ALGORITMOS GENETICOS

ÍNDICE

1. Introducción

1.1. Antecedentes

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad en todo el mundo durante los últimos años

Esta técnica se basa 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, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos más deseables (i.e., los que le permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente.

Un investigador de la Universidad de Michigan llamado John Holland era consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que permitió incorporarla a un programa. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente "planes reproductivos", pero se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su libro en 1975.

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

"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. "

1.2. Definición

Los Algoritmos Genéticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Un **algoritmo genético** consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuales de ellos deben generar descendencia para la nueva generación.

Versiones más complejas de algoritmos genéticos generan un ciclo iterativo que directamente toma a la especie (el total de los ejemplares) y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño. Una de sus características principales es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano, principal defecto de otros métodos para solucionar problemas, como los Sistemas Expertos.

1.3. Problemática

Los principios básicos de los Algoritmos Genéticos fueron establecidos por Holland, y se encuentran bien descritos en varios textos . Goldberg, Davis, Michalewicz, Reeves.

En la naturaleza los individuos de una población compiten entre sí en la búsqueda de recursos tales como comida, agua y refugio. Incluso los miembros de una misma especie compiten a menudo en la búsqueda de un compañero. Aquellos individuos que tienen más éxito en sobrevivir y en atraer compañeros tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario individuos poco dotados producirán un menor número de descendientes. Esto significa que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes ancestros, puede a veces producir descendientes "superindividuos", cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor ó puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos . descendientes de los anteriores . los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.

De esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior. Así a lo largo de las generaciones las buenas características se propagan a través de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las áreas más prometedoras del espacio de búsqueda. Si el Algoritmo Genético ha sido bien diseñado, la, población convergerá hacia una solución óptima del problema.

1.4. Ventajas y Desventajas

No necesitan conocimientos específicos sobre el problema que intentan resolver.

- Operan de forma simultánea con varias soluciones, en vez de trabajar de forma secuencial como las técnicas tradicionales.
- Cuando se usan para problemas de optimización maximizar una función objetivo- resultan menos afectados por los máximos locales (falsas soluciones) que las técnicas tradicionales.
- Resulta sumamente fácil ejecutarlos en las modernas arquitecturas masivamente paralelas.
- Usan operadores probabilísticos, en vez de los típicos operadores determinísticos de las otras técnicas.
- Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta medida de los parámetros que se utilicen tamaño de la población, número de generaciones, etc.-.
- Pueden converger prematuramente debido a una serie de problemas de diversa índole.

1.5. Limitaciones

El poder de los Algoritmos Genéticos proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el Algoritmo Genético encuentre la solución óptima, del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria. En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al Algoritmo Genético, tanto en rapidez como en eficacia. El gran campo de aplicación de los Algoritmos Genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los Algoritmos Genéticos.

1.6. Como Saber si es Posible usar un Algoritmo Genético

La aplicación más común de los algoritmos genéticos ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables. Sin embargo, no todos los problemas pudieran ser apropiados para la técnica, y se recomienda en general tomar en cuenta las siguientes características del mismo antes de intentar usarla:

- Su espacio de búsqueda (i.e., sus posibles soluciones) debe estar delimitado dentro de un cierto rango.
- Debe poderse definir una función de aptitud que nos indique qué tan buena o mala es una cierta respuesta.
- Las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar en la computadora.

El primer punto es muy importante, y lo más recomendable es intentar resolver problemas que tengan espacios de búsqueda discretos aunque éstos sean muy grandes. Sin embargo, también podrá intentarse usar la técnica con espacios de búsqueda continuos, pero preferentemente cuando exista un rango de soluciones relativamente pequeño.

La **función de aptitud** no es más que la función objetivo de nuestro problema de optimización. El algoritmo genético únicamente maximiza, pero la minimización puede realizarse fácilmente utilizando el recíproco de la función maximizante (debe cuidarse, por supuesto, que el recíproco de la función no genere una división por cero). Una característica que debe tener esta función es que tiene ser capaz de "castigar" a las malas soluciones, y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez.

La **codificación** más común de las soluciones es a través de cadenas binarias, aunque se han utilizado también números reales y letras. El primero de estos esquemas ha gozado de mucha popularidad debido a que es el que propuso originalmente Holland, y además porque resulta muy sencillo de implementar.

Los Algoritmos Genéticos resuelven los problemas generando poblaciones sucesivas a las que se aplican los operadores de mutación y cruce. Cada individuo representa una solución al problema, y se trata de encontrar al individuo que represente a la mejor solución.

La Programación Genética funciona igual que la técnica anterior pero se centra en el estudio de problemas cuya solución es un programa. De manera que los individuos de la población son programas que se acercan más o menos a realizar una tarea que es la solución.

La Programación Evolutiva es otro enfoque de los algoritmos genéticos, en este caso el estudio se centra en conseguir operadores genéticos que imiten lo mejor posible a la

naturaleza, en cada caso, más que en la relación de los padres con su descendencia. En este caso no se utiliza el operador de cruce, tomando la máxima importancia el operador de mutación.

Estrategias Evolutivas se centran en el estudio de problemas de optimización e incluyen una visión del aprendizaje en dos niveles: a nivel de genotipo, y a nivel de fenotipo. Y por último los Sistemas Clasificadores engloban el estudio de problemas en los que la solución buscada se corresponde con toda una población.

Para finalizar se muestra un esquema en el que se sitúan las técnicas mencionadas con respecto a otros procedimientos de búsqueda conocidos.

2.Extensiones y Modificaciones del Algoritmo Genético Simple

En este apartado se introducirán algunas extensiones y modificaciones del Algoritmo Genético Simple. Se comenzará dando un pseudocódigo para un Algoritmo Genético Abstracto (AGA), para a continuación introducir algunas variantes que se han ido proponiendo en trabajos desarrollados en estos últimos años.

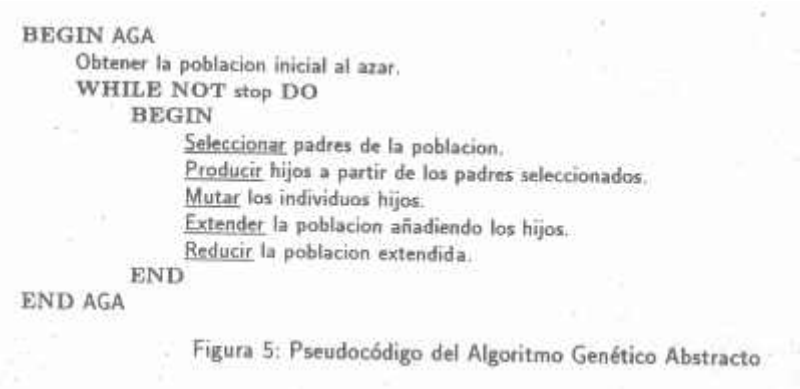


Figura 5: Pseudocódigo del Algoritmo Genético Abstracto

Figura 5

2.1. Población

2.1.1. Tamaño de la población

Una cuestión que uno puede plantearse es la relacionada con el tamaño idóneo de la población. Parece intuitivo que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que el trabajar con poblaciones de gran tamaño puede acarrear problemas relacionados con el excesivo costo computacional.

Goldberg efectuó un estudio teórico, obteniendo como conclusión que el tamaño óptimo de la población para ristras de longitud I, con codificación binaria, crece exponencialmente con el tamaño de la ristra.

Este resultado traería como consecuencia que la aplicabilidad de los Algoritmos Genéticos en problemas reales sería muy limitada, ya que resultarían no competitivos con otros métodos de optimización combinatoria. Alander, basándose en evidencia empírica sugiere que un tamaño de población comprendida entre l y 21 es suficiente para atacar con éxito los problemas por el considerados.

2.1.2. Población inicial

POBLACION

Población es el conjunto de elementos de referencia sobre el que se realizan las observaciones. Es necesario realizar un análisis de este concepto para el proceso del Algoritmo Genético, se consideran los siguientes puntos:

El tamaño de la población. Se pueden considerar dos factores.

- El primero es que si optamos por una población inicial pequeña puede que no abarquemos adecuadamente el espacio de búsqueda.
- El segundo es que si creamos una población muy extensa se pueden tener problemas en el excesivo costes computacionales. Acerca del tamaño de la población hay que decir que no existen reglas fijas, normalmente una población de 25 a 100 individuos es perfectamente válida para la mayoría de los casos.

Características de la población inicial. Ahora tenemos una problemática en cuanto a los valores de la población inicial. Esto es como se le asignaran los valores a la población inicial. Habitualmente la población inicial se escoge generando ristras al azar, pudiendo contener cada gen uno de los posibles valores del alfabeto con probabilidad uniforme. En algunas investigaciones donde los valores no son generados al azar se constata que se puede acelerar la convergencia del algoritmo genético, sin embargo en algunos casos es una desventaja ya que se llega a una convergencia prematura del Algoritmo.

2.3 Función de Evaluación.

Un buen diseño de la función de evaluación (también conocida como función objetivo o función de adaptación) resulta extremadamente importante para el correcto funcionamiento de un AG. Esta función determina el grado de adaptación o aproximación de cada individuo al problema y por lo tanto permite distinguir a los mejores individuos de los peores.

Dos aspectos que resultan cruciales en el comportamiento de los Algoritmos Genéticos son la determinación de una adecuada función de adaptación o función objetivo, así como la codificación utilizada

Por funcion entendemos una magnitud cuyo valor depende de una u otras variables , por ejemplo:

función objetivo f1(x1,x2,....xn), las variables estando sujetas a restricciones (f2(x1,x2,...xn)= 0 o f2(x1,x2,...xn) ≥ 0 ...)

2.4 Selección.

Existen distintos métodos para la selección de los padres que serán cruzados para la creación de nuevos individuos, uno de los más utilizados se denomina *función de selección proporcional a la función de evaluación*. Se basa en que la probabilidad de que un individuo sea seleccionado como padre es proporcional al valor de su función de evaluación. Esto hace que los mejor individuos sean los seleccionados para el proceso de reproducción. Esta técnica

puede producir el inconveniente de que la población converja prematuramente hacia un resultado óptimo local. Esto significa que pueden aparecer "superindividuos" muy similares entre si, de forma que la diversidad genética sea bastante pobre y el algoritmo se estanque en una solución buena (óptimo local) pero no la mejor.

Trataremos éste y otros problemas más adelante con mayor detalle. Otro método de selección es el de la *Ruleta*, consiste en asignar una porción de la "ruleta" a cada individuo de forma que el tamaño de cada porción sea proporcional a su *fitness*. Los mejores individuos dispondrán de una porción mayor y por lo tanto de más posibilidades de ser seleccionados. El método de *Torneo* consiste en hacer competir a los individuos en grupos aleatorios (normalmente parejas), el que tenga el *fitness* más elevado será el ganador. En el caso de competición por parejas se deben realizar dos torneos. Con este método de selección nos aseguramos de que al menos dos copias del mejor individuo de la población actuarán como progenitores para la siguiente generación. Este método tiene evidentes similitudes con el mundo animal, en donde los machos combaten por el control del grupo.

2.5 Cruzamiento

Consiste en el intercambio de material genético entre dos cromosomas (individuos). El *Cruzamiento* es el principal operador genético, hasta el punto que se puede decir que no es un algoritmo genético si no tiene *Cruzamiento*, y, sin embargo, puede serlo perfectamente sin mutación, según descubrió Holland. El *teorema de los esquemas* confía en él para hallar la mejor solución a un problema, combinando soluciones parciales.

Para aplicar el Cruzamiento, entrecruzamiento o recombinación, se escogen aleatoriamente dos miembros de la población. No pasa nada si se emparejan dos descendientes de los mismos padres; ello garantiza la perpetuación de un individuo con buena puntuación (y, además, algo parecido ocurre en la realidad; es una práctica utilizada, por ejemplo, en la cría de ganado, llamada *inbreeding*, y destinada a potenciar ciertas características frente a otras). Sin embargo, si esto sucede demasiado a menudo, puede crear problemas: toda la población puede aparecer dominada por los descendientes de algún gen, que, además, puede tener caracteres no deseados. Esto se suele denominar en otros métodos de optimización *atranque en un mínimo local*, y es uno de los principales problemas con los que se enfrentan los que aplican algoritmos genéticos.

En cuanto al teorema de los esquemas, se basa en la noción de *bloques de construcción*. Una buena solución a un problema está constituido por unos buenos bloques, igual que una buena máquina está hecha por buenas piezas. El Cruzamiento es el encargado de mezclar bloques buenos que se encuentren en los diversos progenitores, y que serán los que den a los mismos una buena puntuación. La presión selectiva se encarga de que sólo los buenos bloques se perpetúen, y poco a poco vayan formando una buena solución. El *teorema de los esquemas* viene a decir que la cantidad de *buenos bloques* se va incrementando con el tiempo de ejecución de un algoritmo genético, y es el resultado teórico más importante en algoritmos genéticos.

El intercambio genético se puede llevar a cabo de muchas formas, pero hay dos grupos principales

Cruzamiento por N-puntos:

Los dos cromosomas (individuos) se cortan por n puntos, y el material genético situado entre ellos se intercambia. Lo más habitual es un Cruzamiento de un punto o de dos puntos.

En el operador de cruce basado en dos puntos, los cromosomas (individuos) pueden contemplarse como un circuito en el cual se efectúa la selección aleatoria de dos puntos, tal y como se indica en la siguiente figura.

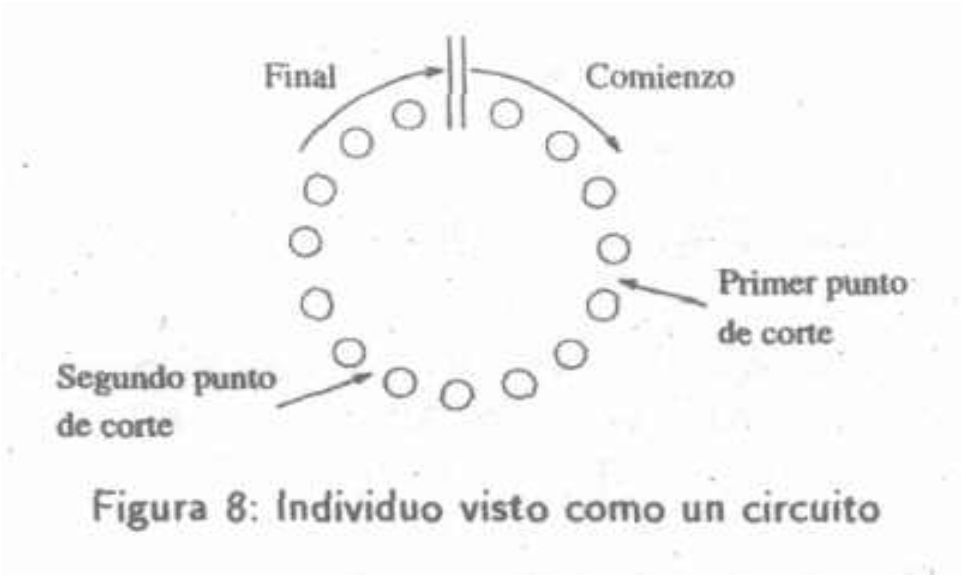


Figura 8: Individuo visto como un circuito

Ejemplo:

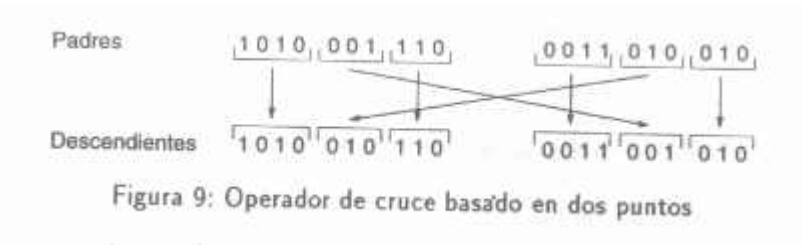


Figura 9: Operador de cruce basado en dos puntos

El cruce basado en dos puntos, representaba una mejora mientras que añadir más puntos de cruce no beneficiaba el comportamiento del algoritmo. La ventaja de tener más de un punto de cruce radica en que el espacio de búsqueda puede ser explorado más fácilmente, siendo la principal desventaja el hecho de aumentar la probabilidad de ruptura de buenos esquemas.

Cruzamiento uniforme:

Se genera un patrón aleatorio de 1s y 0s, y se intercambian los bits de los dos cromosomas (individuos) que coincidan donde hay un 1 en el patrón. O bien se genera un número aleatorio para cada bit, y si supera una determinada probabilidad se intercambia ese bit entre los dos cromosomas.

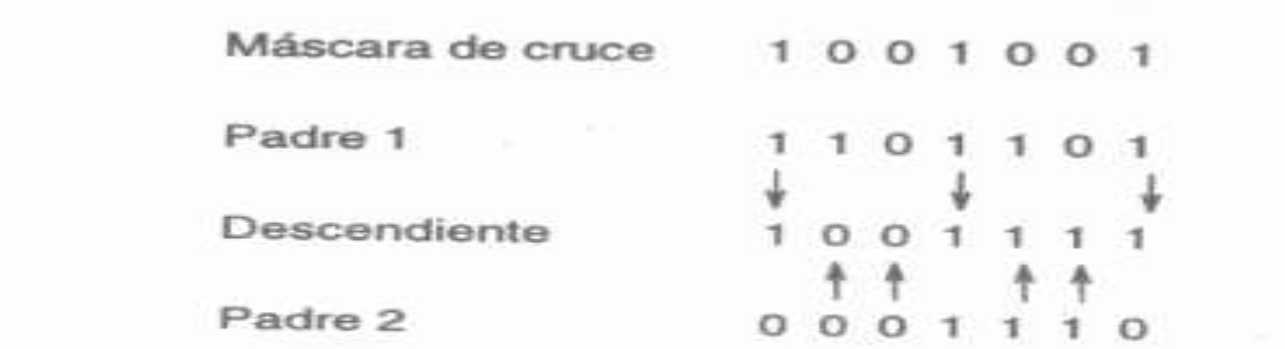
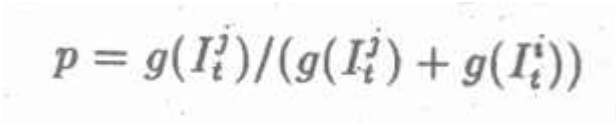


Figura 10: Operador de cruce uniforme

Hablando en términos de la teoría de la probabilidad la máscara de cruce está compuesta por una muestra aleatoria de tamaño A extraída de una distribución de probabilidad de Bernouilli de parámetro I/2.

Si tuviésemos en cuenta el valor de la función de adaptación de cada padre en el momento de generar la "máscara de cruce", de tal manera que cuanto mayor sea la función de adaptación de un individuo, más probable sea heredar sus características, podríamos definir un operador de cruce basado en la función objetivo, en el cual la "máscara de cruce" se interpreta como una muestra aleatoria de tamaño l proveniente de una distribución de Bernouilli de parámetro P obtenido de:



$$p = g(I_t^j)/(g(I_t^j) + g(I_t^i))$$

Función

Donde (I superíndice j subíndice t) y I(superíndice i subíndice t) denotan los padres seleccionados para ser cruzados.

2.6 Mutación

La mutación se considera un operador básico, que proporciona un pequeño elemento de aleatoriedad en la vecindad (entorno) de los individuos de la población. Si bien se admite que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, también parece desprenderse de los experimentos efectuados por varios investigadores que el operador de mutación va ganando en importancia a medida que la población de individuos va convergiendo (Davis).

La búsqueda del valor óptimo para la probabilidad de mutación, es una cuestión que ha sido motivo de varios trabajos. Así, De Jong recomienda la utilización de una probabilidad de mutación del bit de (l super -1), siendo l la longitud del string. Schaffer y col. utilizan resultados experimentales para estimar la tasa óptima proporcional a l /(lambda super 0.9318),(l super 0.4535), donde lambda denota el número de individuos en la población.

Una vez establecida la frecuencia de mutación, por ejemplo, uno por mil, se examina cada bit de cada cadena cuando se vaya a crear la nueva criatura a partir de sus padres (normalmente se hace de forma simultánea al Cruzamiento). Si un número generado aleatoriamente está por debajo de esa probabilidad, se cambiará el bit (es decir, de 0 a 1 o de 1 a 0). Si no, se dejará como está. Dependiendo del número de individuos que haya y del número de bits por individuo, puede resultar que las mutaciones sean extremadamente raras en una sola generación.

No hace falta decir que no conviene abusar de la mutación. Es cierto que es un mecanismo generador de diversidad, y, por tanto, la solución cuando un algoritmo genético está estancado, pero también es cierto que reduce el algoritmo genético a una búsqueda aleatoria. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población, o garantizar la aleatoriedad de la población inicial.

Si bien en la mayoría de las implementaciones de Algoritmos Genéticos se asume que tanto la probabilidad de cruce como la de mutación permanecen constantes, algunos autores han obtenido mejores resultados experimentales modificando la probabilidad de mutación a medida que aumenta el número de iteraciones. Pueden consultarse los trabajos de Ackley, Bramlette, Fogarty y Michalewicz y Janikow.

Este operador, junto con la anterior y el método de selección de ruleta, constituyen un *algoritmo genético simple,* sga, introducido por Goldberg en su libro.

2.7 Reducción

Una vez obtenidos los individuos descendientes de una determinada población en el tiempo t, el proceso de reducción al tamaño original, consiste en escoger lambda individuos de entre los Lambda individuos que forman parte de la población en el tiempo t, y los lambda individuos descendientes de los mismos. Dicho proceso se suele hacer fundamentalmente de dos formas distintas.

O bien los lambda individuos descendientes son los que forman parte de la población en el tiempo t + 1, es lo que se denomina reducción simple, o bien se escogen de entre los 2lambra individuos, los lambda individuos más adaptados al problema, siguiendo lo que podemos denominar un criterio de reducción elitista de grado lambda. Podemos también considerar otros procedimientos de reducción que se colocan entre los anteriores, por ejemplo, si escogemos los (lambda sub 1) mejores de entre padres y descendientes, escogiéndose los lambda . (lambda sub 1)y restantes de entre los descendientes no seleccionados hasta el momento.

El concepto de reducción está ligado con el de tasa de reemplazamiento generacional, (t sub rg) es decir en el porcentaje de hijos generados con respecto del tamaño de la, población.

Si bien en la idea primitiva de Holland [22] dicho reemplazamiento se efectuaba, de l en 1, es decir (t sub gr)=(lambda super -1), habitualmente dicho reemplazamiento se efectúa en bloque, (t sub gr)= 1. De Jong [13] introdujo el concepto de tasa de reemplazamiento generacional con el objetivo de efectuar un solapamiento controlado entre padres e hijos. En su trabajo, en cada paso una proporción, t,∼, de la población es seleccionada para ser cruzada. Los hijos resultantes podrán reemplazar a miembros de la población anterior. Este tipo de Algoritmos Genéticos se conocen bajo el nombre de SSGA (Steady State Genetie Algorithm), un ejemplo de los cuales lo constituye GENITOR (Whitley y Kauth, Whitley).

Michalewicz introduce un algoritmo que denomina Algoritmo Genético Modificado, (MOD sub GA), en el cual para llevar a cabo el reemplazamiento generacional, selecciona al azar r1 individuos para la reproducción, así como r2 individuos (distintos de los anteriores) destinados a morir. Estas selecciones aleatorias tienen en consideración el valor de la función objetivo de cada individuo, de tal manera que cuanto mayor es la función objetivo, mayor es la probabilidad de que sea seleccionado para la reproducción, y menor es la probabilidad de que dicho individuo fallezca. El resto de las lambdas. (r1 + r2) individuos son considerados como neutros y pasan directamente a formar parte de la población en la siguiente generación

Ejemplos de Aplicación

Ejemplo1

Veamos cómo funciona un algoritmo genético:

Vamos a partir de una función f(x) muy sencilla:

$$f(x)=x^2$$

Imagina que deseas encontrar el valor de x que hace que la función f(x) alcance su valor máximo, pero restringiendo a la variable x a tomar valores comprendidos entre 0 y 31. Aún más, a x sólo le vamos a permitir tomar valores enteros, es decir: 0,1,2,3,...,30, 31.

Obviamente el máximo se tiene para x = 31, donde f vale 961. No necesitamos saber algoritmos genéticos para resolver este problema, pero su sencillez hace que el algoritmo sea más fácil de comprender.

Lo primero que debemos hacer es encontrar una manera de codificar las posibles soluciones(posibles valores de x). Una manera de hacerlo es con la codificación binaria. Con esta codificación un posible valor de x es (0,1,0,1,1). ¿Cómo se interpreta esto? Muy sencillo: multiplica la última componente (un 1) por 1, la penúltima (un 1) por 2, la anterior (un 0) por 4, la segunda (un 1) por 8 y la primera(un 0) por 16 y a continuación haz la suma: 11. Observa que (0,0,0,0,0) equivale a x = 0 y que (1,1,1,1,1) equivale a x = 31.

A cada posible valor de la variable x en representación binaria le vamos a llamar individuo. Una colección de individuos constituye lo que se denomina población y el número de individuos que la componen es el tamaño de la población. Una vez que tenemos codificada la solución, debemos escoger un tamaño de población. Para este ejemplo ilustrativo vamos a escoger 6 individuos.

Debemos partir de una población inicial. Una manera de generarla es aleatoriamente: coge una moneda y lánzala al aire; si sale cara, la primera componente del primer individuo es un 0 y en caso contrario un 1. Repite el lanzamiento de la moneda y tendremos la segunda componente del primer individuo (un 0 sí sale cara y un 1 sí sale cruz). Así hasta 5 veces y obtendrás el primer individuo. Repite ahora la secuencia anterior para generar los individuos de la población restantes. En total tienes que lanzar 5 * 6 = 30 veces la moneda.

Nuestro siguiente paso es hacer competir a los individuos entre sí. Este proceso se conoce como selección. La tabla 1 resume el proceso.

Tabla 1.- SELECCION

(1)	(2)	(3)	(4)	(5)
1	(0,1,1,0,0)	12	144	6
2	(1,0,0,1,0)	18	324	3
3	(0,1,1,1,1)	15	225	2
4	(1,1,0,0,0)	24	576	5
5	(1,1,0,1,0)	26	676	4
6	(0,0,0,0,1)	1	1	1

Cada fila en la tabla 1 está asociada a un individuo de la población inicial. El significado de cada columna de la tabla es el siguiente:

(1) = Número que le asignamos al individuo.

(2)= Individuo en codificación binaria.

(3) = Valor de x.

(4) = Valor de f(x).

Observa que el mejor individuo es el 5 con f = 676. Calcula la media de f y obtendrás fmed=324.3. En cuanto a la columna (5) ahora te lo explico. Una manera de realizar el proceso de selección es mediante un torneo entre dos. A cada individuo de la población se le

asigna una pareja y entre ellos se establece un torneo: el mejor genera dos copias y el peor se desecha. La columna (5) indica la pareja asignada a cada individuo, lo cual se ha realizado aleatoriamente. Existen muchas variantes de este proceso de selección, aunque este método nos vale para ilustrar el ejemplo.

Después de realizar el proceso de selección, la población que tenemos es la mostrada en la columna (2) de la tabla 2. Observa, por ejemplo, que en el torneo entre el individuo 1 y el 6 de la población inicial, el primero de ellos ha recibido dos copias, mientras que el segundo cae en el olvido.

Tabla 2.- CRUCE

(1)	(2)	(3)	(4)
1	(0,1,1,0,0)	5	1
2	(0,1,1,0,0)	3	3
3	(1,0,0,1,0)	2	3
4	(1,0,0,1,0)	6	1
5	(1,1,0,1,0)	1	1
6	(1,1,0,1,0)	4	1

Tras realizar la selección, se realiza el cruce. Una manera de hacerlo es mediante el cruce 1X: se forman parejas entre los individuos aleatoriamente de forma similar a la selección. Dados dos individuos pareja se establece un punto de cruce aleatorio, que no es más que un número aleatorio entre 1 y 4 (la longitud del individuo menos 1). Por ejemplo, en la pareja 2-3 el punto de cruce es 3, lo que significa que un hijo de la pareja conserva los tres primeros bits del padre y hereda los dos últimos de la madre, mientras que el otro hijo de la pareja conserva los tres primeros bits de la madre y hereda los dos últimos del padre. La población resultante se muestra en la columna (2) de la tabla 3.

Tabla 3.- POBLACION TRAS EL CRUCE

(1)	(2)	(3)	(4)
1	(0,1,0,1,0)	10	100
2	(1,1,1,0,0)	28	784
3	(0,1,1,1,0)	14	196
4	(1,0,0,0,0)	16	256
5	(1,1,0,1,0)	26	676
6	(1,0,0,1,0)	18	324

En la columna (3) tienes el valor de x; en la siguiente tienes el valor de f correspondiente. Fíjate en que ahora el valor máximo de f es 784 (para el individuo 2), mientras que antes de la selección y el cruce era de 676. Además fmed ha subido de 324.3 a 389.3. ¿Qué quiere decir esto? Simplemente que los individuos después de la selección y el cruce son mejores que antes de estas transformaciones.

El siguiente paso es volver a realizar la selección y el cruce tomando como población inicial la de la tabla 3. Esta manera de proceder se repite tantas veces como número de iteraciones tú fijes. Y ¿cuál es el óptimo?. En realidad un algoritmo genético no te garantiza la obtención del óptimo pero, si está bien construido, te proporcionará una solución razonablemente buena. Puede que obtengas el óptimo, pero el algoritmo no te confirma que lo sea. Así que quédate con la mejor solución de la última iteración. También es buena idea ir guardando la mejor solución de todas las iteraciones anteriores y al final quedarte con la mejor solución de las exploradas.

Consideraciones adicionales

En problemas reales en los que se aplican los algoritmos genéticos, existe la tendencia a la homegeinización de la población, es decir a que todos los individuos de la misma sean idénticos. Esto impide que el algoritmo siga explorando nuevas soluciones, con lo que podemos quedar estancados en un mínimo local no muy bueno.

Existen técnicas para contrarrestar esta "deriva genética". El mecanismo más elemental, aunque no siempre suficientemente eficaz, es introducir una mutación tras la selección y el cruce. Una vez que has realizado la selección y el cruce escoges un número determinado de bits de la población y los alteras aleatoriamente. En nuestro ejemplo consiste simplemente en cambiar algunos(s) bit(s) de 1 a 0 ó de 0 a 1.

2. Ejemplo 2

Con la finalidad de aclarar mejor los conceptos cubiertos previamente, presentaremos ahora una sencilla aplicación del algoritmo genético a un problema de optimización:

Un grupo de financieros mexicanos ha resuelto invertir 10 millones de pesos en la nueva marca de vino "Carta Nueva". Así pues, en 4 ciudades de las principales de México se decide iniciar una vigorosa campaña comercial: México en el centro, Monterrey en el noroeste, Guadalajara en el occidente y Veracruz en el oriente. A esas 4 ciudades van a corresponder las zonas comerciales I, II, III y IV. Un estudio de mercado ha sido realizado en cada una de las zonas citadas y han sido establecidas curvas de ganancias medias, en millones de pesos, en función de las inversiones totales (almacenes, tiendas de venta, representantes, publicidad, etc.) Estos datos se ilustran en la tabla 2 y en la figura 4. Para simplificar los cálculos, supondremos que las asignaciones de créditos o de inversiones deben hacerse por unidades de 1 millón de pesos. La pregunta es: ¿en dónde se deben de asignar los 10 millones de pesos de los que se dispone para que la ganancia total sea máxima?

Inversión (en millones)	Beneficio I	Beneficio II	Beneficio III	Beneficio IV
0	0	0	0	0
1	0.28	0.25	0.15	0.20
2	0.45	0.41	0.25	0.33
3	0.65	0.55	0.40	0.42
4	0.78	0.65	0.50	0.48
5	0.90	0.75	0.62	0.53
6	1.02	0.80	0.73	0.56

7	1.13	0.85	0.82	0.58
8	1.23	0.88	0.90	0.60
9	1.32	0.90	0.96	0.60
10	1.38	0.90	1.00	0.60

Tabla 1. Datos obtenidos con la investigación de mercado en cada una de las regiones en estudio

1) **Representación:** Para poder aplicar el algoritmo genético, lo primero que necesitamos determinar es cuál será el esquema a utilizarse para representar las posibles soluciones del problema. En este caso necesitamos 4 bits (2^4 = 16) para representar cada solución, porque cada una admite 11 valores posibles (de 0 a 10). Como existen 4 valores independientes (uno por cada zona de estudio), se requieren entonces 16 bits (4 x 4) por cada cromosoma. Es importante hacer notar que se requiere una función de codificación (i.e., que transforme el valor de la inversión a binario) y una de decodificación (i.e., que realice el proceso inverso). Debido a que en este caso los 4 bits utilizados para representar una solución pueden producir más valores de los que se necesitan, se usará una función de ajuste que haga que los resultados producidos siempre se encuentren en el rango válido.

2) **Función de Aptitud:** Dado que el objetivo es obtener las inversiones que sumen 10, y que tengan un beneficio máximo, podemos usar la siguiente función de aptitud penalizada:

$$F(x)=\frac{c1+c2+c3+c4}{500*V+1}$$

donde c1, c2, c3 y c4 son las ganancias por zona, que se calculan de acuerdo a los valores de la tabla 2, y v es el valor absoluto de la diferencia entre la suma obtenida de las inversiones y 10. Nótese que cuando no se viole ninguna restricción (i.e., cuando la suma de inversiones sea exactamente 10) la función de aptitud no será "castigada".

3) **Operadores:** Se usará una cruza de 2 puntos. La probabilidad que se dará a la misma será del 80%. En cuanto a la mutación, se le asignará una probabilidad baja, que será del 1%. El tamaño de población manejado para este ejemplo será de 50 cromosomas, y se correrá el algoritmo genético durante 20 generaciones.

4) **Resultados:** El resultado obtenido en una corrida típica es un beneficio de 1.81 millones de pesos, correspondiente a invertir 4 millones en la zona comercial I, 3 millones en la zona II, 1 millón en la zona III y 2 millones en la zona IV. Esta es la solución óptima, la cual se obtuvo originalmente mediante programación dinámica. El tiempo que le tomó al algoritmo genético encontrar este valor fue de sólo 13 segundos. Debe hacerse notar que, en este caso, si deseáramos analizar inversiones que sumen otra cantidad, y en unidades menores al millón, el algoritmo genético tendría que modificarse de manera mínima, mientras que la programación dinámica requeriría una cantidad tal de trabajo que prácticamente se volvería inoperante.

Bibliografía

"Aplicaciones de la Computación Evolutiva"

http://www.geocities.com/SiliconValley/7491/aplce_c.htm

- "Evaluación del comportamiento de los Algoritmos Genéticos"

http://www.geocities.com/SiliconValley/9802/3d5ca400.htm

- "Introducción a la vida artificial"

http://içgeocities.com/CapeCanaveral/8104/ivan.htm

- "Construcción de bases de conocimiento con Computación Evolutiva"

http://www.fciencias.unam.mx/revista/soluciones/N17/Vlad1.html

- "Introducción a los algoritmos genéticos"

http://www.fciencias.unam.mx/revista/soluciones/Coello2.html

- "Los placeres existenciales de los algoritmos genéticos"

http://www.fciencias.unam.mx/revista/soluciones/N17/Gold3.html

- "Algoritmos Genéticos"

http://www.iamnet.com/users/jcontre/genetic/ag.htm

- "Operadores genéticos"

http://www.iamnet.com/users/jcontre/genetic/operadores.htm

- "Travelling Salesman problem using genetic algoritms"

http://www.lalena.com/ai/TSP

- "Algoritmos genéticos"

http://www.uv.es/~rmarti/genet.html

- "Introducción.(Tantas veces va el cántaro al agua hasta que se rompe)"

http://homepages.mty.itesm.mx/~lagrado/al-gen.htm

- "Algoritmos Genéticos"

http://www.iamnet.com/users/jcontre/genetic/ag.htm

- "MannaMouse"

http://www.caplet.com/MannaMouse.html#parameters

- "The flying circus" (contiene acceso al programa ejecutable Match)

http://www.wi.leidenuniv.nl/~gusz/Flying_Circus/index.html

- "The Hitch-Hiker. s guide to Evolutionary Computation"

http://gnomics.udg.es/~encore/www/top.htm

[1] Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

[2] Koza, J.R., "Genetic Programming. On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992, 819 p.

[3] Buckles, B.P., and Petry, F.E., "Genetic Algorithms", IEEE Computer Society Press, 1992, 109 p.

[4] Goldberg, D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison- Wesley Publishing Company, 1989, 412 p.

[5] Booker, L.B.; Goldberg, D.E., and Holland, J.H., "Classifier Systems and Genetic Algorithms", Artificial Intelligence, 40, 1989, pp. 235-282.

[6] Davis, L. (Editor), "Handbook of Genetic Algorithms", Van Nostrand Reinhold, 1991, 385 p.

[7] Forrest, S., "Genetic Algorithms: Principles of Natural Selection Applied to Computation", Science, Vol. 261, No. 5123, August 13, 1993, pp. 872-878.

[8] Smith, R.E.; Goldberg, D.E., and Earickson, J.A., "SGA-C : A C-language Implementation of a Simple Genetic Algorithm", TCGA Report No. 91002, The Clearinghouse for Genetic Algorithms, The University of Alabama, May 14, 1991.

[9] Ribeiro Filho, J.L.; Treleaven, Ph.C., and Alippi, C., "Genetic-Algorithm Programming Environments", IEEE Computer, June 1994, pp. 28-43.

[10] Srinivas, M., and Patnaik, L.M., "Genetic Algorithms : A Survey", IEEE Computer, June 1994, pp. 17-26.

[11] Rawlins, G.J.E. (Editor), "Foundations of Genetic Algorithms", Morgan Kaufmann Publishers, 1991, 341 p.

[12] Whitley, L.D. (Editor), "Foundations of Genetic Algorithms 2", Morgan Kaufmann Publishers, 1993, 322 p.

[13] Kaufmann, A., y Faure, R., "Invitación a la Investigación de Operaciones", Segunda Edición, CECSA, México, 1977, 311 p.

[14] Porter, K., "Handling Huge Arrays", Dr. Dobb's Journal of Software Tools for the Professional Programmer, Vol. 13, No. 3, 1988, pp. 60-3.

[15] Koza, J., "Genetic Programming II : Automatic Discovery of Reusable Programs", The MIT Press, 1992, 746 p.

[16] Francisco J. Varedas&Francisco J.Vico ,"Computación Evolutiva Basada en un modelo de codificaión implicita", Inteligencia Artificial, Nº 5, pag 20-25.