simple de triple porosidad, está dada por: (ver ref.[13]).

$$k\frac{1}{r_D}\frac{\partial}{\partial r_D}\left(r_D\frac{\partial p_{Df}}{\partial r_D}\right)+\lambda_{mf}\left(p_{Dm}-p_{Df}\right)+\lambda_{vf}\left(p_{Dv}-p_{Df}\right)=\omega_f\frac{\partial p_{Df}}{\partial t_D} \tag{4.1}$$

Para los bloques de matriz el coeficiente de almacenamiento está dada por:

$$-\lambda_{mv}\left(p_{Dm}-p_{Dv}\right)-\lambda_{mf}\left(p_{Dm}-p_{Df}\right)=\left(1-\omega_f-\omega_v\right)\frac{\partial p_{Dm}}{\partial t_D} \tag{4.2}$$

Y para los vúgulos el coeficiente de almacenamiento está dada por:

$$\left(1-k\right)\frac{1}{r_D}\frac{\partial}{\partial r_D}\left(r_D\frac{\partial p_{Dv}}{\partial r_D}\right)+\lambda_{mv}\left(p_{Dm}-p_{Dv}\right)-\lambda_{vf}\left(p_{Dv}-p_{Df}\right)=\omega_v\frac{\partial p_{Dv}}{\partial t_D} \tag{4.3}$$

Donde:

ω_f = coeficiente de almacenamiento (fracturas)
ω_v = coeficiente de almacenamiento (vúgulos)
λ_{mf} = coeficiente del flujo interporoso (entre matriz y fracturas)
λ_{mv} = coeficiente del flujo interporoso (entre matriz y vúgulos)
λ_{vf} = coeficiente del flujo interporoso (entre vúgulos y fracturas)

Y las variables adimensionales estan dadas por:

$$p_{Dj} = 2\pi \left(k_f + k_v\right) h \left(p_i - p_j\right) / \left[q\mu B\right]$$

donde j = fracturas o vugulos, y

$$t_D = \left(k_f + k_v\right) t / \left[\left(\phi_f c_f + \phi_m c_m + \phi_v c_v\right) \mu r_m^2\right]$$
$$k = k_f / \lfloor k_f + k_v \rfloor$$
$$\lambda_{mf} = \sigma_{mf} k_m r_w^2 / \left[k_f + k_v\right]$$
$$\lambda_{mv} = \sigma_{mv} k_m r_w^2 / \left[k_f + k_v\right]$$
$$\lambda_{vf} = \sigma_{vf} k_{vf} r_w^2 / \left[k_f + k_v\right]$$

con $k_{vf} = k_v$ si $p_v > p_f$
$k_{vf} = k_f$ en otro caso.

Los cuales están relacionadas con la permeabilidad del medio poroso. σ_{ij} es el factor de flujo interporoso entre el medio i y el j. Los radios de almacenamiento relacionados con la porosidad para los vúgulos y las fracturas estan dadas respectivamente por:

$$\omega_f = \phi_f c_f / \lfloor \phi_f c_f + \phi_m c_m + \phi_v c_v \rfloor$$
$$\omega_v = \phi_v c_v / \lfloor \phi_f c_f + \phi_m c_m + \phi_v c_v \rfloor$$

En el problema de identificación de parámetros se tienen datos medidos en los pozos de **tiempo presión y su derivada** y se trata de encontrar el valor de los coeficientes $(\lambda_{mf}, \lambda_{vf}, \omega_f, \lambda_{mv}, \omega_v)$ = X que al sustituírlos en las ecuaciones 4.1, 4.2, 4.3 la solución del sistema (la presión y su derivada) produzcan una buena aproximación a esos datos dado un margen de error permitida.

Para un juego de valores de los coeficientes (X), la solución del modelo diferencial Mod(X) produce una gráfica para la presión y la derivada en diferentes intervalos de tiempo como se muestra en la figura 4.1:

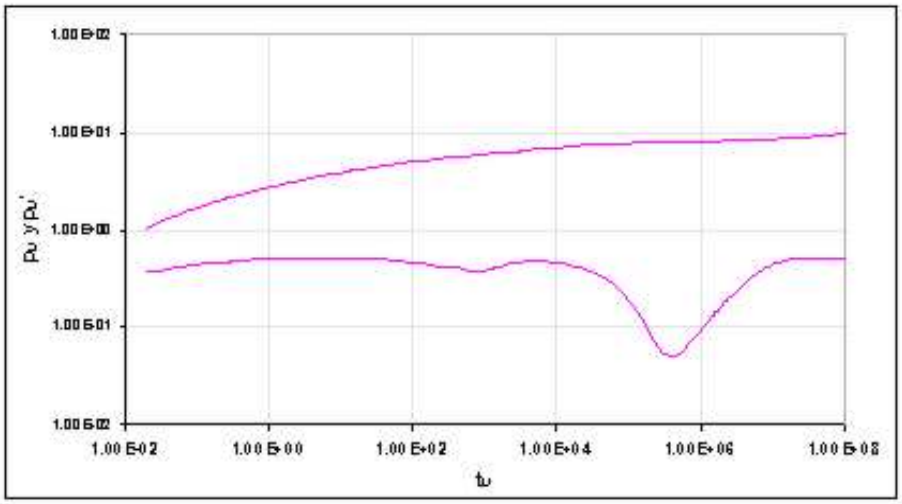

Fig. 4.1: Tiempo-presión-derivada.

Caracterizar un yacimiento tipo es hallar los coeficientes para un conjunto de datos medidos en los pozos (presión y derivada en varios intervalos de tiempo); para los cuales la solución del sistema de ecuaciones diferenciales p = Mod(coef) ajusta lo mejor posible a los pDatos.

Normalmente este proceso se realiza con prueba y error, pero el objetivo de este trabajo es hacerlo en forma automática. Y una forma de resolver este problema es minimizar el error entre el Mod(X) = p(X) y la pDdatos.

La identificación de los coeficientes $X = \omega_v, \omega_f, \lambda_{mf}, \lambda_{mv}, \lambda_{vf}$ que caracterizan al yacimiento se puede realizar rosolviendo un problema de optimización de mínimos cuadrados como:

$$MinF(X) = \|Mod(X) - pDatos\|_2^2 \tag{4.4}$$

Sujeto a restricciones relacionadas con la física del problema (tipo de roca) $X_1 + X_2 \leq 1$ y cotas de $X_{min} \leq X \leq X_{max}$. Mod(X) corresponde a la solución calculada del modelo para un conjunto de coeficientes X.

El ejercicio o problema inverso es entonces dados estos datos (tiempo y presión) encontrar la X. Este problema se resuelve usando el algoritmo genético con subpoblaciones con los paramétros analizados en el capítulo 3.

4.2. Experimentación numérica

Para realizar la experimentación numérica y considerando que el problema a resolver corresponde a problemas con mínimos muy estrechos según la clasificación mencionada en el capítulo 2 y con el objetivo de poner en práctica el algoritmo genético desarrollado en este trabajo, es necesario generar ejemplos en los que se conoce la solución. Para esto, se forma un juego de parámetros que describen a un yacimiento tipo, (por ejemplo doble porosidad) que será la solución exacta y usando ese valor de los coeficientes se resuelve el sistema de ecuaciones diferen-

ciales y se obtiene la presión a diferentes tiempos que se denomina la presión exacta. El ejercicio consiste en tomar esos datos de presión y tiempo para encontrar el valor de los coeficientes que aproximen mejor al orden de magnitud permitida en la solución exacta de los coeficientes. Y que la curva de presión calculada y de la derivada se parezcan a las curvas exactas. Y con los parámetros que se muestran en la tabla 4.1.

Subp.	Topologia	M_Seleccion	M_Cruza	PC	PM	SP	%MIG	%Elitismo
1	No_Restringida	SUS	Intermedia	M	M	M	25	30
2	No_Restringida	Ruleta	Intermedia	M	M	M	25	30
3	No_Restringida	Torneo	Intermedia	M	M	M	25	30
4	No_Restringida	R_Lineal	Intermedia	M	M	M	25	30
5	No_Restringida	R_Exponencial	Intermedia	M	M	M	25	30

Tab. 4.1: Parámetros para el problema de aplicación.

4.3. Presentación de resultados.

Ahora se comparan los resultados obtenidos contra las soluciones exactas para cada caso.

Analizamos el primer modelo de porosidad simple con la solucion exacta, la calculada y las cotas en la solución para considerarla como válida se muestran en la tabla 4.2

Parámetro	Solucion-Exacta	Solucion-Calculada	Cotas
ω_v	0.00E+00	9.843330975E-04	$0.00E{+}00 \leq \omega_v \leq 0.01E{-}02$
ω_f	1.00E+00	8.999999999E-01	$0.90E{+}00 \leq \omega_f \leq 1.00E{+}00$
λ_{mf}	1.00E-05	9.999999185E-04	$1.00E{-}06 \leq \lambda_{mf} \leq 1.00E{-}05$
λ_{vf}	1.00E-07	9.999645235E-06	$5.00E{-}08 \leq \lambda_{vf} \leq 5.00E{-}07$
λ_{mv}	1.00E-07	1.498838694E-07	$5.00E{-}08 \leq \lambda_{mf} \leq 5.00E{-}07$

Tab. 4.2: Comparación numérica del modelo de porosidad simple.

Las gráficas para la solución exacta y la calculada se muestran en la figura 4.2.

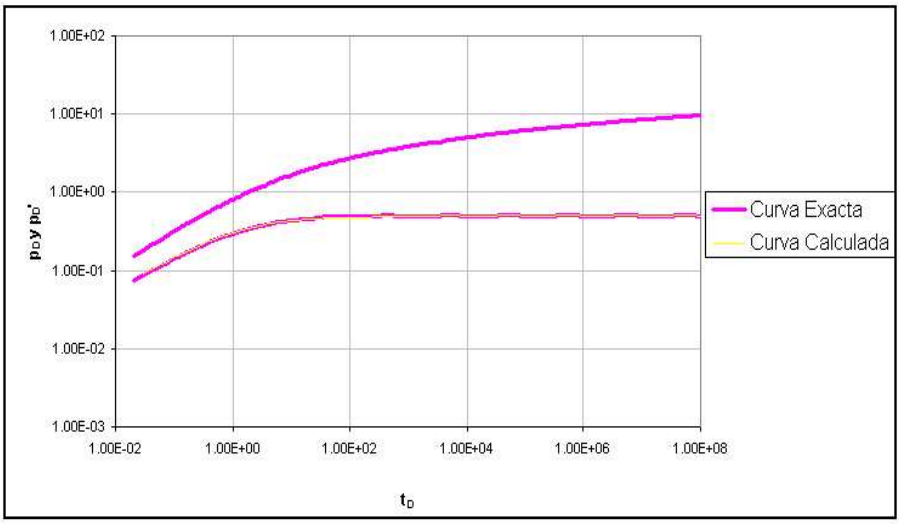

Fig. 4.2: Comparacion gráfica del modelo de porosidad simple

Observando la tabla 4.2 y la figura 4.2 del modelo de porosidad simple donde $\omega_v = 0$ es decir cuando el flujo se presenta únicamente a través del sistema de fracturas, la solución calculada se aproximó a la solución permitido como válido en la mayoría de los coeficientes. El comportamiento gráfico también se aproximó perfectamente a la solución exacta. En la figura 4.2 se grafica la gráfica calculada sobre la gráfica de la solución exacta con el fin de hallar concordancia entre ellas y hacer el análisis de resultados más riguroso

Ahora veamos el modelo de doble porosidad con la solución exacta, la calculada y las cotas donde pueden variar las soluciones, se presentan en la tabla 4.3

.

Parámetro	Solución-Exacta	Solución-Calculada	Cotas
ω_v	0.00E+00	2.12493998E-06	$\omega_v \leq$0.01E-002
ω_f	1.00E-03	9.99332076E-04	1.00E-04 $\leq \omega_f \leq$ 1.00E-03
λ_{mf}	1.00E-05	7.04753833E-06	1.00E-06 $\leq \lambda_{mf} \leq$ 1.00E-05
λ_{vf}	1.00E-07	8.44951107E-06	5.00E-08 $\leq \lambda_{mf} \leq$ 5.00E-07
λ_{mv}	1.00E-07	4.65966377E-06	5.00E-08 $\leq \lambda_{mf} \leq$ 5.00E-07

Tab. 4.3: Comparación numérica del modelo de doble porosidad.

Las gráficas para la solución exacta y calculada se muestran en la figura 4.3.

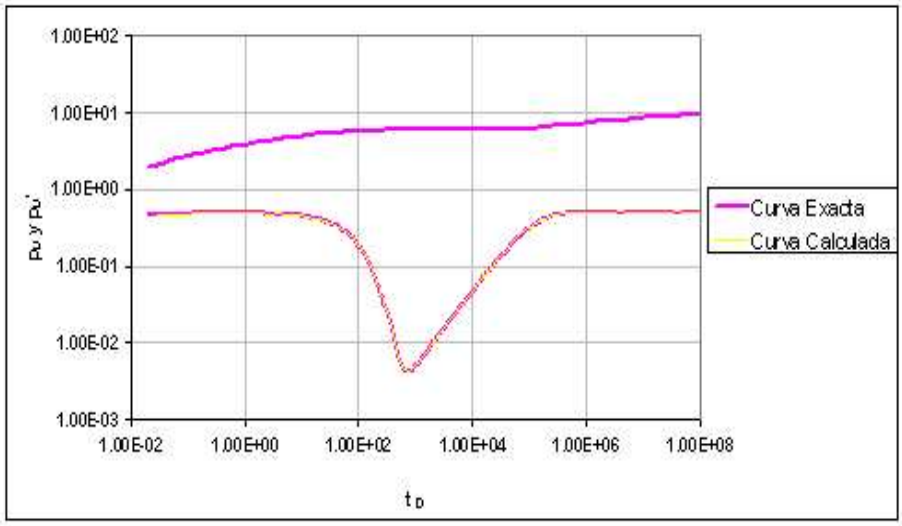

Fig. 4.3: Comparación gráfica del modelo de doble porosidad.

Observando la tabla 4.3 y la figura 4.3 vemos que para el modelo de doble porosidad donde el flujo se presenta entre la matriz y el sistema de fracturas los valores de los coeficientes calculados se aproximaron suficientemente.

Ahora analizaremos el primer caso de triple porosidad donde se presenta un mayor porcentaje de flujo a través de los vúgulos cuya solución exacta y calculada se muestran en la tabla 4.4.

Parámetro	Solución-Exacta	Solución-Calculada	Cotas
ω_v	1.00E-01	9.84674051E-02	1.00E-02 $\leq \omega_v \leq$ 1.00E-01
ω_f	1.00E-04	9.99929465E-05	1.00E-05 $\leq \omega_f \leq$ 1.00E-04
λ_{mf}	1.00E-07	1.75252649E-07	5.00E-08 $\leq \lambda_{mf} \leq$ 5.00E-07
λ_{vf}	1.00E-05	9.91665090E-06	5.00E-08 $\leq \lambda_{vf} \leq$ 5.00E-07
λ_{mv}	1.00E-07	2.39338480E-08	5.00E-08 $\leq \lambda_{vf} \leq$ 5.00E-07

Tab. 4.4: Comparación numérica del modelo de triple porosidad 1.

Las gráficas para la solución exacta y calculada se muestran en la figura 4.4.

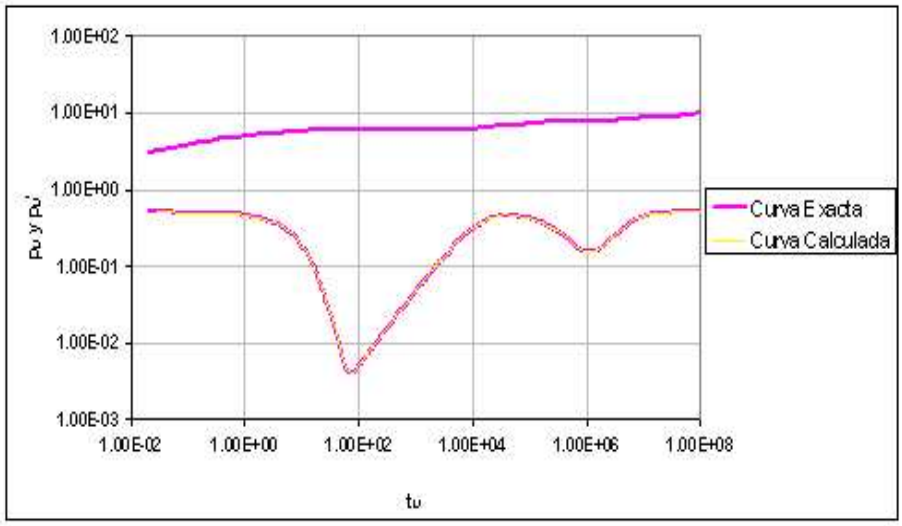

Fig. 4.4: Comparación gráfica del modelo de triple porosidad 1

Nuevamente observando la tabla 4.4 y la figura 4.4 vemos que el valor de los coeficientes calculados se aproximaron al orden de magnitud permitido.

Para el segundo caso de triple porosidad donde la difusión de flujo a través de los vúgulos y el sistema de fracturas es la misma y cuya solución exacta y calculada se muestran en la tabla 4.5.

Parámetro	Solución-Exacta	Solución-Calculada	Cotas
ω_v	1.00E-02	1.00013053E-02	1.00E-03 $\leq \omega_v \leq$ 1.00E-02
ω_f	1.00E-02	1.00021680E-02	1.00E-03 $\leq \omega_f \leq$ 1.00E-02
λ_{mf}	1.00E-07	6.40878666E-08	5.00E-08 $\leq \lambda_{mf} \leq$ 5.00E-07
λ_{vf}	1.00E-05	9.96541757E-06	5.00E-06 $\leq \lambda_{vf} \leq$ 5.00E-05
λ_{mv}	1.00E-07	1.36669641E-07	5.00E-08 $\leq \lambda_{mv} \leq$ 5.00E-07

Tab. 4.5: Comparación numérica del modelo de triple porosidad 2.

Las gráficas para la solución exacta y calculada se muestran en la figura 4.5.

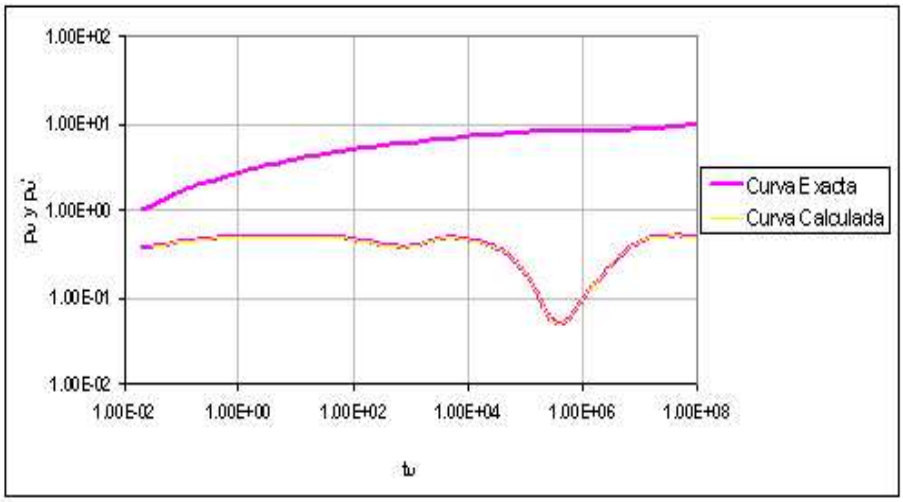

Fig. 4.5: Comparación gráfica del modelo de triple porosidad 2

Nuevamente observando la tabla 4.5 y la figura 4.5 vemos que los valores de los coeficientes calculados se aproximaron al orden de magnitud permitido.

Para el tercer caso de triple porosidad donde la difusión de flujo a través de los vúgulos es mucho mayor al del sistema de fracturas y cuya solución exacta y calculada se muestran en la tabla 4.6.

Parámetro	Solución-Exacta	Solución-Calculada	Cotas
ω_v	5.00E-01	4.97560337E-01	5.00E-02 $\leq \omega_v \leq$ 5.00E-01
ω_f	1.00E-03	1.00043821E-03	1.00E-04 $\leq \omega_f \leq$ 1.00E-03
λ_{mf}	1.00E-05	9.99999990E-04	5.00E-06 $\leq \lambda_{mf} \leq$ 5.00E-05
λ_{vf}	1.00E-03	8.52296119E-06	5.00E-04 $\leq \lambda_{vf} \leq$ 5.00E-03
λ_{mv}	1.00E-07	1.56115314E-06	5.00E-08 $\leq \lambda_{mv} \leq$ 5.00E-07

Tab. 4.6: Comparación numérica del modelo de triple porosidad 3.

Las gráficas para la solución exacta y calculada se muestran en la figura 4.6.

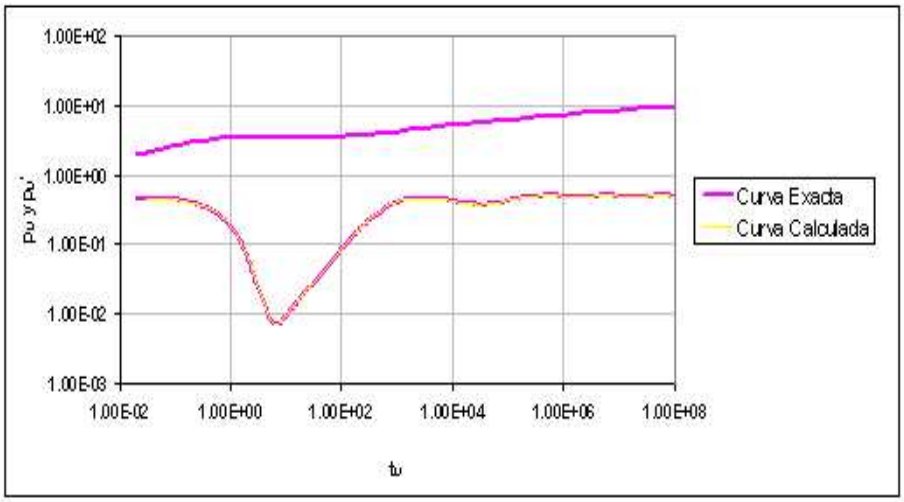

Fig. 4.6: Comparación gráfica del modelo de triple porosidad 3

Nuevamente observando la tabla 4.6 y la figura 4.6 vemos que en al menos tres de los casos se aproximaron al orden de magnitud permitido.

Y para el cuarto caso de triple porosidad donde la difusión de flujo a través λ_{mf}, λ_{vf} y λ_{mv} son iguales y cuya solución exacta y calculada se muestran en la tabla 4.7.

Parámetro	Solución-Exacta	Solución-Calculada	Orden de Magnitud
ω_v	1.00E-01	2.14429122E-02	1.00E-02 $\leq \omega_v \leq$ 1.00E-01
ω_f	1.00E-02	1.00006534E-02	1.00E-03 $\leq \omega_f \leq$ 1.00E-02
λ_{mf}	1.00E-07	1.57282618E-07	5.00E-08 $\leq \lambda_{mf} \leq$ 5.00E-07
λ_{vf}	1.00E-07	4.33759475E-08	5.00E-08 $\leq \lambda_{vf} \leq$ 5.00E-07
λ_{mv}	1.00E-07	1.74676651E-09	5.00E-08 $\leq \lambda_{mv} \leq$ 5.00E-07

Tab. 4.7: Comparación numérica del modelo de triple porosidad 4.

Las gráficas para la solución exacta y calculada se muestran en la figura 4.7.

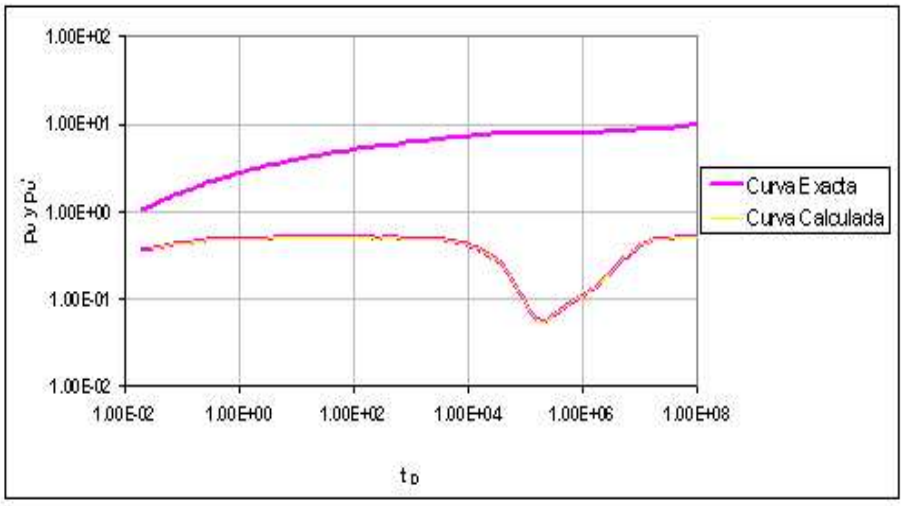

Fig. 4.7: Comparación gráfica del modelo de triple porosidad 4

Nuevamente observando la tabla 4.7 y la figura 4.7 vemos que en la mayoría de los casos los coeficientes se aproximaron al orden de magnitud permitido.

CONCLUSIONES

En base a los resultados obtenidos en este trabajo se llega a las siguientes conclusiones.

Se usó un algoritmo genético como método para resolver el problema de optimización, que nos permite encontrar los coeficientes de un modelo de flujo de petróleo en yacimeintos naturalmente fracturados, usando pruebas de pozo. La función objetivo involucra por tanto, la solución de ecuaciones diferenciales parciales.

El conocimiento de la naturaleza del problema fue muy importante, ya que el buen funcionamiento del algoritmo genético depende de los parámetros establecidos para su funcionamiento y de esta manera es posible acelerar la convergencia y obtener la solución óptima o aproximación deseada.

El algoritmo genético con subpoblaciones, explora al máximo el espacio de búsqueda, analizando un conjunto de soluciones potenciales en subregiones del espacio y no en un sólo punto inicial como sería el caso de los métodos basados en gradientes. Además, la incorporación de las cotas y de la restricción lineal, puede hacerse con facilidad.

El Algoritmo Genético desarrollado en este trabajo según lor resultados obtenidos, demostró ser muy eficiente, ya que el problema de caracterización de yacimientos naturalmente fracturados petroleros usando el modelo de triple porosidad es un problema muy difícil, que no se ha podido resolver usando técnicas que utilizan derivadas como los del tipo Newton.

Debido a la complejidad del problema analizado en este trabajo es recomendable pensar en la paralelización del algoritmo genético ya que la función objetivo es muy costosa.

BIBLIOGRAFíA

[1] Aldem H. Wright, *Genetic Algorithms for real parameter optimización*, Foundations of Genetic Algorithms, Morgan Kaufman Publisher, pp 205-217, 1991.

[2] Bäck Thomas et al, *Evolutionary Computation 1, Basic Algorithms and Operators*, Institute of Physics Publishing, Bristol and Philadelphia, 2000.

[3] Bäck Thomas et al, *Evolutionary Computation 2, Advanced Algorithms and operators*, Institute of Physics Publishing, Bristol and Philadelphia, 2000.

[4] Bäck Thomas, *Generalized convergence models for tournament and (m,l)-selection*, Proceedings of the sixth international conference on Genetic Algorithms, University of Pittsburgh, Morgan Kaufmann Publishers San Francisco California, (pp. 2-8), 1985.

[5] Bäck Thomas, *Extended selection mechanisms in Genetic Algorithms*, Proceedings of the fourth international conference on Genetic Algorithms, University of California, Morgan kaufmann Publishers, San Mateo California, pp. 92-99, 1991.

[6] Baker J. E., *Reducing bias and inefficiency in the selection algorithm*, Proceedings of the Second International Conference on Genetic Algorithms, Cambridge MA., pp 14-21, 1985.

[7] Banzhaf Wolfgang et al, *Genetic Programming, An Introduction*, Morgan Kaufmann Publishers, 1998.

[8] Barrón C. y S. Gómez, *The exponential Tunneling Method*, Reporte de Investigación, IIMAS, Vol. 1, No. 3, 1991.

[9] Bautista Tomás et al, *Una Descripción de LaTeX2$_\epsilon$*, Manual del Centro de Microelectrónica Aplicada de la Universidad de las Palmas de G.C. 1998.

[10] Blickle Tobias y Thiele Lothar, *A mathematical Analysis of Tournament selection*, Proceedings of the sixth internacional conference on Genetic Algorithms, University of Pittsburgh, Morgan Kaufmann Publishers San Francisco California, pp. 9-16, 1985.

[11] Blickle Tobias and Thiele Lothar, *A comparison of selection schemes used in Genetic Algorithms*, Computer Engineering and Comunication Networks Lab, TIK-Report Version 2, Switzerland, 1995.

[12] Bramlette Mark F., *Initialization, Mutation and selection methods in Genetic Algorithms for Function Optimization*, Proceedings of the fourth international conference on Genetic Algorithms, University of California, Morgan Kaufmann Publishers, San Mateo California, pp. 100-107, 1991.

[13] Camacho Velázquez R. et al, *Pressure Transient and Decline Curve Behavior in Naturally Fractured Vuggy Carbonate Reservoirs*, articulo SPE 77689 presentado en la Conferencia Técnica Anual en San Antonio Texas, U.S.A, del 29 Sep. al 2 Octubre de 2002.

[14] Cantu-Paz Erick, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, 2000.

[15] Cantú-Paz Erick, *Topologies, Migration Rates, and Multi-Population Parallel Genetic Algorithms*, Proceedings of the Genetic and Evolutionary Computation Conference, Orlando Florida, Morgan Kaufmann Publishers, San Francisco California, Vol. 1 (pp. 91-98), 1999.

[16] Cantú-Paz Erick, *On Random Numbers and the Performance of Genetic Algorithms*, Proceedings of the Genetic and Evolutionary Computation Conference, New York city, Morgan Kaufmann Publishers, San Francisco California, Vol. 1 (pp. 311-318), 2002.

[17] Chipperfield Andrew et al, *Genetic Algorithm TOOLBOX For Use with MATLAB*, Department of AUTOMATIC CONTROL and SYSTEMS ENGINEERING, University of Sheffield, 1994.

[18] Coello Coello Carlos A, *Introducción a la Computación Evolutiva*, (Notas de Curso), CINVESTAV-IPN, Departamento de Ingeniería Eléctrica, Mayo 2003.

[19] Eshelman Larry J. y Schaffer J. David, *Preventing premature convergence in Genetic Algorithms by Preventing Incest*, Proceedings of the fourth international conference on Genetic Algorithms, University of California, Morgan Kaufmann Publishers, San Mateo California, pp. 115-122, 1991.

[20] Fogel David B, *Evolutionary computation, The Fossil Record*, IEEE Press, 1998.

[21] Fuentes Cruz Gorgonio et al, *Análisis de datos de presión de pozos parcialmente penetrantes en yacimientos naturalmente fracturados vugulares*, Asociación de ingenieros petroleros de México, Delegación México, XLII Congreso Nacional, 2003.

[22] Goldberg D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading Mass., 1989.

[23] Goldberg David E. y Siegfried Voessner, *Optimization Global-Local Search Hybrids*, Proceedings of the Genetic and Evolutionary Computation Conference, Orlando Florida, Morgan Kaufmann Publishers, San Francisco California, Vol. 1 (pp. 220-228), 1999.

[24] Goldberg David E. y Sastry Kumara A, *Practical Schema Theorem for Genetic Algorithm Design and Tuning*, Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, San Francisco California, Vol. 1 (pp. 328-335),.

[25] Goldberg D. E., *Sizing populations for serial and parallel genetic algorithms*, Proc. 3rd Int. Conf. on Genetic Algorithms, San Mateo CA, pp 70-79, 1989.

[26] Gómez S. y Levy A. V., *The tunneling method for solving the constrained global optimization problem with several non-connected feasible regions*, Lecture Notes in Mathematics, Springer-Verlag, Vol. 909, pág 34-47, 1982.

[27] Gómez S. y Levy A. V., *The tunneling method applied to global optimization*, en P.T. Boggs, R.H. Byrd and R.B. Schnabel (eds), Numerical Optimization, SIAM, pag. 213-244. Por invitación, 1985.

[28] Gómez S. et al, *A global zero residual least squares meted*, Lecture Notes in Mathematics, Springer-Verlag, Vol. 1230, pág. 1-10, 1987.

[29] Gómez S. et al, *The parallel Tunneling Method*, Parallel Computing, Elsevier, En prensa 2002.

[30] Gómez S. et al, *Gradient-based history-Matching with a global optimization method*, Soc. Petr. Eng. Journal, pág. 200-208. June 2001.

[31] Guyaguler Baris et al, *Automated Reservoir Model Selection in Well Test Interpretation*, articulo SPE 71569 presentado en la Conferencia Técnica Anual en New Orleans, Louisiana, U.S.A. del 30 de Sep. al 3 Octubre de 2001.

[32] Hinterding Robert et al, *The Nature of Mutation in Genetic Algorithms*, Proceedings of the sixth internacional conference on Genetic Algorithms, University of Pittsburgh, Morgan Kaufmann Publishers San Francisco, California, pp. 65-72, 1985.

[33] Holland J. H., *Information processing inthe adaptive systems*, Proceedings of the International Union of Physiological Sciences, 3, pp. 330-339, 1962.

[34] Holland J. H., *Adaptation in natural and artificial system*, Ann Arbor, the University of Michigan Press, 1975.

[35] Knuth Donald E., *The Art of Computer Programming*, Second Edition, Addison-Wesley Publishing, 1969.

[36] Koza John R., *On the Programming of computers by means of Natural Selection*, A Bradford Book, The MIT Press, 1994.

[37] Koza John R, *Evolving a computer program to generate random numbers using the Genetic Programming Paradigm*, Proceedings of the fourth international conference on Genetic Algorithms, University of California, Morgan Kaufmann Publishers, San Mateo California, pp. 37-44, 1991.

[38] Matsumoto Makoto y Takuji Nishimura, *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator*, ACM Transactions on Modeling and Computer Simulations: Special Issue on Uniform Random Number Generation, 1998.

[39] Michalewicz Zbigniew, *Genetic Algorithms + Data Structures = Evolution Programs*, Third Revised and Extended Edition, Springer-Verlag, 1996.

[40] Mühlenbein H. y Schlierkamp-Voosen D., *Predective models for the breeder genetic algorithm: I. Continuous optimization*, Evolutionary Computation, 1(1), pp 25-49, 1993.

[41] Mühlenbein H. y Schlierkamp-Voosen D., *The optimal populatión size for uniform crossover and truncation selection*,Technical report GMD-AS-TR-94-11,St. Augustin, Germany, 1996.

[42] Rios Insua David et al, *Simulación Métodos y Aplicaciones*, Editorial RA-MA, Madrid España, 1997.

[43] Sastry Kumara et al, *Don't Evaluate, Inherit*, Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, San Francisco California, Vol. 1 (pp. 551- 558), 2001.

[44] Schaffer J. D. et al, *A study if control parameters affecting online performance of genetic algorithms for function optimization*, Proc. 3rd Int. Conf. on Genetic Algorithms, San Mateo, CA. pp 51-60, 1989.

[45] Z. Cezary et al, *An experimental comparison of binary and floating point representations in Genetic Algorithms*, Proceedings of the fourth international conference on Genetic Algorithms, University of California, Morgan Kaufmann Publishers, San Mateo California, pp. 31-36, 1991.