(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 217 308**

(51) Int. Cl.[7]: **G06F 15/18**

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Número de solicitud europea: **96908579 .4**
(86) Fecha de presentación: **01.03.1996**
(87) Número de publicación de la solicitud: **0898750**
(87) Fecha de publicación de la solicitud: **03.03.1999**

(54) Título: **Método y sistema para programación genética.**

(73) Titular/es: **Worzel, William P.**
214 W. Main Street
Milan, Michigan 48160, US

(45) Fecha de publicación de la mención BOPI:
01.11.2004

(72) Inventor/es: **Worzel, William P.**

(45) Fecha de la publicación del folleto de la patente:
01.11.2004

(74) Agente: **Díez de Rivera y Elzaburu, Ignacio**

DESCRIPCIÓN

Método y sistema para programación genética.

Campo del invento

El presente invento se refiere a sistemas informáticos para descubrir programas eficaces a través de técnicas de programación genética. Más concretamente, el invento combina algoritmos genéticos con técnicas de reducción de grafos para producir un sistema informático que permita la evolución eficaz de programas para resolver problemas cuya entrada es conocida y que tienen una función para determinar si un programa es mejor que otro.

Antecedentes del invento

La mayoría de lo ordenadores que se construyen hoy en día son de un tipo conocido como el de máquinas de von Neuman. Estos ordenadores requieren la transferencia de datos desde posiciones de la memoria, usualmente identificadas por alguna dirección o etiqueta, a una unidad de procesador central (CPU) donde se aplica alguna operación a los datos. Frecuentemente, los datos son movidos de nuevo, ya sea de vuelta a su posición original o ya sea a otra posición en la memoria. Este movimiento constante de los datos es ineficaz y limita la capacidad de construir una máquina verdaderamente paralela.

El cálculo lambda representa otro modo de manipular información para obtener los resultados deseados. En el cálculo lambda, se crean "expresiones lambda" especiales para describir un sistema que enlaza íntimamente los datos con las funciones. Usando estas expresiones lambda, se puede construir una máquina de programación universal que produzca resultados equivalentes a los producidos por las máquinas de von Neuman.

En un artículo publicado por D. A. Turner ("A New Implementation Technique for Applicative Languages" ("Una Nueva Técnica de Ejecución para Lenguajes de Aplicación"), *Software - Practice and Experience,* Vol. 9, nº 1, págs. 31-49, 1979) se describe un método para construir un sistema informático mediante la abstracción de expresiones lambda a una forma más simple y más potente, denominada de combinadores. Los combinados fueron descritos por primera vez por Schonfinkel y Curry en los años 30, pero Turner fue el primero en proponerlos como un modo de construir un sistema informático. A continuación, Clarke, Gladstone, MacLean y Norman describieron en detalle un equipo físico de cálculo que podía ser construido para facilitar la puesta en práctica de tales sistemas. Desde entonces se ha escrito información adicional acerca de combinadores, incluida la patente de EE.UU. Nº 4.734.848 concedida a Yamano y otros, en que se describen equipos físicos de combinadores en paralelo.

Los combinadores son un ejemplo de un sistema de reducción de grafos. Es decir, un sistema donde los programas estén representados por listas de elementos que son transformados por un conjunto de operadores de reducción de grafos. Aunque este invento está enfocado hacia los combinadores, debido a su compacidad y demostrada universalidad, el invento abarca el uso de todos y cada uno de los sistemas de reducción de grafos, incluidos aquellos que trabajan sobre cadenas que representan grafos.

Los combinadores K, S, I, B, C y W están definidos por las siguientes funciones, e ilustrados gráficamente en la Figura 1. En cada caso, f, g, x e y pueden ser expresiones de funciones, operadores, constantes o combinadores.

$$Kxy = x$$

$$Sfgx = fx(gx)$$

$$Ix = x$$

$$Bfgx = f(gx)$$

$$Cfgx = (fx)g$$

$$Wfx = (fx)x$$

Si se contemplan los programas como árboles de expresiones donde cada rama del árbol representa una parte separada del programa. los combinadores alteran directamente los árboles de programas, para cambiar su estructura, cambiando por ello los resultados de la evaluación del árbol. En los sistemas informáticos de combinadores se emplea usualmente la abstracción de horquilla para eliminar las referencias a variables: es decir, los datos de entrada que cambian cada vez que se ejecuta el programa. El método de la abstracción de horquilla produce una expresión de combinador puro que no tiene referencia alguna a los datos de entrada. Por el contrario, los combinadores son aplicados directamente a los datos de entrada dispuestos como un grafo o árbol.

John Holland describió la idea de usar la genética como modelo para resolver problemas informáticos, en su monografía de 1975 titulada *Adaptation in natural and Artificial Systems* (Adaptación en Sistemas naturales y Artificiales). Holland sugería que a partir de varios conjuntos de valores iniciales elegidos aleatoriamente, con tal solamente de que haya un modo de determinar si un valor es "mejor" (tal como se defina en el modo de un problema específico) que otro, se puede obtener una solución, combinando para ello los "genes" que esos valores representan, a través de un mecanismo modelado después de la evolución.

Estos "algoritmos genéticos" proporcionan un método eficaz para converger sobre valores deseados en un gran universo de posibles valores. Si se busca un valor superior entre un gran número de posibles valores, y si los valores pueden ser caracterizados de alguna forma regular, tal como mediante una cadena binaria, es entonces posible moverse rápida y eficazmente hacia la mejor solución seleccionando, haciendo coincidir y, ocasionalmente, mutando para ello esas cadenas de un grupo relativamente pequeño de candidatos, con tal solamente de que sea posible asignar un valor que representa la calidad de un valor cuando se compara con otros valores.

Los algoritmos genéticos se han demostrado usualmente en programas que buscan un valor máximo u óptimo de una función de medición conocida, tal como la de encontrar los máximos o los mínimos de la función. Por ejemplo, el problema del "viajante de ventas", consistente en buscar la ruta de viaje más eficaz entre un número muy grande de posibles rutas, suele usarse frecuentemente para hacer una demostración de los algoritmos genéticos.

Los anteriores intentos hechos de crear programes usando técnicas de algoritmo genético han incluido la combinación de algoritmos genéticos con sistemas basados en reglas para proporcionar un método para construir sistemas complejos para tomar acciones. Esto se puede considerar como una programación por coincidencia de valores binarios con reglas y hacien-

do luego evolucionar los valores para hallar la mejor combinación de reglas para resolver el problema. Esto se ha descrito en la patente de EE.UU. nº 4.697.242 de Holland y Burks, en 1987.

En otro intento anterior, se aplicaron los principios de los algoritmos genéticos a la programación, describiendo para ello programas LISP (de Proceso de Listas) y trozos de programa como elementos de genes potenciales y desarrollando después programas LISP más complejos, a partir de esos elementos, usando las reglas de los algoritmos genéticos. Esto se ha descrito tanto en obras sobre programación genética de J. R. Koza, como en sus varias patentes de programación genética (patentes de EE.UU. Números 4.935.877, 5.136.686 y 5.343.554). En estas patentes se sugiere que las cadenas de genes pueden se de longitud variable y que pueden ser relacionadas con expresiones de ordenador mediante la asignación uno a uno de los genes a fragmentos de programa. Por consiguiente, los sistemas creados por Koza requieren la asignación de los valores de genes usados en un algoritmo genético a expresiones de LISP, con objeto de crear nuevos programas.

Sumario del invento

En el presente invento, tal como se expone en las reivindicaciones 1 y 3 que se acompañan, se describe un sistema de ordenador apto para ser construido con equipo físico diseñado para ese fin, o bien puesto en práctica como una máquina d estado del software, y el correspondiente método, en el que se hace uso de la programación genética y de las técnicas de reducción de gráficos para crear programas que puedan resolver problemas informáticos. Esto se consigue usando para ello un programación genética para producir sucesivamente programas más eficaces, representados en forma de cadenas de operadores de reducción de grafos.

Este invento abarca el uso de cualquier sistema de reducción de grafos como método fundamental para representación del programa. Un sistema de reducción de grafos es uno en el que el programa está representado por una serie de operadores que transforman o reducen una representación de grafos (o lista) de datos o d componentes del programa. Esto incluye los grafos que están representados como cadenas y los sistemas de reducción de grafos basados en las manipulaciones de las cadenas. Debido a su concisión, universalidad y simplicidad, se presenta aquí un sistema de reducción de grafos basado en combinadores. El programa que se esté evaluando se representa como una sola cadena de genes, sin que se requiera asignación o correlación alguna con el presente invento. La evaluación de cada cadena de programas se efectúa aplicando la cadena de genes del programa a datos de entrada y midiendo la "calidad" o "aptitud" relativa de la salida resultante.

El presente invento difiere de los anteriores intentos de construir máquinas de programación genética en que las propias cadenas de genes del programa son programas. En varios intentos se han correlacionado o asignado las cadenas de genes a fragmentos del programa. Esta correlación de las cadenas de genes lleva tiempo, debido a que requiere continuas transferencias de datos a y desde varias posiciones de la memoria, a través del proceso de evolución, con objeto de ensamblar los programas candidatos para prueba. Mediante el uso de combinadores, el presente invento elimina la necesidad de correlacionar las cadenas de genes y las repetidas transferencias de datos asociadas con tal correlación. Por consiguiente, el presente sistema de programación genética es más rápido que otros sistemas de programación genética en los que se emplea la correlación de datos y variables.

Breve descripción de los dibujos

La Figura 1 es una ilustración gráfica de las funciones realizadas por diversos combinadores;

La Figura 2 es un diagrama bloque de una máquina combinadora usada para cargar y evaluar cadenas de combinadores de acuerdo con el presente invento;

La Figura 3 es un diagrama bloque de un sistema de programación genética que tiene múltiples máquinas combinadoras dispuestas en paralelo;

La Figura 4 es un organigrama en el que se ha ilustrado el proceso de evolución utilizado por el sistema de programación genética del invento;

La Figura 5 es un organigrama en el que se ha ilustrado el método usado para entrar los datos de entrada deseados;

La Figura 6 es un organigrama en el que se ha ilustrado el método usado para crear el grupo inicial de cadenas de genes;

La Figura 7 es una representación gráfica de un conjunto de datos de entrada, que se muestran en una estructura de árbol; y

La Figura 8 es un diagrama bloque de un ordenador para fines generales, apto para funcionar como un sistema de programación genética.

Descripción detallada del invento

En la Figura 2, un bus principal 30 proporciona un camino para el flujo de datos entre varias posiciones de la memoria y los demás componentes del sistema. Un bus de control 32 interconecta los diversos componentes de la máquina combinadora y hace posible el flujo de instrucciones de control entre componentes. Una Unidad de Evaluación de Combinador/Controlador (CEU/Controlador) 34 está acoplada eléctricamente a ambos buses 30 y 32. La CEU/Controlador 34 evalúa las cadenas de genes del programa de combinadores. Una memoria de microcódigo 36 es una Memoria de Sólo Lectura (ROM), la cual contiene descripciones para evaluar combinadores individuales. La memoria 36 de microcódigo proporciona las necesarias reglas e instrucciones para evaluar las diversas partes de las cadenas de genes del programa de combinadores.

Además de evaluar combinadores, la CEU/Controlador 34 es capaz de recoger varios elementos de la cadena de genes que esté siendo evaluada. La CEU/Controlador 34 controla también el procesador de I/O (Entrada/Salida) (IOP) 38 y la unidad lógica aritmética (ALU) 40, que están ambos acoplados eléctricamente tanto al bus principal 30 como al bus d control 32. El bus de control 32 es usado por la CEU/Controlador 34 para controlar el funcionamiento del IOP 38 y de la ALU 40. El IOP 38 proporciona una interfaz entre la máquina combinadora y otras partes del sistema de programación genética. La ALU 40 se usa para evaluar simples expresiones aritméticas, tales como la de 2 + 3, y pruebas tales como 3-4.

Una memoria principal, 42 y una memoria de pila 44 están ambas conectadas al bus principal 30. La memoria principal 42 contiene la cadena de genes de programas que está siendo evaluada por la CEU/Controlador 34, así como los datos de entrada a los cuales se ha aplicado el programa. La memoria de pila 44 almacena los estados intermedios y los va-

lores generados como resultado de la evaluación de la cadena de genes del programa. Por ejemplo, la memoria de pila 44 puede almacenar un indicador que recuerda donde fue detenida una evaluación mientras que se había completado una sub-evaluación.

Aunque en la Figura 2 se ha ilustrado una realización de una máquina combinadora individual, esta misma arquitectura puede también ponerse en práctica como una máquina de estado de software que funciona sobre equipo físico (hardware) en un sistema de ordenador tradicional. Usando una ejecución de software, la unidad de procesador central (CPU) del sistema realiza las funciones de la ALU 40 representada en la Figura 2. La CEU/Controlador 34 y la ALU 40 están sustituidas por programas separados en una biblioteca de software, a la cual se invoca cuando sea necesario. La memoria principal 42 y la memoria de pila 44 están realizadas como bloques de memoria dentro de la RAM (Memoria de Acceso Directo) del sistema, y el sistema de procesado de I/O (Entrada/Salida) del sistema sustituye al IOP 38 para la entrada y la salida de expresiones de combinadores y de datos. Se usa un programa evaluador, en vez del microcódigo 36, el cual recupera sucesivamente elementos de la cadena de genes particular que esté siendo evaluada, y llama a la apropiada subrutina de biblioteca de combinadores para ejecutar la evaluación.

Debido a la naturaleza intrínsecamente paralela de los combinadores y de la programación genética, el presente invento es adecuado para uso en sistemas de procesado en paralelo. En la Figura 3 se ha ilustrado un diagrama bloque de un sistema de programación genética en el que se utilizan varias máquinas combinadoras dispuestas en paralelo. Múltiples máquinas combinadoras 50 están dispuestas en una forma en paralelo. Cada máquina combinadora 50 puede ser materializada en ya sea equipo físico (como se ha ilustrado en la Figura 2), o ya sea en software, como se ha descrito en lo que antecede.

Se pueden disponer en paralelo cualquier número de máquinas combinadoras 50; El número de máquinas combinadoras depende de una diversidad de factores. En primer lugar, las limitaciones de coste pueden restringir el número de máquinas combinadoras usadas. Cuando mayor sea el número de máquinas combinadoras usadas, tanto mayor será el coste total del sistema.

Los requisitos de velocidad pueden también imponer el número de máquinas combinadoras usadas en un sistema particular. Un sistema que requiera una evaluación rápida requiere un mayor número de máquinas combinadoras que un sistema de más bajos requisitos de velocidad. Además, la complejidad del problema que haya que resolver afecta al número de máquinas combinadoras requeridas en un sistema particular. Para los problemas relativamente simples se requerirán en general un menor número de máquinas combinadoras que para los problemas complejos.

Finalmente, el número de máquinas combinadoras usadas en un sistema de programación genética varía dependiendo del número de cadenas de genes que estén siendo evaluadas en una generación particular. Un sistema que necesite evaluar muchas cadenas de genes en cada generación se beneficiará de un mayor número de máquinas combinadoras. Por ejemplo, un sistema de programación genética que evalúe 100 cadenas de genes en cada generación puede tener 100 máquinas combinadoras; es decir, una máquina combinadora para evaluar cada cadena de genes. En tal sistema, las 100 cadenas de genes son evaluadas simultáneamente, lo que da por resultado un funcionamiento más rápido de conjunto del sistema.

Como se ha ilustrado en la Figura 3, cada máquina combinadora 50 está conectada a un bus 46 de datos/control y a un bus 48 de cadenas de genes de programa. Un controlador de programación genética (controlador GP) 52 está conectado al bus 46 de datos/control. El controlador GP 52 controla el funcionamiento de conjunto del sistema de programación genética distribuyendo para ello el trabajo de evaluación entre las máquinas combinadoras 50, recuperando los resultados de las máquinas combinadoras, y desarrollando las diversas cadenas de genes.

En la Figura 8 se ha ilustrado un diagrama bloque de un ordenador para fines generales, apto para ser usado con el presente invento. Un bus 10 interconecta los diversos componentes del sistema y proporciona un camino común para el flujo de datos, instrucciones, y similares. Una unidad de procesador central (CPU) 12 está conectada al bus 10 y efectúa las operaciones de cálculo reales. Una memoria de acceso directo 14 está también conectada al bus 10 y proporciona una posición para almacenar datos y otra información. Un dispositivo de almacenamiento de datos 16 está conectado al bus 10 y proporciona almacenamiento no volátil de la información. El dispositivo de almacenamiento de datos 10 puede ser una unidad de disco, una unidad de cinta, o dispositivos de almacenamiento similares.

Un dispositivo de entrada 18 está conectado al bus 10 y permite al usuario de un ordenador dar entrada a datos, órdenes, y demás información, al sistema de ordenador. El dispositivo de entrada 18 puede ser un teclado, un escáner óptico, un micrófono, u otro dispositivo capaz de generar una señal legible por la máquina para el sistema de ordenador. El dispositivo de entrada usado varía dependiendo de la aplicación particular. Si se usa un sistema particular para reconocimiento de escritura a mano, el dispositivo de entrada debe ser capaz de leer o digitalizar la escritura a mano de una persona. En esa situación, el dispositivo de entrada puede ser un escáner óptico, una tableta para escritura sensible a la presión, un lápiz óptico con un lector óptico, u otro dispositivo similar.

En ciertas situaciones, se requieren múltiples dispositivos de entrada. Por ejemplo, un sistema de programación genética usado en un sistema de reconocimiento de voz requiere un micrófono o dispositivo similar para dar entrada a un patrón de voz particular, así como un teclado para dar entrada a diversos parámetros definidos por el usuario.

Con referencia de nuevo a la Figura 8, un dispositivo de presentación 20 está conectado, a través del bus 10, a los demás componentes del sistema. El dispositivo de presentación 20 es un monitor de vídeo usado para presentar parámetros definidos por el usuario, estado de la operación del programa, y resultados del programa. Un dispositivo de salida 33 está también conectado al bus 10 y proporciona una copia permanente de los resultados del programa. Preferiblemente, el dispositivo de salida 22 es una impresora.

En la Figura 4 se ha ilustrado el funcionamiento de conjunto del proceso de evolución utilizado por el sistema de programación genética. Como se ha ilustrado en el paso 58, antes de que el sistema de programa-

ción genética pueda empezar a desarrollar cadenas de genes, se han de proporcionar al sistema varios parámetros y datos.

Con referencia a la Figura 5, en el paso 86 un usuario entra el número de cadenas de genes a ser incluidas en el grupo de genes. A continuación, en el paso 88, el usuario entra una longitud o rango preferido de longitudes para las cadenas de genes iniciales. La longitud de una cadena de genes se refiere al número de genes contenidos en la cadena. En el paso 90, el usuario entra una frecuencia de mutación de las cadenas de genes. En el paso 91, el usuario entra la frecuencia de permutación de las cadenas de genes. En el paso 92 entra la información relativa al régimen de coincidencia de las cadenas de genes.

Todos los pasos 86-92 de entrad de datos pueden ser entrados por el usuario del sistema de programación genética, como se ha descrito en lo que antecede. Una vez entrados, los datos son almacenados en un dispositivo de memoria, tal como una memoria GP (de Programación Genética) 54 descrita en lo que antecede con respecto a la Figura 3. Alternativamente, cualquiera de los datos entrados en los pasos 86-92 puede ser almacenado permanentemente en el dispositivo de memoria y, por lo tanto, no es necesario que sea entrado por el usuario. Otra alternativa proporciona valores de parámetros por defecto. Si el usuario no entra datos para un paso particular, se usa el valor por defecto. Finalmente, los datos entrados en los pasos 86-92 pueden ser determinados aleatoriamente por el sistema de programación genética dentro de un rango especificado de valores para un parámetro dado. Por ejemplo, la probabilidad de mutación puede elegirse aleatoriamente entre los valores de 0,001 y 0,01, la longitud de las cadenas de genes iniciales puede seleccionarse entre 40 y 100 caracteres en longitud.

Como se ha ilustrado en la Figura 5, en el paso 94 el usuario entra los datos de entrada que hayan de ser aplicados a cada cadena de genes del programa. Como se vio en lo que antecede, los datos de entrada variarán dependiendo del problema que haya que resolver. Estos datos de entrada son almacenados en la memoria para uso por el sistema de programación genética.

En el paso 95, el usuario entra todos los valores que hayan de ser usados como constantes. Como se verá en lo que sigue, en la representación de las constantes en la creación de cadenas de genes, las constantes pueden ser valores de un solo carácter, cadenas de caracteres, o números. Esta lista de constantes entradas en este paso se usará en la construcción de las cadenas de genes del programa.

En el paso 96, el usuario entra una función de aptitud que es también almacenada en la memoria. Se usa la función de aptitud para determinar si un resultado es mejor que otro. En lo que sigue se consideran detalles adicionales relativos a la selección y aplicación de las funciones de aptitud.

En el paso 98, se entra un conjunto de criterios de terminación por el usuario. Estos criterios de terminación se usan para determinar cuándo deberá terminarse la evaluación de una cadena de genes particular. En lo que sigue se dan detalles adicionales relativos a la determinación de los criterios de terminación.

Con referencia de nuevo a la Figura 4, en el paso 59 se crea un grupo de genes inicial. El grupo de genes está constituido por cadenas de genes del programa, construidas usando combinadores. Este paso requiere la creación aleatoria de cadenas de combinadores, operadores, y constantes.

Con referencia a la Figura 6, el primer paso para crear un grupo de genes inicial comporta recuperar los parámetros relativos al tamaño del grupo de genes inicial, en el paso 100. El tamaño del grupo de genes inicial se determina usualmente por parte del usuario sobre la base de la estimación de la complejidad del programa y de la necesidad de variedad dentro del grupo de candidatos, pero puede también determinarse usando para ello un valor por defecto o bien un valor generado aleatoriamente. En el paso 102, la rutina determina si el grupo de genes está completo; es decir, si el número de cadenas de genes en el grupo es igual al tamaño del grupo, determinado en el paso 100. Inicialmente, el grupo de genes está vacío, y la rutina bifurca al paso 104, donde se determina la longitud de la siguiente cadena de genes. La longitud de la cadena de genes puede determinarse aleatoriamente, o bien puede ser seleccionada por el usuario, como se ha descrito en lo que antecede. Preferiblemente, se usa un rango de longitudes a fin de crear una gran diversidad de candidatos para el grupo inicial. Por ejemplo, la primera cadena puede tener una longitud de 30 elementos, la siguiente de 43, la siguiente de 22, etc.

En el paso 106, la rutina determina si la cadena de genes es suficientemente larga; es decir, si la longitud de la cadena de genes es igual a la longitud determinada en el paso 104. Inicialmente, la creación de la cadena de genes no ha empezado, y la rutina bifurca al paso 108, donde se selecciona aleatoriamente un gen. El gen seleccionado es o bien un combinador, o una constante, o un operador. Se hace la selección mediante elección aleatoria ponderada, dándose preferencia a los combinadores.

Si en el paso 110 se selecciona un combinador, se añade el combinador al final de la cadena de genes en el paso 118. Si el combinador es el primer gen en la cadena, el mismo forma el punto de partida para construir la cadena de genes.

Si en el paso 112 se selecciona una constante, se añade la constante al final de la cadena de genes en el paso 118. Al igual que con el combinador anterior, si la constante es el primer gen de la cadena, la misma forma el punto de partida para construir la cadena de genes.

La representación de las constantes en la cadena de genes del programa es una aplicación específica, y sin embargo en general debe incluir un modo de diferenciar claramente entre operadores y constantes. Por ejemplo, si el valor de la constante de la letra “C” es parte de las constantes incluidas en las cadenas de genes (por ejemplo, para reconocimiento de escritura a mano), debe ser claramente diferente del combinador “C”.

Hay muchos modos de hacer esto. En la ejecución preferida se hace uso de los convenios de algunos lenguajes de programación, donde se define una constante de carácter como un carácter entre comillas sencillas (por ejemplo, “C”), mientras que el combinador es un carácter C “desnudo” (por ejemplo, C). Análogamente, las cadenas de caracteres que forman las constantes se escriben entre comillas dobles (por ejemplo, “Fred”), mientras que los números son caracteres desnudos en el rango numérico 0 - 9) con una coma de decimales opcional y un carácter de signo opcional (por ejemplo, -3,14159). En el caso de cadenas, las mismas están realmente incluidas den-

tro de las cadenas de genes de programa como árboles secundarios de constantes de carácter (por ejemplo (“F” “r” “e” “d”)) pero pueden ser entradas en las listas de posibles constantes con la notación entre comillas dobles (“Fred”).

Como una alternativa a incluir explícitamente constantes en la cadena de genes del programa, el usuario puede simplemente dar entrada a aquellos valores que parezcan útiles, como parte de los datos a los que se hayan de aplicar las cadenas de genes del programa. Por ejemplo, un programa que esté destinado a calcular valores matemáticos podría usar de modo útil constantes matemáticas tales como las de pi, e, etc. Un programa destinado a analizar sustancias químicas podría tener constantes químicas, etc. Por el hecho de incluir tales constantes como una rama de los datos a los que haya de ser aplicada la cadena de genes del programa, la hipótesis es que la cadena de genes del programa puede desarrollar operaciones que seleccionen esas constantes y las usen en la medida en que sean necesarias para tratar de producir un programa útil. Con este enfoque es posible simplificar las cadenas de genes del programa, de modo que las mismas contengan solamente combinadores y operadores, aunque la realización preferida es la que se ha descrito en relación con la Figura 6, donde las constantes están incluidas directamente en la cadena de genes del programa.

Después de que se hayan añadido los genes a la cadena de genes en el paso 118, la rutina bifurca al paso 106 para determinar si la cadena es suficientemente larga. Los genes se añaden repetidamente a la cadena de genes hasta que se obtenga la longitud deseada. Cuando se haya completado la cadena de genes, el programa bifurca al paso 107, donde se equilibran los paréntesis en las expresiones generadas.

Esto es necesario debido a que los pares de paréntesis, tomados juntos, constituyen un operador que describe la estructura del árbol de una cadena de genes. Por ejemplo, la cadena de genes ((2 3 + K)(4 7 *) 2 1) tiene dos árboles secundarios en la misma, consistentes en las cadenas 2 3 * y 4 7 *, respectivamente, en ramas separadas del árbol, junto con las dos ramas que contienen únicamente el número simple 2 y 1.

Cuando se genera una cadena de genes, el carácter de paréntesis de apertura (“(”) y el carácter de cierre (“)”) se tratan como elecciones separadas en la lista de posibles entradas en una cadena de genes, y por lo tanto al final de paso 106 la cadena generada debe ser examinada, y si hay paréntesis de apertura extra, se añaden suficientes paréntesis de cierre al final de la cadena, para equilibrar el número de los de apertura, y se cierra la cadena. Análogamente, si hay más caracteres de paréntesis de cierre, se añaden caracteres de paréntesis de apertura al principio. Aunque hay otros modos de asegurar este equilibrio, tal como el de añadir tanto un par de apertura como un par de cierre en posiciones aleatorias a la cadena, al mismo tiempo, éste se considera un método superior para añadir la necesaria estructura a las expresiones.

Después del paso 107, el sistema retorna al paso 102 para determinar si el grupo de genes está completo. Si lo está, se completa entonces la creación del grupo de genes inicial. De lo contrario, se crea otra cadena de genes con la que se siguen los pasos descritos en lo que antecede.

Con referencia de nuevo a la Figura 4, después de crear el grupo de genes inicial de cadenas de combinadores, se evalúa la generación inicial aplicando para ello cada miembro a los datos de entrada (paso 60) y evaluando la aptitud (paso 62), aplicando para ello la función de aptitud a los resultados obtenidos en el paso 60.

En el paso 60, la aplicación de los datos de entrada se hace aplicando los datos de entrada a cada cadena de genes del grupo, para obtener un resultado. Inicialmente, la primera generación de cadenas de genes estará compuesta por entero por genes generados aleatoriamente. En las posteriores generaciones, el grupo está constituido por las cadenas de genes que han evolucionado desde las generaciones anteriores. Los datos de entrada suministrados por el usuario son aplicados a cada cadena, para permitir la reducción de combinadores en los datos, mediante una o más máquinas combinadoras. La reducción de combinadores crea un valor o un conjunto de valores.

En un ejemplo sencillo, si la cadena de genes del programa es una expresión de combinador para elevar al cuadrado un número, y se aplica la cadena de programas a la entrada 5, el resultado sería de 25.

En este paso, es necesario considerar lo que ocurre cuando se aplican los operadores a diferentes tipos de datos. Por ejemplo, se ha de efectuar una determinación acerca de cómo ha de evaluarse una cadena de combinadores, tal como “A+3” mediante el SISTEMA DE PROGRAMACIÓN GENÉTICA. Preferiblemente, la definición del operador “+” se extiende para producir un resultado de “D” (otras tres vetas más hacia delante a partir de la “A”). Usando esta definición, se debe definir la función de los operadores a través de todos los tipos de datos definidos en el sistema. Por lo tanto, si se aplica un operador a una lista de muchos elementos, deberá aplicarse la operación uniformemente a través de toda la lista. Por ejemplo, “3 + (1 2 3)” produciría “(4 5 6)” (añadiendo 3 a cada número de la lista).

En una realización alternativa, una expresión tal como “A+3” puede ser tratada como una combinación ilegal, y por consiguiente rechazada, cuando se evalúe la cadena de genes del programa pero se considera que la extensión a los operadores anteriormente sugerida es una solución superior.

En cualquier momento en la aplicación de una cadena de genes a datos de entrada puede producirse un error del programa. En tal ocasión, se aborta la aplicación y se asigna al programa una calificación muy baja de aptitud. Los errores pueden tener lugar por una diversidad de causas, pero la más común es la de que se ha intentado una operación imposible. Puesto que las cadenas de genes están cambiando constantemente debido a la combinación de cadenas, es posible que se produzcan expresiones ilegales, tales como “+ * 3”. Estas expresiones originarán errores cuando se pase el programa, y tales expresiones se eliminan asignando para ello calificaciones bajas de aptitud en el paso 62, donde se valora la aptitud de la cadena del programa.

Después de aplicar los datos de entrada a las cadenas de genes, el sistema continúa en el paso 62, donde s comparan los resultados obtenidos con la salida deseada, usando la función de aptitud. En base a la evaluación de las cadenas de genes, se asigna a cada cadena de genes un valor de su aptitud. El valor de aptitud representa la similitud entre la salida obtenida desde la cadena de genes y el resultado deseado. Preferiblemente, el valor de la aptitud está representado por un valor numérico. La función de la aptitud pro-

porciona una medida objetiva de "cómo es de bueno" el programa, cuando se aplica a los datos de entrada. Esta comparación de la aptitud se aplica a todas las cadenas de genes del programa de una generación y permite la clasificación de todas las cadenas de genes relativamente entre sí. Las cadenas de genes del programa que tengan un valor de aptitud más alto cuando se evalúen mediante la función de aptitud son clasificadas en posiciones superiores a las que tienen valores más bajos. Inicialmente, la mayor parte de las cadenas de genes generadas aleatoriamente tendrán un bajo valor de la aptitud, como resultado de la naturaleza aleatoria del proceso de selección inicial. Sin embargo, a medida que se van creando generaciones sucesivas, el mecanismo de evolución desarrolla cadenas de genes con valores de la aptitud cada vez más altos.

Las funciones de aptitud son específicas para el problema y deben ser suministradas por el usuario, dado que el sistema no puede saber qué es un buen resultado para un problema particular. Por ejemplo, los datos de entrada para un sistema de reconocimiento de manuscritos pueden ser una muestra de un manuscrito digitalizado, a la cual se aplican las cadenas de genes. La salida deseada es la cadena real de caracteres contenidos en la muestra de manuscrito. Cada expresión de combinador se aplica a los datos de entrada para crear y dar salida, la cual se compara con la cadena esperada de caracteres. En este ejemplo, el valor de la aptitud puede determinarse a partir del número de caracteres identificados correctamente, situados en la posición correcta. Dependiendo de los requisitos y de las preferencias del usuario, el orden apropiado de las letras puede ser más importante que la identificación de cada carácter. Por otra parte, un usuario diferente del mismo sistema podría poner más énfasis en identificar correctamente cada carácter, en vez de ordenar apropiadamente cada carácter identificado.

Inicialmente, los resultados serán deficientes, y pueden incluir algunas salidas que no tengan caracteres correctos de ningún tipo, solamente redistribuciones de puntos. Las cadenas de genes cuyos procesos de evaluación produzcan caracteres o cadenas de caracteres tendrán valores de la aptitud más altos que aquéllas cuyas evaluaciones identifiquen menor número de caracteres. Cuanto más se aproxime la salida de un programa a la adaptación a la cadena esperada de caracteres, tanto mayor será el valor de su aptitud.

En el paso 64, el sistema determina si ha de terminar la evolución. En otras palabras, debe haber algún criterio para determinar cuándo una o más de las cadenas de genes desarrolladas es "lo suficientemente buena" como para alcanzar las metas del usuario. Este criterio de terminación depende del problema y debe ser suministrado por el usuario. Se usa una función de aptitud como medida para determinar la superioridad relativa de una solución, comparada con una segunda solución. Los criterios de terminación típicos pueden requerir que se llegue a algún valor umbral cuando se aplique la función de aptitud. Otro posible criterio de terminación puede ser el hecho de que la aptitud de la cadena de genes del programa no haya mejorado apreciablemente durante las "n" últimas generaciones. Alternativamente, los criterios de terminación pueden ser los que comprendan esas dos medidas, de tal modo que se termine la evolución cuando se satisfaga cada uno de los dos criterios.

Si se satisface el criterio de terminación, se termina entonces el programa y se proporcionan las mejores cadenas de genes del programa creadas al usuario del sistema, como las mejores soluciones de programación para el problema particular. A la función de terminar se le da entrada al sistema en el paso 96 de la Figura 5, como se vio en lo que antecede.

Con referencia de nuevo a la Figura 4, si no se llega a cumplir el criterio de terminación, el sistema continúa al paso 66, donde se selecciona una operación para ser usada para construir la siguiente generación de cadenas de genes del programa. Se elige ya sea la coincidencia (paso 68) o ya sea la reproducción (paso 70) sobre la base en parte del régimen de coincidencia entrado como un parámetro del sistema, como se describe en el paso 92 en la Figura 5.

Una vez seleccionada una operación, se aplica la misma a la generación actual. La operación de coincidencia (paso 68) comporta combinar dos cadenas de genes del programa para crear dos cadenas de genes del programa nuevas y diferentes en la siguiente generación. La operación de reproducción (paso 70) comporta elegir una cadena de genes del programa de la generación actual y copiarla en la siguiente generación.

Para estas dos operaciones se han de elegir candidatos de la generación actual. En el caso de coincidencia, se deben seleccionar dos candidatos (paso 72). En el caso de reproducción, se debe seleccionar un solo candidato (paso 74), El método preferido de elección de candidato consiste en usar las clasificaciones de aptitud para ponderar la selección de candidatos de la generación actual. Para hacer esto, se suma el total de todas las clasificaciones de aptitud de la generación actual. La probabilidad de que cualquier candidato sea seleccionado es entonces la relación de su aptitud comparada con la aptitud total de todos los candidatos.

Por ejemplo, si la aptitud total de todas las cadenas de genes fuera de 250, y el valor de la aptitud de la cadena de genes "A" fuera 25, se tendría entonces una posibilidad de 25/250 = 10% de que sea elegida. Análogamente, si la cadena de genes "B" tiene un valor de la aptitud de 12,5, tiene entonces 1/2 de la posibilidad de la "A" (un 5%) de que sea elegida. Hay otras muchas estrategias para seleccionar las cadenas de genes superiores, pero este método se usa aquí como un ejemplo.

En la coincidencia, se divide la cadena de genes por uno o más lugares, denominados los puntos de cruce, y se combina con trozos de otras cadenas de genes satisfactorias que son divididas de un modo similar por uno o más lugares. El modo más simple y más potente de hacer coincidir dos cadenas de genes consiste en elegir un punto arbitrario en cada cadena de genes, romper cada cadena en dos partes, y combinar luego cada parte de la primera cadena de genes con una parte correspondiente de la segunda cadena de genes.

Como se ha ilustrado en la Figura 4, después de que se hayan elegido dos coincidentes en el paso 72, se determina un punto de cruce para cada cadena de genes en el paso 76. Continuando al paso 78, se aplican los trozos de dos o más cadenas de genes entre sí, creándose con ello nuevas cadenas de genes del programa.

En el paso 79, se verifica cada cadena recién creada, para asegurar que el número de paréntesis de apertura en las cadenas está emparejado con un número

igual de paréntesis de cierre. Esto es similar al equilibrio de paréntesis efectuado en el paso 107 de la Figura 6, donde se crean las cadenas de genes iniciales.

Por ejemplo, si la cadena de genes S(S(B+)1C)KI se hace coincidir con B(C(KS *)7K1, se decide mediante elección aleatoria romper la primera cadena de genes después del combinador "B", y romper la segunda cadena de genes después del combinador "I", y luego se hacen coincidir los trozos para crear las cadenas: S(S(B*)7K1 y +)1C)KIB(C(KS, respectivamente. Obsérvese que la primera cadena de genes creada de ea coincidencia: S(S(B*)7K1, tiene más paréntesis de apertura que paréntesis de cierre. Esto expresa un árbol no acabado, y por consiguiente se añade un paréntesis de cierre para crear una expresión acabada S(S(B*)7K1). En el caso de la segunda cadena de genes: +)1C)KIB(C(KS)I, hay dos paréntesis de cierre que están antes de cualquier paréntesis de apertura. Éstos se equilibran añadiendo dos paréntesis al principio de la expresión, de modo que ésta s convierte en: ((+)1C)KIB(C(KS). Continuando con el examen de esta cadena, hay ahora un paréntesis de apertura extra que se equilibra añadiendo un paréntesis de cierre al final de la expresión. Por consiguiente, la cadena se convierte en: ((+)1C)KIB(C(KS)).

En este ejemplo, el resultado de la coincidencia es el de producir dos cadenas de genes que tienen longitudes diferentes a las de sus "progenitoras". Esto es a la vez típico y necesario, debido a que permite que la longitud de las cadenas de genes del programa creadas por coincidencia cambie en las generaciones sucesivas, permitiendo así una complejidad adicional de las cadenas de genes, o bien la simplificación de las cadenas de genes.

Hay varias estrategias disponibles para hacer coincidir cadenas de genes, y el método descrito en lo que antecede es un método corrientemente usado. No obstante, el sistema de programación genética aquí descrito está diseñado para permitir a los usuarios "anular" o sustituir elementos clave del programa genético, tales como las funciones de coincidencia o de selección, dado que los problemas particulares pueden requerir una solución diferente. Además, la capacidad de "anular" el sistema proporciona un mayor nivel de control para los usuarios que deseen tener tal capacidad para "adaptar a la medida" cada programa.

En el paso 70, una cadena de genes de la generación actual se reproduce en la siguiente generación. La reproducción de la cadena de genes es simplemente la copia de una cadena de genes en la generación siguiente. Esto es similar a una supervivencia individual en la generación siguiente en el mundo natural. La cadena de genes no se altera, sino que únicamente es copiada en el grupo de genes para su consideración en la siguiente generación. Las cadenas de genes que son reproducidas en la siguiente generación son, en general, aquellas cadenas de la generación anterior que tienen los más altos niveles de aptitud.

En el paso 74 es donde se hace la elección real de individuos para reproducción. Esta elección se hace por elección aleatoria ponderada, como se ha descrito en el paso 72 anterior para elegir socios coincidentes.

Ya se hagan coincidir las cadenas, o ya se reproduzcan éstas en los pasos 68 ó 70, respectivamente, todas las cadenas que sean candidatos para la siguiente generación son sometidas a las operaciones de permutación y de mutación, representadas en los pasos 80 y 82, respectivamente.

La operación de permutación, paso 80, se efectúa tomando la frecuencia de permutación como entrada en el paso 91 de la Figura 5, y verificando aleatoriamente si ha sido permutada cualquier cadena. Si lo ha sido, se "bate" el orden de los genes candidatos. Por ejemplo, si ha sido reproducida la cadena S(SB(CK)SI)*7K, se verifica aleatoriamente en cuanto a permutación. Si una verificación aleatoria revela que la cadena está permutada en la siguiente generación, la cadena resultante puede ser S(BC7(CK)*I)SK o cualquier otra posible reordenación de la cadena de genes. La permutación proporciona un modo eficaz de transformar una cadena de genes que esté próxima a ser una solución correcta, sin que se tenga que confiar en otros genes para producir una coincidencia que cree tal redistribución. Así, las cadenas de genes del programa que tengan un valor de la aptitud relativamente alto pueden conseguir un valor de la aptitud de los genes todavía más alto, son simplemente redistribuidas con una secuencia diferente.

Según el método de ejecución preferido, se verifica cada gen de la cadena en cuanto a posible permutación, sobre la base de la frecuencia de permutación. Si se decide aleatoriamente permutar el gen, se elige entonces aleatoriamente otro gen y se intercambia la posición de los genes dentro de la cadena de genes del programa. Por ejemplo, si la cadena de genes del programa es S(K*+3)C(4 -)5, y por verificación sucesiva se halla que aunque los dos primeros genes ("S" y "(" respectivamente) no están permutados (sobre la base de la verificación aleatoria frente a la secuencia de permutación) pero que el tercer gen ("K") ha de ser permutado, entonces, por selección aleatoria, se elige otro gen (por ejemplo, el "C") y se intercambian los dos produciéndose la nueva cadena de genes del programa de S(C*+3)K(4 -)5. El resto de los genes serían entonces verificados también antes de completar la operación de permutación.

Puesto que la permutación puede hacer que los paréntesis de la expresión queden intercambiados, con secuencias desequilibradas de paréntesis de apertura y paréntesis de cierre (por ejemplo, puede hacer que la cadena empiece por un paréntesis de cierre), se debe repetir el proceso de equilibrado de paréntesis descrito en el paso 79, como parte de la operación de permutación.

El último operador es el operador de mutación, paso 82. Una mutación es cuando un gen se transforma por sí mismo espontáneamente en otro valor. La posibilidad de mutación está basada en la frecuencia de mutación, como se ha representado en el paso 90 de la Figura 5. Se usa esta frecuencia para determinar la posibilidad de que cualquier gen dado cambie su valor. Por ejemplo, si la cadena de genes S(S(B+)1C)KI tiene una mutación en el combinador C, se elige un nuevo gen de manera similar a la usada para crear la población inicial. El nuevo gen sustituye al combinador C y puede dar por resultado una nueva cadena S(S(B+)1K)KI. Las mutaciones dan por resultado cadenas de genes que son notablemente diferentes y que pueden crear una nueva cadena de genes radicalmente nueva con el potencial de producir una solución totalmente diferente para resolver un problema. Por consiguiente, es lo más probable que la mutación produzca un resultado innovador.

Como parte de la operación de mutación, si se muta un carácter de paréntesis, se debe aplicar la operación de equilibrado de paréntesis descrita en el paso

79 para la coincidencia, a la nueva cadena de genes.

Una vez que los nuevos candidatos hayan sido verificados en cuanto a posibles permutaciones y mutaciones, se añaden a la generación siguiente. En el paso 84 se verifica la nueva generación en cuanto a que esté completa, y si está llena se hace retornar entonces el sistema al paso 60, y empieza el ciclo de nuevo. Si no está llena, el sistema retorna al paso 66 y empieza el proceso de coincidencia o de reproducción, con objeto de continuar el llenado de la generación.

Un aspecto a considerar es el hecho de que no se puede garantizar que los programas generados vayan a tener un final. Esto es conocido en la bibliografía de la ciencia informática como el "Problema de Halting" (Problema de la Detención), y se ha demostrado que un sistema de programación no puede predecir si un programa acabará, o no, alguna vez.

Esto plantea un problema, por cuanto la evaluación de una cadena de genes de combinadores particular se puede producir una evaluación sin fin de los datos de entrada, que jamás llegue a producir una salida. Una solución para este problema de evaluación sin fin comporta temporizar la evaluación y terminar cualquier evaluación que no produzca un resultado dentro de un período de tiempo predeterminado. Una cadena de genes que no pueda ser evaluada dentro del período de tiempo especificado, o bien no es incluida en el grupo de genes candidatos para la siguiente generación, o bien se le asigna una clasificación muy baja de su aptitud.

Es posible otra forma de abordar este problema en un sistema de procesado en paralelo, tal como el ilustrado en la Figura 3. En este caso, un procesador separado (o un proceso si el sistema está realizado como una máquina de estado de software) evalúa cada cadena de genes del programa en el grupo de genes. A medida que se vayan completando las evaluaciones de las cadenas de genes, éstas se convierten en candidatos para las operaciones de reproducción y de coincidencia descritas en los pasos 68 y 70. Inicialmente, solo habrá unos pocos candidatos, pero finalmente aumenta el tamaño del grupo de candidatos. Las cadenas de genes del programa con valores de aptitud bajos son consideradas "no atractivas" para cadenas de genes de aptitud más alta. Las cadenas de genes más aptas se opondrán a la coincidencia con las cadenas de genes de baja aptitud, en un intento de esperar a que puedan entrar cadenas de genes más atractivas en el grupo de genes, más adelante. No obstante, si después de un período de tiempo especificado no aparece ninguna cadenas de genes como apta, o más apta, entonces la cadena de genes de alta aptitud puede coincidir con una cadena de genes de menor aptitud.

Con esta solución se evita el problema de "Halting" al no esperar a que aparezcan todas las cadenas de genes en el grupo de candidatos. En esencia, si una cadena de genes invierte un largo tiempo en entrar en el grupo de candidatos, no está disponible para coincidencia y por lo tanto, a menos que sea excepcionalmente apta, no continuará en la siguiente generación. Aquellas cadenas de genes que nunca aparezcan, nunca coincidirán. Esto es similar a la "selección natural", donde la disponibilidad para la coincidencia es, en esencia, una medida más de la aptitud.

En tal sistema, se debe añadir una verificación adicional para evitar situaciones en las que estén implicados demasiados procesadores en los problemas de evaluación sin fin. Si la mayoría o todas las máquinas combinadoras están evaluando cadenas de genes sin fin, el sistema de programación genética resulta ineficaz y jamás podrá resolver el problema. Esta situación puede evitarse imponiendo un límite de tiempo para la evaluación de cualquier cadena de genes. Si una máquina combinadora no puede evaluar una cadena de genes particular dentro de un período de tiempo predeterminado, se detiene la evaluación y se asigna una nueva cadena de genes al procesador. Por ejemplo, se puede usar un límite de tiempo de cinco segundos para evaluar cualquier cadena de genes del programa dada. Si no se completa la evaluación dentro de los cinco segundos, se termina la evaluación y se desecha la cadena de genes que esté siendo evaluada. Por consiguiente, las cadenas de genes que no puedan ser evaluadas dentro de los cinco segundos, no sobrevivirán en la siguiente generación. El límite de tiempo impuesto para una evaluación puede variar, dependiendo de la complejidad de las cadenas de genes que estén siendo evaluadas.

Un problema similar, pero menos complejo, se plantea cuando una cadena de genes del programa no resuelve por sí misma por completo. Es decir, que después de completada la evaluación el resultado todavía contenga combinadores y operadores. Por ejemplo, la cadena de genes del programa * S * I produciría, al ser aplicada a los datos de entrada 3, la expresión * *3 3, la cual se resolvería en la expresión *9. Suponiendo que no haya definición monádica definida para el operador *, la expresión está incompleta, por cuanto carece de un segundo elemento para la operación *.

El modo más simple de tratar este problema consiste en simplemente asignar una baja aptitud a tales expresiones. Si embargo, la ejecución preferida consiste en volver a aplicar la expresión a los datos de entrada. En el anterior ejemplo, la expresión *9 sería aplicada a la entrada 3, para producir la expresión *9 3, la cual produciría el resultado de 27.

Como se ha descrito anteriormente, en la Figura 2 se ha ilustrado una máquina combinadora para evaluar cadenas de genes del programa que contengan combinadores. La máquina combinadora incluye la memoria principal 42, que está diseñada para almacenar elementos en un formato de árbol, y combinadores que son ejecutados como primitivas máquinas. Tales máquinas dedicadas son eficaces, y pueden ser ejecutadas como un solo circuito integrado, diseñado a la medida, tal como un ASIC (Circuito Integrado para Aplicaciones Específicas), y pueden ser hechas funcionar en paralelo con otras máquinas combinadoras.

Alternativamente, se puede ejecutar una máquina combinadora en software, ejecutando para ello combinadores como manipulaciones de pila. En esta versión, el árbol sobre el cual trabajan los combinadores está construido como un mecanismo de pila de empuje hacia abajo, disponible en la mayoría de los ordenadores actualmente en el mercado. Los árboles de evaluación se construyen como una serie de indicadores en la pila de la máquina, y los combinadores alteran el árbol al redistribuir el orden de los elementos en la pila como sea apropiado. Esto da buen resultado, debido a que la mayoría de los procesadores actuales tienen eficaces operadores de manipulación de pilas, pero tienen operadores menos eficaces para manipular estructuras de árbol.

Además de representar las cadenas de genes con combinadores, los operadores de algoritmos genéti-

cos pueden ser también expresados en forma de combinadores, dado que los combinadores proporcionan una máquina universal. Por lo tanto, si se crea una máquina combinadora especializada, no hay necesidad de tener una máquina que reconozca otra cosa que las expresiones de combinadores.

Como se ha descrito en lo que antecede, en la Figura 3 se ha ilustrado un sistema de programación genética con múltiples máquinas combinadoras dispuestas en paralelo. Debido a la necesidad de tener varias cadenas de genes en el grupo de genes que deben ser evaluadas continuamente cada generación, hay un modo natural y eficaz de crear un sistema de procesado en paralelo. Como se ha ilustrado en la Figura 3, cada máquina combinadora es capaz de evaluar una cadena de genes de combinadores dado un conjunto de datos de entrada. Estas máquinas combinadoras se usan para evaluar cada una de las cadenas de genes en el grupo de genes. Por consiguiente, en vez de una evaluación secuencial de cada cadena de genes del programa en el grupo de genes, se evalúan simultáneamente todas las cadenas de genes, mejorándose con ello el rendimiento y la velocidad.

El controlador GP 52 (de Programación Genética) controla el sistema de programación genética en su totalidad, y hace uso del bus 46 de Datos/control para dirigir las máquinas combinadoras 50. El controlador GP 52 hace uso de los criterios, de los datos y de los parámetros de entrada almacenados en la memoria GP 54, la cual es cargada como parte de la programación del sistema. La memoria GP 54 almacena información tal como la de longitudes de las cadenas, mutación y regímenes de mutación, así como de los datos de entrada, de la función de aptitud y de los criterios de terminación usados para evaluar la eficacia de las cadenas de genes del programa que estén siendo desarrolladas.

Las máquinas combinadoras 50 pueden evaluar cadenas de genes arbitrarias cuando se les presenten datos de entrada. Además, las máquinas combinadoras pueden almacenar instrucciones para ejecutar las necesarias operaciones usadas en el algoritmo genético.

Inicialmente, el controlador GP 52 dirige cada máquina combinadora 50 para generar una cadena de genes aleatoria a partir de un conjunto de combinadores, de constantes y de operadores. El controlador GP 52 controla la coincidencia, la permutación y otras funciones genéticas, analizando para ello los resultados de una generación, tal como son comunicados a lo largo del bus 46 de Datos/control y emitiendo las apropiadas instrucciones a cada máquina combinadora en cuanto a qué función deberá efectuar a continuación con respecto a cada cadena de genes del programa.

En base a estas direcciones, parte o la totalidad de una cadena de genes es hecha pasar entre máquinas combinadoras, dependiendo de que estén siendo reproducidas o hechas coincidir con una cadena de genes en otra máquina combinadora. Estas cadenas de genes o fragmentos de cadenas de genes se desplazan a lo largo del bus 48 de cadenas de genes del programa.

El diseño en paralelo ilustrado en la Figura 3 acelera significativamente el proceso de evolución hacia una solución superior, ya que se pueden evaluar simultáneamente todas las cadenas de genes del programa, en vez de efectuar una evaluación secuencial de cada cadena de genes. Las máquinas combinadoras 50 son económicas y relativamente simples de construir. Por lo tanto, se pueden construir grandes sistemas de cálculo, masivamente, en paralelo, usando cientos o miles de máquinas combinadoras.

Para cadenas de genes del programa largas, o bien para problemas donde haya máquinas combinadoras no usadas; es decir, donde haya menor número de candidatos en el grupo de genes que máquinas combinadoras disponibles, la evaluación de una sola cadena de genes puede subdividirse en trozos y se pueden usar varias máquinas combinadoras para evaluar cada parte de la cadena de genes. Esto es posible debido a que todos los combinadores pueden ser evaluados en una forma independiente del orden, y mantener el mismo resultado cuando se vuelvan a unir todas las piezas de combinadores. En otras palabras, una sola cadena de genes puede ser subdividida en subcadenas A, B y C, cada una de las cuales s evalúa independientemente de las demás. Cuando se vuelvan a combinar las cadenas resultantes, el resultado final será el mismo que si se hubiera evaluado la cadena en un solo proceso.

Ejemplo del viajante de ventas

Un ejemplo del presente invento se ha ilustrado mediante su aplicación al bien conocido problema del viajante de ventas. Este es un problema de optimización, en donde el objetivo es hallar la ruta más corta para que un viajante de ventas se desplace a todas sus plazas de venta. Se suministra una tabla de distancias entre las plazas, como datos de entrada, y la ruta deseada es la que proporcione la distancia más corta, desplazándose a cada lugar al menos una vez.

No existe algoritmo o programa de ordenador alguno conocido que garantice el proporcionar la ruta más corta. Debido al gran número de elecciones (factorial de n-1, donde n es el número de ciudades en el problema), la solución simple de calcular todas las posibles rutas es muy ineficaz. Por ejemplo, en una situación en la que hayan de ser visitadas 15 ciudades diferentes, hay casi 90.000.000.000 de rutas posibles entre todas las ciudades.

El presente invento puede hallar una solución para el programa que produzca una de las mejores rutas, dada una tabla de distancias entre las ciudades. En primer lugar, el usuario crea una estructura de árbol que contenga los datos de entrada, es decir, las distancias entre todas las ciudades que deban ser visitadas. A continuación, el usuario da entrada a la función de aptitud y a la función de terminación para el problema.

Hay dos modos de enfocar el problema. El primer enfoque es el que podríamos denominar como la forma "débil" de programación genética, en la cual se hace uso de métodos conocidos o de soluciones conocidas escritas como cadenas de genes del programa. Estos programas conocidos son usados para "poblar" el grupo de cadenas de genes inicial. Este es un caso de empezar con buenas cadenas de genes iniciales, en un intento de hallar una solución todavía mejor. A esa solución se le denomina la forma "débil" de programación genética, porque es la que tiene más baja probabilidad de producir un resultado innovador, ya que todas las cadenas de genes iniciales representan soluciones conocidas. Puesto que las soluciones conocidas producen ya resultados relativamente satisfactorios, el sistema de programación genética es menos probable que produzca un resultado innovador que difiera significativamente de la "sabiduría usual".

Sin embargo, para este ejemplo del viajante de

ventas, se usará una forma "fuerte" de programación genética. Esta forma "fuerte" comporta "poblar" el grupo de cadenas de genes inicial con cadenas de genes del programa generadas aleatoriamente, las cuales serán evaluadas de acuerdo con su aptitud y seleccionadas como se ha descrito anteriormente. Aunque esta forma "fuerte" de programación genética produce inicialmente resultados inferiores, y lleva más tiempo hallar las buenas cadenas de genes del programa del programa, ofrece la posibilidad de crear una solución del programa radicalmente nueva y potencialmente superior. En esencia, las cadenas de genes del programa iniciales no contienen "nociones preconcebidas" de ningún tipo, en cuanto a lo que el usuario crea que es una buena solución para el problema. Por el contrario, el sistema de programación genética evalúa todas las posibles soluciones del problema, en vez de limitarse a un subconjunto predeterminado de soluciones.

Para empezar a resolver el problema dl viajante de ventas, se crea una estructura de árbol (o bien, matemáticamente, un grafo) que contenga la entrada de la tabla de distancias. Puesto que los programas de combinadores operan sobre grafos, los datos de entrada deben ser representados como un grafo. Tal estructura se ha ilustrado en la Figura 7. En esa figura se ha ilustrado un árbol, cada una de cuyas ramas es un subárbol que describe las distancias entre la ciudad representada por la rama y las otras ciudades que hay en el árbol. La cadena de texto representada en la parte inferior de la Figura 7 es una representación en cadena del mismo grafo. Es simplemente un modo más compacto de describir la misma estructura. Por ejemplo, la primera rama del árbol representa las distancias desde la ciudad "A" a todas las demás ciudades. Cada una de estas ramas tiene "ramitas" que consisten en las distancias y una "ramita de etiqueta" que identifica las ciudades a las que están asociadas esas distancias. Así, la distancia de "A" a "B" es de 12, la distancia de "A" a "C" es de 17, y la distancia desde "A" a "D" es de 45. Análogamente, la siguiente rama subiendo por el árbol representa las distancias desde la ciudad "B" a las demás ciudades; es decir, que de B a A, es de 12, de B a c es de 27, y de B a D es de 32. El árbol continúa hasta el subárbol final, el cual da las distancias desde la ciudad D a las demás ciudades.

Las cadenas de genes del programa generadas mediante el invento se aplican a este árbol de entradas para producir un resultado, El resultado deseado es una lista de ciudades, en el orden en que deberán se visitadas.

La función de aptitud debe dar las calificaciones más altas a aquellas cadenas de genes del programa que produzcan la distancia total más corta. No obstante, la función de aptitud debe también evaluar los candidatos que no produzcan una ruta completa. Por consiguiente, la primera medida de la aptitud debe ser si una cadena de genes del programa particular selecciona la lista completa de las ciudades.

Puesto que la aptitud depende de la distancia total recorrida, una función de aptitud deseable producirá diferentes valores para cada cadena de genes del programa, y cuanto menor sea el valor tanto mejor será la cadena de genes. Por consiguiente, una cadena de genes del programa que produzca una distancia a recorrer corta, recibirá un valor apto de su aptitud. A la inversa, una cadena de genes del programa que produzca una distancia a recorrer larga recibirá un valor bajo de su aptitud.

Para hacer una estimación escalada del rendimiento de una solución, se compara una solución tentativa con una solución que sea probablemente la del peor caso. Una solución de peor caso se genera hallando la distancia más larga entre dos ciudades cualesquiera en la ruta, y multiplicando esa distancia por el número de ciudades, menos uno, en la ruta. Por ejemplo, en la Figura 7 la distancia máxima entre las cuatro ciudades es de 45 (A a D). Esta distancia se multiplica por 4 para obtener una distancia máxima de 180 (ya que 4 x 45 = 180).

Esta solución del peor caso es, como mínimo, tan grande como cualquier solución propuesta, dado que en ninguna ruta que visite todas las ciudades se puede hacer el viaje más largo en todos los casos, a menos que todas las distancias sean la misma; es decir, que todas las ciudades sean equidistantes entre sí, que no es el caso en el problema ilustrado en la Figura 7. Como resultado, se puede usar esta función de aptitud para producir una comparación objetiva de las rutas propuestas. Esto se hace produciendo una relación de rendimiento del recorrido dividiendo para ello la distancia recorrida en una ruta propuesta por la ruta del peor caso.

El criterio para la aptitud se determina entonces mediante la siguiente función:

$$f = (w(n-v)+d)/w$$

donde (valores representados entre paréntesis en la Figura 7):

- w = distancia en el peor caso (180)
- n = número de ciudades en toda la ruta (4)
- v = número de ciudades visitadas en la ruta propuesta
- d = distancia recorrida en la ruta propuesta

Así, para el problema ilustrado en la Figura 7, la función de aptitud se convierte en f=(180(4-v)+d)/180.

Esta función de aptitud establece una prima por completar la ruta, dado que cualquier programa que no finalice tendrá una gran penalidad, en forma de la distancia de la ruta del peor caso multiplicada por el número de ciudades que se hayan omitido en la ruta propuesta. Por otra parte, para todas las rutas propuestas que visiten todas las ciudades en el territorio del viajante de ventas, la primera mitad de la función se reduce a 0 (dado que n-v será cero), y la segunda parte, d/w, pasará a ser el factor decisivo. Puesto que w es una constante, cuanto menor sea el valor de d tanto menor será la relación y tanto mejor será la aptitud. Puesto que la distancia se expresa como una relación, la aptitud relativa de las rutas no se verá influenciada por la escala usada para medir las distancias.

La función de aptitud actúa igualmente bien con kilómetros que con millas o con pies ingleses. No obstante, para producir un resultado exacto, se deben representar todos los valores de las distancias usando la misma unidad de medida; es decir, todas las medidas en millas, o todas las medidas en kilómetros.

Finalmente, el usuario debe proporcionar un criterio de terminación que se use para determinar cuándo una cadena de genes del programa particular sea "suficientemente buena". En este caso, buscaremos simplemente una ausencia de progreso en los resultados.

Si, para 100 generaciones (donde una generación es la ejecución de los pasos 60 a 84 en la Figura 4, el mejor candidato no ha mejorado en más de un 5%, se termina la evolución y se usa la cadena de genes del programa que tenga el mejor valor de la aptitud como la solución para el problema.

Sin embargo, puesto que, presumiblemente, estamos buscando la mejor solución de programa para cualquier conjunto de datos, este proceso debe efectuarse repetidamente para los diferentes problemas. Esto es similar al caso de una criatura que se desarrolle naturalmente, sobreviviendo en una serie de diferentes situaciones que ponen a prueba su aptitud. En otras palabras, una solución de programa que funcione bien para un conjunto de dato de entrada puede no funcionar bien para otros, debido a que el único conjunto de datos de entrada usado para crear la solución del programa puede no ser lo suficientemente diverso como para producir una solución general.

En este caso, las mejores soluciones para el conjunto inicial de datos se someten a prueba frente a diferentes conjuntos de datos de entrada. Se repite este procedimiento para crear una sucesión de refinamientos que crearán una solución de programa que produzca buenos resultados para un gran número de problemas.

El anterior ejemplo del viajante de ventas se ha limitado a cuatro ciudades por simplicidad, y tiene únicamente fines ilustrativos. Sin embargo, se puede usar la misma estructura de árbol para representar las distancias entre cualquier número de ciudades. Un mayor número de ciudades se traduce, simplemente, en un árbol mayor para representar los datos de entrada. Se usa la misma función de aptitud ($f = w(n-v) + d)/w$, con independencia del número de ciudades (con los apropiados cambios en cuanto a la distancia en el peor caso w y en cuanto al número de ciudades en toda la ruta, n). Al aumentar el número de ciudades, aumenta el tiempo requerido para generar un solución aceptable, pero permanece invariable el procedimiento usado para crear, desarrollar y evaluar cadenas de genes del programa. Por consiguiente, una vez que se haya desarrollado un sistema de programación genética para resolver un tipo particular de problema, se puede usar el mismo sistema de programación con varios conjuntos de datos de entrada, para determinar el mejor programa para resolver el problema dado.

REIVINDICACIONES

1. Un método ejecutado por ordenador para resolver un problema de programación usando técnicas de programación genética, comprendiendo dicho método los pasos de:

a. definir una función de aptitud (96) para medir la superioridad relativa de una primera solución con respecto a una segunda solución;

b. determinar los datos de entrada (94) a partir de los cuales se resolverá el problema;

c. crear (59) una pluralidad de grafos de programa estando representado cada grafo de programa por una serie de operadores de reducción de grafos operativos para alterar la estructura de dicho grafo de programa, representando cada grafo de programa una solución potencial para el problema a resolver;

d. aplicar (60) cada grafo de programa a dichos datos de entrada, alterando para ello la estructura de dicho grafo de programa de acuerdo con los operadores de reducción de grafos, para generar una solución para dicho problema de programación;

e. usar dicha función de aptitud de un modo específico para el problema, para asignar (62) una aptitud a cada uno de dichos grafos de programa, sobre la base de dicha solución producida aplicando dicho grafo de programa a dichos datos de entrada;

f. desarrollar dichos grafos de programa sobre la base de la evaluación (62) de su aptitud; y

g. repetir los pasos (d) a (f) hasta que se haya satisfecho un criterio de terminación (64).

2. El método según la reivindicación 1, en el que dicho paso de clasificar dicha función de aptitud va seguido de la escalación de la medida de la aptitud producida de un modo específico para el problema.

3. Un sistema de programación genética para hallar un programa para resolver un problema de programación, dada una entrada particular, comprendiendo dicho sistema:

un dispositivo de memoria (42, 44) para almacenar diversa información;

un dispositivo de entrada (18) para entrar un conjunto de datos de datos de entrada que contienen información para los cuales se debe obtener una solución del problema, siendo almacenados dichos datos de entrada en dicho dispositivo de memoria;

una función de aptitud (96) que proporciona una medida para determinar la superioridad relativa de una primera solución con respecto a una segunda solución, siendo almacenada dicha función de aptitud en dicho dispositivo de memoria;

una pluralidad de grafos de programa almacenados en dicho sistema de programación genética, estando representado cada uno de dichos grafos de programa por una serie de operadores de reducción de grafos operativos para alterar la estructura de dicho grafo de programa; y

una unidad de procesador central (12) acoplada eléctricamente a dicho dispositivo de memoria, que comprende respectivos medios para

d. aplicar cada programa a dichos datos de entrada, alterando para ello la estructura de dicho grafo de programa de acuerdo con los operadores de reducción de grafos para generar una solución para dicho problema de programación;

e. usar dicha función de aptitud de un modo específico para el problema, para asignar una aptitud a cada uno de dichos grafos de programa sobre la base de dicha solución producida aplicando para ello dicho grafo de programa a dichos datos de entrada;

f. desarrollar dichos grafos de programa sobre la base de la evaluación de su aptitud; y

g. repetir estos pasos (d) a (f) hasta que se haya satisfecho un criterio de terminación.

4. El sistema según la reivindicación 3, en que dicho sistema está realizado en un circuito electrónico construido para el caso.

5. El sistema según la reivindicación 3, en el que dicha unidad de procesador central comprende un controlador de programa (52).

6. El sistema según la reivindicación 3, en el que dicho dispositivo de memoria par almacenar diversa información comprende además una unidad (54) de memoria de programa genético para almacenar los datos asociados con la operación de dicho sistema de programación genética, estando dicha unidad de programa genético acoplada electrónicamente a dicha unidad de procesador central.

7. El sistema según la reivindicación 3, en el que dichos grafos de programa son desarrollados para abordar una solución deseada.

8. El método según la reivindicación 1, que comprende además el paso de efectuar uno o más de los pasos A-G utilizando una pluralidad de unidades de procesado (50) de ordenador en paralelo.

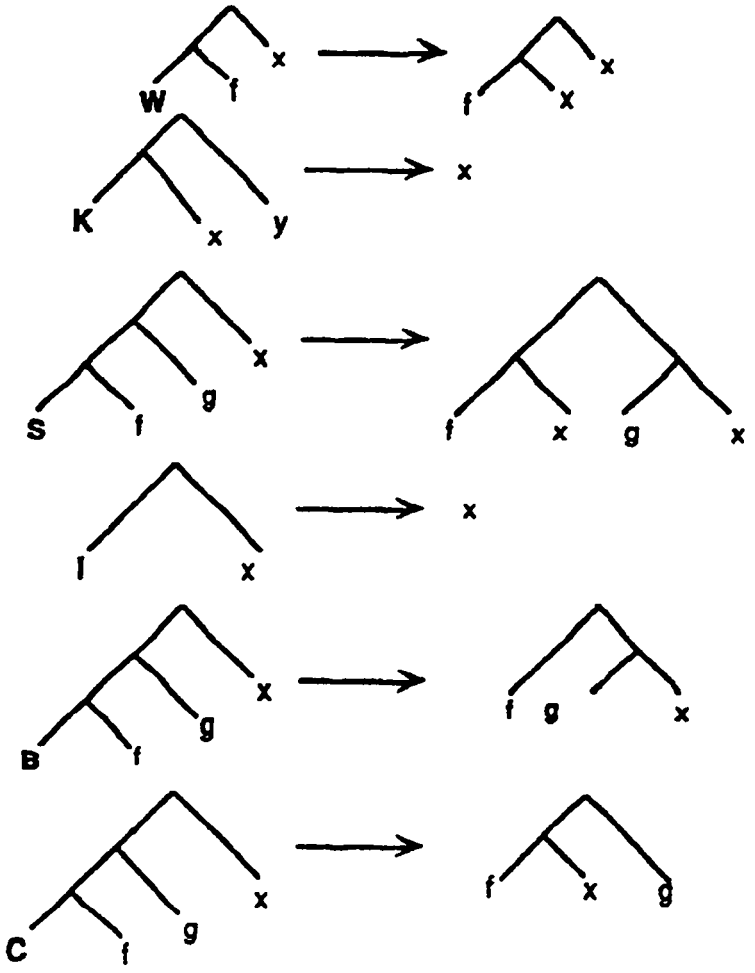


Fig. 1

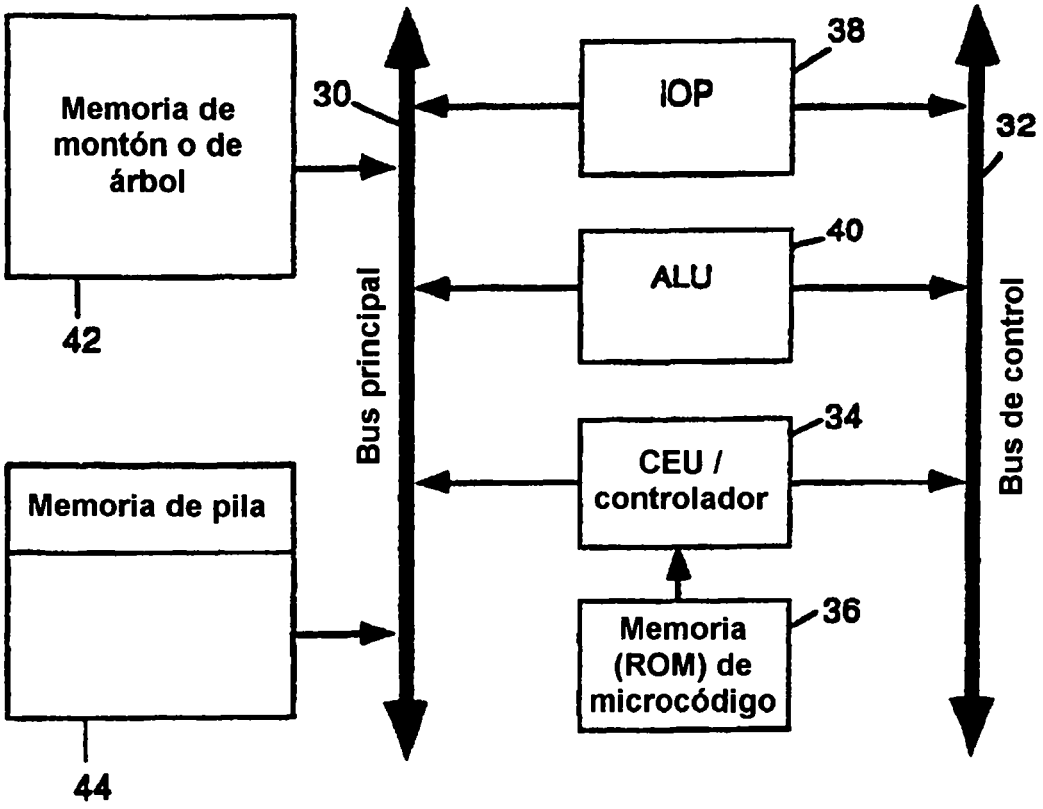

Memoria de montón o de árbol
42
Memoria de pila
44
30
Bus principal
IOP
38
ALU
40
CEU / controlador
34
Memoria (ROM) de microcódigo
36
32
Bus de control

Fig. 2

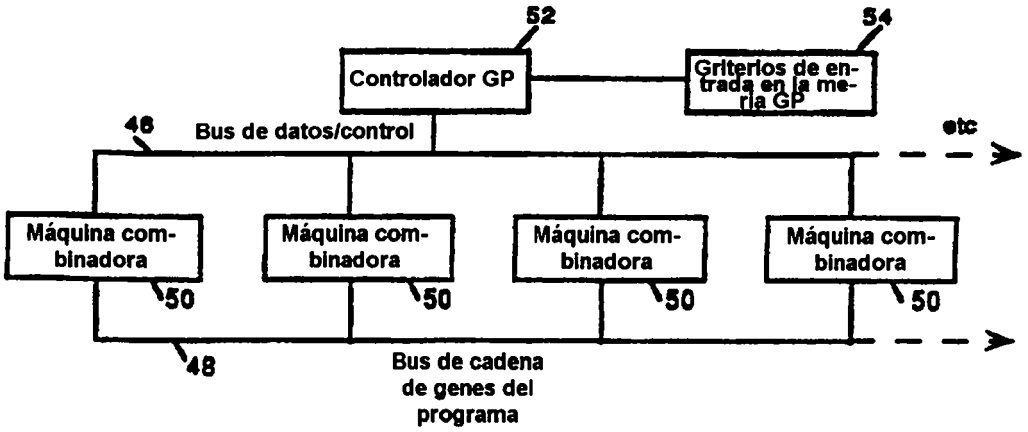

52
54
Controlador GP
Criterios de entrada en la memoria GP
48
Bus de datos/control
etc
Máquina combinadora
50
Máquina combinadora
50
Máquina combinadora
50
Máquina combinadora
50
48
Bus de cadena de genes del programa

Fig. 3

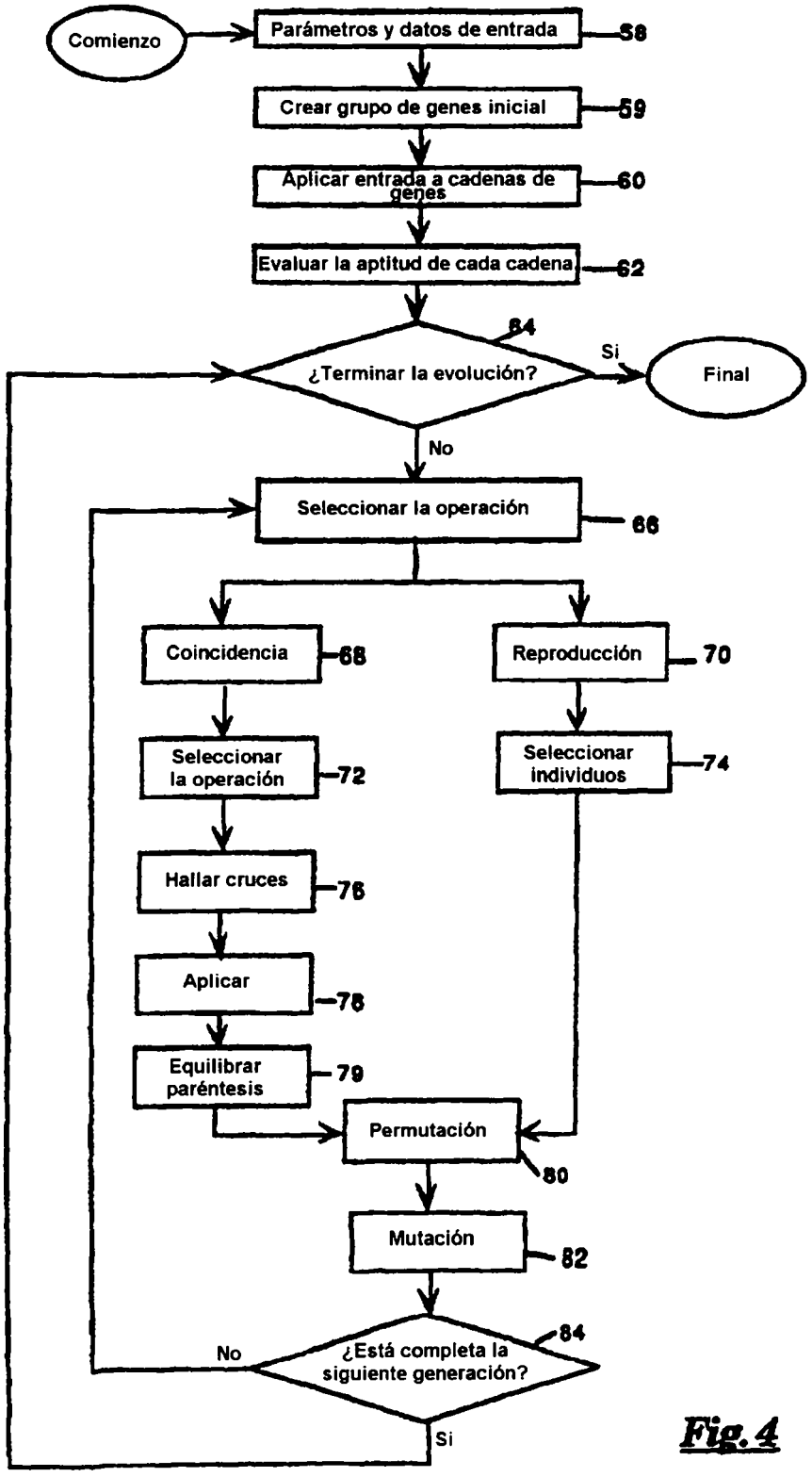

Comienzo
Parámetros y datos de entrada
58
Crear grupo de genes inicial
59
Aplicar entrada a cadenas de genes
60
Evaluar la aptitud de cada cadena
62
64
¿Terminar la evolución?
Si
Final
No
Seleccionar la operación
66
Coincidencia
68
Reproducción
70
Seleccionar la operación
72
Seleccionar individuos
74
Hallar cruces
76
Aplicar
78
Equilibrar paréntesis
79
Permutación
80
Mutación
82
84
No
¿Está completa la siguiente generación?
Si

Fig. 4

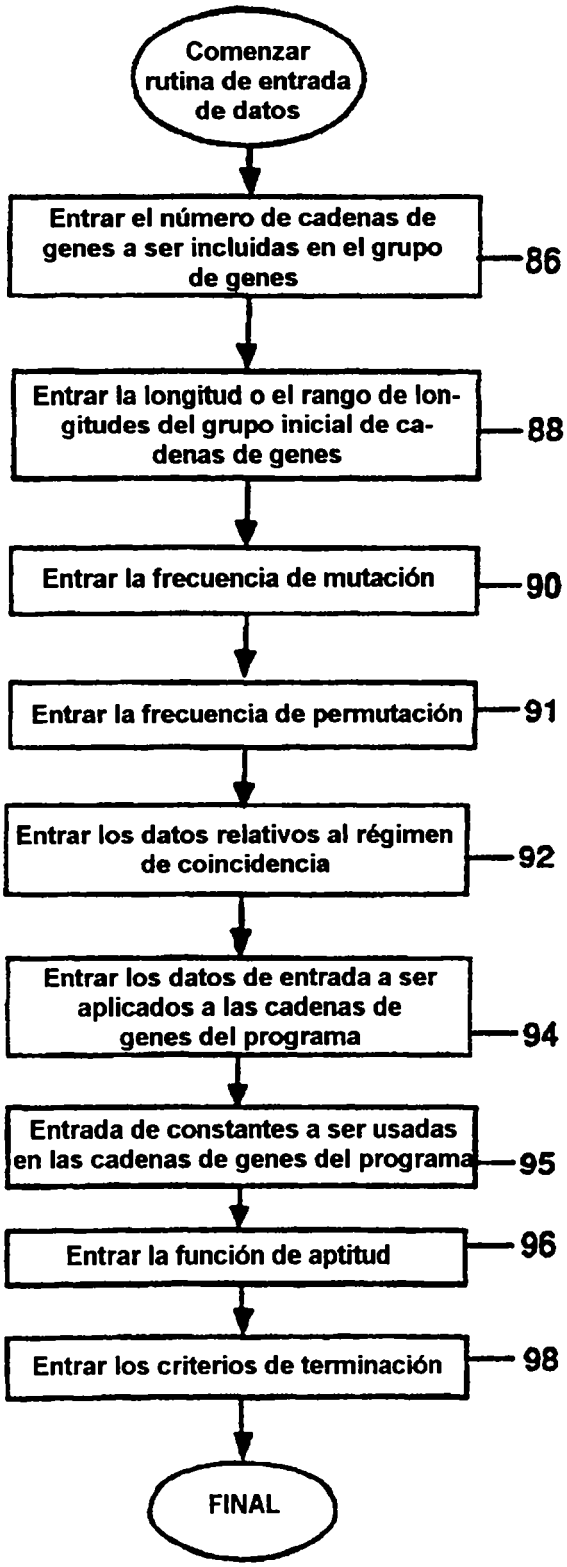

Comenzar rutina de entrada de datos
Entrar el número de cadenas de genes a ser incluidas en el grupo de genes
86
Entrar la longitud o el rango de longitudes del grupo inicial de cadenas de genes
88
Entrar la frecuencia de mutación
90
Entrar la frecuencia de permutación
91
Entrar los datos relativos al régimen de coincidencia
92
Entrar los datos de entrada a ser aplicados a las cadenas de genes del programa
94
Entrada de constantes a ser usadas en las cadenas de genes del programa
95
Entrar la función de aptitud
96
Entrar los criterios de terminación
98
FINAL

Fig. 5

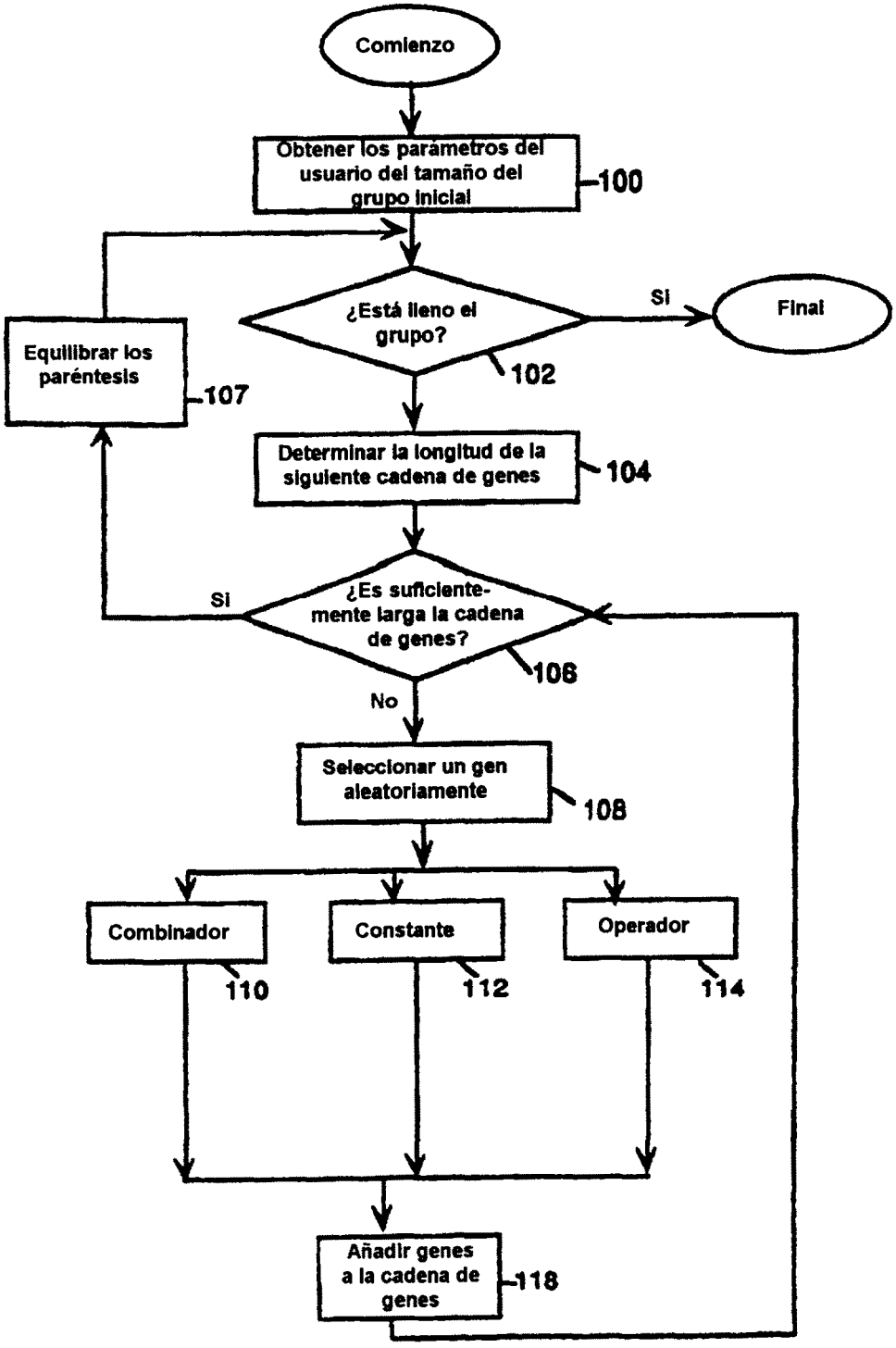
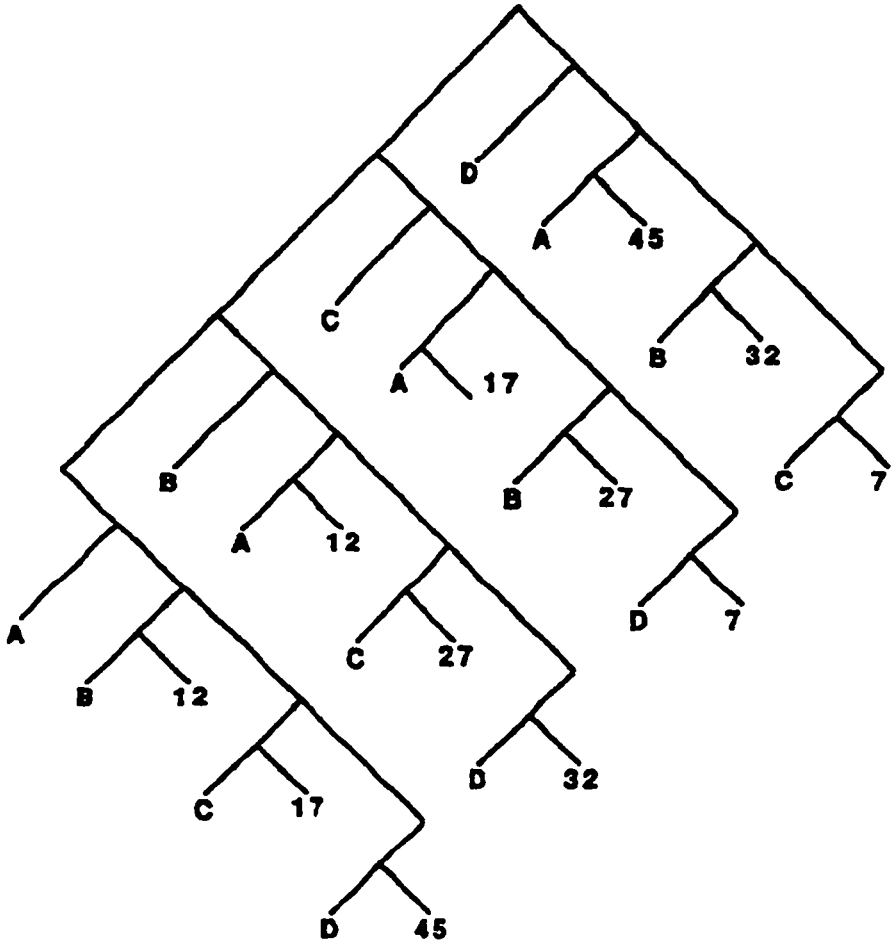

Comienzo
Obtener los parámetros del usuario del tamaño del grupo inicial
100
¿Está lleno el grupo?
102
Si
Final
Equilibrar los paréntesis
107
Determinar la longitud de la siguiente cadena de genes
104
¿Es suficiente-mente larga la cadena de genes?
106
Si
No
Seleccionar un gen aleatoriamente
108
Combinador
110
Constante
112
Operador
114
Añadir genes a la cadena de genes
118

Fig. 6




```
(  (A(B  12)(C  17)(D  45))
(B(A  12)(C  27)(D  32))
(C(A  17)(B  27)(D  7))
(D(A  45)(B  32)(C  7)))
```

Fig. 7

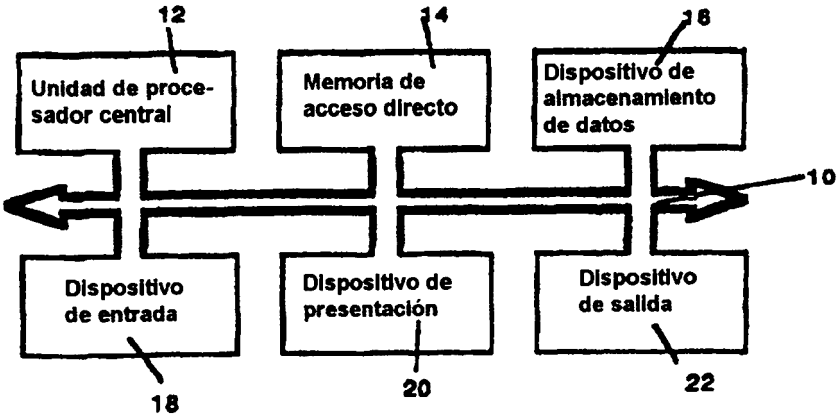

12
14
16
Unidad de procesador central
Memoria de acceso directo
Dispositivo de almacenamiento de datos
10
Dispositivo de entrada
Dispositivo de presentación
Dispositivo de salida
18
20
22

Fig. 8