

¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
 Ese apoyo que siempre quiso tener...

1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
 HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
 Gestor de contenidos (Alfresco)
 Aplicaciones híbridas

Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

Control de autenticación y
 acceso (Spring Security)
 UDDI
 Web Services
 Rest Services
 Social SSO
 SSO (Cas)

JPA-Hibernate, MyBatis
 Motor de búsqueda empresarial (Solr)
 ETL (Talend)

Dirección de Proyectos Informáticos.
 Metodologías ágiles
 Patrones de diseño
 TDD

BPM (jBPM o Bonita)
 Generación de informes (JasperReport)
 ESB (Open ESB)

adictos al trabajo

Encuentra un trabajo que te guste y no volverás a trabajar ni un sólo día de tu vida
Confucio

E-mail:

Contraseña:

Entrar

[Inicio](#) [Quiénes somos](#) [Tutoriales](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#)
[Charlas](#) [Más](#)

Estás en:
[Inicio](#) [Tutoriales](#)

Introducción a los algoritmos genéticos: como implementar un algoritmo gené...

DESARROLLADO POR:
 Jose Carlos López

Ingeniero Técnico en Informática

Catálogo de servicios Autentia



Anuncios Google

[Herencia](#)

[Genetica](#)

[Antecedentes](#)

Fecha de publicación del tutorial: 2009-02-26



Share |

[Regístrate para votar](#)

Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA

Índice de contenidos.

- 1. ¿Qué son los algoritmos genéticos?.
- 2. Antecedentes históricos.
- 3. Representación.
- 4. Algoritmo.
- 5. Función de evaluación y función de aptitud (fitness).
- 6. Operadores genéticos.
- 7. ¿Y cómo puedo implementar un algoritmo genético en Java?.

1. ¿Qué son los algoritmos genéticos?.

Basados en modelos computacionales de la evolución biológica natural, los algoritmos genéticos pertenecen a la clase de los algoritmos evolutivos, junto con la programación evolutiva, la evolución de estrategias y la programación genética.

Los **algoritmos genéticos (AGs)** son mecanismos de búsqueda basados en las leyes de la selección natural y de la genética. Combinan la supervivencia de los individuos mejor adaptados junto con operadores de búsqueda genéticos como la mutación y el cruce, de ahí que sean comparables a una búsqueda biológica. Fueron desarrollados por John Holland y Rechemberg que crearon algoritmos de optimización imitando los principios básicos de de la naturaleza. Estos algoritmos se utilizan con éxito para gran variedad de problemas que no permiten una solución eficiente a través de la aplicación de técnicas convencionales.

Últimas Noticias

[Comic Flash sobre Las factorias de software retos y oportunidades](#)

[Mi primer coderetreat, Chispas!!!](#)

[Entregamos nuestro primer diploma ...](#)

[Comic Flash de Head Hunting](#)

[XI Charla Autentia - Mule - Recordatorio](#)

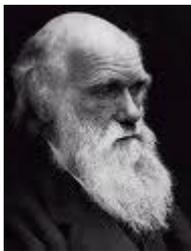


Tiene asimismo aplicaciones variadas en ingeniería, como es el diseño de piezas (turbinas, rotores, etc.), diseño y control de redes, planificación de tareas, síntesis de mecanismos, diseño de tolerancias, etc. y aplicaciones en otros gremios como en sistemas de computación paralelos (paralelización automática de programas secuenciales), química (optimización de procesos de producción, diseño tecnológico y de instalaciones), negocios y comercio (modelización de sistemas económicos complejos, predicción de mercados), medicina (análisis de datos en medicina, diagnóstico automático) o gestión (análisis de datos en gestión, asistentes de gestión, sistemas automáticos de decisión).

Para la ingeniería, los algoritmos genéticos y la programación evolutiva en general presentan oportunidades de plantearse problemas que no permiten una solución eficiente a través de la aplicación de técnicas convencionales.

2. Antecedentes historicos.

Los algoritmos genéticos tienen sus antecedentes en la biología y comienzan con Charles Darwin, que con su libro El origen de las especies por medio de la selección natural o la preservación de las razas favorecidas en su lucha por la vida, nos habla sobre los principios de la selección natural.



Los principios básicos de los algoritmos genéticos se derivan de las Leyes de la Vida Natural descritos por Darwin:

- Existe una población de individuos con diferentes propiedades y habilidades. Así mismo existe una limitación sobre el número de individuos que existen en una determinada población.
- La naturaleza crea nuevos individuos con propiedades similares a los individuos existentes.
- Los individuos más prometedores se seleccionan más a menudo para la reproducción de acuerdo con la selección natural.

Los algoritmos genéticos imitan los principios de la vida descritos y los utilizan para propósitos de optimización.

Una de las principales deficiencias del argumento de Darwin es que, a pesar de que la herencia juega un papel preponderante en su teoría, no ofrece una explicación acerca de su funcionamiento. Sin embargo, desde Mendel se conoce que la herencia se produce a través del código genético presente en las células reproductivas.

3. Representación.

Todos los organismos vivos están constituidos por células, y cada célula contiene uno o más cromosomas (cadenas de ADN), que le sirven como una especie de "plano" al organismo. Un cromosoma puede ser conceptualmente dividido en genes cada uno de los cuales codifica una proteína. En términos generales, se puede decir que un gen se codifica como si fuera un rasgo, como puede serlo el color de ojos. Cada gen se encuentra en una posición particular del cromosoma, y está formado por alelos.

[Histórico de NOTICIAS](#)

Últimos Tutoriales

 [Cómo alcanzar el éxito en el sector de la informática.](#)

 [IAQ \(Interesting Asked Questions\), recordando la posición del scroll con el soporte de jQuery.](#)

 [Publicar un repositorio Mercurial con Apache](#)

 [Liferay IDE](#)

 [Rendimiento en espacio y transferencia de un servidor Subversion](#)

Últimos Tutoriales del Autor

 [Tutorial de la API de Google Maps](#)

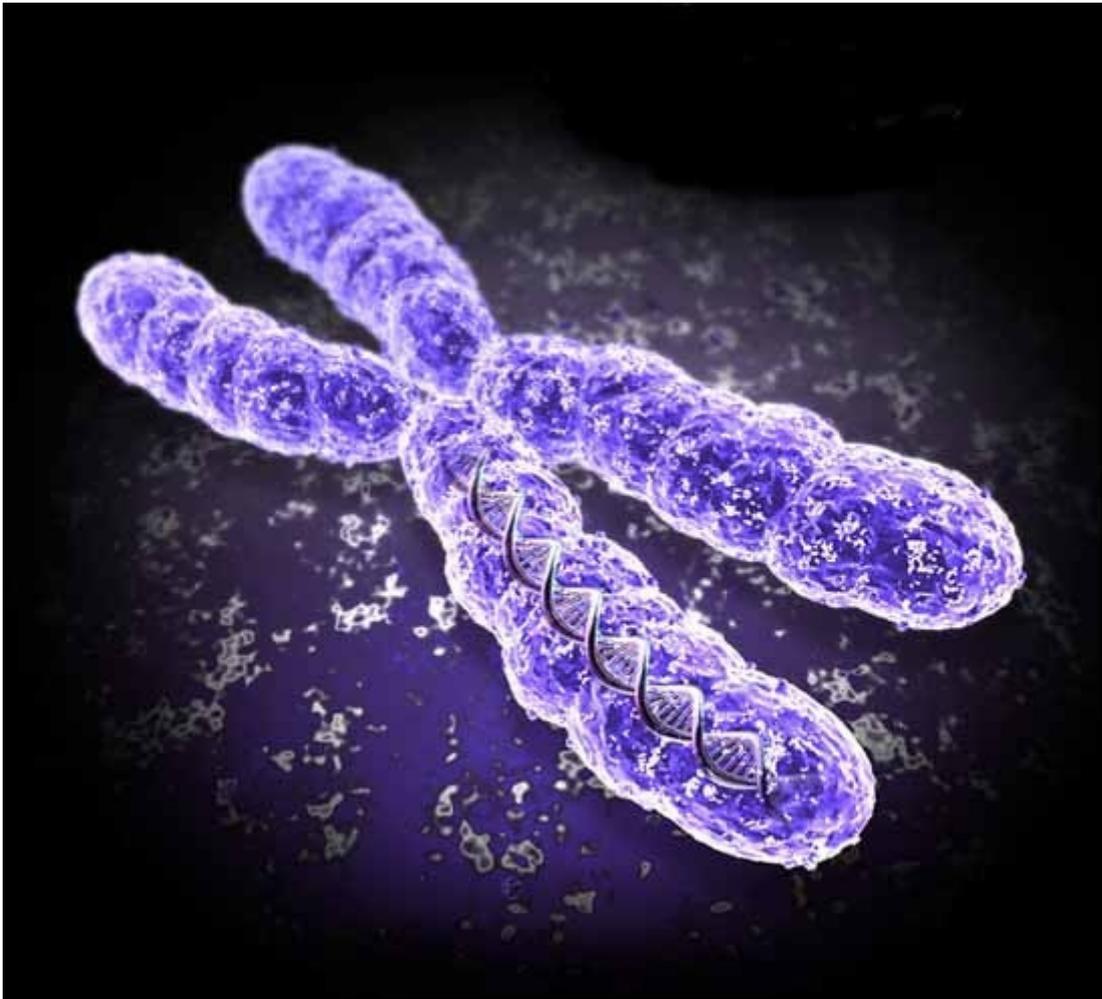
 [Validando XML contra Schema](#)

 [Crear un archivo JAR con java.util.jar](#)

 [Accediendo a rutinas C y C++ desde Oracle](#)

 [Hibernate 3 y los tipos de datos para cadenas largas](#)

Síguenos a través de:



ofertas de

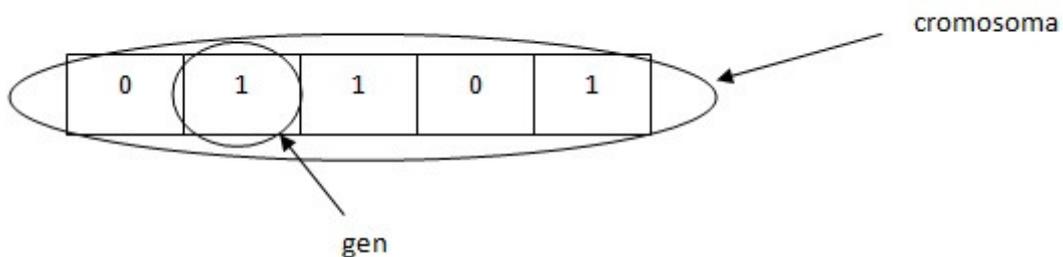
-30
as -
tricidad -
ONA.

-24
as Sin
logar -

-25
nformación
alista /
nador -
ONA.

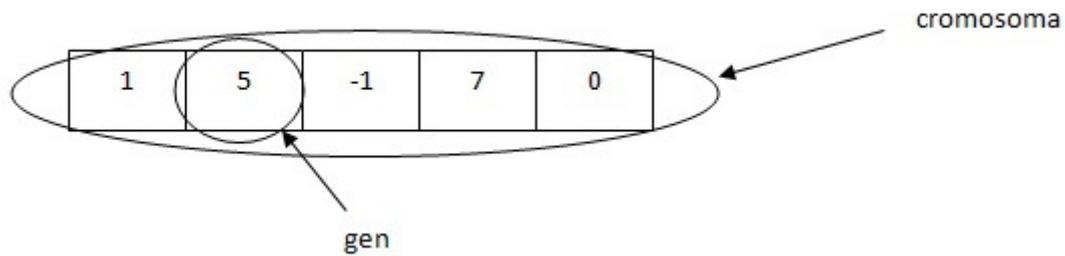
Se supone que los individuos (posibles soluciones del problema), pueden representarse como un conjunto de parámetros (que denominaremos genes), los cuales agrupados forman una ristra de valores, a menudo referida como cromosoma. Debe existir una representación de estos genes para poder utilizarlos posteriormente en el algoritmo genético y dotarles de unos valores. Se pueden considerar tres tipos básicos de representación o codificación de los genes:

- Binaria: en ella se utiliza un vector cuya longitud es la del número de genes de cada individuo y el valor que puede tomar cada elemento es un número binario.

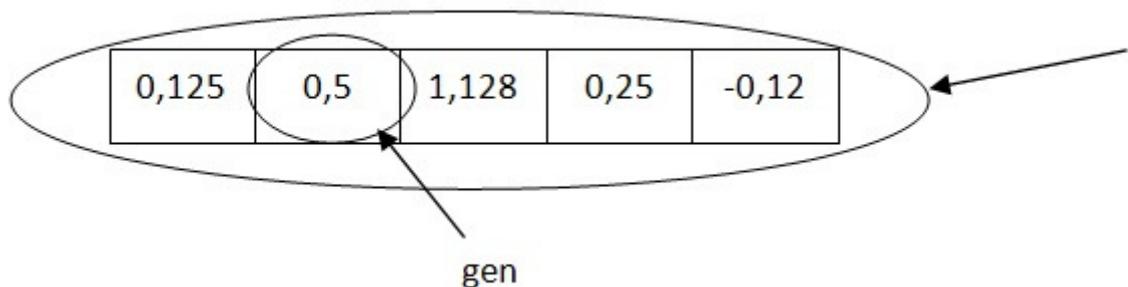


- Entera: en ella se utiliza un vector cuya longitud es la del número de genes de cada individuo y el valor que puede tomar cada elemento es un número

entero.



- Real: en ella se utiliza un vector cuya longitud es la del número de genes de cada individuo y el valor que puede tomar cada elemento es un número real.



Un individuo es una solución potencial al problema que se trata. Cada individuo contiene un cromosoma. A un conjunto de individuos se le denomina población. El fitness de un individuo es la evaluación de la función de evaluación e indica **qué tan bueno es el individuo (es decir, la solución al problema) con respecto a los demás.**

4. Algoritmo.

Desarrollado por John H. Holland, el algoritmo genético opera entonces a nivel de genotipo de las soluciones mediante la siguiente secuencia:

1. Comenzar con una población inicial, la cual puede ser generada de manera aleatoria.
2. Calcular el fitness (aptitud) de cada individuo.
3. Aplicar el operador de selección con base en el fitness de la población.
4. Aplicar los operadores genéticos de reproducción, cruce y mutación a la población actual para generar a la población de la siguiente generación.
5. Ir al paso 2 hasta que la condición de parada se satisfaga.
6. Cuando se cumple la condición de parada, se devuelve al mejor individuo encontrado (bien el mejor de todas las generaciones, bien el mejor de la última generación).

Al igual que en muchas otras heurísticas, el comportamiento del algoritmo genético es altamente dependiente de los parámetros iniciales (tamaño de la población, porcentaje de cruce, porcentaje de mutación, número de generaciones, etc.), por lo que será necesario ajustar esos parámetros para tratar de mejorar la solución para los objetivos del problema.

A cada iteración de este proceso se le denomina una generación. Un algoritmo genético típicamente se itera de 50 a 500 o incluso más generaciones. El conjunto entero de generaciones se denomina una ejecución. Al final de una ejecución existen a menudo uno o varios cromosomas altamente adecuados en la población, y que pueden ser elegidos como solución al problema.

La función de evaluación o de fitness de un problema es realmente la función que se desea optimizar. **Su diseño es junto con el del genotipo, una de las características más importantes a la hora de encontrar la mejor solución a un problema.**

Como se ha podido ver en el algoritmo, en cada generación se selecciona a un conjunto de los mejores individuos (paso 3) y se les modifica para generar la siguiente generación mediante los llamados operadores genéticos (paso 4). Estos operadores son tres: reproducción, cruce y mutación. El operador de cruce intenta simular la reproducción sexual, de tal manera que los individuos resultantes del cruce contendrán información de varios individuos. El operador de mutación simula la mutación biológica, de tal manera que los individuos mutados serán ligeramente diferentes de los individuos originales. El operador de reproducción simplemente copia sin modificación un individuo de una generación a la siguiente. Así en la siguiente generación, algunos individuos simplemente habrán sido copiados, y/o cruzados y/o mutados.

5. Función de evaluación y función de aptitud (fitness).

La función de evaluación generalmente es la función objetivo, es decir, **es lo que se quiere llegar a optimizar** (ej: número de aciertos, número de movimientos, etc.). Es necesario decodificar la solución presente en el cromosoma para evaluarla.

La función de aptitud (fitness) es la que permite valorar la aptitud de los individuos y debe tomar siempre valores positivos.

Ambas funciones suelen ser iguales, pero puede ser que la función objetivo sea muy compleja, tome valores negativos, o no proporcione un valor numérico y, por lo tanto, sea necesario definir una función de aptitud diferente.

En un algoritmo genético la información ha de codificarse para poder trabajar adecuadamente con ella. Como hemos visto, existen numerosos sistemas de codificación, aunque nosotros utilizaremos la codificación real.

Una vez definido el sistema de codificación a emplear se verá cómo actúan los operadores básicos de selección, cruce y mutación sobre este código.

6. Operadores genéticos.

En su forma más simple, un algoritmo genético consta de los siguientes operadores genéticos: selección, reproducción, cruce (crossover) y mutación.

Selección

El proceso de selección sirve para escoger a los individuos de la población mejor adaptados, para que actúen de progenitores de la siguiente generación. En la naturaleza existen varios factores que intervienen para que un individuo pueda tener descendencia. El primero de todos es que consiga sobrevivir, ya sea porque no es devorado por depredadores, o porque sea capaz de procurarse alimento. Lo segundo es que encuentre pareja para reproducirse. El último factor es que la combinación de ambos individuos sea apta para crear un nuevo individuo.

Sin embargo, en la realidad es posible que "el mejor" individuo no pueda reproducirse, pero otro individuo de "peor calidad" pueda conseguirlo. Aunque este hecho es menos probable, sigue siendo posible.

En los algoritmos genéticos, la selección es un conjunto de reglas que sirven para elegir a los progenitores de la siguiente generación. Estos progenitores se reproducirán (cruzamiento genético) y generarán descendencia.

Un sistema muy utilizado en los algoritmos genéticos es la selección por torneo. Este sistema consiste en escoger aleatoriamente de la población un cierto número de individuos. De esos individuos se escoge el mejor de todos para ser el padre. Para escoger la madre se repite el proceso: se escoge aleatoriamente a un número de individuos de la población y se elige al individuo con mejor calidad. Este sistema garantiza un mínimo de diversidad, ya que no siempre se elegirá al mejor individuo de la población para tener descendencia. Pero, por el contrario, existen grandes posibilidades de que éste tenga descendencia, ya que si es escogido en algún torneo, será el vencedor.

Reproducción

En este contexto, se entenderá por "reproducción" la clonación de un individuo. Es decir, un individuo pasará a la siguiente generación sin modificación. De esta manera, la reproducción es un operador genético que se contrapone al cruce y la mutación, puesto que estos últimos modifican los individuos que pasan a la siguiente generación. El objetivo de la reproducción es mantener en la siguiente generación a individuos con fitness alta de la presente generación.

Relacionado con el concepto de reproducción está el de "elitismo", el cual mantiene a los mejores individuos de una generación a la siguiente, para que no se pierda su información.

Cruce

Durante esta fase se cruzan o mezclan los individuos seleccionados en la fase anterior. Es decir, los genes de los dos padres se mezclan entre sí para dar lugar a los diferentes hijos. Existen diversos métodos de cruce, pero los más utilizados son los siguientes:

- **Cruce basado en un punto:** los dos individuos seleccionados para jugar el papel de padres, son recombinados por medio de la selección de un punto de corte, para posteriormente intercambiar las secciones que se encuentran a la derecha de dicho punto. Es decir, los genes del padre1 a la izquierda del punto de corte forman parte del hijo1 y los situados a la derecha formarán parte del hijo2, mientras que con el padre2 sucederá lo contrario.
- **Cruce punto a punto:** este tipo de cruce es similar al anterior pero realizándose para cada gen de los padres. Por tanto, en este cruce los genes pares del padre1 formarán parte del hijo1 y los genes impares formarán parte del hijo2, mientras que para el padre2 sucederá lo contrario.
- **Cruce multipunto:** en este tipo de cruce se selecciona aleatoriamente la cantidad de puntos que se van a utilizar para el cruce. De esta forma, y de manera análoga al anterior cruce, se irán intercambiando los genes para formar los dos nuevos hijos.
- **Cruces específicos de codificaciones no binarias:** Para este tipo de codificación se pueden definir, además de los anteriores, otros tipos de operadores de cruce:
 - **Media:** el gen de la descendencia toma el valor medio de los genes de los padres. Tiene la desventaja de que únicamente se genera un descendiente en el cruce de dos padres.
 - **Media geométrica:** cada gen de la descendencia toma como valor la raíz cuadrada del producto de los genes de los padres. Presenta el problema añadido de qué signo dar al resultado si los padres tienen signos diferentes.
 - **Extensión:** se toma la diferencia existente entre los genes situados en las mismas posiciones de los padres y se suma el valor más alto o se resta del valor más bajo. Solventa el problema de generar un único descendiente.

Mutación

La mutación se considera un operador básico, que proporciona un pequeño elemento de aleatoriedad en los individuos de la población. Si bien se admite que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, el operador de mutación es el responsable del aumento o reducción del espacio de búsqueda dentro del algoritmo genético y del fomento de la variabilidad genética de los individuos de la población. Existen varios métodos para aplicar la mutación a los individuos de una población, pero el más comúnmente utilizado es el de mutar un porcentaje de los genes totales de la población.

Este porcentaje de genes a mutar se puede seleccionar de dos maneras, de forma fija, especificando el mismo porcentaje de mutación a todas las generaciones del algoritmo genético y de forma variable, es decir, modificando el porcentaje de mutación de una generación a otra, por ejemplo reduciéndolo. De esta manera, se consigue hacer una búsqueda más amplia y global al principio e ir reduciéndola en las siguientes generaciones.

Con otro tipo de codificaciones (por ejemplo codificación real) existen otras opciones de mutación, aplicadas con una probabilidad generalmente pequeña:

- Mutación al azar: Modifica el valor de un gen asignando con un nuevo valor que se encuentra dentro de un determinado rango. El nuevo valor es independiente del valor previo del gen.
- Mutación gaussiana: Dado un cromosoma p con un gen seleccionado para la mutación i, se le aplica una distribución normal N de media pi y desviación estándar s (parámetro). Alternativamente se puede disminuir el valor de s a medida que aumenta el número de generaciones.

7. ¿Y como puedo implementar un algoritmo genético en Java?.

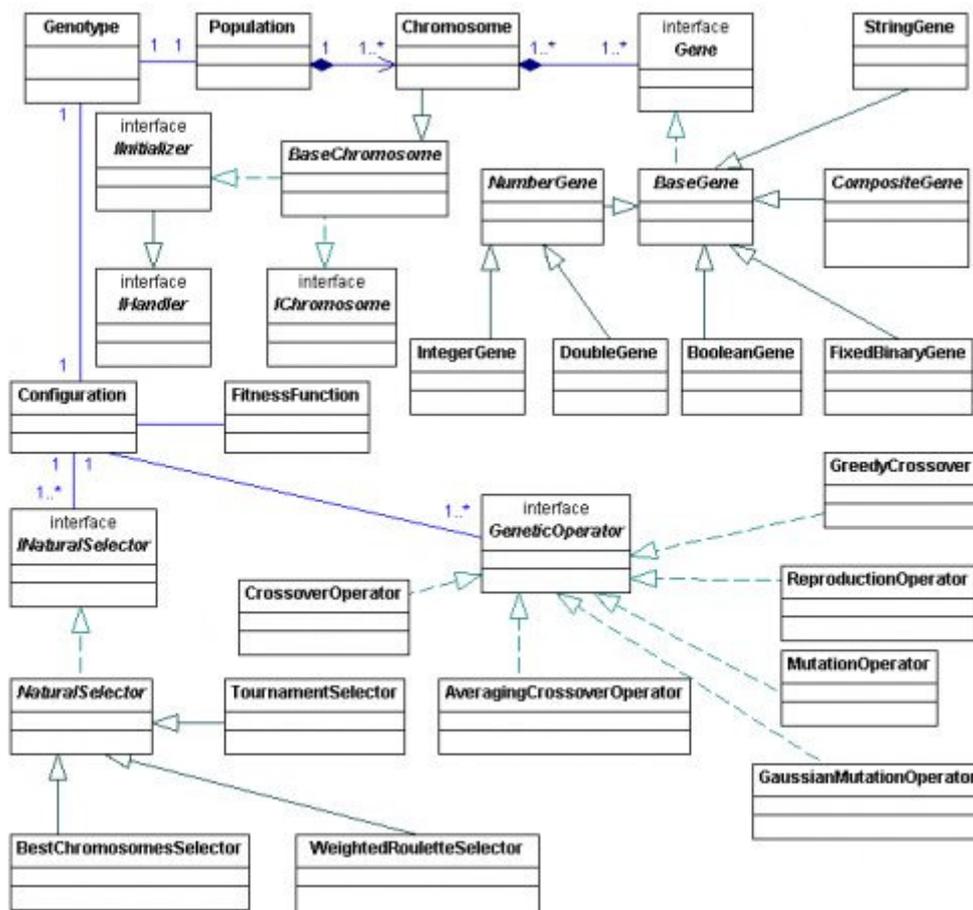
Para la implementación de algoritmos genéticos, disponemos de un framework para Java llamado

JGAP (Java Genetic Algorithms Package)

JGAP proporciona los métodos y mecanismos básicos para implementarlo. Fue diseñado para que fuera muy fácil de usar, siendo altamente modular y configurable, llegando a ser realmente fácil crear incluso nuevos y personalizados operadores genéticos.

Se distribuye bajo licencia GPL. En su página web podemos encontrar gran variedad de ejemplos de uso y documentación.

A continuación mostramos el diagrama de clases implementado en JGAP:



Como veis, ya están implementados la mayoría de operadores de los que hemos hablado anteriormente (cruce, selección, etc) y los tipos de genes más comunes. De todas formas, siempre podrás crear los operadores y tipos de gen que necesites.

El framework JGAP se encarga de ejecutar los operadores en cada ejecución, devolviendo el individuo con mayor Fitness (permitiéndonos incluso obtener cualquier individuo de la población en cualquier ejecución).

En la siguiente dirección, podéis ver lo fácil que sería implementar un algoritmo genético desde cero.

<http://jgap.sourceforge.net/doc/tutorial.html>

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

Copyright 2003-2010 © All Rights Reserved. [Inicio](#) | [Contacto](#) | [condiciones de uso](#) | [Banners](#) |

