Todo

Año 1 • Número 12 • **6,50** euros

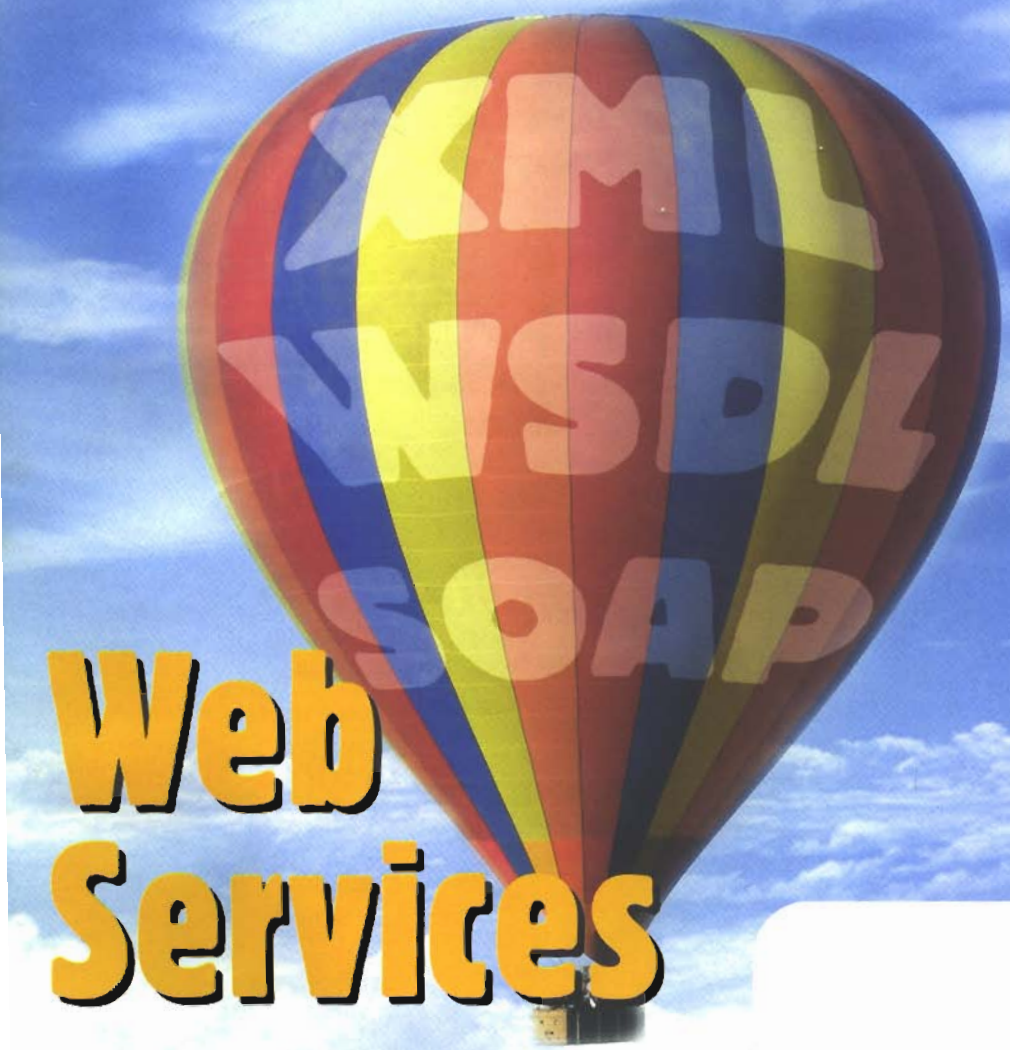
Studio Press

8 413042 751977 00012

Programación

La Revista bimestral para entusiastas de la programación

www.iberprensa.com



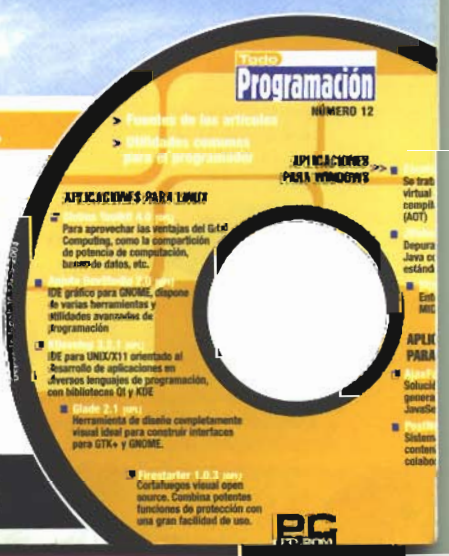
Web Services

Sesiones en PHP con PostNuke

Uso de Generics en el lenguaje Java

Diseño de interfaces gráficos con Glade

■ CD-ROM: ESPECIAL ENTORNOS DE PROGRAMACIÓN LINUX: ANJUTA, GLADE, KDEVELOP, ETC.



Zona Linux

- C#: Servicios Web, ejemplos con Google y Amazon
- Cómo crear repositorios con Subversion

Zona Windows

- F#: Programación funcional para .NET
- Programación basada en algoritmos genéticos

¿QUÉ ES LA PROGRAMACIÓN ROBUSTA Y CÓMO OBTENER CÓDIGO ROBUSTO EN JAVA

Todo Programación
La Revista bimestral para entusiastas de la programación

DIRECTOR
Eduardo Toribio
etoribio@iberprensa.com

REDACCIÓN
Yenifer Trabadela
yenifer@iberprensa.com

COLABORADORES
Antonio M. Zugaldía
(azugaldia@iberprensa.com)
David Santo Orcero
(orcero@iberprensa.com)
Manuel Domínguez
(mdominguez@iberprensacom)
Fernando Escudero
(fescudero@iberprensa.com)
José Manuel Navarro
(jnavarro@iberprensa.com)
Marcos Prieto
(mprieto@iberprensa.com)
Guillermo "el Guille" Som
(elguille@iberprensa.com)
Santiago Márquez
(smarquez@iberprensa.com)
José Rivera
(jrivera@iberprensa.com)
Alejandro Serrano
(aserrano@iberprensa.com)
Jordi Massaguer
(jmassague@iberprensa.com)
Pedro Agulló
(pagullo@iberprensa.com)
Moisés Díaz
(mdiaz@iberprensa.com)

DISEÑO PORTADA
Antonio Gª Tomé

MAQUETACIÓN
Antonio Gª Tomé

DIRECTOR DE PRODUCCIÓN
Carlos Peropadre
cperopadre@iberprensa.com

ADMINISTRACIÓN
Marisa Cogorro

SUSCRIPCIONES
Tel: 91 628 02 03
suscripciones@iberprensa.com

FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: I. G. Printone
DUPLICACIÓN CD-ROM: M.P.O.

DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Studio Press
www.iberprensa.com

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, Nave 13.
Polígono "El Nogal"
28110 Algete (Madrid)
Tel.: 91 628 02 03*
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-13679-2004
Número 12 • Año 1

Eduardo Toribio

EDITORIAL

Web Services

Este mes dedicamos nuestro tema de portada a los Web services, un término que ya hace tiempo se emplea con frecuencia en distintos ámbitos. Estudiaremos desde una perspectiva teórica qué son los servicios web, para posteriormente pasar a ver algunos ejemplos de implementación práctica en la sección de Mono. Por lo demás seguimos incrementando nuestra oferta en el bloque Linux, ahora con programación para KDE con las librerías Qt, que también compilan para Windows, y las habituales secciones de Mono y GTK#, además del curso de Globus. En lo que a Windows se refiere, y concretamente a .NET, dedicamos un reportaje especial a la programación funcional con F#. Finalmente, y para que aparezcan todos los actores principales en este nuestro mundillo, incluimos dos tutoriales centrados en Java, el primero sobre el uso de Generics y el segundo sobre programación robusta.

SUSCRIPCIONES

Como oferta de lanzamiento existe la posibilidad de suscribirse durante 12 números a **Todo Programación** por solo 61 euros lo que significa un ahorro del 20% respecto el precio de portada. Además de regalo puedes elegir entre una de las dos guías que aparecen abajo.
Más información en: www.iberprensa.com

SERVICIO TÉCNICO

Todo Programación dispone de una dirección de correo electrónica y un número de Fax para formular preguntas relativas al funcionamiento del CD-ROM de la revista.
e-mail: todoprogramacion@iberprensa.com
Fax: 91 628 09 35

LECTORES

Comparte con nosotros tu opinión sobre la revista, envíanos tus comentarios, sugerencias, ideas o críticas.

Studio Press
(Todo Programación)
C/ Del Río Ter, Nave 13
Pol. "El Nogal"
28110 Algete. Madrid

DEPARTAMENTO DE PUBLICIDAD

Si le interesa conocer nuestras tarifas de publicidad no dude en ponerse en contacto con nuestro departamento comercial:
Tel. 91 628 02 03
e-mail: publicidad@iberprensa.com

Sumario

Número 12

A quién vamos dirigidos

Todo Programación (TP) es una revista para programadores escrita por programadores y con un enfoque eminentemente práctico. Trataremos de ser útiles al programador, tanto al profesional como al estudiante. Si hay algo cierto en este sector es que nunca podemos parar, vivimos en un continuo proceso de reciclaje. Ahí es donde tratará de encajarse TP: información, actualidad, cursos y prácticas de los lenguajes más demandados y formación en sistemas.

En portada

Web Services: introducción y escenarios para su uso

10 Desde hace algún tiempo ha surgido una nueva tecnología que parece estar en todas partes: los servicios web. En el tema de portada de este mes vamos a tratar de definir qué son y su ámbito de aplicación. Aprovecharemos también para dar algunas definiciones básicas, mientras en la sección habitual de Mono, dentro de la zona Linux, implementaremos varios ejemplos prácticos.

ZONA LINUX

Actualidad: Novedades en lenguajes y compiladores >>

21 Analizamos la actualidad sobre lenguajes y herramientas para programadores Linux. Han sido novedades estos días la publicación de la nueva versión del IDE de Anjuta y algunas noticias relacionadas con los programas para control de versiones.

C#: Introducción a los servicios web >>

24 Enlazamos con nuestro reportaje del mes, ahora vamos a ver un par de ejemplos prácticos para la implementación de servicios web con C#. Aprovecharemos los APIs abiertos y disponibles para el programador de Google y Amazon. Manos a la obra.

GTK#: Diseño de interfaces con Glade >>

28 Finalizamos nuestra serie dedicada a GTK con un ejercicio práctico con Glade. También comentaremos algunas alternativas como Gazpacho, incluyendo una entrevista con el coordinador de dicho proyecto.

Ejemplos y código fuente

Cada CD-ROM de la revista incluye una carpeta denominada *fuentes* en la que se encuentra el material complementario para seguir cada uno de los cursos: los listados completos, los ejemplos desarrollados en diversos lenguajes, compiladores, editores, utilidades y en general, cualquier herramienta que se mencione o cite en la respectiva sección.
Todo con la finalidad de completar la formación y facilitar el seguimiento de cada artículo por parte del lector.

CONTENIDO DEL CD-ROM

Herramientas y recursos para el programador

65 En el CD de **Todo Programación** incluimos este mes una recopilación de herramientas para programadores en ambas plataformas (Windows/Linux), además del código fuente y los ejemplos de todos los artículos de la revista. Este número también incluye un especial dedicado a los entornos RAD de desarrollo para Linux, destacando entre todos Glade. También se incorporan diversas utilidades para desarrolladores Java como JDebugTool y ExcelsiorJet.

TALLER PRÁCTICO

Programación robusta en Java >>

Programación de Grids: Arquitectura del Globus toolkit

Extendiendo el framework de Qt: El juego Memory

Bases de datos en el cliente con JavaScript DB

Generics en Java: uso y ejemplos prácticos

ZONA WINDOWS

F#: Programación funcional para .NET >>

REPORTAJE: Programación de algoritmos genéticos

Y ADEMÁS...

DESARROLLO WEB: sesiones en PHP con PostNuke >>

JUEGOS JAVA: Aspectos avanzados >>

NOTICIAS

6. **Java** Expo 2005.
6. **Séptima** edición del foro ENSA@WORK.
7. **Sun** renueva su convenio con la UC3M.
7. **JDBC** Manager será integrado en NetBeans.
8. "**Summer** of Code" de Google.
8. **Nuevas** herramientas de desarrollo para Longhorn.
8. **AMD** presenta "Pacifica".
8. **Borland** Day.
9. **Nuevos** servidores Bull NovaScale 9162 y 9322.
9. **Servidor** Sun Fire V40z.
9. **Dell** Precision 380.

NIVEL DE DIFICULTAD:

MATERIAL INCLUIDO EN EL CD-ROM:
Código fuente de los ejemplos

Programación de algoritmos genéticos

MANUEL DOMÍNGUEZ
mdominguez@iberprensa.com

Aunque muchas películas de cine y los sueños de muchos visionarios coinciden desde hace años en que la informática permitiría construir sistemas capaces de ser más inteligentes que nosotros, lo cierto es que el ser humano está muy bien construido y no es fácil mejorarlo.

Es obvio que cuando debemos tomar una decisión, no empezamos a pensar como si tuviésemos un bucle en nuestro cerebro, evaluando todas las posibilidades; tendemos a elegir una solución a los problemas que sea rápida y buena, aunque no sea óptima. En la mayor parte de los casos esto es una buena elección. Por ejemplo, si vamos por la carretera con nuestro coche y otro vehículo que va delante frena en seco, instintivamente tendemos a girar el volante a la izquierda para no chocar con él ni salimos de la carretera. La mayor parte de las veces esto evitará un golpe, pero no es una opción óptima; sobre todo si viene un coche por el carril contrario. Sin embargo, es una buena decisión, dado el tiempo que teníamos para pensar qué hacer. Quizás en cinco minutos hubiésemos encontrado una solución perfecta, pero ya no nos valdría de nada, ya que habríamos chocado con el vehículo de delante.

Técnicas bioinspiradas

La informática había obviado este enfoque durante mucho tiempo, pero hace algunos años surgió lo que hoy en día se llaman técnicas bioinspiradas. Son técnicas que intentan imitar el funcionamiento de los procesos en los seres humanos para resolver problemas en el ámbito informático. Procesos como la evolución, el razonamiento, la toma de decisiones, la adaptación, etcétera.

La inteligencia artificial aporta diversas técnicas bioinspiradas para la solución de problemas, como por ejemplo, el concepto de neurona, las redes neuronales, los algoritmos evolutivos, o los algoritmos genéticos, en los que nos vamos a centrar en este reportaje. Los algoritmos genéticos intentan copiar el modo en que la especie humana evoluciona, y utilizar este modelo para conseguir soluciones buenas y rápidas a problemas complejos. Para entender y poder programar un algoritmo genético, hay que tener claras algunas nociones simples de genética y biología; así que vamos con ello.

Evolución humana

¿Cómo hemos llegado a ser lo que somos? Durante años los hombres se han realizado esta pregunta. Charles Darwin (1809-1882), mediante la observación de muchas especies, llegó a la innovadora teoría de que éstas evolucionan mediante su capacidad para adaptarse al medio que las rodea, esto es, los que mejor se adaptan, sobreviven y se reproducen, los peor adaptados, mueren y generalmente sin descendencia.

Por su parte, Johann Mendel (1822-1884) sentó las bases de lo que años después sería la genética. Averiguó cómo funcionaban de forma muy básica las leyes de la herencia, y que existían caracteres que se pasaban (no sabía cómo) de padres a hijos, de hijos a nietos, etcétera. Y además, comprobó que los hijos tenían una mezcla de las características de los padres. Hoy en día se han fusionado ambas teorías en lo que se llama neodarwinismo que, junto con la genética avanzada, ha dado muchas respuestas. Así, un individuo de una especie está determinado por la composición genética de sus cromosomas (obviando los factores sociales). Aquellos a los que sus genes les impidan adaptarse al medio, desaparecen; el resto se reproduce y se entremezcla, dando lugar a nuevos individuos con una combinación de los genes paternos, con características distintas (pero parecidas) que de nuevo tienen más o menos posibilidades de adaptarse. Así, tras un cierto tiempo, la especie habrá evolucionado.

De vez en cuando, por diversos factores, la composición genética de un individuo sufre una mutación. La mayor parte de las veces esta mutación es perjudicial, impide que el individuo se adapte a su entorno, y por tanto muere y desaparece. Pero en raras ocasiones ocurre que la mutación provoca una mejora en el propio individuo que le hace adaptarse mucho mejor, con lo cual sobrevive, se reproduce y sus genes son traspasados a generaciones futuras.

Por ejemplo, una composición genética que provoque que un individuo sea más atractivo que el resto, permitirá que éste tenga más posibilidades de éxito para reproducirse y, a la vez, transmitir sus genes a una nueva generación. Algo parecido ocurre con una composición genética que haga que la persona tenga más facilidad para ser afectado por enfermedades; probablemente esa persona fallezca prematuramente o, en cualquier caso, tendrá menos posibilidades de sobrevivir.

En esta introducción teórica hemos sacado a la luz todos los términos importantes en la genética que nos van a hacer falta para programar un algoritmo genético: individuo, cromosoma, gen, combinación, mutación y adaptación. En la Tabla 1 se muestra una definición más concreta de cada uno de ellos.

Deriva genética

Dado lo que hemos contado hasta ahora, está claro que la mejor forma de evolucionar es tener mucha diversidad y descendencia. Las poblaciones tienden a estabilizarse pasado un tiempo, cuando la mayor parte de la población comparte un mismo conjunto de genes y solo se reproducen entre ellos. Existen casos de regiones aisladas en las que todos los habitantes comparten, por ejemplo, el tener seis dedos en un pie. También hay otra población en China donde

El objetivo del algoritmo genético es buscar la **mejor solución** dentro de un tiempo concreto

todos los habitantes mueren con más de ciento diez años. En otros casos el efecto es más drástico, como por ejemplo, lugares donde existe una probabilidad de desarrollar una cierta enfermedad. Incluso en las familias reales de todo el mundo donde históricamente se han reproducido entre ellos, es raro no encontrar una característica común: hemofilia, daltonismo...

Los casos anteriores son debidos a un efecto llamado *deriva genética*. Consiste en que en poblaciones reducidas las posibilidades de combinación genética son pocas, puesto que todos los individuos tienen casi los mismos genes y esto no permite la evolución salvo, claro está, que ocurra una mutación en alguno de los individuos de estas zonas reducidas.

En el Gráfica 1, se puede observar este fenómeno. Se muestra una estrella que indica el estado de evolución de una población concreta. Por el proceso de selección natural y la teoría de evolución que hemos visto, salir de ese máximo local es complicado, no hay ningún individuo que pueda aportar un gen nuevo que haga que la descendencia sea mejor. Existen estados de evolución con muchísima más facilidad para adaptarse al entorno que el actual, pero para ello hay que evolucionar a estados más débiles, lo cual es improbable; siempre sobreviven los mejor individuos, no los peores. Así que la población sigue continuamente estancada.

Gráfica 1. Deriva genética.

¿Cuál es la forma de salir de esta situación? Pues solo hay dos soluciones: que vengan individuos de otras familias con "genes nuevos" y se entremezclen con la población local, como ocurre en los procesos de inmigración y/o emigración entre países; o bien que ocurra una mutación genética que haga que de repente un individuo se encuentre directamente mucho mejor adaptado, y a partir de él, sus descendientes también. Este último hecho se puede observar en la Gráfica 2.

Gráfica 2. Salida de la deriva por mutación.

Aplicación de la genética en la informática

Una vez que hemos dejado claras unas nociones de genética y evolución, es hora de pensar para qué nos vale esto en el ámbito informático y de la programación, pues para mucho. Aunque socialmente sea muy discutible, el hecho es que cada vez la especie humana es más sofisticada, mejor. Se han producido catástrofes climáticas, epidemias, guerras, etcétera, y la especie humana sigue avanzando, o sea que el resultado de los refinamientos sucesivos de un estado, para obtener otro estado mejor, nunca peor.

Si esto que, en principio parece cosa de magia en algo tan desconocido y variable como la vida humana, ha producido individuos tan sofisticados, ¿por qué no podría usarse la misma técnica de reproducción, selección y adaptación a procesos de cualquier tipo para obtener resultados más refinados cada vez? Esto es precisamente lo que persiguen los algoritmos genéticos.

Todo algoritmo genético, igual que la vida misma, se basa en dos principios básicos que hacen posible su funcionamiento:

- En condiciones normales, cerca de una solución buena, hay otra buena, muy buena o casi buena, al igual que al lado de una solución mala hay otra muya mala, mala o casi mala. Esto implica que no es probable que los descendientes de dos individuos muy bien adaptados, si no hay mutación, estén mal adaptados.
- Conseguir averiguar las características óptimas para un individuo es un problema complejo. Tanto es así que los algoritmos genéticos no garantizan una solución óptima a un problema, sino una que al menos es la mejor de todas las que hasta un momento dado se han estudiado y que, por tanto, puede estar cerca de una solución buena o muy buena.

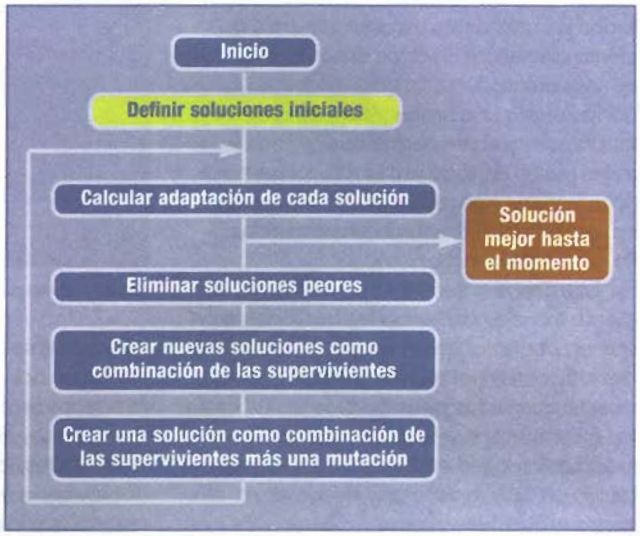
Tabla 1. Glosario de genética

Término	Descripción
Individuo	Un componente dentro de una especie. Por ejemplo, en la especie humana, una persona.
Cromosoma	Un conjunto de genes agrupados en una estructura concreta. Un cromosoma dota al individuo de una serie de características que, en conjunto, permiten que se adapte mejor o peor a las características de su entorno. Un individuo puede tener varios cromosomas.
Gen	Es la pieza mínima dentro de un cromosoma. Un gen dota al individuo que lo porta, en algunos de sus cromosomas, de una característica concreta: color de ojos, altura, tamaño de las pestañas, etc.
Combinación	Obviando muchos detalles que no nos interesan para nuestro propósito, es el proceso por el cual, cuando dos individuos se cruzan y tienen descendencia, ésta tiene una composición genética que es la mezcla de los padres. Un ejemplo, no del todo correcto, sería una pareja que está formada por una persona de raza blanca y otra de raza negra y tiene hijos mulatos. Esto es debido a que los hijos tienen una composición genética que es la mezcla de la de los padres.
Mutación	En una combinación de genes, los genes resultantes pertenecen o bien a la madre o bien al padre, aunque estén ordenados y distribuidos de otra forma, o mezclados. Una mutación es un proceso por el cual aparece espontáneamente, muy de vez en cuando, un gen en un cromosoma de un individuo, que no pertenece a ninguno de sus padres sino que es resultado de la perturbación de los genes por algún factor (radiación solar, radiación nuclear, etcétera). Por ejemplo, los nacidos en Chernobil tras el desastre nuclear; muchos de ellos nacieron con cánceres o deformaciones.
Adaptación	Dado un individuo, con sus características genéticas, se dice que está bien adaptado cuanto más le ayude ndichas características genéticas a sobrevivir en el ambiente donde se mueve. Se dice que está peor adaptado cuanto más le desfavorezcan esas características. Por ejemplo, el primer pájaro sería un reptil al que por una mutación le salieron alas. Esa característica le permitió escapar mejor de los depredadores y así sobrevivir y reproducirse, pasando a sus hijos, al mismo tiempo, la característica de las alas.

ESTRUCTURA TÍPICA DE UN ALGORITMO GENÉTICO

Nuestro objetivo es trasladar todo lo que hemos visto, a un programa informático. El objetivo del algoritmo genético va a ser buscar la mejor solución que sea posible dentro de un tiempo concreto, para un problema con un número de factores influyentes lo suficientemente elevado como para que probar todas las opciones no sea viable. Por tanto, lo primero que vamos a hacer es traducir los términos sobre genética de la Tabla 1, a conceptos informáticos. El resultado se puede ver en la Tabla 2.

La estructura típica de un algoritmo genético, es la que se muestra en la Grafica 3, donde se ve claro que el algoritmo genético en sí es sencillo de entender.



Gráfica 3. Aspecto general de un algoritmo genético.

Individuos y poblaciones

Lo primero que debemos hacer a la hora de crear un algoritmo genético es ver qué aspecto va a tener un individuo. Ya hemos visto que un individuo va a ser, en un algoritmo genético, una solución a al problema pero, ¿qué estructura de datos debemos usar? Eso depende mucho del problema seleccionado. Generalmente cada individuo se representa por un vector de valores donde cada uno de los valores representa una característica de la solución. Para verlo claro, supongamos que vamos a crear un algoritmo genético para calcular en poco tiempo cuáles deben ser las dimensiones de una caja para poder almacenar la mayor cantidad de arena posible. El volumen de una caja viene dado por la fórmula siguiente:

```
Volumen = ancho *
largo * alto
```

Entonces, un individuo estaría representado por un vector con tres elementos (genes) enteros donde cada uno de ellos representa una de las variables que influyen en que el volumen que permite la caja sea más o menos. En el número 6 de **Todo Programación**, Jaime Anguiano explica claramente cómo hay que optimizar la forma de representar las características de un individuo para que el programa no consuma excesivos recursos. En este artículo no nos vamos a preocupar de problemas de optimización puesto que está fuertemente ligado a lenguaje de programación utilizado y a la naturaleza del problema que se desea solucionar, pero es una característica que hay que tener bien presente.

En cuanto a la población, se define como el número de individuos (soluciones) que vamos a tener en un momento dado, cada uno con sus características y su capacidad de adaptación. También depende del problema que estemos tratando. A más soluciones o individuos simultáneos, más riqueza genética tendremos y más posibles soluciones buenas podremos conseguir al tener un amplio espectro de posibilidades, además de evitar la deriva genética de forma eficiente. Sin embargo, más recursos consumirá el algoritmo, tanto de memoria como de proceso.

En el caso más trivial debería haber cuatro individuos en una población: un padre, otro padre, un descendiente que sea combinación de ambos y otro más al que se le crea una mutación. Además, podemos establecer que la población siempre contenga un número determinado y fijo de individuos (los más buenos) o por el contrario, que la población pueda variar su número dinámicamente. Nosotros supondremos un tamaño fijo, puesto que es más sencillo para aprender.

Función de adaptación

Es el método o función que se tiene que encargar de evaluar cómo de buena es cada una de las soluciones (individuos) de nuestra población, y devuelve un número. Realmente es el grueso del algoritmo genético y es sin duda la parte más delicada, complicada y que más esfuerzos requiere por parte del desarrollador.

Evaluar una solución es fácil en ejemplos como el de la caja, donde la misma fórmula del volumen expresada unas líneas atrás puede valer. Sin embargo, en casos complejos, donde la relación entre las variables no está lo suficientemente clara, no hay una fórmula matemática directa y las variables que entran en juego son muchas, la cosa

Tabla 2. Glosario para algoritmos genéticos

Término	Descripción
Individuo	Cada una de las soluciones posibles a un problema que hemos planteado.
Cromosoma	Si la solución a un problema pasa por determinar el valor que deben tener todas las variables que influyen en el mismo, el estado concreto de todas ellas en un momento dado es un cromosoma. En nuestro ejemplo supondremos que cada individuo solo tiene un cromosoma, que será un vector de enteros.
Gen	Si la solución a un problema pasa por determinar el valor que deben tener todas las variables que influyen en el mismo, cada una de esas variables es un gen.
Combinación	Si tenemos el estado de todas las variables para dos soluciones de un problema y éstas son las que hasta el momento se adaptan mejor (solucionan mejor el problema), una combinación es una solución que tenga valores de ambas, lo cual, en principio, parece un buen camino para encontrar una salida mejor.
Mutación	Si tenemos el estado de todas las variables para dos soluciones de un problema, y éstas son las que hasta el momento se adaptan mejor (solucionan mejor el problema), una mutación es una solución que tenga valores de ambas. Esto, en principio, parece un buen camino para encontrar una solución mejor y tras ello, se cambia de forma aleatoria el valor de uno de los genes para probar nuevos dominios de soluciones y evitar la deriva genética.
Adaptación	Es un método para valorar cómo de bueno es un cromosoma (solución a un problema con muchos factores). Devuelve un número que indica cómo de buena es la solución que se está evaluando.

cambia. Por ejemplo, la predicción meteorológica o el diagnóstico médico. Todavía es más complicado cuando el estado de una variable influye en el valor de otras, como la temperatura y la humedad o el cansancio y la disminución de glóbulos rojos. Para la creación de un algoritmo genético, como para casi cualquier cosa en Inteligencia artificial, se requiere la ayuda de un experto en el tema que nos pueda asesorar sobre cómo evaluar las soluciones, pues es necesario un profundo conocimiento del modelo que queremos programar.

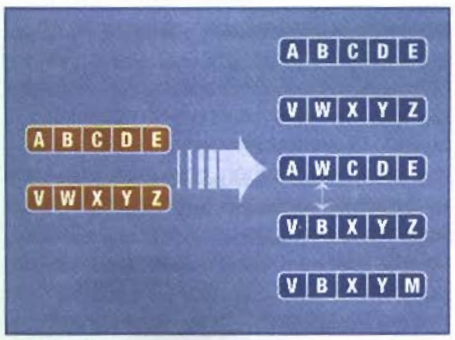
Función de selección

La función de selección se encarga de eliminar de la población (conjunto de soluciones) aquellas soluciones que se adaptan peor, o lo que es lo mismo, que son peores soluciones al problema. Es aquí donde se hace la selección natural que Darwin explicó en sus teorías. Los peores adaptados no sobreviven. En el ejemplo de la caja, si tuviésemos como soluciones cinco posibles cajas de distintas dimensiones, eliminaríamos aquellas que permitan menor volumen.

Función de reproducción

La función de reproducción es la que permite la recombinación genética. Simula la reproducción humana haciendo cruces entre los individuos que han sobrevivido a la aplicación de la función de selección. Generalmente, para ejemplos sencillos como los que vamos a ver, con poblaciones de tamaño fijo lo que se hace es crear tantos individuos nuevos como haga falta para recuperar la información. Generalmente el proceso se lleva a cabo como observamos en la Gráfica 4.

- Los supervivientes se quedan todos. Por algo han demostrado estar bien adaptados.
- Recombinando los genes de los supervivientes, creamos nuevos individuos. La recombinación se suele realizar cambiando un gen de un individuo, por otro del mismo tipo del individuo con el que se cruza (ver Gráfica 4). Se repite el proceso hasta que se recupera el tamaño de la población.
- Creamos una mutación en un gen de uno de los nuevos individuos. Esto es, ponemos un valor válido, pero aleatorio en uno de los genes, que no provenga de los supervivientes sino que "aparezca de la nada".



Gráfica 4. Creación de descendientes a partir de supervivientes. Mutación.

Con esto ya hemos creado una nueva población con una nueva generación de individuos, descendientes de los que sobrevivieron en la última selección. También aparece un individuo "raro", con una mutación que a lo mejor le sirve para adaptarse mejor (o todo lo contrario).

Ya hemos visto todas las partes de un algoritmo genético típico, por lo que ya podemos realizar un ejemplo completo.

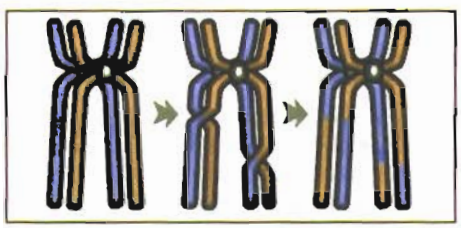
EJEMPLO PRÁCTICO: CONDICIONES BUENAS PARA PESCAR

Vamos a crear un algoritmo genético que nos sirva para averiguar rápidamente unas condiciones buenas para pescar, dado que entran muchas variables en juego. Vamos a definir todo lo que hemos explicado, pero para este ejemplo concreto.

Programaremos el algoritmo genético en Java, aunque podríamos usar cualquier otro lenguaje.

Número de cromosomas

¿Cuántos individuos vamos a tener en cada generación? Podríamos tener los que quisiéramos, pero cuantos más, más se complican todas las funciones que hemos comentado, así que cogeremos como modelo un caso muy simple, el que se muestra en la figura inferior. Cada generación tendrá dos progenitores, dos hijos por combinación y un hijo con mutación. Además vamos a suponer un número de individuos en cada población fijo: siempre cinco.



Modo en que se entremezclan genes de dos individuos para formar otro.

Definición de los genes

Vamos a suponer que hemos realizado las siguientes observaciones tras muchos años pescando en lagos:

- Entre -10 y 50 grados de temperatura, las pescas mejores se han realizado con temperaturas más altas, pues los peces se acercan más a la orilla.
- Cuando la presión atmosférica es alta, las pescas son mejores que cuando ésta es baja o normal. Presiones altas suelen coincidir con altas temperaturas, pero hay veces que no ocurre así.
- Si la noche anterior ha habido una luna brillante y grande (nueva o llena), la luz nocturna hace que los peces coman durante la noche y por tanto las pescas son malas al día siguiente. Con lunas pequeñas (menguante y creciente), este hecho se minimiza.
- Por la noche no se puede pescar en la mayoría de las ocasiones, por ley. Durante el día, las mejores horas de pesca varían según el tipo de peces y el tipo de embalse/lago.
- Si el lago tiene algas, los peces se alimentan de ellas y acuden menos a los cebos. Además, es más difícil pescar habiendo algas en medio.
- Si no llueve, los peces pican como de costumbre. Si llueve poco, los peces no pican demasiado, y si por el contrario hay lluvia abundante, el agua arrastras todo tipo de insectos y barro hasta el lago y los peces pican mucho porque se acercan a comérselos.
- Por último, la época del año buena para pescar va de abril hasta octubre. En abril comienza la temporada de pesca, que alcanza en junio su mejor momento y de ahí desciende hasta octubre. El resto de meses ocurre lo mismo pero a la inversa, se pesca cada vez menos.

Así que tenemos siete genes: temperatura, presión, luna, hora, algas, lluvia y mes. Usaremos por tanto un vector de enteros

Tabla 3. Posibles valores de los genes

Gen	Valores
Temperatura	(-10,...,50) °C
Presión	Baja, alta, normal
Luna anterior	Llena, nueva, creciente, menguante, sin luna (nublado)
Hora	Amanecer, mañana, mediodía, tarde, anochecer
Algas en el lago	Muchas, normal, pocas
Lluvia	Mucha, poca, ninguna
Mes	Enero, febrero,...,diciembre

para cada individuo. Todos los valores no son numéricos, pero habrá que sustituirlos por un número para almacenarlos en el vector. La sustitución que hemos hecho consta de números negativos y positivos, y puede verse en el código fuente en el CD que acompaña a la revista.

Función de adaptación

Como no somos expertos en condiciones de pesca, haremos una función de adaptación lo mejor que podamos, pero seguro que no obtendremos resultados todo lo precisos que hubiéramos deseado. Para ello sería imprescindible contar con la opinión de un experto.

De entre todos los genes que hemos definido para un individuo, no todos tienen la misma importancia o el mismo peso. Ahora mismo tenemos una representación numérica de cada gen, que será un valor positivo cuando ayuda a pescar más, y un valor negativo cuando empeora las capturas. La función de adaptación podría ser la suma de los valores de los genes, pero no sería correcto del todo, ya que no todos los genes son igual de importantes. De hecho, hay que ponderar los valores de los genes para que tengan el peso correcto. La ponderación (proveniente al 100 por cien de mi imaginación) se puede ver en la Tabla 4.

Por tanto, vamos a considerar como función de adaptación la suma ponderada (con los datos de dicha tabla) de los valores de todos los genes de cada individuo. Hay que observar que esto es lo que permitirá saber qué individuo muere, cuál se reproduce, etc. De lo expertos que seamos y de lo afinada que sea esta función, dependerá por completo el éxito del algoritmo genético. Generalmente suele ser necesario documentarse mucho sobre el problema antes de realizar esta función.

Función de selección

Para este ejemplo, la función de selección es bastante sencilla, puesto que la población con la que contamos es de cinco individuos (soluciones) y además es fija. Una vez que tengamos el valor resultante de la función de adaptación para los cinco individuos de la población, eliminaremos tres de ellos, los que peor función de adaptación ofrezcan. En caso de igualdad, elegiremos al azar uno de los empatados, pero en cualquiera de los casos debemos quedarnos tras el proceso de "selección natural" exclusivamente con dos individuos, los más preparados, que posteriormente tendrán que dar lugar a la regeneración de la población. En ejemplos con poblaciones variables y con un mayor número de individuos, el proceso de selección se complica, pero no nos merece la pena añadir dificultad a este ejemplo que es más que nada didáctico.

Tabla 4. Peso de cada gen del cromosoma

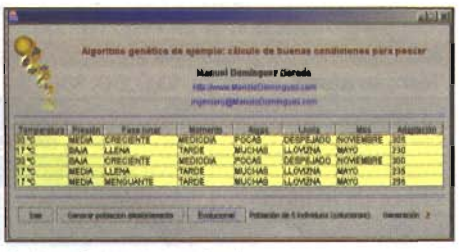
Gen	Peso en la función de adaptación
Temperatura	10%
Presión	5%
Luna anterior	30%
Hora	20%
Algas en el lago	10%
Lluvia	5%
Mes	20%

Función de reproducción

Siempre se llama a esta función cuando se ha realizado la selección de los individuos más adaptados. Por tanto, solo contaremos con dos individuos que consideraremos como "padres" para crear una nueva generación. Lo haremos de la siguiente forma:

- Cada hijo "normal" tendrá en común seis genes con uno de sus padres, y el gen restante será del otro padre. Lo más fácil es que seleccionamos el gen que deseamos modificar y lo intercambiemos entre "el padre" y "la madre" dando lugar a dos hijos, uno igual al padre excepto en un gen que será de la madre, y otro igual a la madre excepto en un gen que será del padre. Así tendremos dos hijos y dos padres ya en la población.
- De los dos hijos que hemos creado, tomaremos uno de ellos y modificaremos aleatóriamente uno de sus genes, dándole, eso sí, un valor dentro del rango permitido. Así ese individuo será hijo y tendrá parte de los padres, pero además presentará una mutación que puede modificar su adaptabilidad.

Con este proceso, ya tendremos de nuevo los cinco individuos de la población, todos partiendo de dos individuos buenos, que fueron seleccionados. Con esto se espera aprovechar el principio que comentábamos de que al lado de una buena solución, es posible encontrar otra similar o muy buena.



Captura del programa de ejemplo.

Deriva genética y gemelos

Ya hemos explicado lo que es la deriva genética, y dos gemelos son dos hijos de los mismos padres que tienen la misma composición genética entre sí, fruto de una división tras la fecundación. En nuestro ejemplo, aparecerán rápidamente gemelos (individuos genéticamente idénticos) pero serán fruto de la deriva genética, ya que cinco individuos con siete genes cada uno y cada gen con tres o cuatro posibles valores no dan mucho juego. Es deseable que no haya gemelos, ya que no aportan diversidad a la población; se pueden y deben poner medios en la función de reproducción para que no aparezcan. En este ejemplo se han dejado para mostrar que la deriva genética puede ser un gran problema para alcanzar la solución a un problema si no se establece un sistema de mutaciones.

Si se ejecuta y prueba el ejemplo, se puede ver que tras unas pocas generaciones se alcanzan soluciones más que aceptables, y que alcanzar el valor óptimo puede requerir mucha evolución en la población: ésa es la clave de los algoritmos genéticos.

Código del ejemplo

Dado lo farragoso que puede resultar el código de un algoritmo genético si no se explican los conceptos teóricos que hemos comentado, se ha optado por no poner código en el artículo sino explicar qué debe hacer cada función. En su lugar, el código está en el CD de la revista y el código importante del algoritmo genético se encuentra enmarcado claramente entre dos comentarios que dicen "CODIGO IMPORTANTE" (el resto es para el interfaz de usuario). Las funciones se llaman como las hemos nombrado en este artículo, y está todo lo suficientemente explicado como para entenderlo fácilmente viendo el código con la revista delante.

CONCLUSIÓN

¿Realmente funcionan los algoritmos genéticos? Pues la verdad es que sí, y lo hacen muy bien. No obstante, su utilización está aconsejada para resolver problemas complejos, con muchas variables, cuya resolución exacta consume un tiempo muy elevado, tanto que la solución ya no nos es útil cuando la averiguamos. Por ejemplo, la dirección de un huracán, no nos sirve de nada poder averiguar su dirección exacta si tardamos un mes. Con un algoritmo genético se encuentran soluciones siempre mejores en cada iteración, y en relativamente poco tiempo podemos solventar un problema con éxito (aunque no necesariamente de forma óptima).

1ª entrega, CD-ROM y Presentación de la obra por solo 5,99 euros

Vídeodigital

El primer curso para usuarios de cámaras digitales

- Aprende a trabajar con los programas de edición de vídeo más importantes.
- Cómo grabar, editar y producir vídeo digital paso a paso.
- Saca provecho a la cámara digital y al PC.
- No se requieren conocimientos previos para seguir esta obra.

Composición de la obra

20 coleccionables

divididos en:

Teoría
Sección donde se abordan las técnicas de grabación, el funcionamiento de las cámaras, el estudio de las principales aplicaciones profesionales de edición y en general todo lo relacionado con este campo.

Práctica
Ejercicios para aprender técnicas de trabajo, cómo utilizar Adobe Premiere a la perfección y cómo montar los primeros clips de vídeo propios.

Tutorial
Área donde se complementa la parte práctica con el estudio de herramientas adicionales, como son: conversores, codecs, programas de retoque y efectos, aplicaciones de edición de audio, y todo tipo de utilidades relacionadas con la producción de vídeo.

Contenido CD-ROM
Descripción del contenido del CD-ROM y explicación de cómo instalar el software.

20 CD-ROMs

Todo el software actual para la edición de vídeo, demos de programas profesionales, aplicaciones completas, utilidades, etc.

Todos los CD-ROMs incorporan un asistente para mayor comodidad en la búsqueda de contenidos e instalación de programas.

Studio PRESS

C/ del Río Ter, Nave 13
Polígono Industrial "El Nogal"
28110 Algete (Madrid)
Tel.: 916280203
Fax: 916280935
e-mail: video@iberprensa.com
www.iberprensa.com

ya a la venta **semanalmente** en quioscos y centros comerciales