Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Algoritmos genéticos binarios con control de diversidad

Tesis presentada para obtener el título de
Licenciado en Ciencias de la Computación

Por: Alejandra Di Rado, Gabriela Podestá

Director: Dr. Tomás Tetzlaff

Diciembre del 2014

Resumen

El comportamiento que muestra un algoritmo evolutivo depende del balance entre la explotación local y la exploración global del espacio de búsqueda. Con el objetivo de mantener un equilibrio adecuado implementamos un algoritmo genético básico, analizamos métodos elitistas para mejorar el desempeño del algoritmo genético e incluimos un procedimiento para controlar automáticamente la diversidad poblacional. Para llevar a cabo este control proponemos una medida de dispersión poblacional que nos permite detectar prematuramente el problema de homogeneización en un óptimo local. Para obtener el valor de dispersión, y de esta manera conocer la diversidad poblacional, utilizamos el método de Sidaner, Bailleux y Chabrier que utiliza las distancias en torno a un punto medio, y le agregamos nuestra propuesta de disminuir el valor de dispersión ante la presencia de individuos repetidos. Luego desarrollamos un método para obtener la cota que le permite al algoritmo decidir y tomar acción automáticamente en caso de que se deba agregar diversidad. Cuando se decide aumentar la diversidad, se incrementa la tasa de mutación, lo cual hace más probable la aparición de nuevos individuos en la población. Presentamos un desarrollo teórico y los resultados de las ejecuciones del algoritmo genético binario básico y con control de diversidad programados en C++. Las corridas en problemas de prueba muestran que el número de generaciones para alcanzar el óptimo global es menor con control de diversidad. Finalmente demostramos que el algoritmo con control de diversidad converge con probabilidad 1 a un óptimo global.

Abstract

The behavior of an evolutionary algorithm depends on the balance between local explotation and global exploration of the search space. In order to keep the balance adequate we implemented a basic genetic algorithm, analyzed elitist methods to improve the genetic algorithm's performance and included a procedure for automatic population diversity control. To perform this control we propose a population dispersion measure that allows early detection of the homogenization problem at a local optimum. In order to obtain the dispersion value and this way know the population diversity we use the method of Sidaner, Bailleux and Chabrier that uses the distances to a middle point and incorporate our proposal of lowering the dispersion in the presence of repeated individuals. Then we develop a method to obtain the bound that allows the algorithm to decide and automatically take action in case diversity must be increased. When this decision is made, the mutation rate is incremented, making it more probable that new individuals appear in the population. We introduce a theorical development and the results of the basic binary genetic algorithm runs and those of the algorithm with diversity control implemented in C++. Test problems runs show that the number of generations to reach a global optimum is lower with diversity control. Finally we prove that the algorithm with diversity control converges with probability 1 to a global optimum.

Agradecimientos

En primer lugar, quisiéramos agradecer a nuestro director de tesis Tomás, por su esfuerzo y dedicación, quién con sus conocimientos, su paciencia y su motivación ha logrado que nosotras podamos finalizar nuestros estudios con éxito.

A la facultad y a los docentes que tuvimos por habernos brindado la formación que nos hace hoy profesionales.

A quienes se tomaron el trabajo de leer y corregir nuestra tesis minuciosamente.

A todos aquellos que nos acompañaron e incentivaron durante todo este tiempo, que no nos permitieron bajar los brazos y que nos acompañaron en los buenos y malos momentos.

Agradecimientos de Alejandra

Quiero agradecerle a Florencia por haber estado siempre, desde muy pequeña, viendo a su madre entre papeles y computadoras perdonando su ausencia.

A mi hermano Marcelo por haberme anotado en esta carrera cuando yo no sabía que camino seguir.

A mi hermano Luciano por estar incondicionalmente apoyándome.

A la gente buena que conocí en esta facultad y me permitió compartir con ellos aprendizaje y vivencias personales.

A Gabi, mi compañera de tesis, por no bajar los brazos cuando el final parecía tan lejano.

Por último a mis padres porque me brindaron todas las posibilidades para que llegara hasta acá y además me hicieron nacer en un país que cuenta con el acceso a una educación pública, gratuita y de primer nivel académico.

Agradecimientos de Gabriela

Quiero agradecer a mi esposo Jorge y a mis hijos Sofía y Joaquín quiénes fueron un gran apoyo emocional durante el tiempo en que escribía esta tesis.

A mis padres Jorge y Alicia quiénes me acompañaron siempre en forma incondicional.

A Alejandra, mi compañera de tesis, quien me apoyo y alentó para continuar, cuando parecía que me iba a rendir.

Gracias a todos los que pasaron por mi camino, que me acompañaron en el crecimiento tanto personal como profesional y me ayudaron a llegar hasta donde llegué.

Dedicado a Florencia, Sofía, Joaquín, Martina y Camilo.

Índice general

Capítulo 1

Introducción

Los algoritmos evolutivos son una de las técnicas heurísticas de optimización más utilizadas en los últimos años para resolver problemas complejos. Conforme avanza la investigación sobre el mecanismo algorítmico evolutivo y se incrementa el poder de cómputo disponible, los científicos e ingenieros intentan aplicarlos como herramienta de resolución para problemas de optimización cada vez más difíciles. En este trabajo se investiga cómo evitar que los algoritmos genéticos se estanquen en soluciones que corresponden a óptimos locales y de esta manera no lleguen a la solución óptima. Los objetivos fundamentales de esta tesis son la comprensión del mecanismo de los algoritmos genéticos como técnica de resolución de problemas de optimización y además evaluar y mejorar la convergencia de los algoritmos genéticos binarios incorporando control de diversidad poblacional. Con este fin definimos un método para conocer la diversidad de individuos presente en la población y eventualmente aumentar esa diversidad. Luego hemos realizado pruebas comparativas para analizar el comportamiento de nuestra propuesta y verificar así si los resultados obtenidos mejoran la convergencia del algoritmo.
Para estudiar las generalidades de algoritmos genéticos se consultaron libros y artículos de referentes como Holland [HOL/75] y D. Goldberg [GOL/89]. Respecto del problema de diversidad poblacional utilizamos el método presentado por Sidaner, Bailleux y Chabrier [SID/02]. Además, debido a que los algoritmos genéticos están inspirados en aspectos biológicos, se han consultado publicaciones del ámbito de la biología como la obra de Jorge M. Lobo, Métodos para Medir la Biodiversidad [LOB/01].
Este trabajo está compuesto por siete capítulos, el primero de los cuales es la presente *Introducción*. En el capítulo dos *Conceptos básicos*, damos una breve reseña de estos algoritmos, su historia, evolución y características principales. También realizamos una comparación con otros algoritmos de optimización enunciando las posibles ventajas de los algoritmos genéticos.
En el capítulo tres, *Implementación del algoritmo genético canónico*, exponemos la implementación que realizamos de un algoritmo genético básico. Luego explicamos el Problema de la Mochila (un problema de optimización combinatoria NP-Completo) y un problema de optimización de una función continua. Realizamos corridas de los algoritmos genéticos utilizando distintos parámetros y comparamos los resultados obtenidos para estos problemas, con el algoritmo canónico y el algoritmo con elitismo.
En el capítulo cuatro, *Diversidad poblacional*, realizamos una breve introducción sobre la importancia de la biodiversidad y la relación con la diversidad genética tanto en la naturaleza como en los algoritmos genéticos. Luego proponemos un método para evaluar diversidad poblacional mediante

una medida de dispersión. Por último desarrollamos la fórmula del cálculo de la esperanza para la dispersión y calculamos el valor esperado para poblaciones iniciales como valor de referencia.
En el capítulo cinco, *Algoritmos genéticos con control de diversidad*, utilizamos los métodos del capítulo anterior para decidir si la población es poco diversa. Luego aplicamos un método para controlar la tasa de mutación y a través de ella la diversidad en la población. Exponemos la resolución del Problema de la Mochila e incorporamos la resolución de un problema adicional que es el de reconocimiento de imágenes, a través del algoritmo genético con control de la diversidad.
En el capítulo seis, *Convergencia del algoritmo genético*, demostramos convergencia del algoritmo genético con elitismo y control de la diversidad.
En el capítulo siete se exponen las *Conclusiones*.
Por último, el apéndice A define términos de uso frecuente en el contexto de algoritmos genéticos, el apéndice B describe una familia de funciones de difícil optimización de interés para la teoría de convergencia al óptimo y el apéndice C muestra los lotes de pruebas con los resultados del problema de la mochila.

Capítulo 2

Conceptos básicos

2.1. Fundamentos de algoritmos genéticos

2.1.1. Motivación y fundamentos de su diseño

Según el conocido concepto de evolución darwiniana, cualquier especie animal o vegetal en sucesivas generaciones es capaz de adaptarse mediante cambios en su forma de vida, e incluso en su estructura, para compensar cualquier agente que hiciera peligrar su supervivencia. Los cambios al azar junto con la selección producen la aparición de individuos mejor adaptados que sus antecesores. Ante esto, surge la idea de emular la naturaleza para la resolución de ciertos problemas computacionales, en especial aquellos para los cuales no se puede encontrar soluciones por métodos determinísticos. Como en la naturaleza, se quiere que aquello que sea favorable para la supervivencia sea asumido por la población, mientras que lo que represente poco valor de adaptación sea descartado. Así surge la computación evolutiva, una rama de la inteligencia artificial que involucra problemas de optimización y se inspira en los mecanismos de la evolución biológica. Los algoritmos genéticos, que forman parte de la computación evolutiva, se sustentan en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven con mayor probabilidad por estar mejor adaptados al entorno. Por este proceso, son capaces de ir creando soluciones para un problema. La evolución lleva las posibles soluciones hacia valores óptimos.

Dado que los algoritmos genéticos tienen su origen tanto en la genética natural como en la ciencia computacional, la terminología usada en ellos proviene de ambas áreas. Al conjunto de los genes de un individuo se lo llama *genotipo*. El equivalente a los cromosomas en los sistemas genéticos artificiales son secuencias de dígitos o caracteres. En los sistemas naturales, al organismo formado por la interacción del genotipo con el medio ambiente se lo denomina *fenotipo*. En el contexto de los algoritmos genéticos, el fenotipo representa la traducción de la información contenida en el genotipo. Otros términos de esta área de investigación se aclararán en el texto. Incluimos además un glosario.

2.1.2. Diferencias entre los algoritmos genéticos y otros métodos de optimización

Los algoritmos genéticos son capaces de crear soluciones para problemas computacionales mediante la búsqueda de un óptimo de una función objetivo. En la manera de abordar los problemas de optimización difieren de otros procedimientos de búsqueda fundamentalmente en lo siguiente:

- Trabajan con una codificación del conjunto de los valores de las variables del problema, a menudo binaria. En muchos casos, no con los valores en sí mismos del dominio de la función a optimizar.
- Buscan una solución óptima a partir de un conjunto de puntos (población) y no desde uno solo.
- Usan como única información las evaluaciones de la función objetivo en vez de otras propiedades como linealidad o derivabilidad de la función u otros conocimientos adicionales.
- Utilizan reglas de transición probabilísticas, no reglas determinísticas.

Los algoritmos genéticos requieren que los elementos del conjunto de variables naturales del problema de optimización se codifiquen como una secuencia finita sobre algún alfabeto finito.
Trabajan a partir de una serie de puntos simultáneamente (una población de secuencias), recorriendo así en paralelo el relieve de la función, pudiendo aventajar en algunos casos a los métodos que se mueven de un punto a otro. Sin embargo, no están exentos de la posibilidad de estancarse en un óptimo local, quedando la población reducida a repetidas copias de un mismo individuo.

2.1.3. Trabajos fundacionales

Según relata Goldberg, discípulo de John Holland [GOL/89], los algoritmos genéticos fueron desarrollados por John Holland, al comienzo de los años sesenta, en la Universidad de Michigan. Holland investigó el comportamiento de los sistemas naturales con el objeto de explicar y estructurar rigurosamente su proceso de adaptación para diseñar sistemas artificiales que conserven los mecanismos en que se fundamentan los sistemas naturales. Esta propuesta ha facilitado avances importantes en el campo de inteligencia artificial. Como ejemplo del trabajo pionero de Holland podemos citar Adaptation in Natural and Artificial Systems [HOL/75].
Goldberg continúa relatando que el objetivo central de estos algoritmos de búsqueda ha sido la capacidad de adaptación, es decir, una vez que el algoritmo es capaz de resolver eficientemente un problema en ciertas condiciones, aunque éstas se modifiquen, el algoritmo sigue encontrando de forma eficaz la solución del problema.
Los algoritmos genéticos quedaron incorporados al área de metaheurísticas, que son familias de estrategias heurísticas de alto nivel.
Han sido desarrollados para una amplia gama de problemas, desde planteos como ecuaciones diferenciales hasta cuestiones concretas como problemas de ingeniería.
Para nombrar algunos ejemplos de áreas variadas podemos citar:

- Química: un pulso láser ultracorto de alta energía puede romper moléculas complejas produciendo moléculas más sencillas, un proceso con aplicaciones importantes en la química

orgánica y la microelectrónica. Los productos específicos de una reacción pueden controlarse modulando la fase del pulso láser. Sin embargo, para moléculas grandes, obtener la forma del pulso deseado de manera analítica es demasiado difícil. Se resolvió este problema utilizando un algoritmo evolutivo para diseñar la forma del pulso [ASS/98].

- Ingeniería aeroespacial: Williams, Crossley y Lang [WIL/01] aplicaron algoritmos genéticos a la tarea de situar órbitas de satélites para minimizar los apagones de cobertura.
- Telecomunicaciones: las redes movibles ad hoc son desarrolladas y evaluadas en ambientes genéricos de simulación. Las redes reales varían mucho en términos de topología, tráfico y algunas características específicas como sobrecarga, energía, etc. En respuesta a esto las redes ad hoc deben ser diseñadas con muchos parámetros modificables. Se utilizó un algoritmo genético para automatizar la selección de parámetros de un sistema de red ad hoc. [MON/05]
- El problema de Steiner: consiste en encontrar el árbol mínimo que interconecta varios puntos de una red. Fue propuesto por el matemático alemán Jacob Steiner a principios del siglo XIX. [THO/06]
- El problema del viajante: el objetivo es encontrar una ruta que, comenzando y terminando en una ciudad, pase una sola vez por cada una de las ciudades que debe recorrer el viajante y minimice la distancia. Se trata de un problema clásico de optimización combinatoria para el cual no se conoce solución en tiempo polinomial [HAU/04].

2.2. Componentes principales de un algoritmo genético

Un problema de optimización consiste en encontrar la solución que maximiza o minimiza una función dada. Dada una función F (a valores reales) de n variables, optimizar la función consiste en encontrar un vector $(x_1, .., x_n)$ tal que $F(x_1, .., x_n)$ es máximo (o mínimo) sobre todos los valores posibles de las n variables. Se puede hablar indistintamente de minimizar o maximizar, ya que maximizar F es minimizar $-F$. Los algoritmos genéticos son métodos de optimización de una función objetivo que suele denominarse *fitness* por analogía con la aptitud biológica.
Hay cinco componentes básicos de todo algoritmo genético, que a su vez distinguen a un algoritmo de otro según las variantes que se implementen. Estas componentes que describiremos a continuación son: representación de los elementos del espacio de búsqueda, procedimiento para crear la población inicial de individuos de ese espacio, función objetivo o de evaluación, operadores de evolución (selección y métodos de cambio) y configuración paramétrica (tamaño de la población, probabilidad de cruzamiento, probabilidad de mutación, criterio de parada, etc.).

2.2.1. Representación de los elementos del espacio de soluciones

El algoritmo genético requiere como primer paso un método para codificar elementos del espacio de búsqueda de forma que una computadora pueda procesarlos.
Un procedimiento usual es codificar estos elementos como cadenas binarias (secuencias de 0s y 1s), donde el dígito de cada posición representa el valor de algún aspecto de la solución. Tiene la ventaja para el proceso de búsqueda evolutiva que se pueden introducir cambios (mutación) relativamente pequeños porque la información se representa con una cantidad máxima de posiciones. Este es el

método original de representación, el cual estudiamos en nuestro trabajo.
Existen otras maneras de representación que consisten en codificar las soluciones como cadenas de enteros, números decimales o letras de un cierto alfabeto. Estas se aplican a problemas específicos, como la representación de permutaciones para el problema del viajante. Lo usual es representar las soluciones candidatas como una cadena (llamada $cromosoma$ o simplemente $individuo$) de datos (valores de variables a veces llamadas $genes$) de una longitud fija.

2.2.2. Población inicial

La población inicial es el conjunto de individuos con los que se inicia el proceso de búsqueda. El proceso de generar la población inicial de individuos puede ser aleatorio o arbitrario. En este último caso, el algoritmo podría partir de un conjunto de soluciones aceptables, conocidas, producto de otro algoritmo de optimización o cualquier otra técnica que pueda servir como base.

2.2.3. Función objetivo

Si bien la función objetivo está dada en el planteo del problema, es posible que pueda ser reformulada de manera que se logre más rápido la obtención del óptimo. Es conveniente trabajar con funciones objetivo que verifiquen que, para dos individuos que se encuentren cercanos en el espacio de búsqueda, sus respectivos valores en las funciones objetivo sean similares, lo cual corresponde a un relieve sencillo de la gráfica de la función, la cual va a favorecer su optimización. Pero en muchos problemas de optimización combinatoria, donde existe gran cantidad de restricciones, buena parte de los puntos del espacio de búsqueda representan individuos no válidos. Además, una dificultad en el comportamiento del algoritmo genético puede ser la existencia de gran cantidad de óptimos locales, así como el hecho de que el óptimo global se encuentre muy aislado.
Para el planteo en el que los individuos están sometidos a restricciones, se han propuesto varias soluciones. Una posibilidad es que aquellos individuos que no verifican las restricciones, no son considerados como tales, y se siguen efectuando cruces y mutaciones hasta obtener individuos válidos, o bien, a dichos individuos se les asigna un valor de la función objetivo igual a cero o alguna cota inferior de los valores de la función a maximizar. Otra posibilidad consiste en reconstruir aquellos individuos que no verifican las restricciones. Dicha reconstrucción suele llevarse a cabo por medio de un nuevo operador que se acostumbra denominar $reparador$.
Otro enfoque está basado en agregar penalización a la función objetivo. La idea general consiste en dividir la función objetivo del individuo por una cantidad (la penalización) que guarda relación con las restricciones que dicho individuo viola. Dicha cantidad puede simplemente tener en cuenta el número de restricciones violadas o bien el costo esperado de reconstrucción, es decir, el costo asociado a la conversión de dicho individuo en otro que no viole ninguna restricción.
En cuanto al caso en que el máximo global se alcanza en varios puntos y se desea conocer a todos, se ha propuesto modificar el pico correspondiente a un punto óptimo ya visitado recortándolo o transformándolo en un pozo, como se expone en el trabajo de Carpintero y Gularte [CAR/05]. Esto se aplica en distintas corridas del algoritmo en el caso de funciones que tienen varios óptimos globales, con el fin de dar oportunidad al procedimiento para que pueda encontrarlos a todos.

2.2.4. Métodos de selección

El objetivo principal de los operadores de selección es conservar preferentemente las mejores soluciones, es decir, los individuos con mayor fitness. Esto se alcanza mediante dos pasos principales, la reproducción y la selección de la nueva población.
Imitando lo que ocurre en la naturaleza, se otorga una mayor oportunidad de reproducción a los individuos más aptos. Por lo tanto la selección de un individuo estará relacionada con su valor de fitness. Sin embargo, no se debe eliminar por completo las posibilidades de reproducción de los individuos menos aptos, pues en pocas generaciones la población se volvería homogénea. Una vez realizada la reproducción, se elige de los individuos disponibles (progenitores y sus hijos) una cantidad fija que será la nueva generación de progenitores. Esta elección también puede realizarse dando mayor probabilidad a los individuos más aptos. Estos dos pasos principales dan lugar a variados métodos de implementación. Algunos de estos métodos son mutuamente excluyentes, pero otros pueden utilizarse en combinación, algo que se hace a menudo. En cuanto a la selección de individuos para reproducirse los métodos más utilizados son:

- **Selección proporcional**: cada individuo tiene una probabilidad de ser seleccionado como padre que es proporcional al valor de su función objetivo. Esta selección permite que los mejores individuos sean elegidos con una mayor probabilidad, pero al mismo tiempo permite a los peores individuos ser elegidos, lo cual puede ayudar a mantener la diversidad de la población. Dos métodos de muestreo usuales utilizados en selección proporcional son *selección por rueda de ruleta* y *muestreo universal estocástico*. En la selección por rueda de ruleta la probabilidad que tiene un individuo de reproducirse es proporcional a su valor de función de evaluación, es decir, a su adaptación. Para seleccionar a los individuos se realizan n experimentos aleatorios independientes. El muestreo universal estocástico utiliza un único giro de la ruleta siendo los sectores circulares proporcionales a los valores de la función objetivo. Los individuos son seleccionados a partir de marcadores igualmente espaciados y con comienzo aleatorio. Se respetan las proporciones pero los muestreos individuales no son independientes de modo tal que es menor probable la homogeneización. En la **Figura 2.1** vemos este método de selección. Notar en el ejemplo de la figura que ningún individuo puede escogerse cuatro veces.

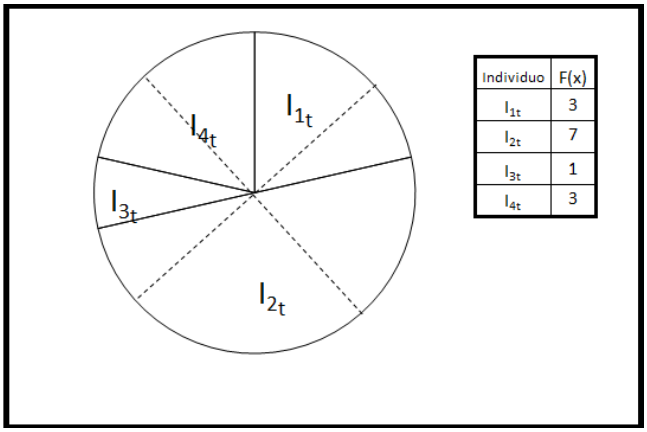

Figura 2.1: Método de selección de padres denominado muestreo universal estocástico. El individuo I_{1t} se escoge dos veces mientras que I_{3t} y I_{4t} son elegidos una única vez.

- **Selección elitista**: se garantiza la selección de los miembros más aptos de cada generación. En la mayoría de los algoritmos genéticos no se utiliza elitismo puro, se suele utilizar una forma modificada por la que el individuo mejor, o algunos de los mejores, son copiados hacia la siguiente generación.

- **Selección por torneo**: la idea principal de este método consiste en realizar la selección en base a comparaciones directas entre individuos. Existen dos versiones: determinística y probabilística. En la primera se toma un número T al azar de indiviuos. De los T individuos se selecciona el más apto para pasarlo a la siguiente generación. La versión probabilística difiere en el paso de selección del ganador del torneo. En vez de escoger siempre el mejor se genera un número aleatorio del intervalo [0,1] si es mayor que el parámetro p (fijado para todo el proceso evolutivo) se escoge el individuo más apto y en caso contrario el menos apto.

- **Selección por rango**: a cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en este ranking, en lugar de las diferencias absolutas en aptitud. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia al principio a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar la búsqueda de una solución aceptable.

Los siguientes procesos corresponden a la selección que se realiza para el reemplazo de una población por una nueva generación:

- **Selección generacional**: la mayor parte de los algoritmos genéticos son generacionales, lo que significa que cada una de las nuevas generaciones está formada por los hijos de la generación previa. La descendencia de los individuos seleccionados en cada generación se convierte en toda la siguiente generación.

- **Selección por estado estacionario**: en contraposición con la selección generacional, en este caso sólo unos pocos individuos son reemplazados en cada generación. Normalmente una pequeña cantidad de los individuos con menor fitness (peores individuos) son reemplazados por los hijos resultantes de las operaciones de cruce y mutación de los mejores individuos.

- **Selección jerárquica**: los individuos atraviesan múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente. Este método puede reducir el tiempo total de cálculo al utilizar una evaluación más rápida y menos selectiva para eliminar a la mayoría de los individuos que se muestran poco o nada prometedores, y sometiendo a una evaluación de aptitud más rigurosa y computacionalmente más costosa sólo a los que sobreviven a esta prueba inicial.

2.2.5. Métodos de cambios

Una vez que la selección ha elegido a los individuos más aptos, éstos deben ser alterados aleatoriamente con la expectativa de mejorar su aptitud para la siguiente generación. Existen dos estrategias básicas para llevar esto a cabo. La primera y más sencilla se llama $mutación$. Al igual que una mutación en los seres vivos cambia un gen, una mutación en un algoritmo genético también causa pequeñas alteraciones en puntos concretos de la cadena correspondiente a un individuo.

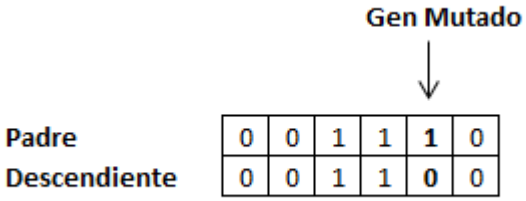

Figura 2.2: Operador de mutación

El segundo método se llama *cruzamiento*, e implica elegir dos individuos para que intercambien segmentos de su código, produciendo una descendencia cuyos individuos son combinaciones de sus padres. Este proceso pretende simular el proceso análogo de la recombinación que se da en los cromosomas durante la reproducción sexual.
Los individuos originales y los resultantes de los cambios conforman un grupo que puede ser objeto de selección para decidir quiénes integran la nueva población, lo cual muestra que selección y cambios pueden intercalarse de diferentes maneras.

Operador de Mutación La mutación es el cambio en unos pocos bits en un cromosoma pasándolos de valor 0 a 1 o viceversa. Con un concepto de distancia entre cromosomas proporcional al número de bits en los que difieren, podemos decir que la mutación proporciona la posibilidad de un movimiento aleatorio en un entorno de los individuos de la población. El operador de mutación va ganando en importancia a medida que la población de individuos va convergiendo.
El operador de mutación produce nuevas soluciones a partir de la modificación de un cierto número de genes de una solución existente, de modo que fomenta la variabilidad dentro de la población.
Existen muy diversas formas de realizar la mutación. En la más sencilla, cada gen muta aleatoriamente con independencia del resto de los genes y existe una probabilidad fija para todos los bits de que la mutación se produzca en un bit. Esta probabilidad es usualmente pequeña, por ejemplo 0,01. Si fuera muy alta el algoritmo se asemejaría a una simple búsqueda al azar. Existen variantes más complejas donde se tienen en cuenta la estructura del problema y la relación entre los distintos genes.
La **Figura 2.2** muestra la mutación del quinto gen del cromosoma. Si bien puede pensarse que el operador de cruce es más importante que el operador de mutación, ya que proporciona una exploración rápida del espacio de búsqueda, este último asegura que ningún punto del espacio de búsqueda tenga probabilidad cero de ser examinado, y es de capital importancia para asegurar la convergencia de los algoritmos genéticos.

Operador de Cruce El operador de cruce permite utilizar la información almacenada hasta el momento en la población y combinarla para crear mejores individuos. Dentro de los métodos habituales de cruce para codificación binaria destacamos los siguientes:

- **Cruce de un punto**: es el método de cruce más sencillo. Se elige al azar una posición en

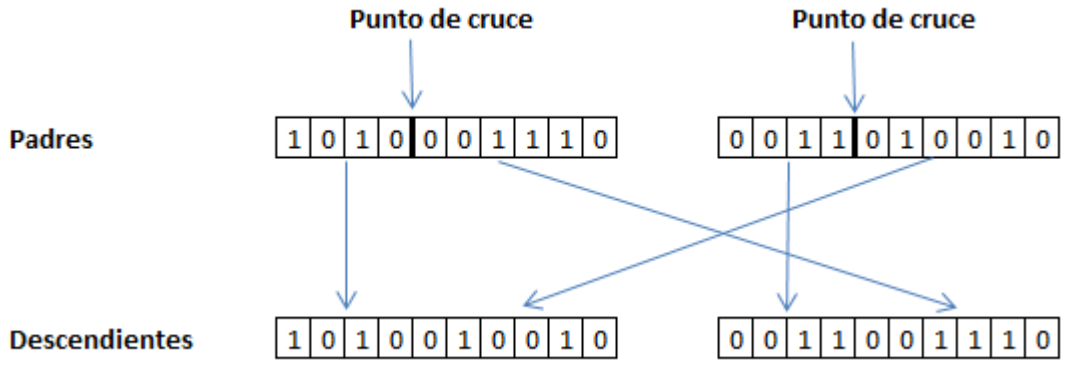

Figura 2.3: Operador de cruce basado en un punto

las cadenas de dos individuos progenitores, y se intercambian los genes a la izquierda de esta posición obteniendo dos individuos (descendientes). En la **Figura 2.3** se muestra un ejemplo de cruce en un punto.

- **Cruce de n puntos**: es una generalización del método anterior. Se eligen varias posiciones (n) en las cadenas de dos progenitores y se intercambian subcadenas definidas por esos cortes.

- **Cruce Uniforme**: con este método cada gen de la descendencia tiene las mismas probabilidades de provenir a uno u otro padre, sin correlación con un gen vecino.
  Aunque se puede implementar de muy diversas formas, la técnica implica la generación de una máscara de cruce con valores binarios al azar. Si en una de las posiciones de la máscara hay un 1, el gen situado en esa posición en uno de los descendientes se copia del primer padre, si hay un 0, se copia del segundo padre. Para producir el segundo descendiente se intercambian los papeles de los padres, o bien se intercambia la interpretación de los unos y los ceros de la máscara de cruce.
  En la **Figura 2.4** se muestra un ejemplo de cruce uniforme.

2.2.6. Configuración paramétrica

El objetivo de nuestra implementación es el de realizar un diseño flexible, por este motivo, los siguientes parámetros importantes quedan abiertos para ser configurados por el usuario: criterio de parada, tasa de mutación, longitud del individuo, cantidad de individuos de la población y elitismo (si o no).

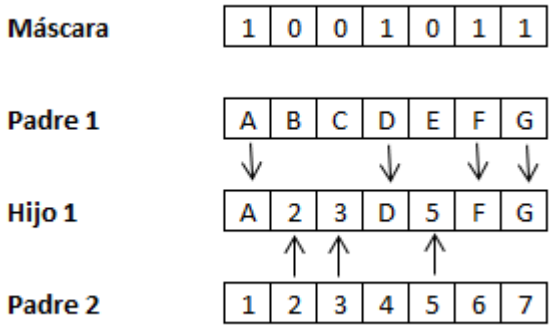

Figura 2.4: Cruce uniforme

2.3. El algoritmo genético canónico

Existen múltiples propuestas y variantes de algoritmos genéticos. Explicaremos en detalle la propuesta original de Goldberg [GOL/89], conocida como *algoritmo genético simple*, también denominado *canónico*, que es en la que basamos nuestro desarrollo.

2.3.1. Representación de los elementos del espacio de soluciones

La estructura de datos es una población de n individuos. Cada individuo corresponde a un cromosoma que es una secuencia de l bits.

2.3.2. Población inicial

Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que dentro de la población inicial, se tenga la diversidad necesaria para tener una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura.

2.3.3. Selección

Si bien el método original [GOL/89] utiliza selección por rueda de ruleta, nosotros implementamos el método de selección por torneo binario que es más sencillo de implementar y que también cumple con la propiedad de que un individuo de mayor fitness tienen mayor probabilidad de ser seleccionado.

2.3.4. Cruce

El método que se utiliza es el cruce en un punto. Es el operador genético que representa la reproducción sexual. Opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las características de ambos cromosomas padres.

2.3.5. Mutación

En la mutación se cambia el valor de cada uno de los bits de los individuos de acuerdo con una probabilidad de mutación. Es decir que se recorre bit a bit al cromosoma decidiendo, para cada uno independientemente, cambiar su valor binario con probabilidad $pmut$.

2.3.6. Descripción del algoritmo

1. **Inicialización**: se genera la población inicial al azar. Se le asigna un valor aleatorio (0 o 1) a cada uno de los bits del cromosoma de cada individuo.

2. **Fitness**: a cada uno de los cromosomas de esta población se aplicará la función de aptitud para saber qué tan "$buena$" es la solución que se está codificando.

3. **Condición de detención**: el algoritmo genético se deberá detener cuando se alcance la solución óptima, pero puede tomar mucho tiempo llegar a ella y además, según el problema, puede no haber forma de saber si se alcanzó. Por eso se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el algoritmo genético un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de detención se hace lo siguiente:

   a) **Selección**: después de calcular el fitness de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor fitness tienen mayor probabilidad de ser seleccionados.

   b) **Cruzamiento**: se generan parejas al azar, luego para cada pareja se genera un número al azar correspondiente al punto de corte y por último se cruzan formando dos hijos que reemplazaran a los individuos originales.

   c) **Mutación**: modifica al azar parte del cromosoma de los individuos, y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.

   d) **Reemplazo**: una vez aplicados los operadores genéticos, se seleccionan los mejores individuos para conformar la población de la generación siguiente.

Capítulo 3

Nuestra implementación del algoritmo genético canónico

3.1. Lenguaje utilizado

Implementamos el algoritmo genético canónico en C++ para aprovechar las ventajas del lenguaje C. Permite la redacción de programas fuente muy concisos, debido en parte al gran número de operadores que incluye el lenguaje. Tiene un conjunto de instrucciones básicas relativamente pequeño, aunque incluye numerosas funciones de biblioteca que mejoran las instrucciones básicas. Además los usuarios pueden escribir bibliotecas adicionales para su propio uso. Genera programas objeto que son pequeños y muy eficientes. Al utilizar C++ se utiliza Objetos con todas sus bondades (encapsulamiento, herencia, polimorfismo, ocultación, etc.). En nuestro caso, para realizar la codificación del algoritmo genético, generamos el programa principal y cuatro clases: Población, Individuo, Cromosoma y Parejas; de esta manera obtuvimos objetos pequeños y fáciles de entender.

3.2. Archivo de configuración paramétrica

El algoritmo que desarrollamos lee los parámetros de entrada de un archivo. Cada registro del mismo contiene un parámetro que representa:

- Cantidad de generaciones a realizar durante una ejecución ($numGen$). Este es el criterio de detención del algoritmo.
- Tasa de mutación ($pmut$, que debe tomar algún valor entre 0 y 0.5).
- Longitud del individuo ($longInd$).
- Cantidad de individuos de la población ($nind$).
- Elitismo ($elit$). (0=No, 1=Si). En caso de que se ingrese 1, al momento de seleccionar los individuos se guardará una copia del individuo con más alto fitness. Este individuo no participará en la selección para el cruce y la mutación. Pasará directamente a la próxima generación.

3.3. Programa principal

En el archivo fuente del programa principal, sólo se encuentran los pasos a seguir enunciados en el algoritmo genético canónico, dejando la resolución de los mismos a las clases invocadas que se enuncian en los párrafos posteriores.

1. Se leen los parámetros enunciados en el item "Configuración Paramétrica"
2. Inicializamos la Población
3. **Mientras** la cantidad de generaciones sea menor a la pedida ($numGen$) por parámetro y el usuario desee continuar
   a) Se realiza la selección (se invoca al método selección de la clase Población).
   b) Se cruza y muta (se invoca al método cruzarmutar de la clase Población).
   c) Se calcula el Fitness (se invoca al método fitness de la clase Población).
   d) Se le consulta al usuario si quiere continuar.

   FinMientras

3.4. Clases

Construimos cuatro clases, las cuales fueron distribuidas en tres archivos fuente (archivos extensión cpp) para mayor claridad.
Las clases $Población$ y $Parejas$ están separadas en dos fuentes. Las clases $Cromosomas$ e $Individuos$ se ubican juntas en otro archivo fuente.

Clase Población: vector de Individuos de longitud $nind$.

Métodos:

inicializar : inicializa la población con valores 0 y 1 al azar. Para esto, recorre todos los individuos de la población e inicializa cada bit de cada Cromosoma correspondiente a los Individuos (ver estructura de Individuo más abajo).

mostrar : muestra por pantalla el contenido de la población.

selección : genera $nind/2$ parejas de individuos al azar según el método calcularParejas de la clase Parejas, luego compara el Fitness de cada pareja y copia el individuo de mayor Fitness en el de menor Fitness, quedando $nind$ individuos. Es decir que el método de selección implementado fue selección por torneo, agrupando siempre de a dos y seleccionando el mejor.

cruzarmutar : genera $nind/2$ parejas de individuos al azar según el método calcularParejas de la clase Parejas para realizar el cruce y la mutación. Calcula un número entero que corresponde a la posición de cruce (s). Recorre el vector de Parejas generadas, realiza el cruce en la posición de cruce (s) y la mutación con la tasa de mutación ($pmut$) para cada individuo de cada pareja.

fitness : obtiene el Fitness de cada Individuo. Recorre todos los Individuos de la Población y calcula su Fitness de acuerdo a la función objetivo establecida. Nótese que este fitness es un método, mientras que Fitness es componente de la clase Individuo, que se verá más abajo.

Clase Cromosoma: forma parte del Individuo de la Población.
El Cromosoma está representado por un vector de bits, de dimensión parametrizable ($longInd$). Cada posición contiene un valor booleano, el valor True indica el 1 binario, el valor False el 0 binario.

Métodos:

set : cambia el valor del bit
Parámetros:
val: nuevo valor a asignar.
pos: posición del vector donde se debe asignar el valor.

get : obtiene información sobre el bit.
Parámetro:
pos: es un parámetro de entrada, recibe la posición del bit que se desea obtener (retorna el valor del bit en la posición pos).

mostrar : muestra por pantalla el contenido del Cromosoma.

mutar : para cada bit del Cromosoma: genera un valor al azar en el intervalo [0,1], si este valor es menor a la tasa de mutación ($pmut$) muta el bit, es decir, si el valor es True lo cambia por el valor False y viceversa. Recibe como parámetro la tasa de mutación ($pmut$).

Clase Individuo: representa a un Individuo de la Población.
Está compuesta por su Genotipo (de tipo Cromosoma), Fenotipo y Fitness. El Genotipo es la secuencia binaria que se utiliza para seguir los pasos del algoritmo. El Fenotipo es una función del Genotipo, varía según el problema a resolver, como detallaremos en los problemas desarrollados en esta tesis, en la práctica se trata de un cálculo intermedio. Fitness es el valor de la función objetivo.

Métodos:

getCromosoma : obtiene el valor del Genotipo.
Retorna el valor del Genotipo.

setCromosoma : setea el valor del Cromosoma.
Parámetros: valor del Cromosoma.

setFenotipo : setea el valor del Fenotipo (varía según el problema).
Parámetros: valor del Fenotipo.

getFenotipo : obtiene el valor del Fenotipo (varía según el problema).
Retorna el valor del Fenotipo.

setFitness : setea el valor del Fitness.

getFitness : obtiene el valor del Fitness.
Retorna el Fitness.

cruce : dados dos Individuos realiza el cruce, es decir para cada bit del Genotipo del Individuo, si la posición del bit es mayor a la posición de cruce intercambia el bit por el bit del Individuo que viene como parámetro. Recibe como parámetro la posición de cruce y los Individuos a cruzar. Retorna un Individuo resultante del cruce.

Clase Parejas: es un vector de longitud igual a la cantidad de individuos ($nind$). Si la posición i del vector contiene el valor j entonces una pareja está formada por los Individuos i y j.

Métodos:

SetPareja : setea el valor de la pareja.
Parámetros:
nuevo valor a asignar.
pos: posición del vector donde se debe asignar el valor.

Mostrar : muestra por pantalla las parejas.

CalcularParejas : selecciona las parejas para realizar el cruce o la selección por torneo binario. Recibe como parámetro la dimensión $nind$ del vector, cantidad de individuos de la población. Las parejas se registrarán en un vector de pares, donde en la posición i del vector (posición donde se encuentra un individuo) se registrará el valor de la posición donde se encuentra el segundo individuo. El algoritmo setea cada elemento del vector con un valor mayor a la cantidad de individuos ($nind$+1). Si la cantidad de individuos es impar, se seleccionará el individuo que no tendrá pareja y se registrará en la posición del vector el valor de dicha posición. Luego, comienza a seleccionar las parejas, recorriendo el vector secuencialmente, de forma tal que, el primer individuo sea el valor i aún no seleccionado, y el segundo individuo sea calculando al azar con un número entero entre la posición i+1 y $nind$. Si el valor encontrado ya posee pareja, se descarta hasta que se encuentre un valor que no posee pareja. El algoritmo finaliza cuando el sistema recorrió todo el vector obteniendo $nind/2$ parejas. El resultado es que todas las parejas posibles son igualmente probables y no puede haber parejas repetidas (salvo que haya individuos repetidos).

Con esta implementación, dado un problema nos ocupamos de determinar cómo representarlo en el individuo, es decir establecer la representación del Genotipo (representación binaria de cada posible solución), cálculo del Fenotipo (se detallará en cada problema) y función objetivo (Fitness). El resto de las clases no varía.

3.5. Aplicaciones y resultados

3.5.1. Problema de la mochila

El problema de la mochila, que a continuación explicaremos, será resuelto con el algoritmo que desarrollamos con el fin de testear nuestros métodos con un problema de optimización combinatoria. Se tiene n objetos y una mochila para transportarlos. Para $j = 1, 2, ..., n$ el objeto j tiene un peso positivo p_j y un beneficio positivo c_j. La mochila puede llevar un peso total que no sobrepase una

cota P.
Se pretende llenar la mochila de forma que el valor total de los objetos transportados sea máximo.

Para un vector $(x_1, .., x_n)$ de componentes binarios, supongamos que $x_j = 0$ representa «no colocar el objeto j en la mochila» y que $x_j = 1$ representa «sí colocar el objeto j en la mochila» entonces el problema se puede formular de la siguiente manera:

$$\text{Maximizar} \sum_{j=1}^{n} x_j c_j$$

sujeto a

$$\sum_{j=1}^{n} x_j p_j \leq P$$

donde $c_j \in N \ \forall j = 1, .., n$; $P \in N$ y $p_j \in N \ \forall j = 1, .., n.$, donde N representa el conjunto de números enteros positivos.

En cuanto a la complejidad se trata de un problema NP-Completo. Esto motiva el interés por utilizar un método heurístico para su resolución. A continuación veremos la resolución mediante un algoritmo genético.

Implementación

Cada cromosoma está formado por n bits de tipo booleano. El bit j toma valor 1 si está presente el objeto y 0 en caso contrario, es decir, cada cromosoma representa una combinación de objetos disponibles a ingresar en la mochila.
Para un cromosoma $x = (x_1, .., x_n)$, notemos que $\sum_{j=1}^{n} c_j x_j$ es el valor de los objetos indicados por x. Definiremos el fitness como:

$$f(x)=\begin{cases} \sum_{j=1}^{n} c_j x_j & si\ g(x) \leq P \\ P - g(x) & si\ g(x) > P \end{cases}$$

donde $g(x)$ es el peso de los objetos indicados por x, o sea:

$$g(x)= \sum_{j=1}^{n} p_j x_j$$

Es decir, el fitness representa el beneficio dentro de la mochila, si el peso no supera P y el exceso de peso (con signo negativo) si el peso supera a P. De esta manera el exceso de peso permite ordenar por fitness aún a los cromosomas que superan el peso permitido. El fenotipo de un individuo x en este problema es un par ordenado cuya primer componente es el beneficio y la segunda componente es el peso:

$$\text{fenotipo} = (\sum_{j=1}^{n} x_j c_j, \sum_{j=1}^{n} x_j p_j)$$

A continuación se detallan los parámetros que se utilizaron para los dos primeros ejemplos. Se trabaja con el mismo lote de pruebas pero variando los algoritmos, se ejecuta primero el algoritmo genético canónico y luego el algoritmo genético con elitismo. Para realizar estas pruebas se adiciona a la implementación estándar, en el archivo de parámetros, la Capacidad y la Cantidad de objetos.

Parámetros

	Sin elitismo	Elitista
Cantidad de generaciones a realizar	1000	1000
Tasa de mutación	0,1	0,1
Longitud del individuo	30	30
Cantidad de individuos de la población	34	34
Elitista (0=No, 1=Si)	0	1
Capacidad	2001	2001

El siguiente lote de $Valores$ y $Pesos$ se generó en forma aleatoria:

$Valores = \{485, 326, 1248, 1421, 322, 1795, 43, 845, 955, 1252, 1009, 1901, 1122, 1094, 738, 574, 1715, 882, 1367, 1984, 1299, 1433, 1682, 72, 1874, 138, 1856, , 1145, 1995, 1529\}$

$Pesos = \{1094, 506, 416, 992, 649, 1237, 457, 1815, 1446, 1422, 791, 1359, 1667, 1598, 7, 544, 1334, 766, 1994, 1893, 633, 1131, 1428, 700, 617, 1874, 1720, 419, 1794, 196\}$

En la **Figura 3.1** se observa la comparación de los dos métodos hasta ahora descriptos.
Podemos confirmar que el algoritmo canónico sin elitismo no ha llegado a la solución esperada, ya que, gracias al algoritmo determinístico de programación dinámica que hemos desarrollado sabemos que el resultado correcto es 6688 (ver **Figura 3.1 izquierda**). Esto se debe, a que no se ha aplicado en el algoritmo ningún tipo de elitismo y aunque en algún momento llegue al resultado deseado, es muy posible que lo pierda. En la **Figura 3.1 izquierda** se puede observar que luego de 1000 generaciones el algoritmo no pasó por la solución óptima. En esta figura, sólo se muestran las primeras 168 generaciones porque el resto de las generaciones tienen un comportamiento similar. A fin de mejorar el algoritmo se agregó elitismo, que consistió en conservar para la siguiente generación el elemento con mayor valor.
Se ejecutó el algoritmo con el mismo lote de pruebas y los mismos parámetros del algoritmo canónico, excepto que se modificó el parámetro elitismo, se cambió el 0 por el 1.
En la **Figura 3.1 derecha**, podemos observar que a partir de la generación 168 se llega al resultado óptimo, es decir, se llega al valor 6688. Como posee elitismo no pierde el resultado óptimo. Queda claro que el algoritmo con elitismo ha mejorado la solución notablemente. Más adelante verificaremos que el algoritmo con control de la diversidad llega con mayor rapidez al resultado deseado. En este caso, no fue necesario incluir las generaciones siguientes dado que desde la 168 en adelante el resultado es 6688.
Cabe aclarar que para los algoritmos genéticos en general, no siempre el algoritmo elitista es superior. En efecto el elitismo puede conducir rápidamente a una solución buena pero no óptima, en la cual el algoritmo queda estancado.

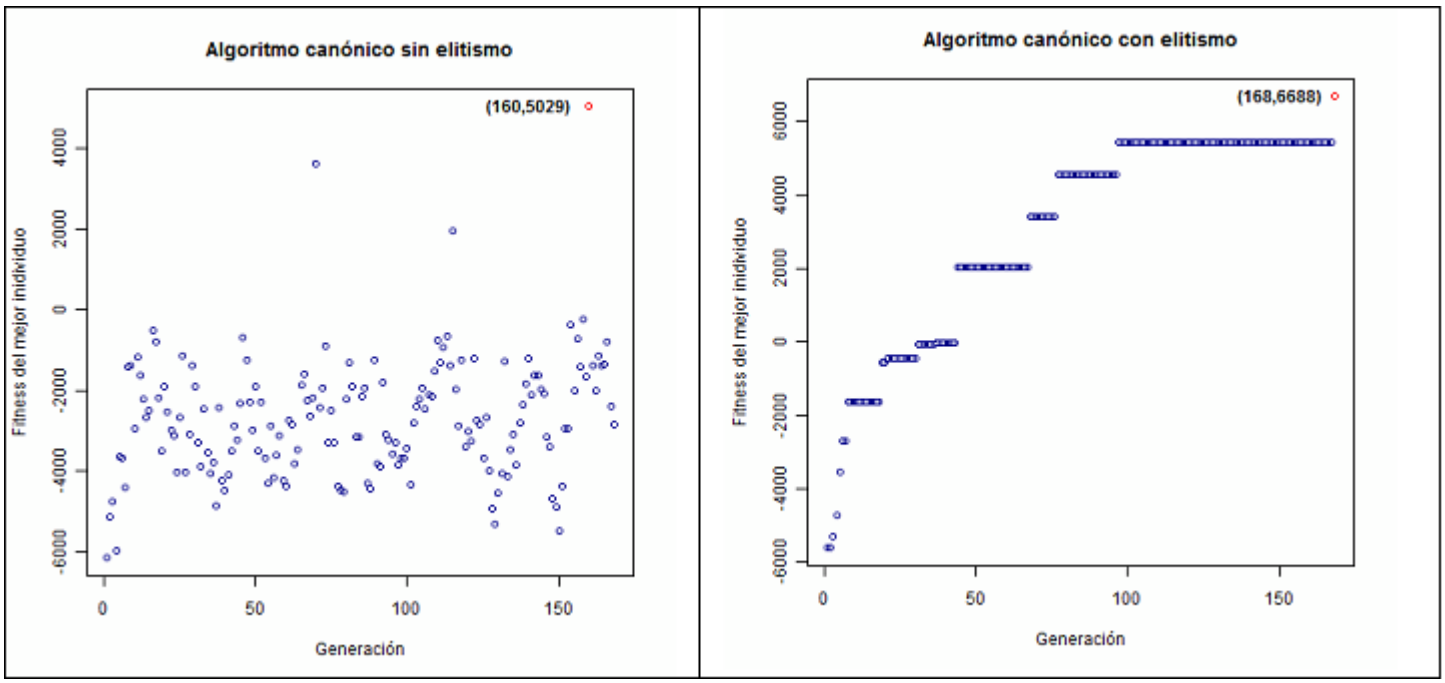

Figura 3.1: AG - Algoritmos de la Mochila

3.5.2. Función continua con varios mínimos locales

Seleccionamos para resolver el problema de minimización de la siguiente función continua:

Encontrar el mínimo de la función: $f(x, y) = x\ sin(4x) + 1,1y\ sin(2y)$
con $0 \leq x \leq 10$ y $0 \leq y \leq 10$ (**Figura 3.2**)

Para calcular la función de fitness el cromosoma del individuo contendrá las componentes x e y, las $n/2$ primeras posiciones corresponderán a x y las restantes a y. En la representación binaria de x las cuatro primeras posiciones serán para la parte entera del número y el resto para los decimales, lo mismo ocurre con y. Por ejemplo el cromosoma 0001010001111000 de longitud 16 representa x=1,25 e y=7,5. De la misma manera que en el ejemplo anterior, ejecutaremos el algoritmo con el parámetro de Elitismo en 0 y en 1.

Parámetros

	Sin elitismo	Elitista
Cantidad de generaciones a realizar	4000	4000
Tasa de mutación	0,1	0,1
Longitud del individuo	16	16
Cantidad de individuos de la población	32	32
Elitista (0=No, 1=Si)	0	1

Los resultados arrojados se pueden visualizar en la **Figura 3.3**.

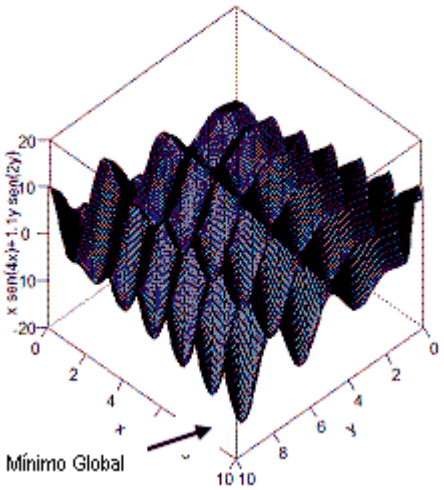

Figura 3.2: Función continua

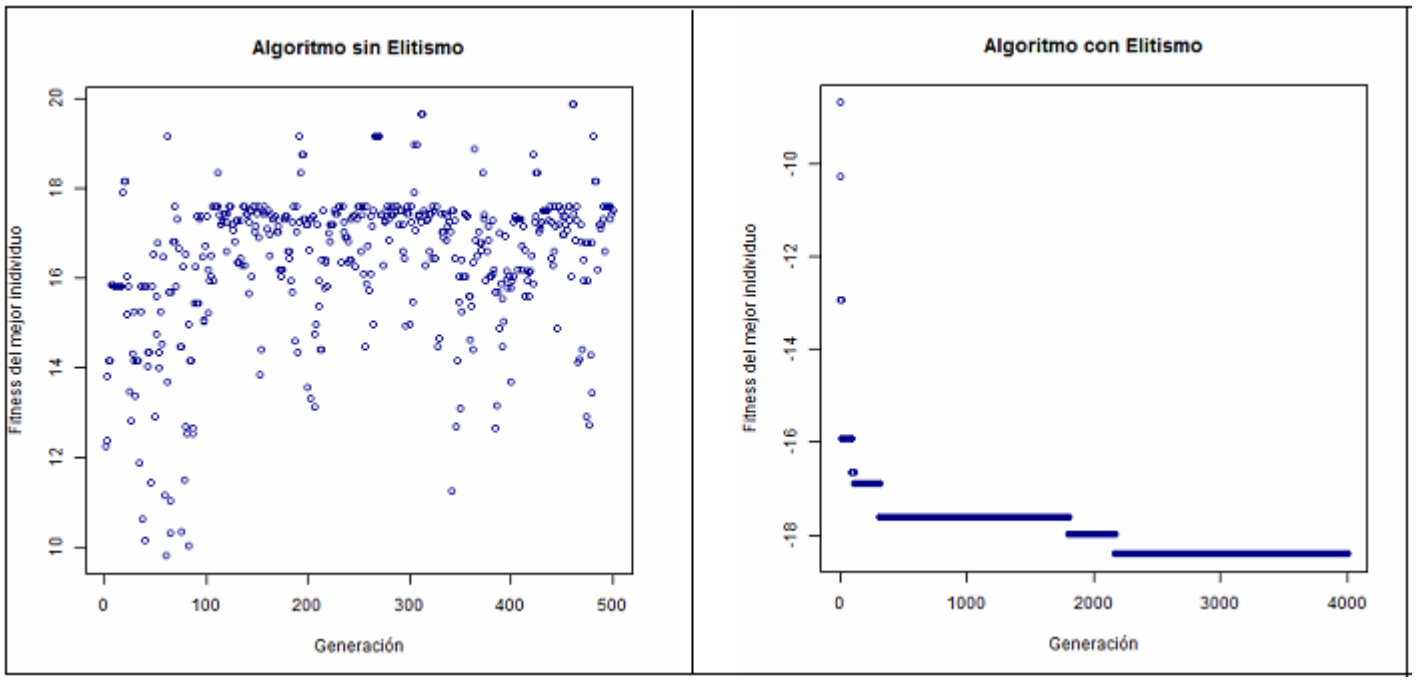

Figura 3.3: AG - Algoritmos de la Función Continua

Si bien con ninguno de los dos métodos se llegó al óptimo global, en el caso del algoritmo canónico sin elitismo los resultados están muy alejados de este óptimo. El óptimo goblal es -18.5547 y se alcanza en el punto (9,039, 8,668). En este caso la codificación binaria con la distancia de Hamming no representa adecuadamente la distancia euclídea y no permite aprovechar la continuidad de la función, por eso no se utilizará para comparar los métodos de dispersión que se analizarán en los capítulos siguientes.

Capítulo 4

Diversidad poblacional

4.1. Biodiversidad y su importancia para la evolución

El concepto de biodiversidad es empleado por los estudiosos de la naturaleza como los naturalistas, los biólogos, los ecólogos o los botánicos, que se dedican a analizar las poblaciones en cuanto a sus características morfológicas, funcionales, genéticas y evolutivas. Para ellos significa la variedad de los organismos biológicos.
La variación genética es importante para las adaptaciones que se producen en una especie, en respuesta al medio ambiente y a las adaptaciones de otras especies, con las que interactúan.
La selección natural opera reduciendo la frecuencia de ciertos fenotipos en la población que peor se ajustan a las condiciones del ambiente y aumentando la frecuencia de los fenotipos mejor ajustados, siempre que existan opciones para seleccionar. La ausencia de variabilidad genética impide la selección natural, lo que puede llevar a la extinción de la población.
Cualquier estrategia de protección del medio natural debe asegurar la salvaguardia de la biodiversidad. El conjunto de los seres vivos que habita un país constituye un patrimonio insustituible porque cada especie, e incluso cada población, alberga en su genoma la información de millones de años de adaptaciones evolutivas. Poblaciones y especies enteras están desapareciendo debido a la perturbación ejercida sobre el medio por las actividades humanas y ése es quizás el mayor reto ambiental al que se enfrenta la humanidad.
Ante el riesgo evidente de pérdida de diversidad biológica que las actividades humanas están produciendo, el propósito esencial de las reservas naturales es la protección de la biodiversidad. Sin embargo, tanto para decidir dónde debemos situar nuestras reservas como para evaluar su estado, es necesario medir su variación en el espacio y en el tiempo [LOB/01].
La biodiversidad se mide a nivel genético, a nivel de especies y a nivel de comunidades.
En esta tesis queremos mostrar que en el área de algoritmos genéticos, medir la diversidad y tomar acción aplicando algún método que aumente la variabilidad en la población en caso de que ésta se encuentre reducida, también es muy conveniente.
La medición de diversidad se realiza en este caso a escala genética, no a nivel de especies o comunidades, dado que en el estado actual de los algoritmos genéticos, lo más usual es trabajar con una sola población.

4.2. Distancias entre individuos

Con el objetivo de medir la diversidad de la población se debe obtener en primer lugar una distancia entre individuos. Más adelante veremos que esta distancia es utilizada en el método para cuantificar la diversidad.
Adoptamos la distancia de Hamming, lo cual es natural y frecuente para comparar cadenas binarias en ausencia de supuestos sobre el significado de cada bit. Sin embargo, nuestro estudio de la diversidad podría adaptarse a otras distancias porque las variantes que presentamos para cuantificar la diversidad se generalizan a cualquier espacio métrico.

4.2.1. Distancia de Hamming

Definición

La distancia de Hamming entre dos palabras binarias de igual longitud, es el número de bits en que difieren una de la otra.
Por ejemplo con las dos cadenas siguientes:

$$\begin{array}{cccccccc} 0 & 0 & \mathbf{1} & \mathbf{0} & 1 & 0 & 1 \\ 0 & 0 & \mathbf{0} & \mathbf{1} & 1 & 0 & 1 \end{array}$$

la distancia de Hamming es 2.

4.3. Medidas de diversidad de un grupo

Dadas las distancias entre pares de un conjunto de individuos, el siguiente paso consiste en cuantificar la diversidad de ese conjunto. Si bien hay muchos métodos posibles de utilizar, como contar el número de individuos no repetidos, calcular la máxima distancia entre dos individuos del conjunto, etc., la opción que adoptamos es promediar las distancias a cierto punto central del conjunto. Esta opción, generaliza la manera usual de cuantificar la dispersión de un conjunto de números reales, por ejemplo cuando se calcula la varianza.
Un antecedente importante para nuestro trabajo es el artículo de Sidaner, Bailleux y Chabrier [SID/02], donde se detalla un método para medir la dispersión de un conjunto de puntos en un espacio métrico, es decir, un conjunto donde se encuentra definida una distancia entre sus elementos. Este método permite cuantificar la diversidad de la población a fin de analizar y/o mejorar la convergencia de los algoritmos evolutivos. A este método, que utiliza el punto medio, le agregamos nuestra idea de disminuir el valor de la dispersión ante la presencia de individuos repetidos. De esta manera, combinamos una medida estándar con una modificación adecuada a nuestra problema.
A continuación describimos entonces dos métodos para medir la dispersión. El primero corresponde al descripto en el artículo recientemente referenciado, el segundo es el propuesto por nosotros que consideramos bien adaptado al problema de los algoritmos genéticos porque busca evitar que haya demasiados individuos repetidos, típicos del estancamiento de la población en un óptimo local.

4.3.1. Método de cálculo de dispersión 1

A continuación presentaremos un conjunto de definiciones que utilizaremos para describir el método del artículo ([SID/02]);

Definiciones y propiedades

Llamamos $multiconjunto$ a un conjunto que admite elementos repetidos.
Sea Y un multiconjunto finito de puntos del espacio $E = \{0,1\}^l$ (los vectores de l componentes binarias). Si bien Y puede tener elementos repetidos, existe un conjunto $\{x_1,..,x_p\} \subset Y$ que contiene los elementos distintos que hay en Y. El multiconjunto Y representará una población de cromosomas.
Sea $noc(x_i)$, $1 \leq i \leq p$, el número de ocurrencias de x_i en Y.
Sea $|Y| = \sum_{i=1}^{p} noc(x_i)$, es decir, $|Y|$ es la cantidad total de elementos que contiene el multiconjunto Y.
Para un $\alpha, \beta \in \{0,1\}$, denotamos $dif(\alpha,\beta) = max(\alpha,\beta) - min(\alpha,\beta)$.
Para un $x_i \in \{0,1\}^l$ y $j = 1,..,l$, x_{ij} denota la j-ésima componente de x_i.
Para un par $(x_i, x_k) \in \{0,1\}^l \times \{0,1\}^l$, $h(x,y) = \sum_{j=1}^{l} dif(x_{ij}, x_{kj})$, denota entonces la distancia de Hamming entre x_i y x_k.

Definimos la proximidad entre $x \in E$ e $Y \subset E$ mediante $\overline{H}(x,Y) = \frac{1}{|Y|} \sum_{\alpha \in Y} h(x,\alpha)$, es decir, el promedio de distancias entre x y los puntos de Y.
Llamamos $punto\ promedio$ de Y a un punto $\overline{x}$ en E (que puede no ser único y además puede no estar en Y) que minimiza $\overline{H}(x,Y)$.
La $dispersión\ de\ un\ multiconjunto\ Y$, que llamaremos $disp_1$ para distinguirla de la que definiremos luego, es el valor que toma $\overline{H}(x,Y)$ cuando se alcanza ese mínimo.
Claramente, una dispersión nula ocurre sólo cuando Y colapsa al mismo punto con cualquier número de ocurrencias.
A continuación definiremos el vector de dispersión, que facilitará el cálculo de la dispersión de un multiconjunto.

Vector de dispersión

Sea Y el multiconjunto de puntos que contiene $x_1,..,x_n$, donde $x_i \in \{0,1\}^l$ con $1 \leq i \leq n$, y $|Y| = n$. Llamamos $vector\ de\ dispersión$ de Y al vector $W = (w_1,...,w_l)$, tal que,
$w_j = \frac{1}{|Y|} \sum_{i=1}^{|Y|} x_{ij}$, es decir, que la componente j de W es el promedio de las componentes j de los elementos de Y.

Daremos un ejemplo para facilitar la interpretación:

Representaremos con una matriz a un conjunto Y de cromosomas. Cada cromosoma x_i de Y es una fila, donde x_i representa (x_{i1},..,x_{il}), y cada elemento w_j del vector W es el resultado de aplicar la fórmula anterior a cada columna j de la matriz.

Sea $Y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ entonces

$|Y| = 3$, es decir, n=3

$x_1 = (0,1,0,0)$ $\qquad$ $x_2 = (1,0,0,0)$ $\qquad$ $x_3 = (1,1,1,1)$

$w_1 = \frac{1}{3}(0+1+1)$ $\qquad$ $w_2 = \frac{1}{3}(1+0+1)$ $\qquad$ $w_3 = \frac{1}{3}(0+0+1)$ $\qquad$ $w_4 = \frac{1}{3}(0+0+1)$

El siguiente teorema de [SID/02] da una fórmula sencilla para el cálculo de la proximidad de un punto de x a Y, usando el vector de dispersión.

Teorema

Sea $x \in \{0,1\}^l$ con componentes binarios $x_1,..,x_l \in \{0,1\}$ y $W = (w_1,..,w_l)$ el vector de dispersión de un multiconjunto Y.

Entonces $\overline{H}(x,Y) = \sum_{i=1}^{l}((1-x_i)w_i + x_i(1-w_i))$.

Demostración: Por definición, $\overline{H}(x,Y) = \frac{1}{|Y|}\sum_{\alpha \in Y} h(x,\alpha) = \frac{1}{|Y|}\sum_{\alpha \in Y}\sum_{i=1}^{l} dif(x_i,\alpha_i)$.

Entonces $\overline{H}(x,Y) = \frac{1}{|Y|}\sum_{i=1}^{l}\sum_{\alpha \in Y} dif(x_i,\alpha_i)$.

$\overline{H}(x,Y) = \frac{1}{|Y|}\sum_{i=1..l/x_i=0}(\sum_{\alpha \in Y} dif(x_i,\alpha_i)) + \frac{1}{|Y|}\sum_{i=1..l/x_i=1}(\sum_{\alpha \in Y} dif(x_i,\alpha_i))$.

$\overline{H}(x,Y) = \frac{1}{|Y|}\sum_{i=1..l/x_i=0}(\sum_{\alpha \in Y} \alpha_i) + \frac{1}{|Y|}\sum_{i=1..l/x_i=1}(\sum_{\alpha \in Y}(1-\alpha_i))$.

$\overline{H}(x,Y) = \sum_{i=1..l/x_i=0}\frac{1}{|Y|}\sum_{\alpha \in Y}\alpha_i + \sum_{i=1..l/x_i=1}(1 - \frac{1}{|Y|}\sum_{\alpha \in Y}\alpha_i)$

$\overline{H}(x,Y) = \sum_{i=1..l/x_i=0} w_i + \sum_{i=1..l/x_i=1}(1-w_i)$

Este último es el resultado que buscamos porque si $x_i = 0$, entonces $(1-x_i)w_i = w_i$ y $x_i(1-w_i) = 0$, mientras que si x_i=1, entonces $(1-x_i)w_i = 0$ y $x_i(1-w_i) = 1-w_i$.

Fórmula para medir la dispersión de Y

Recordemos que llamamos dispersión de Y al promedio de distancias entre los puntos de Y y un punto en E que minimiza este promedio.

Es decir que la dispersión es $min\{\overline{H}(x,Y)\} = min\left\{ \sum_{i=1..l/x_i=0} w_i + \sum_{i=1..l/x_i=1} (1-w_i) \right\}$ con $x \in E$.
Ahora notamos que $\overline{H}(x,Y)$ consiste en sumar para i desde 1 hasta l, w_i o $1-w_i$ según si $x_i = 0$ o $x_i = 1$. Luego, el valor mínimo de $\overline{H}(x,Y)$ será aquel en el que para cada i se suma el valor más pequeño entre w_i y $1-w_i$. Así nos queda la fórmula

$\boxed{disp_1(Y) = \sum_{i=1}^{l} min(w_i, 1-w_i)}$, donde w_i es el promedio de la columna i

donde Y es el multiconjunto de cadenas binarias con vector de dispersión $W = (w_1, .., w_l)$.

Volviendo al ejemplo anterior tenemos

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

Calculamos la dispersión de Y.

$|Y| = 3$
$w_1 = \frac{1}{3}(0+1+1) = \frac{2}{3} \quad w_2 = \frac{1}{3}(1+0+1) = \frac{2}{3} \qquad w_3 = \frac{1}{3}(0+0+1) = \frac{1}{3} \quad w_4 = \frac{1}{3}(0+0+1) = \frac{1}{3}$

entonces
$disp_1(Y) = \sum_{i=1}^{l} min(w_i, 1-w_i) = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = \frac{4}{3}$

4.3.2. Método de cálculo de dispersión 2

El método presentado en [SID/02], detallado en el item 4.3.1, obtiene el vector de dispersión de una población, sin más análisis de los individuos que la componen. Por este motivo, el resultado obtenido cuando la población posee poca diversidad no es adecuado si hay pocos grupos homogéneos en la población. Para comprender esto notemos primero que el valor más grande que puede tomar $min(w_i, 1-w_i)$ es $\frac{1}{2}$. Ahora consideramos una población Y con la mitad de individuos iguales entre sí y la otra mitad también iguales entre sí pero a distancia máxima de las anteriores, por ejemplo $(1,0,1,0)$ y $(0,1,0,1)$. En este caso se tiene una alta dispersión según la medida anterior. En efecto, se tiene $w_i = \frac{1}{2}$ para todo i y por lo tanto se trata de una población con dispersión máxima. Pero la gran cantidad de individuos repetidos no se corresponde con una alta diversidad.
Nuestra propuesta presenta entonces una variación en el cálculo de la dispersión a fin de mejorar el resultado obtenido, teniendo en cuenta no sólo dicho valor, sino también, penalizando los casos en que la población presente poca diversidad.
Para definir la nueva dispersión definimos previamente:
Sea Y un multiconjunto de puntos que contiene todos los puntos $x_1, .., x_n$, donde $x_i \in \{0,1\}^l$ con $1 \leq i \leq n$.

Sea $\|Y\|= \sum_{j=1}^{n}\sum_{i=1}^{n}\begin{cases} 1 \text{ si } x_i = x_j \\ \\ 0 \text{ si } x_i \neq x_j \end{cases}$

Notemos que $\|Y\|$ tomará un valor alto si hay muchos elementos repetidos en Y y que toma valores entre n y n^2.

Y'=conjunto de elementos de Y en el que no se incluyen las repeticiones, es decir que Y'=Y si no hay elementos repetidos.

$|Y'|$= cantidad de elementos de Y'.

$v_j = \sum_{x_i \in Y'}^{|Y'|} x_{ij}$, es decir, que la componente j de v es la suma de las componentes j de los elementos de Y'.

Ahora definimos la nueva medida de dispersión mediante:

$disp_2(Y) = \frac{1}{\|Y\|}\sum_{i=1}^{l} min(v_i, |Y'| - v_i)$ que es igual a $\frac{|Y'|}{\|Y\|}\sum_{i=1}^{l} min(\frac{v_i}{|Y'|}, 1 - \frac{v_i}{|Y'|})$

Es decir, la nueva dispersión es:

$$\boxed{disp_2(Y)= \frac{|Y'|}{\|Y\|} disp_1(Y')}$$

Cabe observar que si todos los elementos de Y son iguales $disp_1$ y $disp_2$ coinciden en 0. Por otro lado, si Y no tiene elementos repetidos, $disp_1$ y $disp_2$ coinciden en el mismo valor porque Y=Y' y $|Y'| = \|Y\| = n$, por lo tanto, $disp_1(Y') = disp_1(Y)$. Es decir que, ambas maneras de calcular la dispersión se mantienen en el mismo rango de valores. Además, si existen elementos repetidos $disp_2 \leq disp_1$. Para demostrarlo, vamos a escribir nuevamente $disp_1$ y $disp_2$.

Recordemos: $disp_1(Y) = \sum_{i=1}^{l} min(w_i, 1 - w_i)$ y $w_i = \frac{1}{|Y|}\sum_{j=1}^{|Y|} x_{ji}$. Por lo tanto, podemos escribir $disp_1(Y) = a/|Y|$, donde a representa $min(\sum_{j=1}^{|Y|} x_{ji}, |Y| - \sum_{j=1}^{|Y|} x_{ji})$.

De manera similar podemos escribir $disp_2(Y) = b/\|Y\|$, donde b=$\sum_{i=1}^{l} min(v_i, |Y'| - v_i)$.

Veamos ahora que $b/\|Y\| \leq a/|Y|$, esto es $b\,|Y| \leq a\,\|Y\|$. Sabemos que $|Y| < \|Y\|$ si hay elementos repetidos. Sólo resta ver que $b \leq a$. Si hay elementos repetidos significa que Y' contendrá menos filas que Y entonces $b \leq a$ por construcción. En efecto, notemos que:

$\sum_{j=1}^{|Y|} x_{ji}$ es la suma de 1s en a columna i de Y.

$|Y|$-$\sum_{j=1}^{|Y|} x_{ji}$ es la suma de 0s en a columna i de Y.

v_i es la suma de los 1s en la columna i de Y'.

$|Y'| - v_i$ es la suma de los 0s en la i de Y'.
Pero la columna i de Y' resulta de extraer algunos 1s y/o 0s de la columna i de Y. Por lo tanto:

$\sum_{j=1}^{|Y|} x_{ji} \geq v_i$ y

$|Y|$-$\sum_{j=1}^{|Y|} x_{ji} \geq |Y'| - v_i$.

Así resulta que

$min(\sum_{j=1}^{|Y|} x_{ji}, |Y| - \sum_{j=1}^{|Y|} x_{ji}) \geq min(v_i, |Y'| - v_i)$ como queríamos.

Ejemplo:

Sea $Y = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ entonces

$\|Y\|$ =9

Las filas de la matriz Y que se tienen en cuenta para el cálculo son las siguientes de la matriz:

$Y' = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. Entonces $|Y'| = 3$

$v_1 = (0 + 0 + 0)$ $\quad$ $v_2 = (1 + 0 + 0)$ $\quad$ $v_3 = (0 + 0 + 1)$ $\quad$ $v_4 = (1 + 1 + 0)$

$disp_2(Y) = \frac{1}{9}(0 + 1 + 1 + 1) = \frac{3}{9} = \frac{1}{3}$.

A continuación presentamos dos ejemplos que permiten comparar $disp_1$ y $disp_2$ en presencia o no de repeticiones.

Sean las siguientes matrices:

$M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ y $M_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

$disp_1(M_1) = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = 1$

$disp_2(M_1) = \frac{1}{4}(1 + 1 + 1 + 1) = 1$

$disp_1(M_2) = \frac{2}{4} + \frac{2}{4} + \frac{0}{4} + \frac{0}{4} = 1$

$disp_2(M_2) = \frac{1}{8}(1 + 1 + 0 + 0) = \frac{1}{4}$.

En la matriz M_1 se muestra que el resultado obtenido aplicando el método de $disp_1$ o de $disp_2$ es el mismo. Esto ocurre porque no existen individuos repetidos. Ambos métodos obtienen resultados parecidos cuando encontramos diversidad en la población.
En la matriz M_2 se muestra que es más apropiado el valor calculado aplicando el método de $disp_2$ que $disp_1$, esto ocurre porque encontramos menor diversidad en la población. El método $disp_1$ no penaliza las filas repetidas. Si bien los términos "dispersión" y "diversidad" no poseen una definición objetiva generalmente aceptada y podrían tomarse como sinónimos, podemos concluir esta sección observando que $disp_1$ da una idea de dispersión espacial, mientras que $disp_2$ se adecúa más a la diversidad poblacional.

4.4. Cota superior para la dispersión

Como vimos en la sección anterior, del cálculo de la dispersión se obtiene un número que refleja la diversidad poblacional. Es necesario analizar este número para poder determinar si estamos en presencia de una población diversa o no y con esta información decidir si debería aplicarse un aumento de la tasa de mutación. La cota inferior para la dispersión es cero, que corresponde al caso en el que todos los individuos son iguales. Necesitamos conocer una cota superior que le permita al programa evaluar si el valor de dispersión calculado en cada ejecución se encuentra dentro de un rango de dispersiones adecuado. La cota que no quisiéramos superar y que nos servirá como referencia es la dispersión esperada para una población generada al azar, como lo es la población inicial.

A continuación, deducimos la fórmula para obtener la esperanza E_1 de la dispersión para el método 1, para una población en la que cada bit puede ser 0 o 1 con igual probabilidad y su valor es independiente de los demás bits. Consideramos como antes que $Y = (x_1, .., x_n)$ es un multiconjunto de puntos $x_i \in \{0, 1\}^l$ y que $W = (w_1, .., w_l)$ es su vector de dispersión. Este vector es ahora aleatorio. Para facilitar el cálculo definimos $T = (t_1, .., t_l)$, donde $t_j = w_j n$, es decir que t_j es simplemente el número de bits con valor 1 en las componentes j de los n cromosomas aleatorios de Y.

Tenemos que $disp_1 = \sum_{i=1}^{l}(\text{mín}(w_j, 1 - w_j)) = \frac{1}{n}\sum_{i=1}^{l}(\text{mín}(t_j, n - t_j)\,)$

Luego la esperanza de esta variable aletoria es

$E_1 = E(disp_1) = E(\sum_{j=1}^{l}(\text{mín}(w_j, 1 - w_j)) = E(\frac{1}{n}\sum_{j=1}^{l}(\text{mín}(t_j, n - t_j)),$

que por linealidad de la esperanza da

$\frac{1}{n}\sum_{j=1}^{l} E(\text{mín}(t_j, n - t_j))$

El valor de cada sumando es igual para todo j porque el valor esperado de unos no depende de la componente que analicemos. Veamos que además es fácilmente calculable. Por definición de

esperanza y dado que el rango de valores posibles de $\min(t_j, 1-t_j)$ va de 0 a $[n/2]$ (la parte entera de $n/2$), se tiene para todo j:

$E(\min(t_j, n-t_j)) = \sum_{i=0}^{[n/2]} i \cdot P(\min(t_j, n-t_j) = i)$

Si $i < n/2$, el evento $\min(t_j, n-t_j) = i$ se compone de $t_j = i$ y $n - t_j = i$, es decir que su probabilidad es la suma de dos probabilidades binomiales iguales, por lo cual:

$P(\min(t_j, n-t_j) = i) = P(t_j = i) + P((n-t_j) = i) = 2\binom{n}{i}(1/2)^n.$

Si $i = n/2$, lo cual ocurre sólo si n es par, tenemos:
$P(\min(t_j, n-t_j) = n/2) = P(t_j = n/2) = \binom{n}{n/2}(1/2)^n.$

De modo que obtenemos si n es impar

$$E_1 = \frac{1}{n}\sum_{j=1}^{l}\sum_{i=0}^{(n-1)/2} i \cdot 2\binom{n}{i}(1/2)^n = \frac{l}{n}\sum_{i=0}^{(n-1)/2} 2i\binom{n}{i}(1/2)^n$$

Y si n es par

$$E_1 = \frac{1}{n}\sum_{j=1}^{l}\left(\frac{n}{2}\binom{n}{n/2}(1/2)^n + \sum_{i=0}^{(n/2)-1} i \cdot 2\binom{n}{i}(1/2)^n\right) = \frac{l}{n}\left(\frac{n}{2}\binom{n}{n/2}(1/2)^n + \sum_{i=0}^{(n/2)-1} 2i\binom{n}{i}(1/2)^n\right)$$

En cuanto a la esperanza de la dispersión calculada por el método 2 de una población al azar, la fórmula es de cálculo más complicado porque involucra la probabilidad de repeticiones. Sin embargo, utilizaremos la fórmula de E_1 como cota, también para nuestros experimentos con la dispersión $disp_2$. Esto se justifica en que $disp_1 = disp_2$ si no hay repetición de cromosomas y como veremos, en una población generada al azar todos los individuos son distintos con probabilidad prácticamente igual a 1. Esto último se debe a que en un algoritmo genético, el número de genes por cromosoma suele ser lo suficientemente alto como para que sea poco probable que se generen dos cromosomas iguales por azar, como veremos a continuación.

Calcularemos la probabilidad de ausencia de repeticiones utilizando el argumento del llamado "problema del cumpleaños". Se trata de obtener la probabilidad de que en un grupo de n personas no haya siquiera dos que cumplan años el mismo día, o bien la probabilidad del evento complementario, que haya alguna coincidencia en los cumpleaños. Se supone que cada persona tiene la misma probabilidad de haber nacido en cualquier día del año y se ignoran los años bisiestos para simplificar el problema. Llamemos P_n a la probabilidad de que en un grupo de n personas, no haya dos que cumplan el mismo día. Para calcular esta probabilidad notemos que los casos posibles son 365^n. Por otro lado, los casos favorables (ninguna coincidencia) son $365 \cdot 364 \cdot ... \cdot (365 - (n-1))$. En efecto, una de las n personas puede cumplir años cualquiera de los 365 días, pero si no queremos contar coincidencias, otra persona que consideremos del grupo sólo puede cumplir en uno de los 364 días restantes, lo cual da hasta aquí $365 \cdot 364$ posibilidades, y así sucesivamente hasta considerar todas

las personas del grupo obteniéndose $365 \cdot 364 \cdot ... \cdot (365 - (n-1))$. Este número se puede denotar con factoriales mediante $365!/(365-n)!$. De manera que

$$P_n = \frac{365!/(365-n)!}{365^n} = \frac{365!}{365^n(365-n)!}$$

Para el caso de la población generada al azar el número 365 se reemplaza por 2^l donde l es la longitud de los cromosomas y n es el tamaño de la población. Luego la fórmula es

$$P_n = \frac{(2^l)!}{(2^l)^n(2^l-n)!}$$

que toma valores muy pequeños para los rangos de valores usuales de l y n. Esto se debe a que el número n de cromosomas de una población es muy pequeño frente al número total 2^l de cromosomas posibles, de modo que si se toman al azar n de un total de 2^l, la probabilidad de que dos sean iguales es muy pequeña. Mencionemos de paso que esta relación entre el pequeño número de cromosomas de una población o aún de muchas generaciones, con el gran número de cromosomas posibles llama la atención sobre la capacidad de los algoritmos genéticos de encontrar óptimos entre tantas posibilidades recorriendo sólo una pequeña porción de ellas.

Capítulo 5

Algoritmos genéticos con control de diversidad

5.1. La diversidad y el equilibrio entre exploración y explotación

Las técnicas estocásticas, al recorrer el espacio de búsqueda del problema, deben tener un balance adecuado entre la exploración y explotación que se hace sobre el espacio de soluciones posibles. La exploración permite que el algoritmo se mueva sobre todo el espacio de soluciones, con lo que evitamos los mínimos locales. Por otro lado, la explotación permite, al encontrar una posible solución óptima, moverse en el espacio circundante a ésta para encontrar la que más se acerque el óptimo en dicho entorno.
De estas técnicas, los algoritmos genéticos son un tipo de algoritmo de búsqueda de propósito general, cuyos operadores pueden establecer un equilibrio adecuado entre exploración y explotación, ya que, aunque el algoritmo genético esté cercano al punto de convergencia, el operador de cruce nos permite explorar individuos, posibles soluciones. Por otra parte, el operador de mutación permite al algoritmo genético una búsqueda sobre un área determinada del espacio de soluciones. Si se aplica una excesiva de tasa de mutación se obtiene una búsqueda aleatoria y el algoritmo comienza a generar en cada paso una población inicial al azar, por el contrario, si se predomina la explotación local con una fuerte presión de selección el algoritmo converge a un óptimo local.
El objetivo de la mutación en los algoritmos genéticos es introducir diversidad (aleatoriamente) para explorar nuevas secciones del espacio de búsqueda; por lo tanto si se tiene el control de la diversidad se podrán tomar decisiones acerca de cambiar el espacio de búsqueda a través de la mutación y así evitar el estancamiento.

5.2. Propuesta para el control de la diversidad

Nuestro objetivo es controlar la diversidad de la población para mejorar la velocidad de convergencia del algoritmo genético y encontrar una solución óptima al problema. Pero como ya conocemos, debemos ser cuidadosos con el mantenimiento de la diversidad porque en caso contrario la presión de selección puede provocar convergencia prematura a un óptimo local.
Para evitar caer en este inconveniente nuestro problema se reduce a encontrar: un valor de dispersión

para conocer la diversidad de la población, una cota de dispersión para decidir si se debe agregar diversidad y un método que nos permita agregar diversidad.

5.2.1. Modificación del algoritmo básico con el fin de controlar la diversidad

Una vez encontrados estos métodos para controlar la diversidad, es necesario incluirlos en el algoritmo básico. En el gráfico que está en la **Figura 5.1** se observan las modificaciones necesarias para incluir el control. Las líneas y leyendas en azul corresponden a la modificación del algoritmo para control de la diversidad.

5.2.2. Cómo controlamos la diversidad en nuestro algoritmo

Para controlar la diversidad en nuestro algoritmo, realizamos las siguientes modificaciones:

- Incorporamos para obtener el valor dispersión, los métodos expuestos en el capítulo 4 ($disp_1$ o $disp_2$, según el parámetro).
- Incorporamos como cota de dispersión la fórmula: $\frac{E}{r}$, donde E es la esperanza (E_1) detallada en el capítulo 4.4, que como ya indicamos sirve para acotar superiormente a las dos dispersiones y $r \in \Re \geq 1$.
  La esperanza representa el valor máximo de dispersión cuando la dimensión de la población es considerablemente grande.
- Incrementamos la tasa de mutación cuando el valor de dispersión es menor a la cota de dispersión.

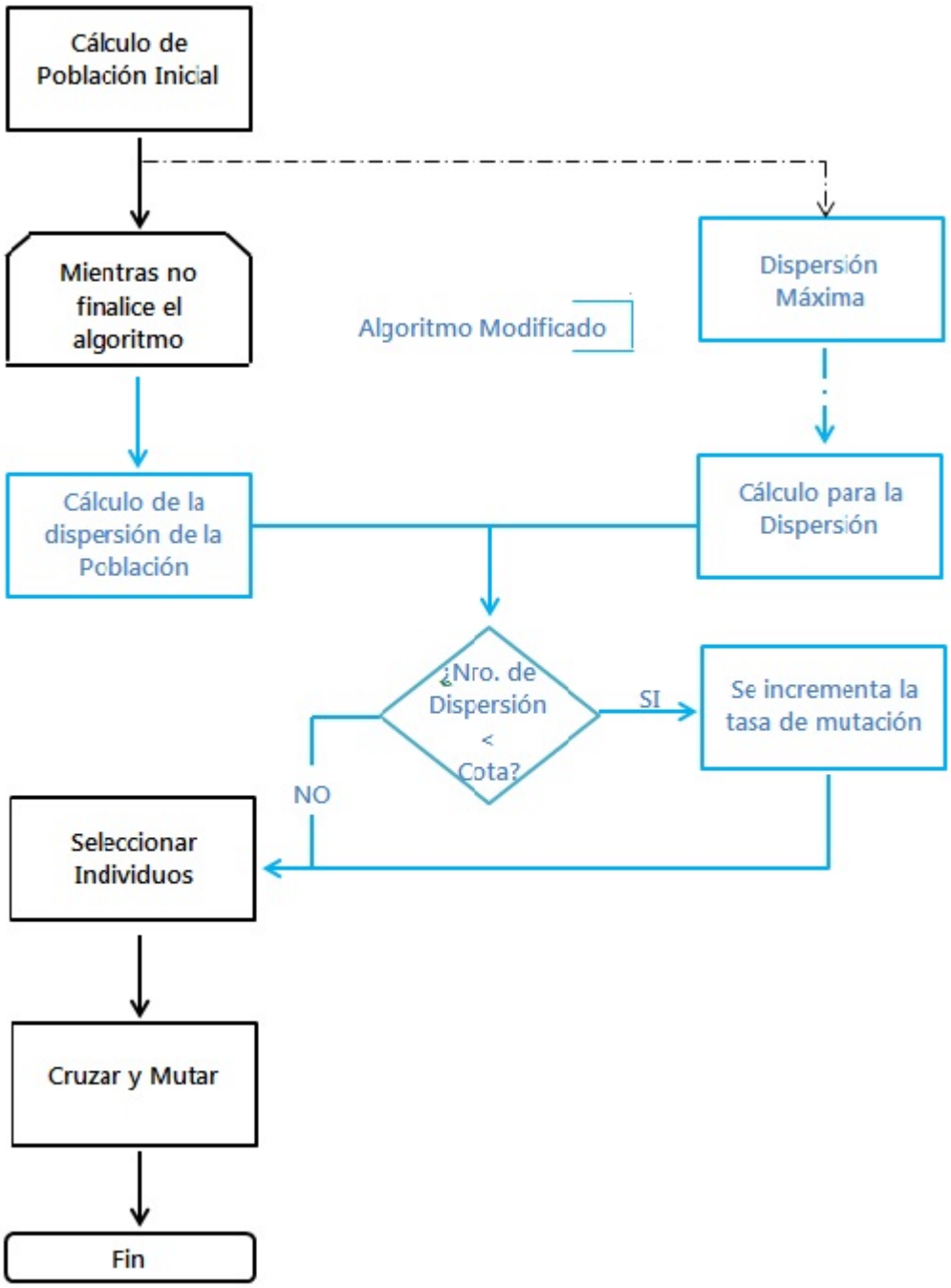

Figura 5.1: Esquema del Algoritmo Genérico con Control de Diversidad

Cómo se ve en el gráfico de la **Figura 5.2**, antes de modificar una población (cruzar y mutar), se calcula la esperanza y el valor de dispersión para poder decidir si se incrementa la tasa de mutación para agregar diversidad a la población.

5.2.3. Consideraciones sobre el valor más conveniente para la cota de dispersión

A continuación presentaremos un conjunto de definiciones que utilizaremos más abajo:
d: representa $disp_1$ y $disp_2$ (según la que se esté utilizando).
E: Esperanza de la dispersión en una población tomada al azar (coincide con la dispersión máxima).
$C(r) = \frac{E}{r}$, la función que calcula la cota de dispersión.

Como nuestro objetivo inicial era encontrar un r apto para cualquier algoritmo de optimización, nos formulamos la siguiente pregunta: ¿Cuál será el r de la función C, tal que, nos permita decidir si es necesario aplicar un método para incorporar diversidad?
Para descubrir el r mencionado realizamos pruebas sobre dos problemas: el problema de la mochila y el problema de clasificación de imágenes. Le aplicamos varios parámetros a cada uno. Los mismos serán detallados en la próxima sección. De acuerdo a estas pruebas concluimos, como era esperable, que no se puede encontrar una única cota natural para todos los problemas.

Notemos que:

- E es la dispersión máxima entonces $d \leq E$
- Si $d = E$ la dispersión es máxima.
- Si $d = 0$ entonces la población es homogénea.

Consideraciones sobre un valor conveniente para r:

- Cuando estamos ante una función con muchos máximos/mínimos locales, se necesita incrementar la mutación con frecuencia alta para evitar los estancamientos. ¿Cómo se logra este efecto en el algoritmo nuestro?
  Si r=1 entonces se va a cumplir siempre la condición $d \leq E/r$ y estaremos aumentando constatemente la tasa de mutación sin control de la dispersión.
  Es conveniente un r lo suficientemente chico para favorecer la mutación pero más grande que 1 para que el aumento no se produzca constantemente. (por ejemplo r=2).
- Si estamos ante una función que no tiene muchos máximos/mínimos locales, no es necesario mutar con tanta frecuencia. Lo que hacemos es tomar un r relativamente grande para que tarde más en cumplirse la condición $d \leq E/r$. Esto favorece la explotación: el procedimiento aprovecha el aprendizaje que hizo para acercarse al óptimo y busca en su entorno para encontrarlo.

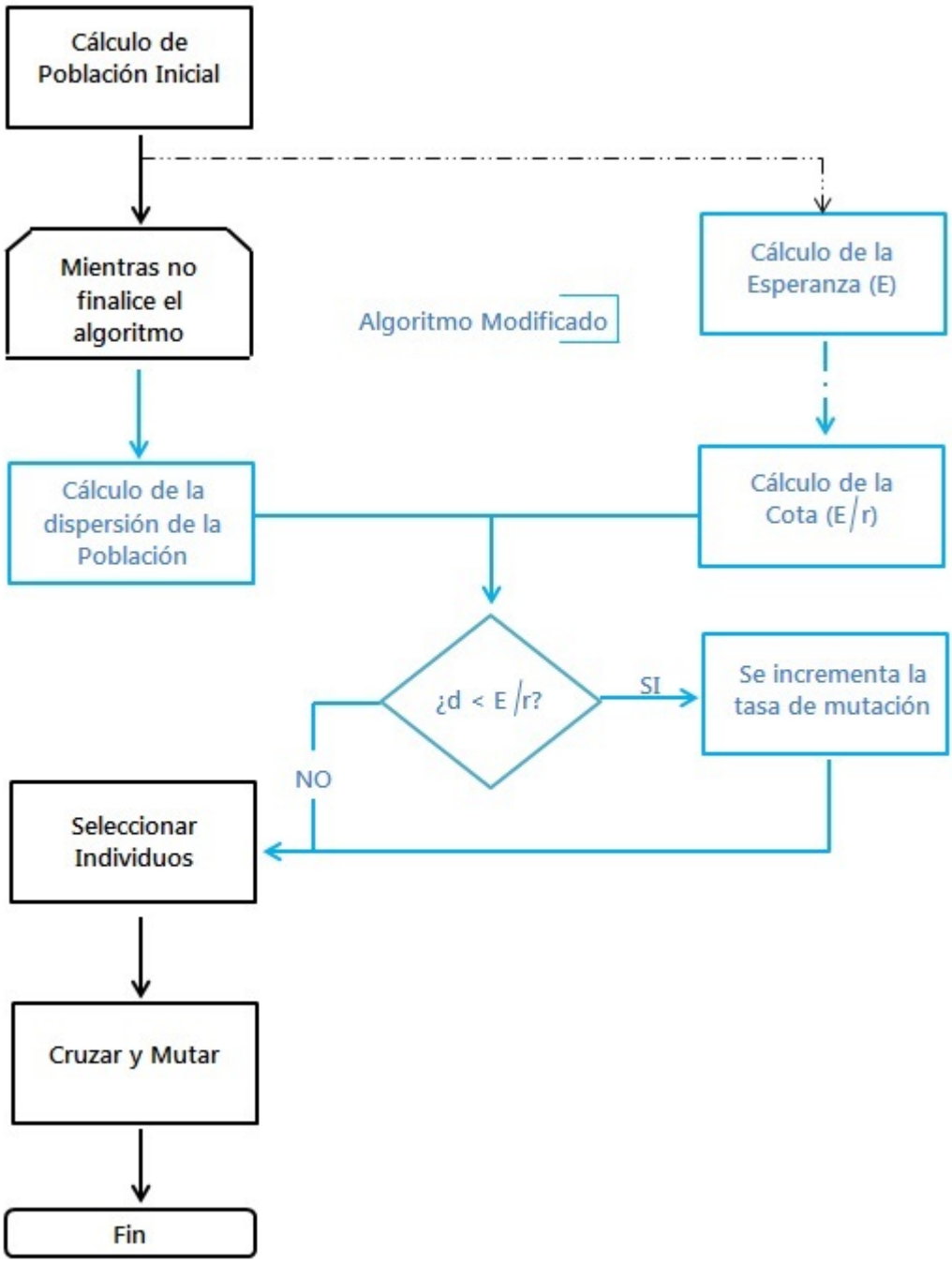

Figura 5.2: Algoritmo Genético con Control de Diversidad

Cabe señalar que estas recomendaciones se basan en que implementamos un algoritmo elitista en virtud del cual los aumentos de dispersión no implican pérdida del mejor individuo.

5.3. Implementación de un algoritmo genético con control de la diversidad

Agregamos a la implementación del algoritmo genético descripta en el capítulo 3 los siguientes parámetros:

1. Permitir utilizar $disp_1$ o $disp_2$.
2. Tasa de mutación inicial (antes Tasa de mutación).
3. Incremento de la tasa de mutación.
4. Divisor para determinar la cota (r).

Ejemplo:

Dispersión: 1
Tasa de mutación inicial: 0,01
Incremento de la tasa mutación: 0,02
r: 4

El algoritmo comenzará con una tasa de mutación de 0,01. Después de cada generación se calcula el número de dispersión y si es menor a la cota se incrementará en 0,02 la tasa de mutación.
Se calcula el valor de la dispersión por el método $disp_1$.
La cota se calcula al inicio del algoritmo. El valor se obtiene al calcular la Esperanza de la dispersión de una población binaria generada al azar divido por 4, el resultado obtenido será la Cota que controlará la dispersión.

5.4. Aplicaciones y resultados

5.4.1. Problema de la Mochila

Este capítulo describe las pruebas que realizamos para evaluar la efectividad de la solución propuesta en el capítulo 4. Para el Problema de la Mochila realizamos las pruebas con cinco lotes diferentes de pares de $\{Valores,Pesos\}$, los cuatro primeros fueron generados al azar el quinto fue generado de manera determinística de modo que se estancara en un óptimo local para ver la eficiencia del algoritmo.
En el Apéndice C se encuentra los pesos y valores utilizados, además de los resultados obtenidos.
Para decidir el incremento de la mutación utilizamos seis r's diferentes de modo que la Cota (C=E/r) sea: 1, 1,5, 2, 4, 6 y 8.
Los parámetros utilizados para todos los casos excepto el caso determinístico correspondiente al resultado óptimo 455, son:

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación	Longitud del Individuo	Cantidad de Individuos	Generaciones
0,01	0,001	50	100	1000

Para el caso cuyo resultado óptimo es 455 cambia el número de generaciones, es necesario aumentarlo para llegar al óptimo.

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación	Longitud del Individuo	Cantidad de Individuos	Generaciones
0,01	0,001	50	100	**2000**

A continuación mostramos a modo de resumen medidas estadísticas (media, mediana, desvío standard, etc) para el número de generaciones hasta alcanzar el óptimo (si el óptimo no se alcanzó toma el valor del parámetro $Generaciones$, es decir 1000 o 2000 según corresponda), con el fin de facilitar la comprensión y la comparación de las diversas características de los lotes de pruebas realizados sobre el problema de la mochila.

Medidas resumen del número de generaciones.

Cota: 1 - Resultado óptimo: 2577.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	392,82	572.34	25,00	1000,00	176,50
Disp 2	50	164,48	132.33	50,00	782,00	121,50

Cota: 1 - Resultado óptimo: 4005.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
DISP 1	50	392,92	323,18	11,00	1000,00	279,50
DISP 2	50	112,86	66,36	30,00	331,00	85,00

Cota: 1 - Resultado óptimo: 3536.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	377,96	278,10	52,00	1000,00	325,50
Disp 2	50	145,46	113,11	32,00	559,00	105,50

Cota: 1 - Resultado óptimo: 6454.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	336,46	285,94	62,00	1000,00	218,50
Disp 2	50	207,88	160,06	53,00	935,00	168,00

Cota: 1 - Resultado óptimo: 455.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	216,54	92,22	108,00	518,00	181,00
Disp 2	50	360,96	400,07	104,00	2000,00	222,50

Cota: 1.5 - Resultado óptimo: 2577.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	231,34	242,43	59,00	1000,00	144,00
Disp 2	50	188,34	235,51	39,00	1000,00	122,00

Cota: 1.5 - Resultado óptimo: 4005.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	184,82	139,64	36,00	674,00	130,50
Disp 2	50	128,36	80,21	36,00	380,00	106,00

Cota: 1.5 - Resultado óptimo: 3536.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	146,78	87,94	52,00	446,00	112,50
Disp 2	50	103,00	69,64	30,00	437,00	86,00

Cota: 1.5 - Resultado óptimo: 6454.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	252,40	210,30	52,00	1000,00	203,00
Disp 2	50	217,92	223,29	53,00	1000,00	127,00

Cota: 1.5 - Resultado óptimo: 455.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	300,86	192,41	96,00	893,00	248,50
Disp 2	50	337,52	261,02	108,00	1188,00	228,00

Cota: 2 - Resultado óptimo: 2577.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	163,54	103,82	31,00	685,00	151,00
Disp 2	50	152,74	88,18	45,00	453,00	130,00

Cota: 2 - Resultado óptimo: 4005.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	144,06	85,52	49,00	463,00	116,00
Disp 2	50	116,12	59,89	30,00	318,00	112,00

Cota: 2 - Resultado óptimo: 3536.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	136,12	78,18	32,00	412,00	115,50
Disp 2	50	110,76	76,83	41,00	487,00	94,50

Cota: 2 - Resultado óptimo: 6454.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	237,96	185,36	63,00	1000,00	169,00
Disp 2	50	255,30	237,16	44,00	1000,00	160,50

Cota: 2 - Resultado óptimo: 455.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	441,02	448,42	86,00	2000,00	229,00
Disp 2	50	442,88	442,30	15,00	2000,00	307,00

Cota: 4 - Resultado óptimo: 2577.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	271,66	246,12	51,00	1000,00	202,50
Disp 2	50	227,18	201,75	32,00	1000,00	157,50

Cota: 4 - Resultado óptimo: 4005.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	197,94	158,35	39,00	911,00	138,00
Disp 2	50	220,44	205,31	33,00	1000,00	163,50

Cota: 4 - Resultado óptimo: 3536.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	219,12	222,09	26,00	912,00	126,50
Disp 2	50	203,54	198,57	35,00	1000,00	143,00

Cota: 4 - Resultado óptimo: 6454.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	251,70	250,96	18,00	957,00	132,00
Disp 2	50	351,36	314,37	53,00	1000,00	217,00

Cota: 4 - Resultado óptimo: 455.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	896,84	595,47	17,00	2000,00	795,50
Disp 2	50	910,66	718,45	115,00	2000,00	651,50

Cota: 6 - Resultado óptimo: 2577.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máxim	Mediana
Disp 1	50	526,20	339,44	53,00	1000,00	518,00
Disp 2	50	422,20	330,08	16,00	1000,00	299,50

Cota: 6 - Resultado óptimo: 4005.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	590,66	370,19	31,00	1000,00	593,00
Disp 2	50	606,28	383,71	34,00	1000,00	659,00

Cota: 6 - Resultado óptimo: 3536.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	455,34	374,91	26,00	1000,00	357,00
Disp 2	50	538,28	398,68	30,00	1000,00	457,50

Cota: 6 - Resultado óptimo: 6454.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	603,40	370,11	69,00	1000,00	541,50
Disp 2	50	687,90	367,73	73,00	1000,00	887,00

Cota: 6 - Resultado óptimo: 455.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	1774,30	481,86	389,00	2000,00	2000,00
Disp 2	50	1690,52	503,26	143,00	2000,00	2000,00

Cota: 8 - Resultado óptimo: 2577.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	831,04	256,79	145,00	1000,00	1000,00
Disp 2	50	664,08	351,88	28,00	1000,00	775,50

Cota: 8 - Resultado óptimo: 4005.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	886,84	284,53	40,00	1000,00	1000,00
Disp 2	50	842,20	307,07	31,00	1000,00	1000,00

Cota: 8 - Resultado óptimo: 3536.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	826,06	341,10	32,00	1000,00	1000,00
Disp 2	50	789,88	353,88	29,00	1000,00	1000,00

Cota: 8 - Resultado óptimo: 6454.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	805,96	304,69	60,00	1000,00	1000,00
Disp 2	50	858,62	278,63	29,00	1000,00	1000,00

Cota: 8 - Resultado óptimo: 455.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	1946,38	211,44	704,00	2000,00	2000,00
Disp 2	50	1952,90	192,52	732,00	2000,00	2000,00

Podemos observar que con Cota 8 el valor obtenido para la mediana prácticamente coincide en todos los casos con la cantidad de generaciones que se utilizó como parámetro, esto significa que el algoritmo finalizó sin encontrar el resultado esperado.
Por otro lado, podemos observar que el algoritmo siempre arroja mejores resultados con dispersión 2 para las cotas 1,5 y 2, además, con estas cotas el algoritmo llega antes al resultado esperado.
Los resultados arrojados para la media se pueden visualizar en las **Figuras 5.3, 5.4 y 5.5**.

En la **Figura 5.5** se observa que al tener muchos óptimos locales, ambos métodos para medir la dispersión, se comportan de manera similar y llegan al óptimo global, mientras que el método que no tiene control de diversidad se estanca y no logra llegar al óptimo global en ninguna de las cincuenta corridas (lado derecho de la figura, línea verde).

5.4.2. Problema de clasificación de imágenes

En la actualidad un problema con gran variedad de aplicaciones es el reconocimiento de imágenes. En esta tesis vamos a encarar un problema de clasificación de imágenes utilizando el algoritmo genético con control de diversidad. Para evaluar la diversidad utilizamos los métodos Dispersión 1 y Dispersión 2. Cabe destacar que existen métodos más específicos para resolver problemas de reconocimiento de imágenes, por ejemplo la clasificación de imágenes para reconocer patrones gráficos. Pero en este caso, nuestro interés es mostrar que el algoritmo genético binario con control de diversidad es robusto y de fácil adaptación para problemáticas diversas. En este problema queremos discenir entre tres tipos de imágenes: las letras P, L y X. Además, tenemos diez representaciones

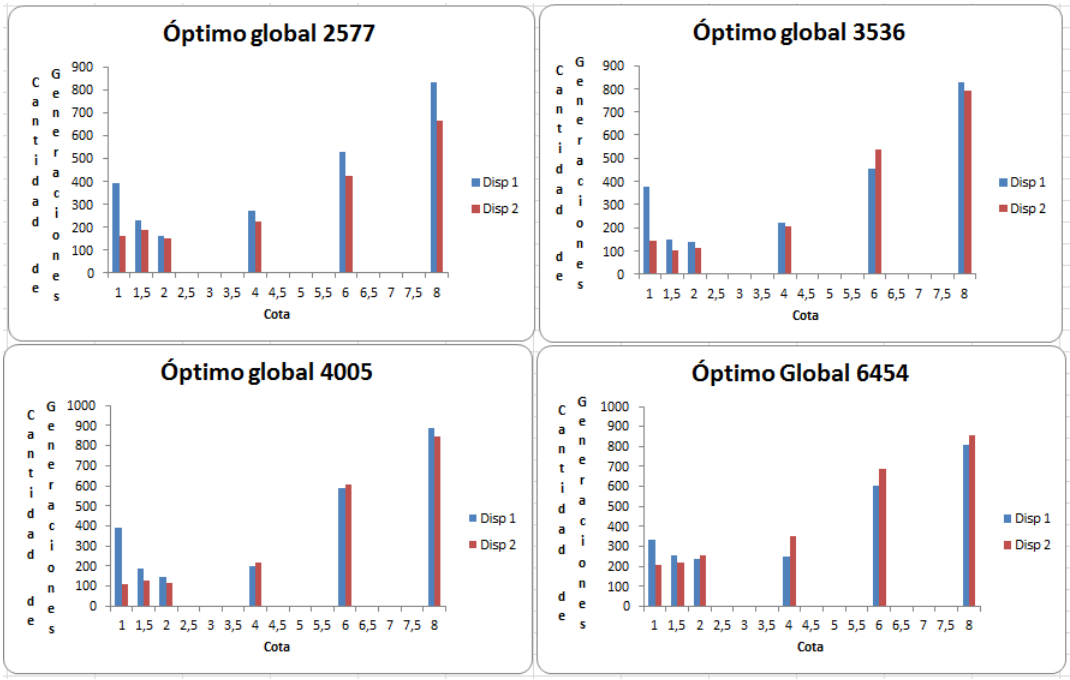

Figura 5.3: Cantidad de generaciones para alcanzar el óptimo con datos al azar. Gráfico de Barras

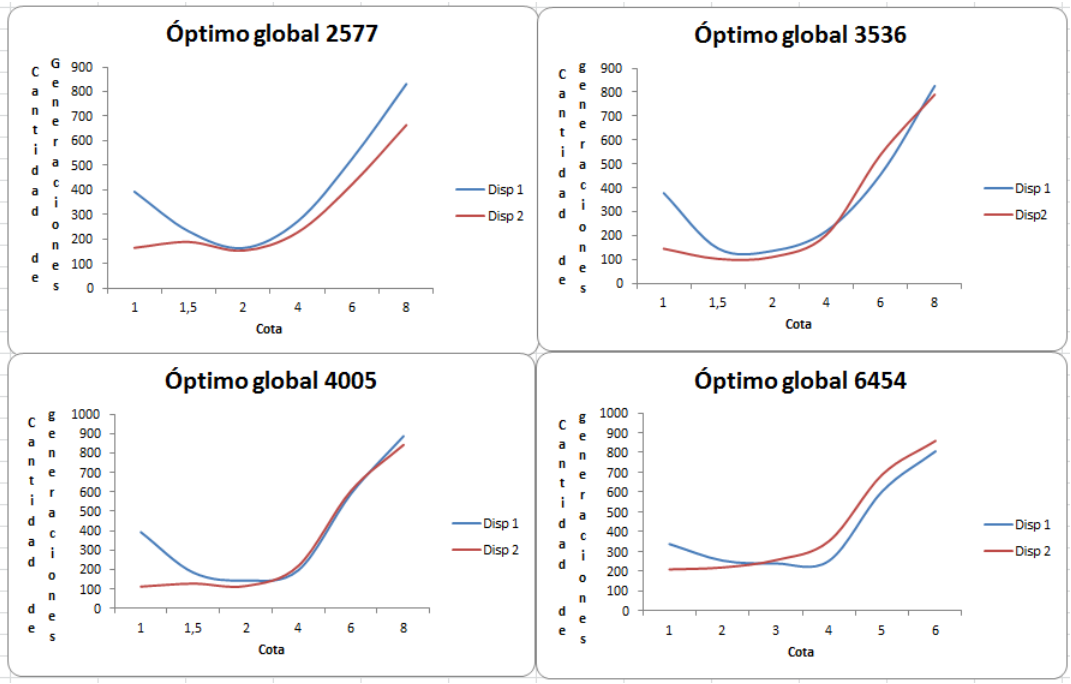

Figura 5.4: Cantidad de generaciones para alcanzar el óptimo con datos al azar. Gráfico de curvas interpolantes

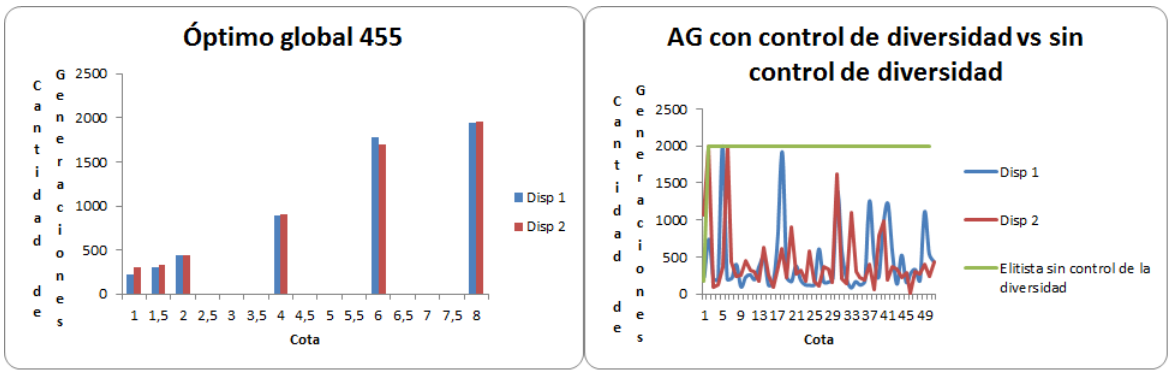

Figura 5.5: Cantidad de generaciones para alcanzar el óptimo con presencia de muchos óptimos locales. Datos determinísticos

distintas de cada imagen que servirán para entrenamiento. En nuestro método permitirán obtener un cromosoma con información apropiada para reconocer si una nueva imagen es una P, una L o una X. Las muestras están codificadas en forma binaria. En la **Figura 5.6** se muestran tres imágenes discretizadas y su codificación con 64 bits. Las imágenes pueden provenir originalmente de una escritura a mano o de la fotografía de un cartel, por ejemplo.

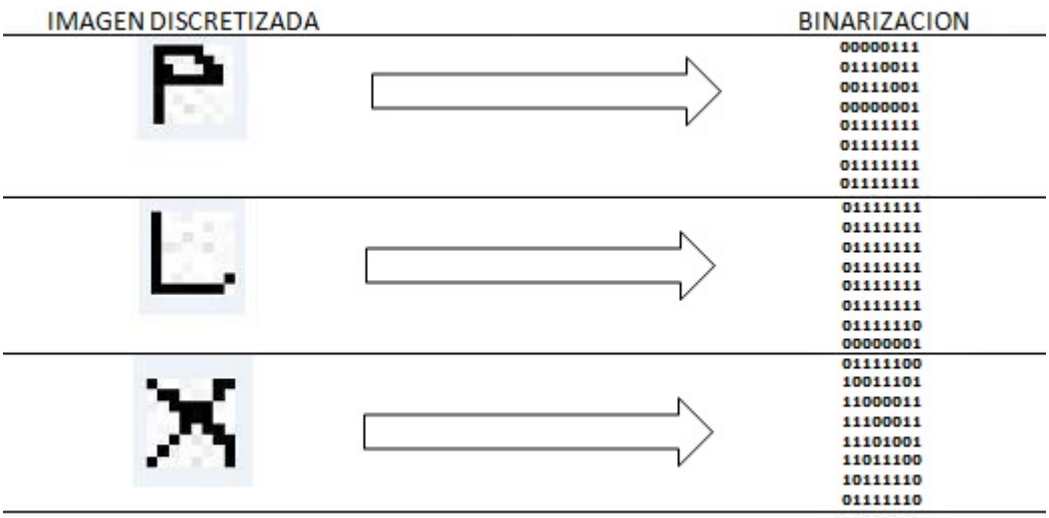

Figura 5.6: AG - Binarización de Imágenes

A continuación explicaremos de qué manera las binarizaciones nos permitirán definir una función de fitness que será mayor para cromosomas que disciernen mejor entre P, L y X. El cromosoma cuenta con tres porciones de 64 bits cada uno, la primera representa la P, la segunda la L y la tercera la X. Además se cuenta con un archivo ("$Muestra.txt$") que tiene 30 muestras que contienen la binarización de diez L, diez P y diez X que se utilizarán para el aprendizaje para el método de clasificación. A modo de ilustración, la **Figura 5.7** contiene las filas correspondientes a las imágenes de la **Figura 5.6**.

Figura 5.7: Ilustración de Muestra.txt

Sin embargo, en la población inicial todos los bits toman valores 0 o 1 al azar. El propósito del algoritmo es que la evolución produzca cromosomas con la primer porción cercana a las letras P de entrenamiento, la segunda cercana a las L y la tercera cercana a las X. Cada una de las 30 filas del archivo "$Muestra.txt$" contiene la digitalización de una letra (10 P, 10 L y 10 X). Denotamos cada fila con l_i ($1 \leq i \leq 30$). El fitness de un cromosoma se obtiene de la manera siguiente. Para cada l_i se calcula la distancia de Hamming contra la porción del cromosoma correspondiente a su letra (P, L, X): Denotamos p_1 a la porción que se adaptará a la imagen P, p_2 a la que se compara con la imagen L y p_3 a la que se adaptará a la imagen X. Luego si h es la distancia de Hamming, definimos $d_{i,P}$, $d_{i,L}$ y $d_{i,X}$ mediante:
$d_{i,P}$=h(p_1,l_i)
$d_{i,L}$=h(p_2,l_i)
$d_{i,X}$=h(p_3,l_i)
que toman valores entre 0 y 64.
Los aportes al fitness se calculan mediante el valor f_i definido por
f_i=64-$d_{i,k}$ + b_i
donde k=1, 2 o 3 según si la fila i corresponde a una letra P, L o X respectivamente y

$$b_i=\begin{cases} 10 & si\ min(\mathsf{d}_{i,1},\ d_{i,2},\ d_{i,3})= d_{i,k} \\ 0 & \text{caso contrario} \end{cases}$$

De manera que 64-$d_{i,k}$ es grande si la porción del cromosoma asignado a la letra que corresponde a la fila i es cercano a l_i. Mientras que b_i es un adicional de 10 puntos de fitness que se agrega si la más pequeña de las distancias de l_i a las tres porciones del cromosoma se alcanza en la porción que corresponde a la letra de l_i. Es decir, se premia con 10 puntos si se produce dicho acierto. La cantidad de 10 podría eventualmente cambiarse.
Finalmente el fitness es:

$$Fitness = \sum_{i=1}^{30} f_i,$$

Resultados obtenidos.

Para interpretar los resultados obtenidos vamos a mostrar cómo realizamos el cálculo determinístico de una cota superior para el fitness máximo para esta muestra en particular. Obviamente, si se alcanza es que el algoritmo genético no se estancó.
Como ya hemos dicho, el archivo "$Muestra.txt$" contiene 30 filas. En primer lugar las separamos en 3 grupos de 10 por cada letra y a cada grupo le calculamos una fp_k(k=1, 2, 3 según la letra P, L, X)=fila promedio de la siguiente forma.
Armamos una matriz de cada grupo y generamos cada componente j de fp_k

$$fp_k(j)=\begin{cases}1 & si \text{ la suma de la componente j} \geq 5\\ 0 & \text{caso contrario}\end{cases}$$

Luego unimos las tres fp_k, le calculamos el fitness mediante $\sum_{i=1}^{30} 64 - d_{ik}$ y sumamos 300 asumiendo que se da b_i=10 para todas las filas. Nótese que la fila promedio se forma como la mayor ocurrencia para cada columna, es decir cada elemento de la fila es el valor que más apareció en cada una de las filas para esa columna. Por este motivo maximiza $\sum_{i=1}^{30} 64 - d_{ik}$ sobre todos los cromozomas posibles. Sumar b_i=10 para cada i nos da una cota superior para $\sum_{i=1}^{30} f_i$ sobre todos los cromosomas posibles pero no tiene porque alcanzarse. El resultado para $Muestra.txt$ es 1946. Nótese que si bien hemos definido que el valor de cada bit de fp es uno cuando la suma de la columna es ≥ 5, podríamos haber definido que el valor es 1 cuando la suma de la columna es > 5 y esto da lugar a más de una solución posible con el mismo fitness y se debe a que hay un número par de filas para cada letra, lo que hace, y en la práctica sucedió en nuestro algoritmo genético, que hubo un conjunto de soluciones posibles con el mismo fitness.
Los parámetros utilizados para todos los casos son:

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación	Longitud del Individuo	Cantidad de Individuos	Generaciones
0,000002	0,000001	192	384	500

Los parámetros «Tasa Inicial de Mutación» e «Incremento de la Tasa de Mutación» son el resultado de muchas pruebas, hasta ajustar y conseguir los resultados satisfactorios.
A continuación mostramos a modo de resumen medidas estadísticas (media, mediana, desvío standard, etc) con respecto al número de generaciones hasta alcanzar el óptimo, con el fin de facilitar la comprensión y la comparación de las diversas características de los lotes de pruebas realizados sobre el problema de la clasificación de imágenes.

Cota: 2 - Resultado óptimo ≤ 1946.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	167,52	35,02	88	270	164
Disp 2	50	147,58	24,71	95	199	144

Cota: 4 - Resultado óptimo ≤ 1946.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	163,69	26,24	115	224	157
Disp 2	50	155,24	28,17	91	203	155,56

Cota: 8 - Resultado óptimo ≤ 1946.

Medidas resumen del número de generaciones

Dispersión	Cantidad de corridas	Media	Desviación Estándar	Mínino	Máximo	Mediana
Disp 1	50	155,57	24,73	110	209	154
Disp 2	50	148,04	29,15	83	206	145

En los resultados alcanzados en todos los casos se puede observar que controlar la diversidad con el método Disp 2 es mejor que la Disp 1, si interpretamos como mejor que alcance antes el óptimo. En la **Figura 5.8** se visualizan las medias de las medidas resumen.

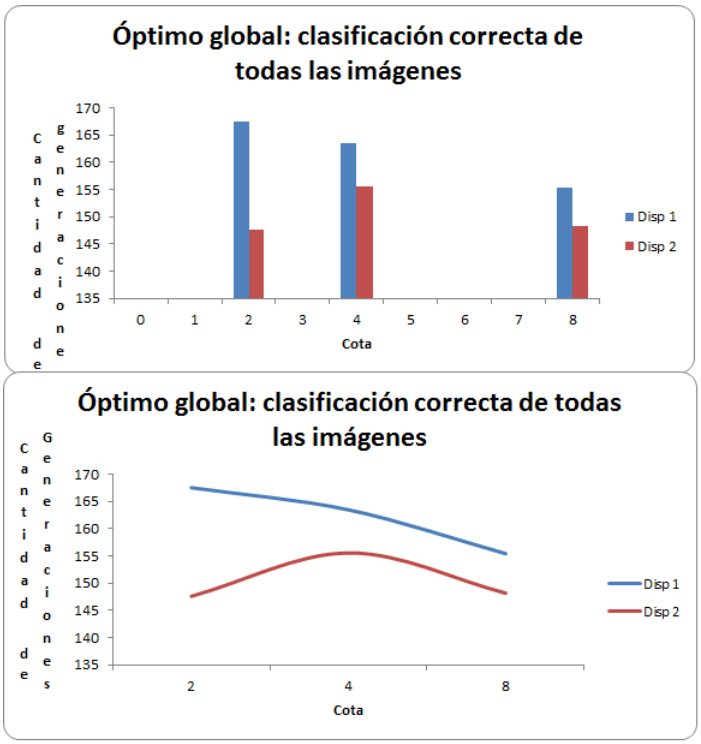

Figura 5.8: Cantidad de generaciones para alcanzar el óptimo

Capítulo 6

Convergencia del algoritmo genético

6.1. Concepto de convergencia

En este capítulo probamos que nuestro algoritmo genético con control de diversidad converge, en un sentido estocástico que luego precisaremos, a un óptimo de la función objetivo. Debemos notar que no incluimos un resultado sobre velocidad o en particular un resultado que muestre una eficiencia superior, para cualquier función objetivo, de nuestro algoritmo con respecto a otro algoritmo determinístico o estocástico. Es natural que así sea porque cualquier comparación de eficiencia de algoritmos, requiere de supuestos sobre la función a optimizar. En efecto, sin supuesto alguno sobre la función a optimizar, podríamos estar en presencia una función de las llamadas *needle in a haystack* (aguja en un pajar), como por ejemplo:

$$f(x) = \begin{cases} 1 & si \quad x = x_0 \\ 0 & si \quad x \neq x_0 \end{cases}$$

para un x_0 desconocido del espacio de búsqueda S. Con una función así, ningún algoritmo puede comportarse mejor que una búsqueda exhaustiva, la cual, como explicamos en el Anexo B, encuentra el óptimo con velocidad exponencial en el número de variables del espacio S. De manera que sin suposiciones sobre f sólo se puede aspirar a demostrar que un algoritmo converge al óptimo, sin poder agregar propiedades ventajosas de convergencia. Esto no quita que para ciertos problemas o familias de funciones se pueda mostrar la eficiencia de un algoritmo genético mediante experimentos computacionales, como lo hicimos anteriormente, o con algún desarrollo teórico específico para la familia de funciones, pero éste no es el caso de este capítulo.

Debemos en primer lugar precisar la idea de convergencia de un algoritmo estocástico, o más exactamente genético. Se dice que algoritmo genético converge [RUD/94] si a lo largo de las generaciones, la probabilidad de que se alcance el óptimo se acerca a 1.
Más precisamente

$$\lim_{h\to\infty} P(Z_h = f^*) = 1 \text{ donde}$$

$Z_h =$ Mejor fitness de un individuo de la generación h $(h = 0, 1, ...)$.
$f^* =$ Máximo valor que puede alcanzar f, es decir $f^* \geq f(x)\ \forall x \in S$ donde $S = \{0, 1\}^l$.

Demostraremos más adelante que esta convergencia se verifica en nuestro algoritmo genético básico elitista, resultado que también se obtiene en [RUD/94] pero nosotros no usaremos argumentos de cadenas de Markov como en dicho artículo. Luego adaptaremos nuestra demostración al algoritmo genético con control de diversidad propuesto en esta tesis. La demostración se basa en que el paso de mutación obtiene con probabilidad 1 el óptimo de la función en un tiempo finito y en que el elitismo asegura la permanencia de este óptimo. No se usa el cruce, en efecto, demostramos que con o sin paso de cruce, la convergencia necesariamente ocurre. Es decir que prescindimos del paso que se estima importante para acelerar la convergencia con ciertas funciones f. Pero recordamos nuevamente que aquí no hacemos supuestos sobre f. Estudiar las propiedades de convergencia restringiéndose a ciertas familias de funciones objetivo sería un interesante trabajo futuro.

Recurriremos a un sencillo lema para sucesiones de eventos independientes. Con esta herramienta y un análisis del evento aleatorio que se produce en el paso de mutación, probaremos que el algoritmo genético elitista converge a una solución. Finalmente adaptamos esta demostración al caso en que la tasa de mutación va a aumentando con el fin de evitar la pérdida de diversidad.

6.2. Un lema de probabilidad

El siguiente lema muestra que si hay una cota inferior positiva para la probabilidad de todo elemento de una sucesión de eventos independientes, entonces con probabilidad 1 ocurre necesariamente alguno de ellos.

Lema: Sea $B_1, B_2, B_3, ...$ una sucesión de eventos independientes. Si para algún $p > 0$ ocurre que para todo t se tiene $P(B_t) \geq p$ entonces $P(B_t$ ocurre para algún $t < \infty) = 1$.

Interpretación: Antes de la demostración, queremos poner de manifiesto que se trata de un resultado de probabilidad elemental, en virtud del cual ocurre, por ejemplo, que si arrojamos repetidas veces una moneda, obtendremos una cara en algún momento, aún si la moneda está desequilibrada y la probabilidad de que salga cara es muy pequeña. En este caso B_t es "La moneda sale cara en el paso t" mientras que podemos tomar como p a la probabilidad de obtener una cara. En un algoritmo genético, B_t será "En el t-ésimo paso de mutación se obtiene un individuo en el que la función alcanza su valor máximo" mientras que la cota $p > 0$ resultará del análisis que haremos del paso de mutación.

Demostración del lema: Basta ver que la probabilidad del complemento de "B_t ocurre para algún $t < \infty$" es igual a cero, es decir $P($Para ningún t ocurre $B_t) = 0$. Ahora bien, "Para ningún t ocurre B_t" es el mismo evento que "Para todo t ocurre B_t^c", el cual se puede expresar como $\bigcap_{t=0}^{\infty} B_t^c$. Notemos ahora que si intersecamos una cantidad menor de conjuntos obtenemos un conjunto que incluye el anterior, es decir que para cualquier k tenemos

$$\bigcap_{t=0}^{\infty} B_t^c \subset \bigcap_{t=0}^{k} B_t^c.$$

Luego las respectivas probabilidades se relacionan mediante

$P(\bigcap_{t=0}^{\infty} B_t^c) \leq P(\bigcap_{t=0}^{k} B_t^c)$

Ahora por independencia

$P(\bigcap_{t=0}^{k} B_t^c) = P(B_1^c).P(B_2^c)...P(B_k^c)$

Aplicando estos dos últimos resultados se tiene

P(Para ningún t ocurre $B_t) = P(\bigcap_{t=0}^{\infty} B_t^c) \leq P(B_1^c).P(B_2^c)...P(B_k^c)$. Como para todo t se cumple que $P(B_t) \geq p$, tenemos que $P(B_t^c) \leq 1-p$ y por lo tanto el producto anterior es menor o igual a $(1-p)^k$, donde $(1-p)^k \longrightarrow 0$ cuando $k \to \infty$. Luego

P(Para ningún t ocurre $B_t) \leq 0$. Como además ninguna probabilidad puede ser negativa, resulta que

P(Para ningún t ocurre B_t) nos queda acotada inferior y superiormente por cero, debiendo ser igual a cero, como queríamos.

6.3. Convergencia del algoritmo genético elitista

En un algoritmo genético binario, cualquiera sea la estrategia de reemplazo de los individuos por los nuevos que el proceso va generando, se tiene una sucesión de pasos de mutación que usualmente, y en particular en nuestro caso, consiste en tomar un individuo, recorrerlo bit a bit y cambiar el valor de cada bit independientemente con un cierta probabilidad p_m (con $0 < p_m < 1/2$). Esto genera un nuevo individuo quizá igual, quizá distinto en algunos bits o quizá totalmente distinto al que fue sometido a mutación. Llamamos *resultado de la mutación en el paso* t al individuo producido por la t-ésima mutación de un individuo que lleva a cabo la ejecución del algoritmo.

En esta sección probamos que si se ejecuta una sucesión de mutaciones, entonces con probabilidad 1 ocurre que en algún paso de mutación t el resultado es un individuo en el que se alcanza el óptimo de f. Debido al elitismo, este individuo (o alguno con igual valor de f) permanecerá para siempre en la población, lo cual implica que se cumple la condición de convergencia a un óptimo. Esto no excluye que en cierta ejecución, el óptimo pueda aparecer como resultado de un cruce (como a menudo es lo esperado en un algoritmo genético), o incluso en la población inicial.

Llamaremos b a un individuo perteneciente a $\{0,1\}^l$ en el que se alcanza el máximo, es decir, $f(b) = f^*$. Se trata de una l-upla de bits $(b_1, ..., b_l)$. Supondremos que la l-upla en la que se alcanza el máximo es única. Las demostraciones se podrían adaptar al caso de más de un máximo global y de hecho es bastante claro que si demostramos que si un único máximo global se alcanza con probabilidad 1, también será 1 la probabilidad de alcanzar un máximo global si hay más de uno. Para $t = 1, 2, ...$ sea A_t el evento "El individuo b aparece en el paso de mutación t". Queremos ver que A_t ocurre para algún t, pero no podemos utilizar directamente el lema porque los eventos A_t no son necesariamente independientes. Esto último se puede justificar con el siguiente argumento. Saber que no ocurrió A_t desde $t = 1$ hasta un cierto k reduce la probabilidad de A_{k+1} con respecto a la probabilidad no condicionada a esa información. O equivalentemente, saber que ocurrió A_t para algún t aumenta la probabilidad de que el individuo b vuelva a aparecer porque su alto fitness hará que su patrimonio genético tienda a difundirse por la población.

Vamos a definir una sucesión de eventos B_t independientes tales que $B_t \subset A_t$. Para ello realizamos una construcción alternativa, pero equivalente, del paso de mutación. El paso de mutación original de un individuo consiste en cambiar b_i, el bit i-ésimo de un individuo b (con i entre 1 y l), con probabilidad p_m y dejarlo igual con probabilidad $1 - p_m$. Este es el procedimiento que aplica el algoritmo. La construcción alternativa es equivalente como luego veremos, por lo cual si se aplicara generaría el mismo proceso estocástico, pero su consideración tiene el valor teórico de permitirnos una demostración de convergencia.

Definición de las variables X_i e Y_i y demostración de convergencia

La construcción alternativa consiste en definir variables aleatorias independientes X_i e Y_i (para i entre 1 y l) de la siguiente manera. La variable X_i vale 1 con probabilidad $2p_m$ y vale 0 con probabilidad $1 - 2p_m$, mientras que para Y_i estas probabilidades son ambas $1/2$. El procedimiento de mutación consite en que, si $X_i = 0$ entonces b_i retiene su valor, mientras que si $X_i = 1$ entonces b_i toma el valor de Y_i. Es decir, si $X_i = 0$ entonces b_i no cambia, mientras que si $X_i = 1$ entonces b_i cambia con probabilidad $1/2$. Veamos la equivalencia con la construcción original. En la nueva construcción, ya sea que $b_i = 0$ o $b_i = 1$, la probabilidad de que b_i cambie es

$$P(X_i = 1, Y_i \neq b_i) = P(X_i = 1)P(Y_i \neq b_i) = 2p_m \cdot 1/2 = p_m.$$

Esto muestra la igualdad de probabilidades **(Figura 6.1)**. Además se verifica la independencia para decidir el cambio de cada bit debido a la independencia de las variables aleatorias X_i e Y_i (para i entre 1 y l) que definimos para decidir el valor de los bits b_i.

Para la t-ésima mutación que se produce durante la ejecución del algoritmo, denominamos a estas variables aleatorias X_i^t e Y_i^t con $i = 1, ..., l$ y $t = 1, 2, ...$, donde todas estas variables aleatorias son independientes.

Ahora, para el paso t de mutación definimos el evento B_t mediante

$$B_t = \left\{(X_1^t, X_2^t, .., X_l^t) = (1, ..., 1), (Y_1^t, Y_2^t, .., Y_l^t) = (b_1, b_2, .., b_l)\right\}.$$

Es decir que B_t es un evento que, de ocurrir, implica que el resultado del t-ésimo paso de mutación es el individuo $b = (b_1, b_2, .., b_l)$. No es el único evento que puede dar este resultado, pero dado que B_t produce b tenemos $B_t \subset A_t$. La independencia de los B_t se debe a que estos eventos están definidos por variables aleatorias independientes entre sí.

Dado que $B_t \subset A_t$, tenemos que si ocurre B_t entonces ocurre A_t, es decir que basta probar que con probabilidad 1 ocurre B_t para algún t para que podamos afirmar lo mismo para A_t. La afirmación para B_t la obtenemos del lema observando que podemos tomar como cota p la probabilidad común a todos los B_t que es (usando la independencia)

$$\begin{aligned} P(B_t) &= P((X_1^t, X_2^t, .., X_l^t) = (1, ..., 1), (Y_1^t, Y_2^t, .., Y_l^t) = (b_1, b_2, .., b_l)) = \\ &= P(X_1^t = 1)P(X_2^t = 1)...P(X_l^t = 1)P(Y_1^t = b_1)P(Y_2^t = b_2)...P(Y_l^t = b_l) = \\ &= (2p_m)^l(1/2)^l = (p_m)^l \end{aligned}$$

que, si bien puede ser un valor muy pequeño, es mayor que cero. Luego por el lema y la inclusión $B_t \subset A_t$ tenemos

$1 = P(B_t$ ocurre para algún $t) \leq P(A_t$ ocurre para algún $t)$

Es decir que $P(A_t$ ocurre para algún $t) = 1$ como queríamos. Una generación puede incluir varios pasos de mutación, pero debido al elitismo, si se produce b en uno de ellos, el mejor individuo de toda población a partir de allí será optimo. Recordando la notación que introdujimos al definir convergencia de un algoritmo genético, podemos ahora afirmar que

$$P(\exists h : \forall m > h : Z_m = f^*) = 1.$$

Este evento que tiene probabilidad 1 se puede escribir también como

$$\cup_{h=1}^{\infty} \cap_{m=h}^{\infty} \{Z_m = f^*\}.$$

Luego usando la propiedad llamada continuidad de la probabilidad tenemos

$$1 = P(\cup_{h=1}^{\infty} \cap_{m=h}^{\infty} \{Z_m = f^*\}) = \lim_{h\to\infty} P(\cap_{m=h}^{\infty}\{Z_m = f^*\}).$$

Finalmente por inclusión de eventos

$$\lim_{h\to\infty} P(\cap_{m=h}^{\infty}\{Z_m = f^*\}) \leq \lim_{h\to\infty} P(Z_h = f^*)$$

Luego $1 \leq \lim_{h\to\infty} P(Z_h = f^*)$ lo cual implica que $\lim_{h\to\infty} P(Z_h = f^*) = 1$ como queríamos probar.

Cabe señalar que esta demostración no es posible sin elitismo. Un algoritmo no elitista puede tener propiedades deseables (como menor tendencia a estancarse en un óptimo local) y ser útil en la práctica, por ejemplo cuando se programa una cantidad fija de generaciones para ver a qué solución se llega. Pero a falta de elitismo, el óptimo puede alcanzarse y luego perderse, incluso esto puede ocurrir varias veces.

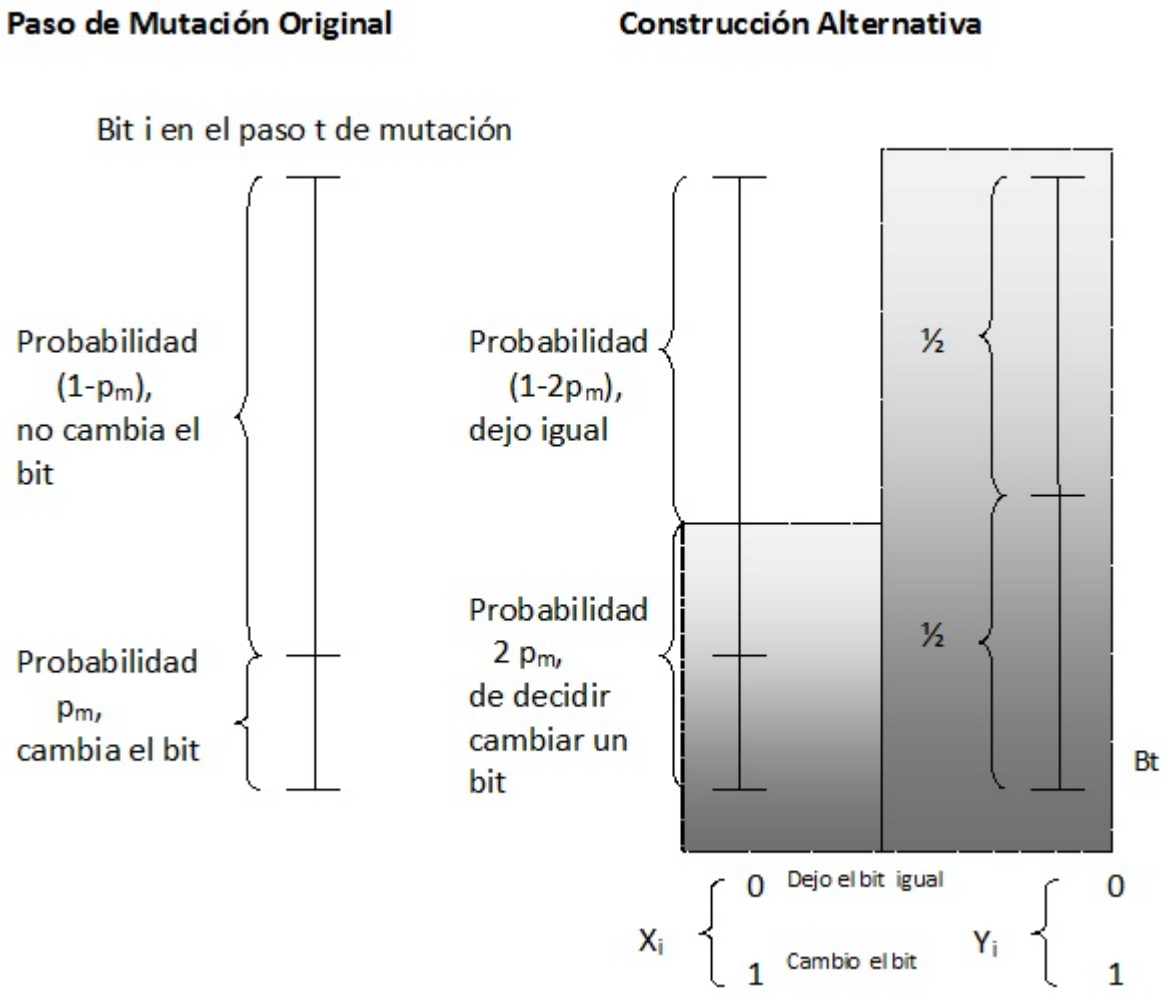

Figura 6.1: Construcción alternativa del paso de mutación

6.4. Convergencia del algoritmo genético elitista con control de diversidad

En este caso tenemos una probabilidad de mutación que va aumentando de acuerdo a nuestra política de control de la diversidad. La probabilidad de mutación de un bit es un valor p_m^0 al comienzo del algoritmo y en el paso t de mutación toma un valor que llamamos p_m^t. Este valor puede mantenerse para varios individuos y durante varias generaciones, según la decisión que el algoritmo toma en base a su medición de la diversidad poblacional. Pero en cualquier caso cumple con que $p_m^t \geq p_m^0$. Esto puede parecer suficiente para poder aplicar el lema también en este caso porque el mismo cálculo hecho anteriormente nos da

$$P(B_t) = (p_m^t)^l$$

donde ahora el supraíndice t nos indica que la probabilidad de cambiar un bit corresponde al paso de mutación t. Ahora en virtud de la acotación

$$P(B_t) = (p_m^t)^l \geq (p_m^0)^l > 0$$

donde $(p_m^0)^l$ sería la cota p que requiere el lema. El problema es que los B_t ya no son necesariamente independientes. En efecto, los eventos B_t posteriores a un paso de mutación t_1 no son independientes de las mutaciones que se produjeron antes de un paso t_1, las cuales influyen en la diversidad de la población, y en nuestro algoritmo esta diversidad determina $p_m^{t_1}$, que elevada al número l de bits que tiene cada cromosoma, es la probabilidad de B_t.

Definiremos entonces eventos B_t^* independientes cuya ocurrencia implique la aparición de un individuo óptimo que llamamos b. Recurriremos otra vez a una construcción del paso de mutación alternativa pero equivalente a la original (cambiar el valor del bit i-ésimo con probabilidad p_m^t). En esta construcción se pondrá de manifiesto que los eventos B_t^* dependen solamente de l ternas de variables independientes X_i^*, Y_i^* y W_i^*.

El paso de mutación de un individuo, en su definición original para el algoritmo con control de la diversidad, consiste en cambiar b_i, el bit i-ésimo de un individuo b (con i entre 1 y l), con probabilidad p_m^t y dejarlo igual con probabilidad $1 - p_m^t$. La construcción alternativa consiste en definir variables aleatorias independientes X_i^*, Y_i^* y W_i^* (para i entre 1 y l).

Definición de las variables X_i^*, Y_i^* y W_i^* y demostración de convergencia

La variable X_i^* vale 1 con probabilidad $2p_m^0$, vale 2 con probabilidad $2p_m^t - 2p_m^0$ y vale 0 con probabilidad $1-2p_m^t$. Las variables Y_i^* y W_i^* ambas valen 0 con probabilidad $1/2$ y 1 con probabilidad $1/2$. Si $X_i^* = 0$ entonces b_i retiene su valor, si $X_i^* = 1$ entonces b_i toma el valor de Y_i^* y si $X_i^* = 2$ entonces b_i toma el valor de W_i^*. Veamos la equivalencia con la construcción anterior. En la nueva construcción **Figura 6.2**, ya sea que $b_i = 0$ o $b_i = 1$, la probabilidad de que b_i cambie es

$$\begin{aligned} &P(X_i^* = 1, Y_i^* \neq b_i) + P(X_i^* = 2, W_i^* \neq b_i) \\ &\quad = P(X_i^* = 1)P(Y_i^* \neq b_i) + P(X_i^* = 2)P(W_i^* \neq b_i) \\ &\quad = 2p_m^0 \cdot 1/2 + (2p_m^t - 2p_m^0) \cdot 1/2 = p_m^t. \end{aligned}$$

Esto muestra la igualdad de probabilidades. Además se verifica la independencia para decidir el cambio de cada bit debido a la independencia de las variables aleatorias que definimos para decidir el valor de los bits b_i.

Para la t-ésima mutación de un individuo que se produce durante la ejecución del algoritmo, denominamos a estas variables aleatorias X_i^{*t}, Y_i^{*t} y W_i^{*t}con $i = 1, ..., l$ y $t = 1, 2, ...$, donde todas estas variables aleatorias son independientes.

Para el paso t de mutación definimos el evento B_t^* mediante

$$B_t^* = \left\{ (X_1^{*t}, X_2^{*t}, .., X_l^{*t}) = (1, ..., 1), (Y_1^{*t}, Y_2^{*t}, .., Y_l^{*t}) = (b_1, b_2, .., b_l) \right\}.$$

Es decir que B_t^* es un evento que, de ocurrir, implica que el resultado del t-ésimo paso de mutación es el individuo b. Dado que B_t^* produce b tenemos $B_t^* \subset A_t$ donde, recordemos, A_t es el evento "El individuo b aparece en el paso de mutación t" La independencia de los B_t^* se debe a que estos eventos están definidos por variables aleatorias independientes entre sí.

Análogamente al caso de los B_t de la sección anterior, aquí tenemos que $B_t^* \subset A_t$ y por lo tanto si ocurre B_t^* entonces ocurre A_t. Así que basta probar que con probabilidad 1 ocurre B_t^* para algún t para que podamos afirmar lo mismo para A_t. La afirmación para B_t^* la obtenemos del lema observando que podemos tomar como cota p la probabilidad común a todos los B_t^* que es (usando la independencia)

$$\begin{aligned} P(B_t^*) &= P((X_1^{*t}, X_2^{*t}, .., X_l^{*t}) = (1, ..., 1), (Y_1^{*t}, Y_2^{*t}, .., Y_l^{*t}) = (b_1, b_2, .., b_l)) = \\ &= P(X_1^{*t} = 1)P(X_2^{*t} = 1)...P(X_l^{*t} = 1)P(Y_1^{*t} = b_1)P(Y_2^{*t} = b_2)...P(Y_l^{*t} = b_l) = \\ &= (2p_m^0)^l(1/2)^l = (p_m^0)^l \end{aligned}$$

Este valor $(p_m^0)^l > 0$ es la cota fija buscada. Luego por el lema y la inclusión $B_t^* \subset A_t$ tenemos

$1 = P(B_t^*$ ocurre para algún $t) \leq P(A_t$ ocurre para algún $t)$

Debido al elitismo, el argumento para probar a partir de aquí la convergencia

$$\lim_{h \to \infty} P(Z_h = f^*) = 1$$

concluye como en la sección anterior.

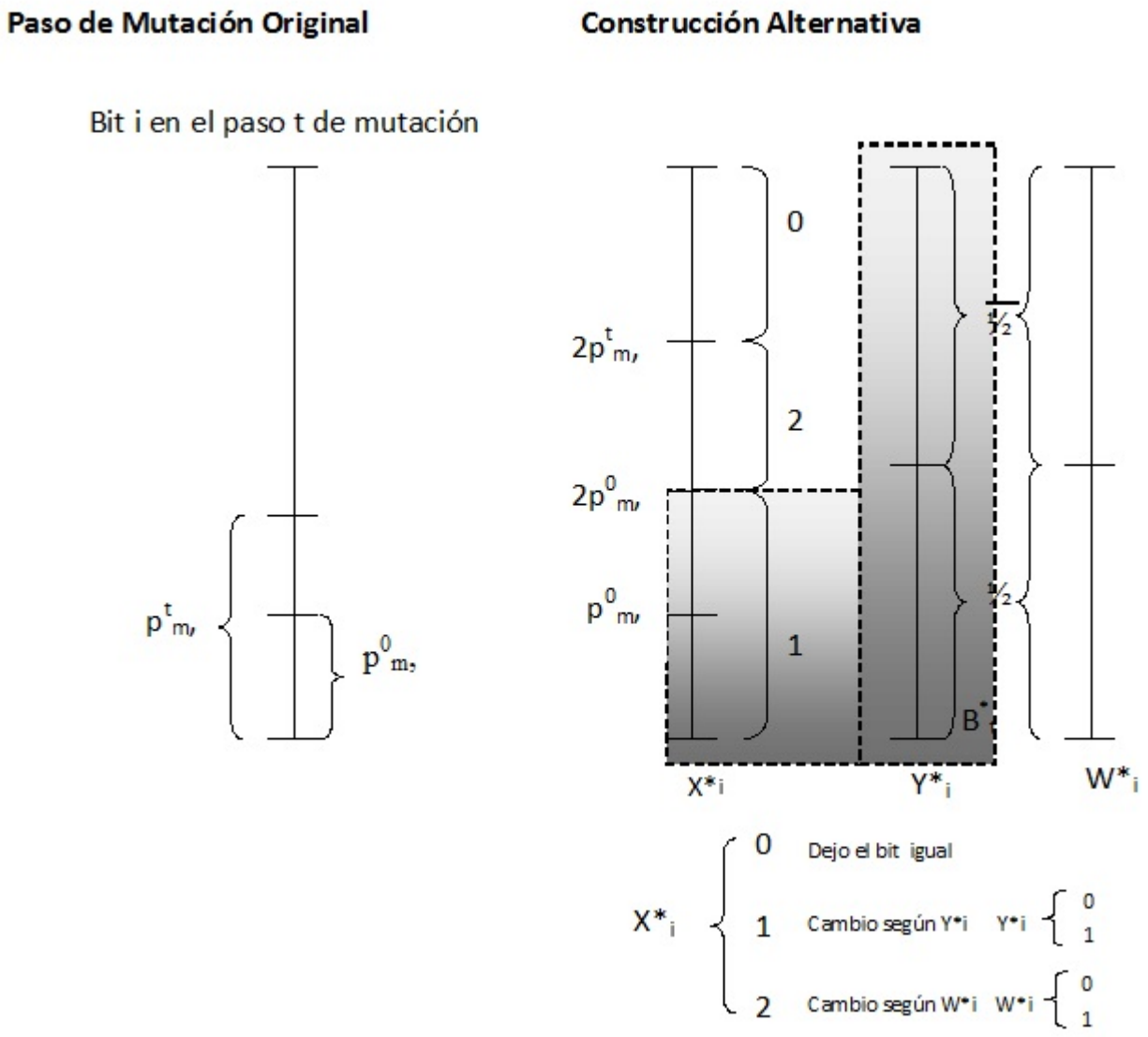

Figura 6.2: Construcción alternativa del paso de mutación para AG con control de diversidad

Capítulo 7

Conclusiones

Hemos programado un algoritmo genético canónico con codificación binaria, al cual le incorporamos el procedimiento de selección llamado elitismo. Luego de estudiar el problema del estancamiento en máximos locales agregamos a nuestro algoritmo el control automático de la diversidad poblacional para evitar que la población se vuelva homogénea prematuramente. Cuantificamos esta diversidad mediante una medida presente en la bibliografía y mediante una variante que propusimos. Ante una gran cantidad de individuos repetidos, nuestra medida se caracteriza por dar un resultado de menor diversidad que la anterior. Realizamos pruebas de nuestros algoritmos con distintas instancias del problema de la mochila. Para ello generamos instancias al azar y algunas instancias determinísticas diseñadas de manera que haya presencia de máximos locales que compitan con el máximo global. También realizamos pruebas para encontrar el óptimo de una función de prueba derivable con dominio en el plano real, con el fin de evaluar cómo se comportan nuestros algoritmos con un problema continuo.

De las pruebas realizadas obtuvimos las conclusiones que enumeramos a continuación. En todos los casos disponíamos de la solución óptima para la instancia del problema a resolver o una cota para la misma. Con este conocimiento comparamos los algoritmos genéticos entre sí, establecimos cuáles consideramos que son los aportes principales de esta tesis y el trabajo que podría continuar nuestra investigación.

1. Sensibilidad al método de codificación y a la distancia utilizada.

   Comprobamos la importancia de encontrar un método adecuado para codificar los elementos del espacio de búsqueda y para definir la distancia utilizada como medida de separación entre esos elementos. Hemos utilizado el método de codificación binaria, considerando las ventajas que posee representar a los individuos de la población con una máxima cantidad de posiciones para que puedan existir mutaciones pequeñas. Como medida de distancia utilizamos la distancia de Hamming (número de bits diferentes). Esta implementación resultó exitosa para nuestro problema de optimización discreta pero no así para el problema de optimización continua (función derivable sobre el plano real). Para esta última aplicación se debería conseguir una codificación y una distancia que se correlacionen adecuadamente con la distancia natural en el dominio de la función, es decir, la distancia euclídea. Es respecto de esta distancia que la función es continua y derivable, de manera que cambios pequeños en el dominio se corresponden con cambios pequeños en el valor de la función, lo cual es deseable para la optimización evolutiva.

2. Bajo desempeño del algoritmo canónico.

   El algoritmo genético básico, sin elitismo, tuvo un bajo desempeño en nuestras pruebas. Los mejores resultados se obtuvieron manteniendo la mejor solución, mientras que el riesgo de homogeneización que esto conlleva se evitó aumentando controladamente la diversidad. Suponemos que el algoritmo no elitista puede ser apropiado en problemas muy extensos como juegos muy complejos o diseños de ingeniería con muchas variables. En estos casos cada generación requiere mucho tiempo de cómputo y sólo se corren unas pocas generaciones sin esperar llegar a un óptimo global sino a alguna solución que represente una respuesta novedosa y satisfactoria. Con elitismo hay mayor riesgo de estancamiento en un óptimo local que en pocas generaciones no podrá ser removido.

3. Buen desempeño de los métodos con control de diversidad

   Incorporamos al algoritmo genético con elitismo un método para cuantificar la diversidad de la población. Para esto implementamos la propuesta de Sidaner, Bailleux y Chabrier cuya medida de dispersión llamamos Dispersión 1. Además propusimos otra medida, Dispersión 2, que ante la presencia de individuos repetidos disminuye más su valor que Dispersión 1. Generamos al azar instancias del problema de la mochila e implementamos la solución de un problema de clasificación de imágenes para probar ambas medidas. Con ellas ejecutamos el algoritmo genético elitista imponiendo un aumento de la tasa de mutación cada vez que la dispersión baja hasta alcanzar cierta cota predeterminada. Comprobamos que con ambas medidas de dispersión los resultados eran mejores que sin control de la diversidad. Observamos que el aplicar el método de Dispersión 2 es levemente mejor que el método de Dispersión 1 en, prácticamente, todos los casos.

4. Importancia del método para acotar la diversidad

   Para el control automático de la diversidad debimos desarrollar un procedimiento que aumente la mutación cuando la dispersión cae debajo de cierta cota. Un aporte que creemos importante de esta tesis es la determinación de esa cota, lo cual requirió cálculos probabilísticos. Hallamos la dispersión esperada en una población generada al azar, que es la situación de la población inicial del algoritmo y por otro lado, según demostramos, prácticamente igual a la dispersión de una población sin individuos repetidos. Entre los valores de la dispersión mínima (población homogénea) y máxima (población generada al azar) ubicamos la cota dividiendo por distintos factores el valor máximo, o sea la esperanza de la dispersión al azar. A partir de las pruebas empíricas realizadas para el problema de la mochila, con tasa de mutación inicial de 0,01 e incrementos de tasa de mutación iguales a 0,0001 (valores pequeños para evitar un incremento demasiado rápido de la mutación) pudimos concluir que:

   - Con un divisor de la esperanza con valor 3 ó 4 obtuvimos las convergencias más rápidas.
   - Con una cota más grande, cuando se utiliza como divisor de la esperanza el valor 2 por ejemplo, la población demora más en alcanzar el óptimo. La situación se asemeja a la de ausencia de control de diversidad.
   - Con una cota pequeña, con un divisor de la esperanza alrededor de 6, la población ya se aleja en varias ocasiones de la solución óptima dado que se llega a altas tasas de mutación y la situación se asemeja a una simple búsqueda aleatoria.

5. Eficiencia razonable en comparación con el método determinístico

   En un algoritmo genético no se sabe a priori cuál será el tiempo de ejecución. Además, terminada la ejecución en base a un criterio de detención, la solución obtenida puede ser más o menos satisfactoria o creativa, pero en muchos casos no se sabe si es un óptimo global. En nuestras pruebas con el problema de la mochila contamos con el dato del óptimo global obtenido por programación dinámica y eso nos permitió obtener la cantidad de generaciones que cada corrida requería para converger al mismo. De manera que si bien la comparación no debe tomarse como para establecer la superioridad de un método u otro, podemos mencionar que en la mayoría de los casos, el número de generaciones hasta la convergencia era menor que el tamaño de la matriz de ítems y pesos que el algoritmo de programación dinámica debe recorrer. El tiempo de ejecución solía ser mayor para el algoritmo genético, lo cual es atribuible a que los problemas estudiados son de relativamente pocas variables por lo cual no llega a percibirse en la práctica que programación dinámica no es polinomial para este problema. Cabe señalar que también experimentamos aumentando el tamaño del problema de la mochila para forzar a que el método de programación dinámica no pudiera correr por falta de memoria, y comprobamos que en estos casos el algoritmo genético podía ejecutarse sin inconvenientes para el problema con esas dimensiones.

Finalmente, ya no como resultado de las pruebas sino de manera analítica, pudimos demostrar que un algoritmo genético elitista con control de diversidad siempre converge a un óptimo global. Este resultado no incluye conclusiones sobre la velocidad de convergencia, pero mostramos con un contraejemplo que cualquiera sea el algoritmo, nada puede decirse a priori sobre la velocidad si no se tiene información sobre el problema particular que se quiere resolver. Aunque no da información sobre la velocidad, el resultado de convergencia nos da la seguridad de que los estancamientos siempre van a ser superados, y la cuestión es si nuestro algoritmo los supera o no con eficiencia en un problema determinado, cuestión que se puede dilucidar, o bien con pruebas estadísticas o con un enfoque teórico que tome en cuenta, además de las propiedades del algoritmo, los supuestos que cumple el problema a resolver. Esto último puede ser una línea de investigación interesante como trabajo futuro, para ser aplicada a ciertas familias de problemas, por ejemplo grupos de funciones de los que se tiene información sobre patrones de correlación entre sus variables.

Anexo A

Glosario de conceptos derivados de la biología

- **Algoritmo genético (AG)**: técnica de programación que imita la evolución biológica como estrategia para resolver problemas. Definición propuesta por John Koza [KOZ/92]: 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.
- **Cruzamiento (Crossover)**: intercambio de material genético de dos cromosomas
- **Fenotipo**: manifestación visible del genotipo en un determinado ambiente.
- **Flujo genético**: o intercambio de material genético entre seres vivos de diferentes especies. Normalmente se produce a través de un vector, que suelen ser virus o bacterias; éstas incorporan a su material genético genes procedentes de una especie a la que han infectado, y cuando infectan a un individuo de otra especie pueden transmitirle esos genes a los tejidos generativos de gametos.
- **Función de aptitud (fitness)**: indica que tan buena es una solución (o individuo) con respecto a las demás.
- **Gen**: secuencia de ADN que constituye la unidad funcional para la transmisión de los caracteres hereditarios.
- **Genoma**: conjunto de material genético.
- **Genotipo**: conjunto de los genes de un individuo, incluida su composición alélica.
- **Mutación**: alteración producida en la estructura o en el número de los genes o de los cromosomas de un organismo transmisible por herencia.

- **Poliploidía**: mientras que las células normales poseen dos copias de cada cromosoma, y las células reproductivas una (haploides), puede suceder por accidente que alguna célula reproductiva tenga dos copias; si se logra combinar con otra célula diploide o haploide dará lugar a un ser vivo con varias copias de cada cromosoma. La mayoría de las veces, la poliploidía da lugar a individuos con algún defecto (por ejemplo, el tener 3 copias del cromosoma 21 da lugar al mongolismo), pero en algunos casos se crean individuos viables. Un caso de mutación fue el que sufrió (o disfrutó) el mosquito Culex pipiens, en el cual se duplicó un gen que generaba una enzima que rompía los organofosfatos, componentes habituales de los insecticidas.

- **Recombinación**: cuando las dos células sexuales, o gametos, una masculina y otra femenina se combinan, los cromosomas de cada una también lo hacen, intercambiándose genes, que a partir de ese momento pertenecerán a un cromosoma diferente. A veces también se produce traslocación dentro de un cromosoma; una secuencia de código se elimina de un sitio y aparece en otro sitio del cromosoma, o en otro cromosoma.

- **Selección Natural**: los individuos que tengan algún rasgo que los haga menos válidos para realizar su tarea de seres vivos, no llegarán a reproducirse, y, por tanto, su patrimonio genético desaparecerá del pool; algunos no llegarán ni siquiera a nacer. Esta selección sucede a muchos niveles: competición entre miembros de la especie (intraespecífica), competición entre diferentes especies, y competición predador-presa, por ejemplo. También es importante la selección sexual, en la cual las hembras eligen el mejor individuo de su especie disponible para reproducirse.

Anexo B

Funciones de comportamiento denominado Aguja en un pajar

Calculamos aquí la velocidad de convergencia al óptimo en una función de las denominadas *Needle in a haystack* (Aguja en un pajar). Se denominan así las funciones que poseen un máximo (o equivalentemente un mínimo) absoluto en un punto x_0 desconocido de un dominio finito y tal que la evaluación de la función en cualquier otro punto no da información sobre la ubicación de x_0. Un ejemplo es:

$$f(x) = \begin{cases} 1 & \text{Si } x = x_0\text{, donde } x_0 \text{ es desconocido} \\ 0 & \text{Si } x \neq x_0 \end{cases}$$

Otro ejemplo es la función $g(x) = X(x)$ donde las $X(x)$, con x variando en el dominio de g (el espacio de búsqueda), son realizaciones independientes de variables aleatorias con distribución uniforme en el intervalo $[0,1]$. Cabe destacar que con probabilidad 1 el máximo es único porque es nula la probabilidad de obtener dos valores iguales en finitas realizaciones. En la práctica pueden aparecer funciones no exactamente iguales a éstas pero si aproximadamente iguales por poseer un pico (valor máximo, mucho más alto que sus vecinos) aislado de otros valores altos.

Con cualquier método, determinístico o estocástico, ninguna búsqueda en estas funciones es mejor que una búsqueda exhaustiva, porque ninguna evaluación de la función da información sobre la ubicación x_0 del máximo, salvo aquella evaluación en la que el máximo es encontrado. Tanto para una búsqueda exhaustiva determinística (con un recorrido sistemático por el dominio) como estocástica (recorriendo el dominio con algún procedimiento estocástico), la mejor estrategia posible sólo puede consistir en no repetir la búsqueda en un punto ya evaluado.

Sea N el número de elementos del domino (el espacio de búsqueda). Si los elementos del dominio son cadenas binarias de longitud l, entonces $N = 2^l$. El número de pasos que le insume a un algoritmo determinístico o estocástico encontrar el óptimo es $N = 2^l$ en el peor caso, es decir, un tiempo exponencial en el número l de variables, que en este caso consideramos binarias, pero el resultado es análogo si toman más de dos valores posibles. Para no considerar sólo el peor caso, podemos suponer que x_0 se encuentra en una posición desconocida que es con igual probabilidad

cualquiera de los N puntos del dominio y calcular la esperanza del número de pasos hasta encontrar x_0. Definimos entonces:

$T =$ Número de pasos hasta encontrar x_0 en un muestreo sin reposición, es decir, sin repetir la búsqueda en un punto ya evaluado.

Buscamos entonces $E(T)$. Para ello notamos que si cualquier ubicación de x_0 es igualmente probable, que es el supueto más razonable ante la ignorancia sobre x_0, tenemos $P(T = n) = 1/N$. Luego

$E(T) = \sum_{n=1}^{N} 1/N = N(N+1)/(2N) = (N+1)/2 = (2^l + 1)/2$

De manera que el tiempo esperado también es exponencial como función del número de variables.

Los algoritmos genéticos suelen usarse para optimizar funciones sobre las que no se hacen supuestos (de linealidad, monotonía, etc.) con lo cual no excluyen funciones de tipo aguja en un pajar (o cercanas a una de ese tipo). Los cálculos anteriores muestran que ninguna variante de algoritmo genético puede ser eficiente para ciertas funciones. Esta conclusión valoriza la posibilidad de probar al menos convergencia al óptimo con probabilidad 1. Esto es lo que hicimos en este trabajo para algoritmos genéticos con control de la diversidad sin obtener conclusiones sobre la velocidad de convergencia, la cual dependerá de la función a optimizar sin que se pueda obtener una conclusión general de eficiencia.

Anexo C

Resumen de los experimentos

A continuación se muestran lotes de prueba y los resultados de los experimentos realizados que sirvieron para desarrollar las conclusiones acerca de los métodos del control de la diversidad.

Lotes de prueba

Resultado: 2577		Resultado: 4005		Resultado: 3536		Resultado: 455		Resultado: 6454	
Tope: 1517		Tope: 1810		Tope: 2335		Tope: 10		Tope: 843	
Pesos	Valores	Pesos	Valores	Pesos	Valores	Pesos	Valores	Pesos	Valores
1	41	1	3	1	34	1	1	1	10
21	33	11	49	111	1	1	2	56	326
33	45	4	87	8	6	1	3	129	12
52	26	10	46	30	3	1	4	52	142
19	11	109	21	50	44	1	5	132	12
18	4	34	52	17	109	1	6	17	15
15	10	34	72	21	34	1	7	34	43
3	109	1	124	10	34	1	8	815	322
98	34	6	194	65	1	1	9	106	21
49	34	3	275	32	6	1	10	422	457
85	1	44	70	144	3	1	11	791	1
87	6	107	4	207	4	1	12	15	12
13	3	13	100	22	45	1	13	16	872
76	44	22	44	13	26	1	14	158	194
191	107	9	94	291	11	1	15	231	738
88	13	129	156	29	41	1	16	54	574
48	22	117	72	111	1	1	17	134	219
52	9	31	51	241	3	1	18	76	82
21	19	159	240	18	21	1	19	198	138
46	61	84	269	156	33	1	20	183	2
141	108	31	2	65	11	1	21	633	100
92	196	49	4	150	49	1	22	234	133
27	180	85	300	297	300	1	23	142	162
5	129	87	250	31	111	1	24	700	72
67	117	13	200	84	34	1	25	617	968
49	31	76	125	91	25	1	26	14	138
26	159	191	150	31	61	1	27	18	856
54	84	13	297	159	84	1	28	419	115
62	31	43	31	84	159	1	29	191	24
81	87	56	84	31	56	1	30	196	159
34	33	121	91	87	121	1	31	69	84
34	52	345	32	33	345	1	32	9	95
1	19	243	66	52	243	1	33	71	122
6	18	4	98	19	4	1	34	129	109
3	85	231	1	22	141	1	35	103	104
44	87	132	7	78	92	1	36	17	78
107	13	23	38	66	27	1	37	83	574
13	76	33	1	82	5	1	38	94	765
21	191	48	15	36	67	1	39	91	882
64	88	79	82	142	59	1	40	5	45
51	48	82	13	178	14	1	41	14	107
77	27	56	27	99	121	1	42	59	18
59	5	73	7	159	345	1	43	49	121
14	67	45	12	51	243	1	44	85	21
71	62	66	62	29	4	1	45	87	171
60	81	81	3	66	231	1	46	41	101
22	34	2	88	34	132	1	47	213	36
33	12	19	48	12	48	1	48	649	43
191	107	650	8	506	2	1	49	13	108
198	108	430	33	437	71	1	50	978	96

Figura C.1: Pesos y valores de los experimentos

Resumen de resultados obtenidos para la COTA de valor 1

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación del Individuo	Longitud del Individuo	Cantidad de Individuos	Generaciones
0,01	0,001	50	100	1000

	Cantidad de generaciones 1000																Cantidad de generaciones: 2000			
	Número de Generaciones hasta alcanzar el óptimo = 2577 Tope: 1517				Número de Generaciones hasta alcanzar el óptimo = 4005 Tope: 1810				Número de Generaciones hasta alcanzar el óptimo = 3536 Tope: 2335				Número de Generaciones hasta alcanzar el óptimo = 6454 Tope: 843				Número de Generaciones hasta alcanzar el óptimo = 455 Tope: 10			
	Div 23,01				Div 23,01				Div 23,01				Div 23,01				Div 23,01			
Cant de Corridas	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Incr	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc
1	71	2	202	6	117	0	51	6	102	1	46	5	96	2	53	5	209	1	239	5
2	96	11	69	17	255	13	133	15	133	12	559	17	113	12	93	15	168	12	405	14
3	799	11	70	16	376	12	61	16	288	10	56	15	132	12	219	15	149	11	233	14
4	34	12	93	17	426	14	72	16	350	12	243	16	588	12	96	18	174	10	113	12
5	273	12	228	17	107	12	188	18	71	11	53	15	87	12	167	14	177	10	181	13
6	1000	12	83	17	463	14	56	17	509	11	118	17	500	12	295	16	449	11	108	13
7	212	11	455	16	947	11	165	17	61	11	103	16	746	13	65	15	187	10	314	13
8	683	12	228	17	114	10	274	17	612	11	66	15	176	12	192	15	298	11	182	12
9	75	12	164	17	233	11	269	16	328	12	72	15	1000		105	14	155	12	192	13
10	25	11	111	19	95	12	87	16	1000	12	104	16	356	12	173	16	166	13	154	14
11	549	13	326	16	1000	12	128	17	836	11	514	17	136	13	441	15	139	11	409	17
12	174	12	160	17	153	12	114	15	700	12	190	15	109	12	222	15	171	12	800	13
13	137	11	101	17	490	12	149	15	52	13	99	17	202	13	212	16	229	10	573	13
14	136	12	119	16	607	12	83	16	197	11	55	15	138	13	104	15	185	12	185	16
15	151	12	112	16	574	11	65	16	154	11	99	16	156	12	184	15	194	11	1847	15
16	198	13	142	16	53	11	42	16	136	11	53	16	94	13	500	15	250	11	146	13
17	139	11	114	16	97	9	114	15	172	11	139	17	287	12	214	16	229	11	260	13
18	89	11	172	16	136	12	205	16	160	11	108	17	1000		302	15	346	11	332	13
19	222	11	81	17	265	13	50	15	139	12	102	16	147	12	440	15	518	12	418	15
20	107	11	113	16	687	11	59	15	706	12	32	16	462	12	169	14	451	11	195	13
21	113	12	65	16	531	13	97	15	120	12	78	15	414	12	209	18	197	13	117	13
22	179	13	159	16	11		81	18	323	11	248	17	77	12	141	15	153	11	161	12
23	736	11	210	16	112	12	67	15	413	12	119	19	183	13	171	14	310	12	812	15
24	187	12	118	16	127	13	68	15	512	12	460	19	221	13	935	17	131	11	350	13
25	156	11	54	17	294	11	72	14	92	11	170	15	238	13	128	14	146	12	302	13
26	277	12	57	16	167	11	185	15	53	11	214	17	338	13	113	14	206	12	367	13
27	126	13	68	16	347	12	227	16	567	12	67	15	419	12	242	14	365	11	220	14
28	1000	11	115	18	55	12	75	15	560	11	242	18	100	13	124	16	175	10	145	13
29	134	11	51	16	1000		161	15	58	11	57	15	97	12	207	15	163	11	207	15
30	66	13	74	17	377	12	114	16	1000	11	100	17	1000		56	14	187	10	151	13
31	310	11	124	16	138	11	99	16	372	11	122	18	278	12	70	18	123	14	368	13
32	151	11	150	16	84	13	109	15	1000	11	196	15	281	12	96	14	166	10	637	13
33	188	12	238	16	809	11	67	15	604	11	82	17	259	12	77	14	347	12	162	14
34	300	12	124	18	78	13	331	16	253	11	65	15	66	13	151	15	167	11	288	14
35	91	12	85	17	782	11	193	14	893	11	184	15	216	12	426	17	160	11	162	15
36	100	11	74	17	563	13	68	15	554	11	276	15	257	12	92	16	223	12	131	13
37	166	12	216	17	1000	13	143	15	138	11	107	15	578	13	121	15	171	11	2000	14
38	209	13	104	16	176	12	79	14	267	13	122	16	418	13	363	18	156	11	384	15
39	580	11	129	16	345	10	66	16	263	12	195	15	183	12	325	14	207	12	1490	14
40	216	11	152	17	12		62	16	109	13	58	16	183	13	164	17	190	11	178	13
41	614	12	782	17	244	11	57	15	368	11	195	15	91	12	175	16	164	12	144	13
42	116	12	537	18	978	11	106	16	415	12	179	15	164	13	105	14	151	11	114	14
43	1000	13	144	16	1000	11	71	15	836	12	72	15	1000		104	15	236	11	196	13
44	1000	11	260	19	148	13	142	15	109	12	87	16	1000		74	16	270	11	253	13
45	199	12	108	16	79	12	59	15	79	11	97	15	62	12	518	16	135	10	154	14
46	1000	12	330	6	706	13	76	16	546	12	117	17	696	14	368	15	331	12	487	12
47	90	13	189	17	125	11	136	17	478	11	93	16	588	14	105	17	167	12	263	14
48	78	13	50	16	849	11	181	15	512	12	242	20	119	12	101	15	353	11	190	13
49	87	12	94	16	429	12	30	16	317	12	176	15	656	12	299	15	108	12	104	14
50	695	12	220	16	885	14	56	16	381	10	42	15	116	12	88	16	125	11	225	13

Figura C.2: Experimento para COTA 1

Resumen de resultados obtenidos para la COTA de valor 1,5

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación de Ind	Longitud del Ind	Cantidad de Ind	Generaciones
0,01	0,001	50	100	1000

	Cantidad de generaciones 1000																Cantidad de generaciones: 2000			
	Número de Generaciones hasta alcanzar el óptimo = 2577 Tope: 1517				Número de Generaciones hasta alcanzar el óptimo = 4005 Tope: 1810				Generaciones hasta alcanzar el óptimo = 3536 Tope: 2335				Número de Generaciones hasta alcanzar el óptimo = 6454 Tope: 843				Número de Generaciones hasta alcanzar el óptimo = 455 Tope: 10			
	Div 15,35				Div 15,35				Div 15,35				Div 15,35				Div 15,35			
Cant de Corridas	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc
1	89	4	78	6	398	4	50	6	116	3	113	6	52	5	765	6	150	1	316	3
2	102	15	67	16	145	12	101	16	284	14	87	16	101	16	85	16	443	12	146	13
3	130	14	73	17	419	14	160	17	231	14	127	16	96	13	98	15	157	13	199	14
4	118	14	1318	16	424	13	84	17	78	13	68	16	201	16	105	15	310	12	659	15
5	705	14	133	17	262	12	36	16	76	14	81	17	213	13	94	16	448	13	754	14
6	103	14	196	16	98	14	85	16	188	13	74	17	56	14	88	17	160	14	284	14
7	321	15	48	17	69	13	380	17	78	13	73	17	137	13	102	15	498	13	1009	13
8	82	15	218	19	81	13	339	17	86	13	71	16	112	14	257	15	250	12	110	14
9	186	13	243	19	36	13	93	16	73	13	150	18	306	13	56	15	214	11	109	14
10	142	15	69	16	338	14	129	16	90	13	46	15	188	13	62	15	281	13	561	13
11	174	14	126	17	230	15	115	16	76	13	177	17	176	15	172	17	304	11	246	14
12	541	13	295	17	146	13	56	17	155	14	85	16	214	15	244	15	206	11	108	14
13	245	14	206	18	87	15	88	15	109	12	129	16	189	14	144	17	419	12	722	14
14	534	15	117	20	196	15	257	17	186	14	98	16	366	15	109	16	121	13	183	14
15	95	14	148	17	100	13	111	16	257	12	64	17	189	14	130	16	417	13	254	14
16	153	13	151	21	42	13	58	16	138	16	141	16	93	16	95	15	96	14	166	14
17	78	13	44	18	227	13	169	16	68	13	61	15	417	14	306	15	161	12	602	14
18	173	14	51	19	400	13	145	16	263	15	87	17	197	13	106	16	388	12	219	13
19	175	14	164	17	74	15	175	17	89	13	110	17	207	14	256	15	285	13	1188	15
20	151	13	97	18	102	15	152	18	81	14	69	16	274	13	595	16	130	11	165	13
21	115	15	86	17	182	15	193	18	229	13	30	16	329	14	184	15	384	11	690	14
22	104	14	1188	17	103	13	46	16	306	12	48	17	165	15	67	17	108	12	135	16
23	347	14	117	17	486	12	73	16	185	15	185	16	195	13	81	16	213	11	452	13
24	1000	14	121	17	220	13	42	16	243	14	99	16	297	13	74	15	169	12	852	14
25	157	13	253	18	133	14	320	20	85	14	55	17	305	14	88	15	619	14	155	13
26	434	14	74	16	269	13	193	18	137	14	64	16	245	13	188	16	511	14	126	14
27	144	14	61	18	674	13	262	18	104	14	62	17	143	14	832	18	140	11	615	14
28	89	16	339	17	65	13	60	19	129	15	88	16	96	15	70	16	263	13	320	13
29	59	15	234	17	422	13	85	16	412	13	172	18	259	14	98	16	249	14	376	15
30	59	13	229	18	307	14	87	17	59	13	108	15	965	16	417	16	109	14	172	13
31	109	15	46	17	128	13	65	16	177	14	114	16	465	13	105	15	326	12	172	13
32	119	15	39	17	360	14	208	17	219	13	139	15	142	14	139	15	303	12	834	13
33	73	13	352	18	169	14	115	16	62	16	76	16	406	14	372	15	246	12	109	14
34	82	17	123	17	144	15	117	15	107	14	34	16	979	13	53	15	268	13	296	13
35	457	14	126	19	75	13	49	17	52	12	53	18	142	15	118	16	893	13	131	15
36	144	15	98	18	209	14	45	16	96	13	94	18	205	13	132	15	864	13	258	14
37	114	16	81	19	91	13	136	17	153	13	84	16	70	13	136	17	204	13	173	13
38	197	14	91	17	113	13	109	17	446	13	94	17	265	13	1000		802	13	337	14
39	1000	14	190	17	97	14	136	16	92	13	437	18	126	13	96	15	116	11	279	14
40	174	13	321	19	69	14	151	15	137	14	55	16	231	15	828	16	201	11	395	16
41	102	14	245	17	152	13	72	16	161	14	110	16	127	13	68	17	143	13	110	14
42	181	13	198	17	224	13	91	16	83	13	69	19	237	13	60	15	647	13	164	15
43	142	13	74	17	116	13	103	16	168	13	55	17	1000		124	18	314	11	407	14
44	168	14	302	17	128	13	182	18	66	13	58	17	217	14	99	16	202	12	213	13
45	125	14	63	16	76	15	184	16	205	14	119	16	113	13	493	16	183	11	223	15
46	81	14	115	17	63	16	239	16	69	13	173	16	444	14	216	17	159	13	121	14
47	145	13	100	18	97	14	41	16	90	13	329	19	205	14	133	15	220	13	143	13
48	115	13	107	17	65	15	71	17	159	15	56	17	129	14	168	18	248	11	177	13
49	198	14	53	18	73	13	74	17	96	13	77	16	226	13	305	15	127	14	233	14
50	845	13	149	16	57	12	86	17	90	13	102	17	108	14	283	15	374	15	208	13

Figura C.3: Experimento para COTA 1,5

Resumen de resultados obtenidos para la COTA de valor 2
Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación de Ind	Longitud del Ind	Cantidad de Ind	Generaciones
0,01	0,001	50	100	1000

	Cantidad de generaciones 1000																Cantidad de generaciones: 2000			
	Número de Generaciones hasta alcanzar el óptimo = 2577 Tope: 1517				Número de Generaciones hasta alcanzar el óptimo = 4005 Tope: 1810				Número de Generaciones hasta alcanzar el óptimo = 3536 Tope: 2335				Número de Generaciones hasta alcanzar el óptimo = 6454 Tope: 843				Número de Generaciones hasta alcanzar el óptimo = 455 Tope: 10			
	Div 11,5				Div 11,5				Div 11,5				Div 11,5				Div 11,5			
Cant de Corridas	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Incr	Disp 1	Incr	Disp 2	Inc
1	180	6	115	8	269	4	48	8	52	4	102	8	71	5	48	8	235	4	1063	4
2	101	16	55	21	130	14	261	17	201	14	64	18	262	18	143	21	742	14	2000	
3	147	14	139	16	347	17	42	17	96	15	233	18	500	14	178	17	213	14	97	15
4	66	15	195	18	186	14	125	17	412	16	84	16	119	15	95	18	197	14	120	14
5	153	15	130	16	131	15	112	19	268	16	62	18	198	14	147	17	2000		365	14
6	60	17	45	18	94	17	83	17	138	15	61	17	160	16	162	20	207	12	2000	
7	115	15	139	19	105	14	75	16	213	14	151	18	383	15	251	18	216	13	431	14
8	73	15	237	17	50	16	119	16	154	14	41	17	78	15	671	18	402	13	244	13
9	241	15	153	19	113	14	118	17	138	15	80	16	695	14	422	17	101	12	264	14
10	239	16	232	17	49	15	183	17	32	17	57	18	489	14	184	17	229	14	447	15
11	64	16	117	17	135	15	133	16	156	14	63	17	95	15	146	18	262	13	318	15
12	220	15	69	17	60	15	112	16	70	15	46	18	133	15	100	18	200	14	307	14
13	174	16	72	16	463	15	128	16	59	16	90	16	237	14	47	18	382	14	179	15
14	119	16	180	17	89	14	69	16	60	18	188	17	224	15	147	19	487	15	624	14
15	175	14	181	16	176	15	145	18	161	14	104	17	291	15	97	18	121	13	261	14
16	150	15	261	17	97	15	162	17	183	16	73	16	134	16	45	19	155	13	90	14
17	181	16	61	10	197	14	75	20	75	16	123	17	327	15	44	18	753	14	365	14
18	274	15	424	20	168	15	48	18	122	14	217	17	253	13	112	16	1916	14	610	16
19	139	14	103	21	65	11	30	16	169	15	99	17	109	16	109	19	229	13	229	13
20	97	16	76	17	302	14	162	18	91	14	56	16	137	16	200	19	170	12	904	14
21	98	15	453	17	57	16	118	17	167	17	114	17	481	15	401	18	380	13	273	15
22	160	15	77	17	158	15	87	18	152	14	105	17	215	14	129	17	194	12	329	14
23	227	15	172	18	226	14	318	18	268	14	43	16	311	13	685	16	128	14	183	15
24	685	15	68	18	229	14	31	18	134	16	140	18	214	14	259	17	122	13	583	15
25	133	15	83	18	262	18	120	17	71	14	245	20	144	13	159	16	131	13	176	14
26	81	16	64	17	93	16	139	17	77	14	63	16	63	16	219	19	610	13	109	14
27	171	17	147	17	96	14	105	17	260	14	81	17	98	14	117	17	168	13	369	14
28	430	15	120	20	57	17	68	18	97	16	87	16	294	13	149	16	163	12	346	16
29	31	17	67	17	106	14	78	17	103	14	51	17	129	14	220	17	215	12	160	14
30	251	15	147	19	77	14	87	16	71	15	112	16	517	14	112	17	1385	15	1616	14
31	163	14	199	16	155	15	73	16	99	14	487	19	79	14	401	17	576	4	208	14
32	116	15	260	21	193	15	124	18	101	15	121	17	125	14	1000	17	206	12	143	15
33	152	15	171	18	258	16	80	17	138	14	76	21	108	15	80	18	86	12	1099	15
34	81	16	324	19	119	15	213	18	88	15	101	19	90	16	103	19	170	15	309	14
35	102	16	86	18	171	14	143	17	87	15	69	19	104	14	271	17	125	14	230	14
36	61	16	108	18	136	16	54	19	127	16	56	17	1000		514	3	180	13	187	14
37	119	16	130	19	62	14	109	18	90	14	113	18	237	16	390	19	1258	16	405	15
38	125	16	305	15	88	15	220	17	45	16	113	20	566	16	958	19	503	15	66	14
39	172	15	82	20	97	15	50	16	54	14	101	16	239	14	295	17	229	13	792	15
40	91	16	155	19	65	15	188	16	276	14	114	17	147	13	237	16	924	15	981	15
41	240	15	194	20	107	14	87	17	68	15	209	17	159	14	288	17	1225	15	184	15
42	75	15	107	18	233	14	56	17	98	14	102	16	72	14	123	18	557	12	373	14
43	132	16	103	17	139	16	93	18	178	14	47	16	169	14	745	17	138	13	335	14
44	175	15	87	19	102	18	155	19	131	14	74	17	292	14	101	17	527	14	232	14
45	114	16	123	18	240	14	233	18	94	14	140	17	107	15	95	18	163	14	287	16
46	161	16	122	17	57	14	124	18	338	17	112	18	169	15	170	18	286	13	15	13
47	178	17	216	16	121	15	125	18	100	14	272	17	87	18	825	21	332	12	307	15
48	207	15	125	16	74	15	139	17	203	15	50	17	193	16	66	19	192	13	281	14
49	220	16	146	17	87	15	97	17	109	17	71	18	473	14	169	17	1113	15	407	13
50	258	15	212	19	112	15	62	18	132	15	75	17	121	14	136	18	548	14	241	15

Figura C.4: Experimento para COTA 2

Resumen de resultados obtenidos para la COTA de valor 4

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación	Longitud del Ind	Cantidad de Ind	Generaciones
0,01	0,001	50	100	1000

	Cantidad de generaciones 1000																Cantidad de generaciones: 2000			
	Número de Generaciones hasta alcanzar el óptimo = 2577 Tope: 1517				Número de Generaciones hasta alcanzar el óptimo = 4005 Tope: 1810				Número de Generaciones hasta alcanzar el óptimo = 3536 Tope: 2335				Número de Generaciones hasta alcanzar el óptimo = 6454 Tope: 843				Número de Generaciones hasta alcanzar el óptimo = 455 Tope: 10			
	Div 5,75				Div 5,75				Div 5,75				Div 5,75				Div 5,75			
Cant de Corridas	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc
1	332	10	165	11	106	9	137	13	128	10	55	10	102	7	872	10	1188	5	313	7
2	115	19	98	21	288	23	297	22	81	20	67	21	318	17	366	18	827	16	703	17
3	263	21	108	21	248	20	243	21	121	20	154	19	19	17	184	18	867	22	309	17
4	664	21	391	21	72	23	138	21	290	21	216	20	19	15	862	17	141	17	328	17
5	105	19	132	19	173	24	34	23	363	18	311	20	255	17	123	18	892	18	2000	19
6	75	19	1000	19	238	19	710	22	188	18	458	21	94	20	776	20	1298	20	1858	16
7	277	22	146	21	206	19	128	22	113	20	38	20	71	17	129	17	1174	17	242	18
8	73	20	88	21	305	20	1000	22	134	19	211	23	157	17	555	19	1080	15	939	18
9	385	21	318	21	81	20	501	23	515	20	68	20	210	18	174	18	1578	17	2000	19
10	254	20	103	21	407	20	207	21	237	18	53	23	117	18	53	19	490	16	567	17
11	73	21	158	23	124	21	238	23	264	19	45	19	70	19	76	18	529	18	2000	20
12	204	22	172	21	400	20	65	23	102	20	68	21	82	16	136	21	195	17	568	17
13	853	21	350	21	68	21	506	20	56	20	35	21	19	19	76	21	2000	17	842	17
14	1000		205	21	86	19	165	21	300	24	97	20	640	21	347	19	869	15	707	17
15	184	18	321	20	233	23	34	21	125	19	59	21	83	17	607	19	186	17	2000	19
16	100	19	131	20	289	21	696	22	66	21	753	20	19	19	253	21	615	16	1383	18
17	88	20	166	19	39	20	221	22	237	19	80	20	290	18	1000	20	668	15	151	17
18	397	21	308	21	97	19	60	21	134	20	307	21	599	18	373	20	1784	20	1482	17
19	94	21	122	19	47	19	473	23	70	18	1000		19	18	1000	20	957	17	1747	16
20	201	21	52	21	176	19	77	23	865	24	202	21	314	17	90	18	477	15	459	17
21	137	19	541	21	76	20	65	21	102	19	262	21	656	17	80	19	2000	15	1327	17
22	124	18	151	19	132	20	255	23	161	18	85	19	19	17	408	18	570	18	151	17
23	306	19	452	21	119	18	55	20	58	19	71	19	199	17	1000	18	672	17	718	17
24	247	20	129	19	394	22	53	21	73	18	145	22	387	20	127	18	286	15	2000	16
25	280	19	101	21	181	21	194	21	78	21	84	20	459	17	54	19	1206	20	1545	17
26	59	21	734	21	138	23	265	21	65	20	446	20	71	18	329	20	875	16	408	18
27	183	20	348	21	331	21	170	20	84	20	78	21	720	17	106	18	165	15	739	16
28	1000		197	21	81	21	524	21	912	21	135	21	196	18	67	18	17	15	606	17
29	51	20	113	21	93	21	33	22	355	18	52	23	119	17	141	19	304	18	531	17
30	1000		104	21	191	21	170	21	93	19	203	20	98	17	102	18	875	15	126	17
31	115	19	132	21	297	22	75	21	26	20	247	20	122	18	1000		2000	16	2000	18
32	154	19	110	20	53	20	185	22	215	19	57	20	22	17	95	19	615	16	346	17
33	220	19	157	22	911	19	168	22	104	19	85	20	558	18	577	20	2000	16	2000	18
34	507	21	386	20	82	19	158	21	56	21	79	21	112	17	396	19	365	17	679	17
35	331	21	246	20	138	21	340	21	196	19	97	20	19	16	112	18	740	15	2000	17
36	237	20	58	21	376	20	550	21	68	20	434	20	142	17	139	18	58	15	577	18
37	61	19	74	20	124	22	100	20	86	20	173	20	957	17	230	19	599	17	641	17
38	477	19	158	21	61	20	80	22	73	19	347	21	262	17	252	18	667	15	152	17
39	211	22	97	19	359	22	125	21	284	20	141	21	771	17	166	19	543	16	122	16
40	172	20	100	20	540	21	58	22	561	20	162	23	20	17	93	19	557	17	238	19
41	87	20	166	23	72	20	99	21	213	21	68	20	457	20	195	18	1758	16	2000	18
42	273	21	43	20	68	19	42	24	123	19	238	20	793	18	254	19	158	16	662	17
43	305	21	288	20	181	20	351	21	264	22	156	20	80	17	674	19	764	16	140	18
44	78	21	216	20	284	20	162	21	739	22	52	21	19	17	1000	19	2000	17	2000	19
45	243	20	45	20	107	20	101	24	908	19	264	20	357	17	105	18	2000	15	372	16
46	115	19	183	19	316	22	205	21	77	18	74	20	409	19	450	17	1185	15	421	17
47	84	21	126	20	86	19	304	22	234	19	185	19	73	17	204	20	1032	16	189	21
48	126	20	32	20	108	20	78	20	61	18	312	22	18		66	19	187	14	2000	16
49	510	19	521	20	42	21	62	21	95	18	659	22	620	17	235	19	1602	16	130	16
50	153	20	817	21	273	19	65	21	203	19	509	20	353	19	859	18	1227	17	115	18

Figura C.5: Experimento para COTA 4

Resumen de resultados obtenidos para la COTA de valor 6

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación de Ind	Longitud del Ind	Cantidad de Ind	Generaciones
0,01	0,001	50	100	1000

	Cantidad de generaciones 1000																Cantidad de generaciones: 2000			
	Número de Generaciones hasta alcanzar el óptimo = 2577 Tope: 1517				Número de Generaciones hasta alcanzar el óptimo = 4005 Tope: 1810				Número de Generaciones hasta alcanzar el óptimo = 3536 Tope: 2335				Número de Generaciones hasta alcanzar el óptimo = 6454 Tope: 843				Número de Generaciones hasta alcanzar el óptimo = 455 Tope: 10			
	Div 3,92				Div 3,92				Div 3,92				Div 3,92				Div 3,92			
Cant de Corridas	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc
1	100	16	16	17	1000	16	34	18	33	15	148	15	1000	23	1000	23	2000	20	2000	21
2	640	24	62	24	1000	23	804	25	1000	24	1000	24	181	23	100	22	2000	19	2000	21
3	309	24	166	27	713	26	34	28	124	24	363	25	1000	22	170	23	2000	21	462	20
4	975	27	293	25	1000	23	640	25	684	24	295	24	1000	22	1000	23	2000	21	2000	21
5	177	23	226	26	99	27	1000	28	40	23	226	26	1000	23	1000	24	543	20	2000	21
6	1000	22	394	24	672	25	1000	27	84	25	581	25	1000	22	1000	23	2000	20	1485	21
7	67	25	1000	26	1000	26	1000	28	582	27	1000	27	473	22	1000	23	1586	19	1902	21
8	394	25	266	25	512	26	1000	28	752	23	30	24	1000	22	931	23	1587	20	2000	22
9	539	26	1000	27	956	26	1000	29	200	25	124	26	1000	21	843	23	2000	19	1084	21
10	583	25	92	25	1000	25	813	27	26	28	1000	29	1000	23	1000	24	497	19	2000	20
11	367	24	1000	25	1000	23	245	25	76	24	702	27	518	23	1000	24	1994	18	790	20
12	88	25	201	24	1000	28	178	29	354	25	1000	26	1000	23	1000	24	2000	19	2000	20
13	518	26	485	25	1000	26	1000	28	258	23	1000	25	310	22	232	22	2000	20	2000	20
14	1000		876	27	487	25	218	27	1000	22	455	25	1000	22	720	23	2000	19	2000	20
15	684	26	297	25	131	25	678	27	160	27	1000	28	152	22	1000	24	2000	21	1231	22
16	53	25	110	24	230	26	256	25	1000	26	1000	28	156	21	171	24	2000	20	2000	20
17	413	25	211	25	450	26	341	27	767	26	337	24	1000	21	1000	23	2000	20	2000	21
18	250	24	290	26	275	26	1000	29	74	24	175	24	1000	22	657	24	2000	19	2000	21
19	607	26	371	28	691	27	1000	28	185	24	191	26	557	23	1000	23	389	20	1929	20
20	1000	25	173	27	1000	24	143	26	1000	25	169	26	667	21	1000	22	838	19	2000	20
21	100	24	163	24	346	26	45	26	989	25	158	24	1000	21	1000	23	2000	21	2000	22
22	1000	23	159	25	31	26	1000	27	1000	25	1000	26	100	22	1000	22	824	22	143	21
23	582	25	1000	26	1000	25	63	27	174	27	1000	28	838	23	341	23	2000	22	2000	21
24	96	25	1000	26	79	28	1000	29	626	24	1000	25	1000	23	801	24	2000	21	2000	22
25	618	26	1000	27	193	25	104	27	386	25	460	24	101	22	572	25	2000	21	2000	21
26	121	23	1000	25	179	26	1000	28	52	26	90	24	170	24	1000	25	2000	19	1778	22
27	357	23	347	26	1000	25	520	27	261	25	132	25	600	24	231	22	2000	19	872	22
28	1000	22	92	24	1000	25	759	27	90	24	1000	25	211	24	103	24	2000	19	2000	22
29	1000	24	109	26	386	27	345	25	401	25	495	25	469	23	1000	24	2000	21	2000	21
30	1000	23	159	25	183	25	725	26	360	25	118	24	899	22	1000	23	1739	19	2000	20
31	1000	23	491	24	514	25	460	26	226	25	1000	26	1000	22	326	23	2000	19	1223	20
32	103	25	707	24	1000	25	69	26	319	24	1000	26	476	24	1000	25	2000	20	926	21
33	574	24	186	24	174	26	546	26	41	25	103	25	212	22	1000	23	2000	20	2000	21
34	147	27	150	27	186	25	1000	26	51	22	1000	23	69	22	1000	22	2000	19	1614	21
35	169	27	448	26	511	25	110	26	396	25	109	25	1000	22	616	23	633	20	1825	20
36	492	24	498	25	719	26	58	29	935	24	339	26	166	21	824	22	2000	20	2000	22
37	1000	23	526	25	1000	26	1000	26	1000	23	159	24	182	27	1000	27	2000	20	1463	21
38	518	24	326	25	1000	27	549	27	1000	25	121	26	75	22	1000	24	2000	19	2000	22
39	1000	24	876	25	1000	25	561	25	1000	26	1000	27	468	21	363	24	2000	20	1358	21
40	873	24	302	24	246	26	65	27	240	26	800	25	1000	21	819	23	2000	19	2000	20
41	236	27	36	26	120	25	1000	27	1000	25	39	26	1000	22	112	23	2000	22	2000	22
42	282	25	88	24	179	26	387	27	406	23	139	24	384	25	73	23	2000	20	2000	21
43	1000	24	346	26	1000	24	52	26	76	26	92	25	352	24	608	22	2000	20	2000	21
44	749	25	244	25	728	25	1000	26	224	24	1000	24	192	21	1000	22	2000	19	2000	20
45	883	24	144	24	1000	26	1000	27	647	25	65	25	238	21	1000	22	817	19	1451	20
46	447	26	508	25	50	26	1000	27	43	25	1000	26	129	21	189	22	2000	19	1147	20
47	294	27	1000	28	800	28	1000	29	27	26	94	26	299	21	113	22	1268	21	1591	20
48	124	23	398	24	339	28	512	26	1000	26	1000	27	526	23	218	23	2000	21	2000	20
49	175	25	1000	27	116	26	1000	26	1000	28	964	29	1000	23	93	24	2000	20	252	21
50	606	24	278	26	238	26	1000	26	398	25	641	24	1000	23	169	25	2000	20	2000	21

Figura C.6: Experimento para COTA 6

Resumen de resultados obtenidos para la COTA de valor 8

Parámetros

Tasa Inicial de Mutación	Incremento de la Tasa de Mutación de Ind	Longitud del Ind	Cantidad de Ind	Generaciones
0,01	0,001	50	100	1000

	Cantidad de generaciones 1000																Cantidad de generaciones: 2000			
	Número de Generaciones hasta alcanzar el óptimo = 2577 Tope: 1517				Número de Generaciones hasta alcanzar el óptimo = 4005 Tope: 1810				Número de Generaciones hasta alcanzar el óptimo = 3536 Tope: 2335				Número de Generaciones hasta alcanzar el óptimo = 6454 Tope: 843				Número de Generaciones hasta alcanzar el óptimo = 455 Tope: 10			
	Div 2,87				Div 2,87				Div 2,87				Div 2,87				Div 2,87			
Cant de Corridas	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc	Disp 1	Incr	Disp 2	Inc
1	738	32	1000	32	1000	31	454	32	1000	31	935	32	317	18	1000	29	2000	27	2000	28
2	1000	33	139	34	1000	31	802	32	1000	31	1000	32	1000	28	1000	29	2000	27	1836	28
3	383	34	392	32	1000	33	1000	33	1000	31	1000	33	1000	28	1000	29	2000	27	2000	27
4	589	33	100	34	1000	33	149	34	1000	31	245	31	865	28	1000	29	2000	27	2000	27
5	1000	33	168	33	1000	33	1000	33	1000	32	1000	31	223	29	1000	30	1536	27	2000	28
6	1000	32	1000	33	1000	34	1000	33	468	31	1000	30	1000	29	296	30	2000	28	2000	27
7	1000	33	230	33	1000	34	1000	35	1000	30	117	30	753	32	1000	32	2000	28	2000	28
8	1000	34	1000	35	1000	33	1000	34	1000	30	1000	30	1000	32	1000	31	2000	28	2000	28
9	1000	32	1000	35	1000	35	1000	34	1000	30	974	32	1000	30	1000	31	2000	27	2000	28
10	1000	31	498	32	1000	31	1000	34	1000	31	927	34	1000	30	1000	31	2000	26	2000	27
11	1000	33	1000	33	1000	31	1000	35	1000	31	1000	34	1000	30	1000	31	2000	26	2000	29
12	817	33	487	32	1000	31	1000	33	1000	31	1000	34	1000	27	1000	31	2000	29	2000	29
13	446	33	118	36	1000	31	1000	34	1000	34	1000	34	1000	28	1000	28	2000	28	2000	29
14	1000	32	893	34	1000	33	1000	33	975	34	1000	34	1000	28	799	28	2000	28	732	29
15	1000	33	676	34	1000	32	1000	33	1000	34	1000	35	1000	30	202	28	2000	28	2000	28
16	439	32	1000	36	1000	31	1000	35	1000	34	1000	33	1000	29	1000	29	2000	27	1557	26
17	1000	31	878	33	1000	31	1000	33	1000	33	1000	34	596	30	1000	29	2000	26	2000	26
18	1000	32	576	33	1000	31	1000	33	1000	33	843	34	149	30	804	29	1452	28	2000	26
19	1000	33	471	35	713	33	619	34	1000	33	1000	35	1000	28	401	30	1627	27	2000	29
20	1000	32	516	32	1000	33	1000	34	1000	32	1000	32	1000	28	1000	29	2000	27	2000	29
21	1000	33	1000	35	1000	33	1000	34	49	32	36	32	60	28	718	29	2000	28	2000	29
22	250	32	1000	36	1000	33	1000	34	1000	31	1000	33	738	27	1000	29	2000	28	2000	29
23	1000	34	1000	36	1000	35	1000	33	1000	33	220	32	1000	28	1000	30	2000	27	2000	28
24	145	32	1000	37	1000	35	1000	34	1000	32	1000	33	1000	29	1000	30	2000	27	1910	28
25	1000	32	312	33	1000	33	31	34	233	33	228	33	1000	28	1000	30	2000	27	2000	28
26	390	33	1000	33	1000	33	834	34	1000	33	29	32	1000	29	1000	30	2000	27	2000	28
27	929	34	335	32	40	35	44	33	660	33	1000	33	1000	29	932	29	2000	28	2000	29
28	1000	34	921	33	1000	34	474	33	1000	33	965	31	1000	29	1000	29	2000	28	2000	29
29	524	35	1000	33	1000	33	144	35	90	32	1000	31	1000	32	1000	29	2000	28	2000	28
30	1000	35	1000	34	165	35	1000	35	1000	34	1000	31	1000	32	1000	31	2000	28	2000	27
31	1000	32	771	33	180	34	1000	35	1000	32	1000	33	1000	31	1000	31	2000	28	2000	29
32	648	31	241	32	87	34	1000	34	1000	32	1000	32	1000	31	1000	31	2000	28	2000	29
33	1000	31	1000	37	1000	33	1000	35	1000	31	1000	31	1000	32	1000	31	2000	28	2000	29
34	1000	33	116	34	1000	34	1000	34	69	31	1000	33	312	32	1000	32	2000	27	2000	29
35	381	35	1000	37	1000	34	1000	34	32	31	1000	34	1000	27	1000	31	2000	27	2000	28
36	1000	33	739	34	1000	34	1000	35	32	31	1000	33	1000	32	693	31	2000	27	2000	28
37	629	33	218	34	1000	35	1000	33	87	32	1000	33	196	27	71	32	2000	27	2000	28
38	1000	34	1000	34	1000	33	1000	35	134	31	364	33	543	29	1000	32	2000	27	1728	28
39	1000	35	811	33	1000	33	1000	35	612	33	1000	31	1000	29	1000	32	2000	27	2000	27
40	1000	34	1000	37	1000	33	1000	38	1000	33	349	31	228	31	712	31	2000	28	2000	28
41	1000	33	28	37	90	34	1000	34	1000	33	34	33	455	29	1000	30	2000	29	2000	29
42	1000	33	222	33	1000	34	1000	34	1000	31	627	33	1000	29	269	29	2000	28	2000	29
43	478	34	543	32	1000	34	812	33	1000	30	166	33	1000	30	234	30	2000	28	2000	29
44	758	32	780	34	795	34	1000	35	862	30	1000	31	255	29	1000	30	2000	28	2000	29
45	1000	31	1000	37	272	38	309	34	1000	31	1000	33	599	30	29	30	704	28	1882	28
46	404	31	705	33	1000	38	1000	38	1000	30	1000	33	1000	30	983	29	2000	29	2000	29
47	1000	31	1000	38	1000	37	1000	38	1000	31	392	34	1000	29	788	28	2000	28	2000	29
48	1000	32	33	34	1000	34	353	35	1000	31	1000	34	1000	28	1000	28	2000	29	2000	29
49	604	34	287	32	1000	37	1000	38	1000	31	1000	34	421	29	1000	28	2000	29	2000	29
50	1000	33	1000	34	1000	36	85	33	1000	31	43	31	588	28	1000	30	2000	27	2000	27

Figura C.7: Experimento para COTA 8

Bibliografía

[ASS/98] Assion, A., T. Baumert, M. Bergt, T. Brixner, B. Kiefer, V. Seyfried, M. Strehle y G. Gerber. (1998) Control of chemical reactions by feedback-optimized phase-shaped femtosecond laser pulses. Science, vol.282, p.919-922.

[CAR/05] Daniel D.Carpintero y Erika Gularte. (2005) Múltiples valores óptimos, MECOM 2005 VIII Congreso Argentino de Mecánica Computacional.

[GOL/89] Goldberg, D. (1989) Genetics Algorithms in Search, Optimization and Machine Learning. Addison Wesley.

[HAU/04] Haupt Randy y Haupt Sue Ellen. (2004) Practical genetic algorithm. John Wiley and Sons.

[HOL/75] John H. Holland. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press.

[KOZ/92] Koza, J. (1992) Genetic Programming. On the Programming of Computers by Means of Natural Selection. The MIT Press.

[LOB/01] Lobo, Jorge M. (2001) Prólogo de Metodos para medir la biodiversidad - M and T Manuales y Tesis SEA, vol. 1.

[MON/05] David Montana y Jason Redi. (2005) Optimizing parameters of a mobile ad hoc network protocol with a genetic algorithm. Genetic and Evolutionary Computation Conference, GECCO. Proceedings, Washington DC, USA.

[RUD/94] Rudolph, G. (1994) Convergence Analysis of Canonical Genetic Algorithms. IEEE Transactions on Neural Networks, volumen 5, pág 96-101.

[SID/01] A Sidaner, O Bailleux, JJ Chabrier. (2001). Measuring the spatial dispersion of evolutionist search processes: application to walksat. 5th International Conference EA, 2001.

[THO/06] David Thomas. (2006) War of the Weasels, An Evolutionary Algorithm Beats Intelligent Design. Volume 34, SKEPTICAL INQUIRER.

[WIL/01] Williams, Edwin, William Crossley y Thomas Lang. (2001) Average and maximum revisit time trade studies for satellite constellations using a multiobjective genetic algorithm. Journal of the Astronautical Sciences, vol.49, no.3, p.385-400.