Capítulo 3

Programación Genética

En este capítulo se describe brevemente la teoría de la programación genética. Se expondrán las pequeñas pero importantes diferencias que existen con los algoritmos genéticos. Además de explicar su funcionamiento, servirá al lector para entrar en contacto con la técnica que realmente se ha utilizado en este proyecto para alcanzar su objetivo.

1. INTRODUCCIÓN

La programación genética es una extensión de los algoritmos genéticos convencionales vistos en el capítulo anterior, introducida por primera vez en el año 1992 por John R. Koza, profesor del departamento de ciencias de la computación en la Universidad de Stanford. En ella, los individuos sometidos al proceso evolutivo son ahora estructuras en forma de árbol, de tamaño y forma variable[1]. El nombre de *Programación Genética* se debe a que, en los primeros ensayos que se hicieron, las estructuras en

[1] En programación genética, se hará referencia a los mienbros de la población en términos de "individuos", "árboles" o "programas" indistintamente.

forma de árbol (o simplemente árboles) estaban representadas por pequeños programas informáticos.

La programación genética intenta afrontar una de las principales cuestiones planteadas en ciencias de la computación: ¿Cómo pueden aprender las computadoras a resolver problemas sin ser explícitamente programadas? En otras palabras ¿cómo se puede hacer que las computadoras realicen una determinada tarea, sin que se les diga explícitamente cómo hacerla? Este capítulo tratará de arrojar un poco de luz a estas cuestiones.

2. FUNDAMENTOS

El espacio de búsqueda en programación genética es el espacio de todos los posibles árboles compuestos de funciones y terminales adecuados al dominio del problema que se está considerando. Pero, ¿qué es todo esto de terminales y funciones?

2.1. Esquema de Representación de Individuos: El Árbol

Considérese el árbol representado en la figura 3.1.

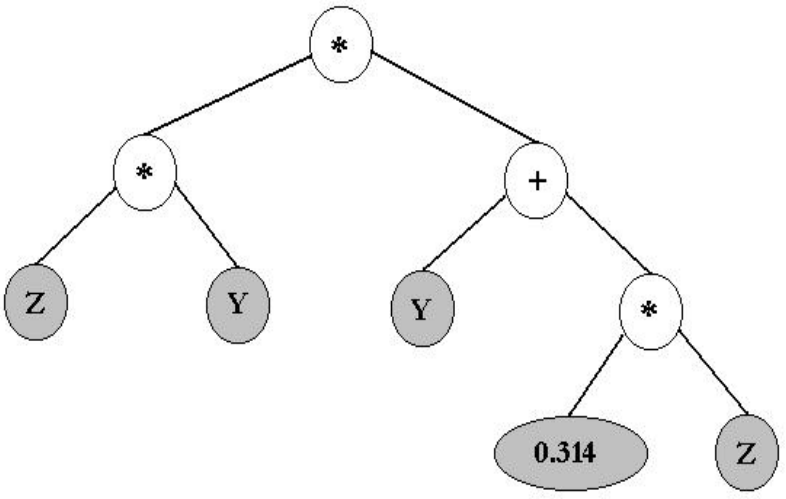


Figura 3.1 Individuo perteneciente al espacio de búsqueda en un problema de programación genética.

Todo árbol está constituido por dos tipos nodos, los *nodos terminales* y los *nodos no terminales* o conjunto de funciones primitivas. En la figura 31 los nodos terminales aparecen sombreados y de ellos nunca cuelga cualquier otro tipo de nodo o subárbol. Normalmente representan una constante numérica o una variable del problema. De los nodos no terminales (no

sombreados en la figura 3.1) siempre cuelga uno o varios subárboles. Éstos ejercen como argumentos de los primeros, ya que este tipo de nodos hacen las veces de funciones (matemáticas o de cualquier otro tipo). Por ejemplo, la figura 3.1 representa la expresión matemática `ZY(Y + 0.314Z)`.

2.2. Pasos Preparativos

Para poder aplicar programación genética a la resolución de un determinado problema, es necesario dar una serie de pasos previos. Estos pasos, cinco en concreto, implican la determinación de:

1) El conjunto de nodos terminales, *T*

2) El conjunto de funciones primitivas, *F*

3) La medida del Grado de aptitud

4) Parámetros de control de la ejecución del programa genético

5) Método para designar un resultado final y criterio de terminación de la ejecución

El primer paso consiste en identificar el conjunto de nodos terminales que se van a usar. Esta elección dependerá de la naturaleza del problema y se debe tener en cuenta que dichos nodos actuarán como argumentos de las funciones definidas en el conjunto de funciones primitivas. Como se dijo en la sección anterior, es habitual que los terminales consistan en constantes numéricas o variables que puedan tomar una serie de valores dentro de un rango definido.

El segundo paso en la fase previa a la utilización de un programa genético, trata de definir el conjunto de funciones primitivas *F*. Las funciones pueden ser simples operaciones aritméticas; operaciones típicas en programación como un `IF`; funciones matemáticas como seno o coseno; funciones lógicas como OR o AND, o funciones específicas para el problema bajo estudio. Estas funciones deben devolver siempre algún valor.

Cada árbol representa una expresión matemática, la cuál estará compuesta por elementos de los conjuntos *T* y *F*. Se debe dar siempre una condición para que un problema pueda ser resuelto mediante programación genética y es que los nodos terminales junto con las funciones primitivas deben satisfacer el *principio de suficiencia* en el sentido de que, combinados de forma adecuada, deben ser capaces de expresar una solución al problema.

Además, cada función definida en el conjunto *F* debería ser capaz de aceptar como argumentos, cualquier valor devuelto por cualquiera otra función de *F* o por cualquier nodo terminal del conjunto *T*. Si ambos

conjuntos cumplen este principio, se dice que se satisface la *condición de cierre*[2].

Si se establece una analogía con los algoritmos genéticos, estos dos primeros pasos se corresponden con la determinación del esquema de representación de individuos. Los otros tres pasos son exactamente iguales.

El proceso evolutivo está dirigido por la medida del grado de aptitud, que evalúa cómo de bien representa un determinado árbol una solución al problema considerado. La medida del nivel de aptitud debería estar completamente definida en el sentido de que debe ser capaz de evaluar a cualquier árbol que encuentre, en cualquier generación de la población.

Los principales parámetros que controlan la ejecución de un programa genético, al igual que en un algoritmo genético, son el tamaño de la población M, y el número máximo de generaciones G, que van a transcurrir. Además existen una serie de parámetros secundarios como pueden ser probabilidades de ocurrencia de un determinado operador genético o el método de selección de individuos que se va utilizar.

Cada ejecución de un programa genético necesita la especificación de un criterio de terminación que decida cuando se detiene el programa, y un método para designar el resultado o solución que se propone al problema. Normalmente se toma como resultado el individuo con el nivel de aptitud más elevado de entre todas las generaciones que han sucedido.

Una vez que se han dado estos pasos preliminares, se está en condiciones de ejecutar el programa genético.

3. LA APTITUD

El proceso de medida del nivel de aptitud va a depender de la naturaleza del problema que se esté tratando. En algunos problemas, el grado de aptitud de un árbol se podrá medir como el error entre el resultado producido por el árbol (se recuerda que un árbol no es más que un programa que devuelve un valor) y el resultado correcto. Cuanto más cercano a cero esté dicho error, mejor será el individuo analizado. Normalmente, para medir este error no se evalúa el individuo una sola vez sino que se hace para un conjunto de casos posibles y luego se calcula un promedio. Por ejemplo, considérese el individuo de la figura 3.1; se podría evaluar para un conjunto de combinaciones posibles de las variables Z e Y (conjunto que habitualmente se denomina *casos de prueba*) y determinar su aptitud como un promedio calculado sobre los casos de prueba. Los pares de valores Z e Y harían las veces de *datos de entrada* del proceso encargado de evaluar al individuo, que posteriormente produciría un valor de salida[3]. Los

[2] En la literatura anglosajona se denomina Closure.

[3] Que en caso del individuo de la figura 3.1 sería el valor numérico de la expresión matemática que representa.

casos de prueba se pueden elegir de forma aleatoria sobre el rango de valores permitidos de las variables independientes o bien de forma estructurada (por ejemplo a intervalos regulares sobre el rango permitido).

En otra clase de problemas, el grado de aptitud no se calcula directamente del valor devuelto en la evaluación del árbol sino que se determina a partir de las consecuencias de la ejecución del programa (que representa al individuo). Por ejemplo, en un problema de control óptimo, el valor devuelto por el controlador afecta al estado del sistema. En este caso, el grado de aptitud de cada individuo (un posible controlador) se basa en la cantidad de tempo (combustible, distancia, dinero, etc.) que se tarda en llevar al sistema a un estado deseado. Cuanto menor sea esta cantidad de tiempo (combustible, distancia, dinero, etc.) mejor aptitud tendrá dicho controlador. Los casos de prueba en problemas de control, consisten, a menudo, en diferentes condiciones iniciales del sistema.

Para problemas que implican la realización de una determinada tarea, la aptitud de un individuo se puede medir en términos de puntos conseguidos (ya sean trabajos correctamente culminados o casos manejados de forma adecuada).

Si uno está intentando reconocer patrones o clasificar objetos, el nivel de aptitud del programa se podría medir en términos del número de instancias tratadas correctamente (por ejemplo calcular el número de verdaderos positivos y negativos producidos) y del número de instancias tratadas incorrectamente (por ejemplo considerar el número de falsos positivos y falsos negativos).

En todos los ejemplos anteriores, el nivel de aptitud se calculaba explícitamente. Sin embargo, la aptitud se puede determinar implícitamente permitiendo a los individuos interactuar (habitualmente en una simulación) con su entorno o entre ellos mismos, donde ciertos comportamientos conducirán a la supervivencia (y en consecuencia a la reproducción y la recombinación con otros seres) y otros no lo harán.

4. OPERADORES EN PROGRAMACIÓN GENÉTICA

Como ya se ha visto, una diferencia sustancial entre los algoritmos genéticos y la programación genética está en la forma de representar a los individuos de la población. Mientras que en los algoritmos genéticos se tienen cadenas de caracteres de longitud fija, en programación genética se trata con programas informáticos de estructura jerárquica (estructuras en forma de árbol) de forma y tamaño variable. Esta diferencia hace que la manera de trabajar de algunos operadores genéticos, como el cruce o la mutación, varíe respecto a los algoritmos genéticos. A continuación se estudian los tres operadores básicos en programación genética: la reproducción, el cruce y la mutación.

4.1. La Reproducción

En el caso de la reproducción no se produce ningún cambio respecto a su homólogo en los algoritmos genéticos. Esta operación implica, en primer lugar, la selección de un árbol de entre todos los que conforman la población basándose en su aptitud. Cuanto más apto sea un individuo, mayor probabilidad de ser seleccionado tendrá. Por último, lo único que hay que hacer es copiar el programa seleccionado diréctamente en la población correspondiente a la siguiente generación, permitiendo así su supervivencia.

La operación se ilustra en la figura 3.2 en la que, arriba aparece el árbol seleccionado, perteneciente a la generación K; en la parte de abajo aparece el individuo reproducido, perteneciente a la generación K+1.

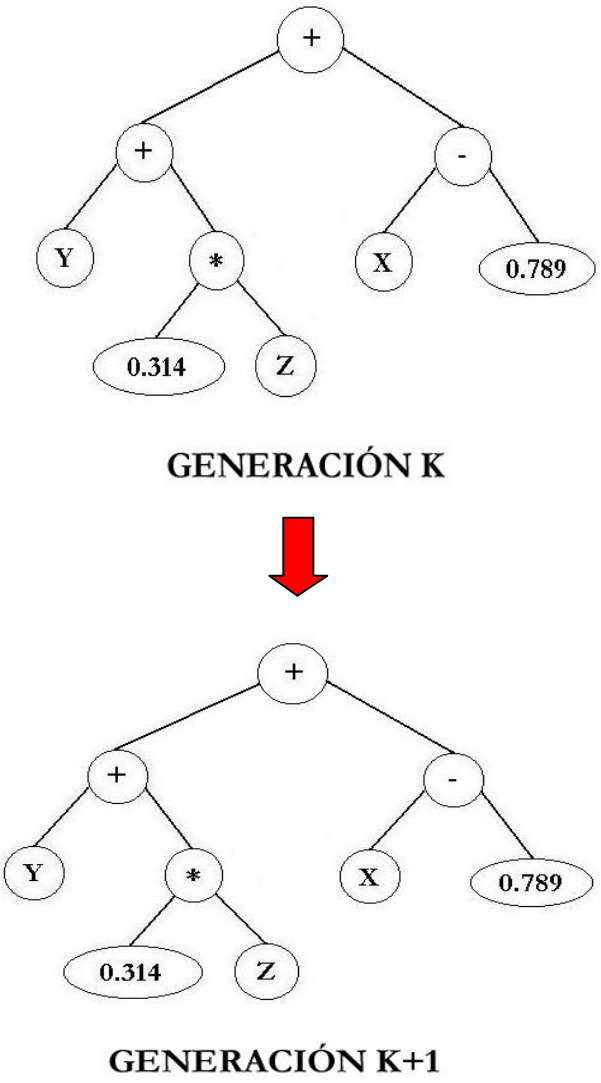


Figura 3.2 Individuo reproducido de la generación K-ésima a la K+1-ésima

4.2. El Cruce

El operador de cruce o recombinación actúa (al igual que en los algoritmos genéticos) sobre dos árboles seleccionados sobre la base de su aptitud, para crear dos nuevos descendientes. Habitualmente, los padres diferirán el uno del otro en tamaño y forma. Los hijos estarán formados por subárboles de cada uno de sus padres y, normalmente, tendrán tamaños y formas distintas de las de sus progenitores. Si dos programas representan de algún modo una buena solución a un determinado problema, entonces algunos de sus bloques constituyentes (subárboles) tendrán algún mérito en ello. Recombinando aleatoriamente estos bloques, se pueden formar nuevas estructuras que incluso mejoren la solución que aportaban sus predecesores.

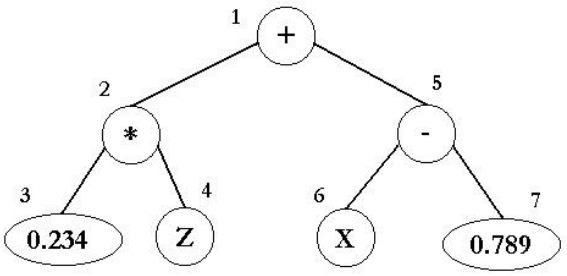
Por ejemplo, considérese el siguiente programa (mostrado aquí como una expresión en lenguaje LISP):

```
(+ (* 0.234 Z) (- X 0.789))
```

Expresión que normalmente escribiríamos como:

```
0.234Z + X – 0.789
```

El árbol que representa esta expresión se muestra en la figura 3.3.



0.234Z + X - 0.789

Figura 3.3 Individuo que actuará como uno de los padres en la operación de cruce

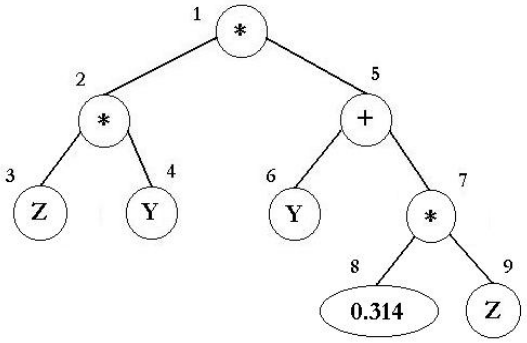
Por otro lado, se presenta un segundo programa dado por la expresión en lenguaje LISP:

```
(* (* Z Y) (+ Y (* 0.314 Z)))
```

Instancia que equivale a escribir:

```
ZY(Y + 0.314Z)
```

Y cuyo árbol se ilustra en la figura 3.4.

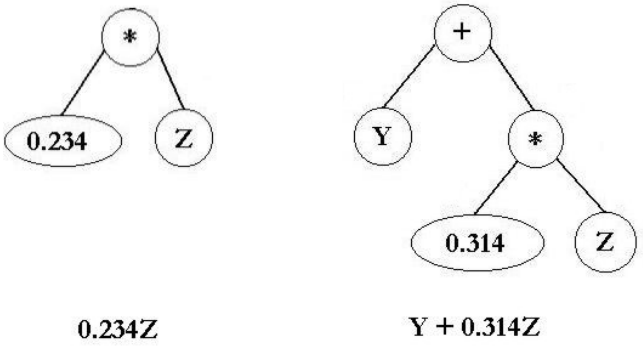


ZY (Y + 0.314Z)

Figura 3.4 Árbol necesario para realizar la operación de cruce

En ambas figuras se han numerado los nodos de los árboles sólo por comodidad, de arriba hacia abajo y de izquierda a derecha. Los nodos internos del árbol se corresponden con las funciones primitivas (en este caso operaciones aritméticas) y los nodos externos (también deneminados hojas) con los terminales.

Como se ha mencionado anteriormente, la operación de recombinación consiste en intercambiar subárboles entre los padres que intervienen en la misma. Estos subárboles son elegidos aleatoriamente. Por ejemplo, suponer que se elige de forma totalmente aleatoria en el individuo de la figura 3.3, el subárbol que cuelga del nodo 2. Del mismo modo, se toma de la figura 3.4 el subárbol que cuelga del nodo 5. Estos puntos de ruptura se denominan puntos de cruce, y los fragmentos seleccionados se muestran en la figura 3.5.



0.234Z Y + 0.314Z

Figura 3.5 Subárboles que van a ser intercambiados en el cruce

Si ahora se realiza el intercambio, se insertaría el fragmento de la izquierda de la figura 3.5, en el árbol de la figura 3.4. Por otro lado, el subárbol de la derecha formaría parte ahora del individuo de la figura 3.3. El resultado son

dos nuevos descendientes, completamente diferentes de sus padres (aunque formados a partir de su material genético) que se ilustran en la figura 3.6.

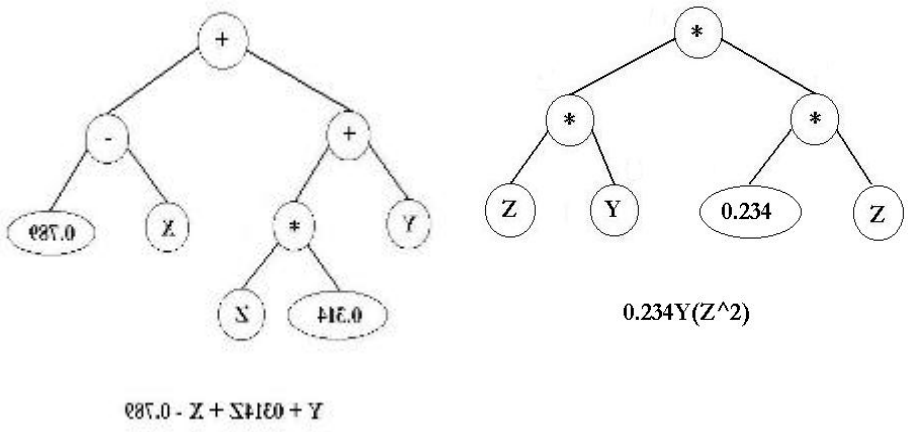


Figura 3.6 Descendientes, producto de la recombinación genética de dos individuos.

Los programas correspondientes a los nuevos árboles (siguiendo con lenguaje LISP) serían para el individuo de la izquierda:

```
(+ (+ Y (* 0.314 Z)) (- X 0.789))
```

y para el de la derecha:

```
(* (* Z Y) (* 0.234 Z))
```

Debido al intercambio de bloques (subárboles) entre individuos que han sido seleccionados en función de su aptitud, el operador de cruce va guiando la búsqueda hacia zonas del espacio de búsqueda donde se encuentran soluciones cada vez mejores.

4.3. La Mutación

De lo visto en la sección anterior se puede intuir la forma de trabajar del operador mutación. En primer lugar se selecciona un programa de entre todos los que forma la población de individuos. Esta elección puede ser totalmente aleatoria o estar basada en el grado de aptitud de los miembros de la población. Por ejemplo, se supone que el individuo seleccionado es el mostrado en la figura 3.4.

A continuación se elige un subárbol perteneciente al individuo seleccionado anteriormente, de forma totalmente aleatoria, y se elimina. Siguiendo con el ejemplo de la figura 3.4, supongamos que se toma el subárbol que cuelga del nodo 4 (que en este caso consiste de un solo nodo). En su lugar, se inserta un nuevo subárbol generado de forma arbitraria. En el caso considerado, se va a introducir el subárbol mostrado en la figura 3.7.

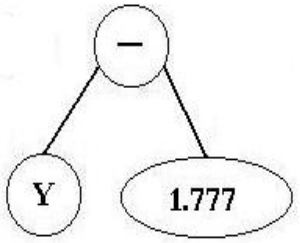
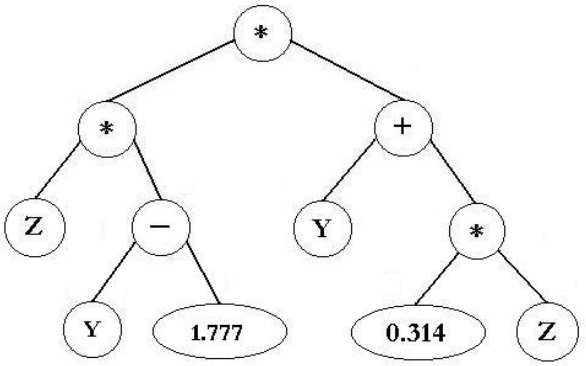


Figura 3.7 Nuevo subárbol introducido en la operación de mutación realizada sobre el programa de la figura 3.4. El punto de mutación elegido es el nodo 4.

El resultado de la operación es un nuevo individuo, con parte de su material genético procedente de su antecesor y otra parte completamente nueva, que se ilustra en la figura 3.8.



$$Z(Y - 1.777)(Y + 0.314Z)$$

Figura 3.8 Resultado de la operación de mutación realizada sobre el individuo de la figura 3.4

La mutación puede resultar útil a la hora de restablecer la diversidad genética de la población en el caso de pérdida de la misma debido a la convergencia prematura, pero es un operador utilizado con muy poca frecuencia.

5. FUNCIONAMIENTO

En programación genética, poblaciones de miles de programas informáticos son sometidas a un proceso evolutivo. El proceso por el cual una población evoluciona hacia otra mediante la aplicación del principio de supervivencia del más apto y de los operadores genéticos, se denomina *breeding*. Como se verá más adelante, un programa que represente una solución aproximada a un determinado problema podría evolucionar hasta convertirse en una solución óptima a dicho problema. Y todo ello, aplicando sucesívamente este proceso de breeding a través de una serie de generaciones.

Un programa genético comienza con una población inicial de programas generados aleatoriamente y compuestos de funciones primitivas y nodos terminales adecuados al dominio del problema que se está analizando. Esta primera población podría ser vista como una búsqueda "ciega" en el espacio de posibles soluciones. Pero el hecho de trabajar con poblaciones de incluso miles de individuos, otorga a la programación genética un elevado grado de paralelismo.

Una vez que se aplica el breeding sobre la población inicial, la población de descendientes reemplaza a la primera. Entonces, cada individuo se somete al proceso de evaluación de su aptitud y a continuación se repite el breeding sucesívamente sobre varias generaciones. Se puede observar como el nivel de aptitud promedio de la población va creciendo a medida que se van sucediendo las generaciones, es decir, debido al proceso evolutivo se van consiguiendo soluciones cada vez mejores para el problema bajo estudio.

En resumen, un programa genético hace evolucionar poblaciones de individuos para encontrar una solución a un problema, ejecutando estos tres pasos:

1) Crear aleatoriamente una población inicial de combinaciones de funciones primitivas y terminales, definidas para el problema (programas informáticos).

2) Realizar de forma iterativa los siguientes pasos sobre la población de inviduos hasta que se satisfaga el criterio de terminación:

   (a) Ejecutar cada programa de la población y asignarle una puntuación haciendo uso del proceso de medida del nivel de aptitud.

   (b) Crear una nueva población de programas aplicando los tres siguientes operadores genéticos. Las operaciones genéticas se aplican sobre individuos de la población que son seleccionados con una probabilidad que se basa en su puntuación o aptitud (un individuo puede resultar seleccionado más de una vez).

      i. Reproducir un individuo copiando directamente su programa en la nueva población.

ii. Crear dos nuevos individuos a partir de otros dos, aplicando la recombinación genética de subárboles de los árboles originales, tomando como referencia un punto de cruce aleatoriamente elegido.

iii. Crear un nuevo programa a partir de otro existente, eligiendo una posición al azar de este último y mutando el subárbol seleccionado.

3) Proporcionar el programa que ha sido identificado por el método de designación de resultados, como la solución de la ejecución del programa genético. Este individuo puede representar una solución (o una solución aproximada) a nuestro problema original.

El carácter jerárquico de los programas que forman la población, es una importante característica de la programación genética. Los resultados de la programación genética son inherentemente jerárquicos. Por ejemplo se pueden tener una serie de tareas a realizar en un determinado orden prioritario, o bien jerarquías en las que un cierto comportamiento implica o descarta otros tipos de comportamientos. También se puede dar el caso de tener expresiones matemáticas complejas, que siempre podrían ser representadas por estructuras en forma de árbol.

El dinamismo con que varían los programas que forman una determinada población, a lo largo de la ejecución de un programa genético, es otra característica importante de la programación genética. Aquellos pueden variar tanto en forma como en tamaño, lo que favorece la diversidad genética de las poblaciones.

Por último destacar el carácter activo de los miembros de las poblaciones. Éstos son programas informáticos que pueden ser ejecutados en sí mismos y producir resultados. Ahora no se manejan estructuras pasivas como eran las cadenas de caracteres en los algoritmos genéticos.

La figura 3.9 ilustra el diagrama de flujo que implementaría un programa genético. Run indica el número de ejecuciones del programa genético y N el número máximo de ejecuciones permitidas. La variable Gen se refiere a la generación actual y M es el tamaño de la población. El índice i denota al individuo considerado de entre todos los que forman la población. La suma de la probabilidad de que se utilice el operador reprodución p_r, la probabilidad del cruce p_c y la probabilidad de mutación p_m debe ser igual a uno.

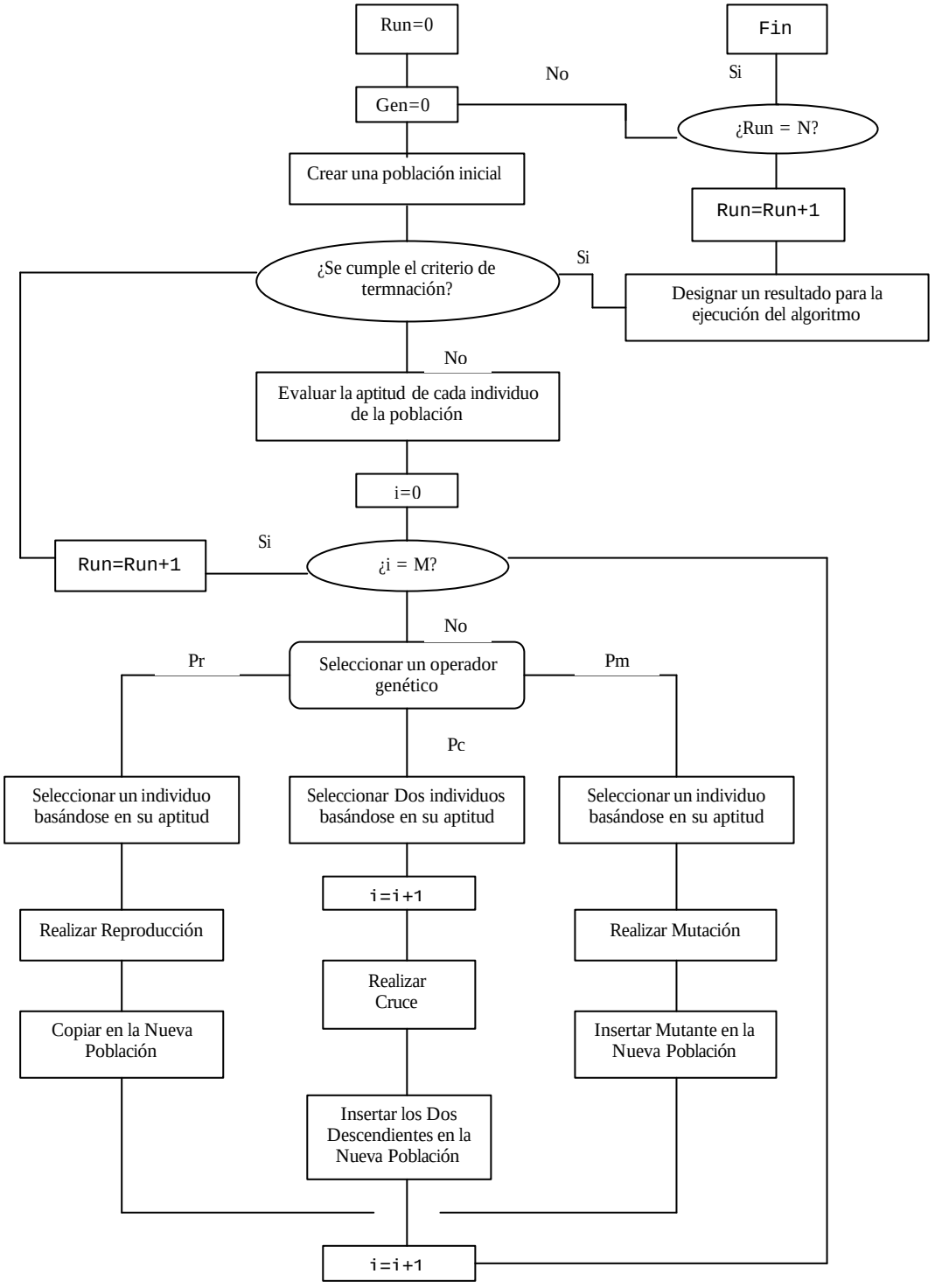


Figura 3.9 Diagrama de flujo de un programa genético convencional