Implementación de operadores genéticos para Programación de Expresiones Genéticas (GEP).

Implementation of genetic operators for Gene Expression Programming (GEP).

Dianet Utria Pérez

Alain Guerrero Enamorado

José Carlos Santiesteban Rojas

Universidad de las Ciencias Informáticas. La Habana. Cuba.

Universidad de las Ciencias Informáticas. La Habana. Cuba.

Universidad de las Ciencias Informáticas. La Habana. Cuba.

Resumen

En los últimos años, el desarrollo de algoritmos evolutivos para las tareas de clasificación, una de las más estudiadas en los campos de aprendizaje de máquinas y minería de datos que permite pronosticar la tendencia futura de los datos, lo cual favorece en gran medida la rapidez y eficiencia en la toma de decisiones en diferentes campos, ha adquirido gran interés. Esta tarea consiste básicamente en encontrar una función capaz de identificar el conjunto de atributos de un objeto (variables predictivas) con una etiqueta o clase de identificación (variable categórica). Disímiles son las aplicaciones apreciables de la clasificación en campos tan diversos entre sí como la investigación médica, la seguridad, la administración pública, logística, relación con el cliente, etc. El presente trabajo se centra en un reciente paradigma de la computación evolutiva, la Programación de Expresiones Genéticas (GEP), específicamente en la implementación sobre el framework JCLEC de los operadores genéticos propuestos en dicho paradigma.

Palabras clave: Programación de Expresiones Genéticas (GEP), operadores genéticos, inversión, JCLEC.

Abstract

In recent years, the development of evolutionary algorithms for classification tasks, one of the most studied in the fields of machine learning and data mining that allows predicting the future trend of data, which greatly favors the speed and efficiency in making decisions in different fields, has acquired great interest. This task consists basically to find a function capable of identifying the set of attributes of an object (predictive variables) with a label or identification class (categorical variable). Dissimilar are the appreciable applications of classification in fields as diverse as medical research, security, public administration, logistics, customer relations, etc. The present work focuses on a recent paradigm of evolutionary computing, the Gene Expression Programming (GEP), specifically in the implementation in the JCLEC framework of the genetic operators proposed in this paradigm.

Keywords: Gene Expression Programming (GEP), genetic operators, inversion, JCLEC.

Introducción

La evolución natural de las especies comenzó a ser vista como un proceso de aprendizaje, mediante el cual la naturaleza dota a las especies de diferentes mecanismos buscando hacerlas más aptas para sobrevivir, con el trabajo del fisiólogo estadounidense Walter Bradford Cannon, The Wisdom of the Body (La sabiduría del cuerpo) (Cannon, 1932) desde los años treinta del siglo XX. En este libro, Cannon plantea que el proceso evolutivo es algo similar al aprendizaje por ensayo y error que suele manifestarse en los humanos. Por su parte el célebre matemático, lógico, científico de la computación, entre otros, de origen inglés Alan Mathison Turing reconoció también una conexión entre la evolución y el aprendizaje de máquina en (Turing, 1950), considerado hoy un clásico en Inteligencia Artificial.

Posteriormente, diversos investigadores desarrollaron algoritmos inspirados en la evolución natural para resolver diferentes tipos de problemas. Así, durante los 1960s y 1970s sucedieron una serie de corrientes de investigación independientes que comenzarían a formar lo que hoy se conoce como computación evolutiva. La computación evolutiva conocida también como algoritmos evolutivos constituye una rama de la inteligencia artificial. Las técnicas de Computación Evolutiva pueden producir soluciones altamente optimizadas en una amplia gama de entornos problemáticos. Existen muchas variantes y extensiones, lo cual hace cada vez más difícil distinguir las diferencias entre los distintos tipos de algoritmos evolutivos existentes, entre los cuales se encuentran principalmente: Programación Evolutiva (EP), Estrategias Evolutivas (ESs), Algoritmos Genéticos (GAs) y Programación Genética (GP). Cada uno de estos paradigmas se originó de manera independiente y con motivaciones diferentes.

En 1999 Cándida Ferreira crea un nuevo tipo de algoritmo evolutivo denominado Gene Expression Programming (Programación de Expresiones Genéticas, GEP) (Ferreira, 2001). La GEP es, al igual que los GAs y la GP, un algoritmo genético que utiliza poblaciones de individuos, los selecciona de acuerdo con la aptitud e introduce variación genética usando uno o más operadores genéticos. La diferencia fundamental entre los tres algoritmos reside en la naturaleza de los individuos: en GAs, los individuos son cadenas lineales de longitud fija (cromosomas); en GP, los individuos son entidades no lineales de diferentes tamaños y formas (árboles de análisis); y en GEP los individuos se codifican como cadenas lineales

de longitud fija (el genoma o cromosomas) que luego se expresan como entidades no lineales de diferentes tamaños y formas (es decir, representaciones de diagrama simple o árboles de expresión). (Ferreira, 2001).

Con GEP es posible crear automáticamente programas de computadora. Estos programas de computadora pueden tomar muchas formas: pueden ser modelos matemáticos convencionales, redes neuronales, árboles de decisión, modelos sofisticados de regresión no lineal, modelos de regresión logística, clasificadores no lineales, estructuras polinomiales complejas, circuitos lógicos y expresiones, así como otras muchas. Pero independientemente de su complejidad, todos los programas GEP están codificados en estructuras lineales muy simples: los cromosomas. Estos cromosomas lineales simples son un gran avance porque, sin importar qué, siempre codifican programas de computadora válidos. Siendo posible mutarlos y seleccionar los mejores para reproducir, luego crear mejores programas, y así sucesivamente. Este es, por supuesto, uno de los requisitos previos para que un sistema evolucione de manera eficiente, buscando cada vez mejores soluciones para todo tipo de problemas.

A través del uso de dicho paradigma se brinda la posibilidad de evolucionar cromosomas multigénicos utilizando funciones de unión entre dichos genes. Adicionalmente se dispone del siguiente conjunto de operadores genéticos: Mutación, Inversión, Transposición IS, Transposición RIS, Transposición de genes, Recombinación de un punto, Recombinación de dos puntos, Recombinación de genes. Siete de estos ocho operadores se encuentran actualmente implementados sobre el framework Java Class Library for Evolutionary Computation (Librería de Clases Java para Computación Evolutiva, JCLEC) (Ventura, y otros, 2007) desarrollado hace poco más de diez años. Sin embargo, en varios de ellos existen inconsistencias, así como falta por implementar el operador de Inversión. Debido a ello, el presente trabajo centra su esfuerzo precisamente corregir las inconsistencias existentes en estos operadores, así como en la implementación sobre JCLEC del operador Inversión.

Desarrollo

Programación de Expresiones Genéticas (GEP)

La GEP, inventada por Cándida Ferreira en 1999 (Ferreira, 2001), tiene como predecesores los GAs y la GP. Incorpora tanto los cromosomas lineales simples de longitud fija similares a los utilizados en GA como las estructuras ramificadas de diferentes tamaños y formas similares a los árboles de análisis de GP. Y dado que las estructuras ramificadas de diferentes tamaños y formas están totalmente codificadas en los cromosomas lineales de longitud fija, esto equivale a decir que, en GEP, el genotipo y el fenotipo se separan finalmente el uno del otro y el sistema puede ahora beneficiarse de todas las ventajas evolutivas que esto produce. (Ferreira, 2006).

Por lo tanto, el fenotipo de la programación de expresiones genéticas consiste en el mismo tipo de estructura ramificada utilizada en GP. Sin embargo, las estructuras ramificadas desarrolladas por GEP (llamadas árboles de expresión) constituyen la expresión de un genoma totalmente autónomo. Así solo el genoma (ligeramente modificado) pasa a la siguiente generación. En consecuencia, ya no es necesario replicar y mutar las estructuras más bien engorrosas ya que todas las modificaciones tienen lugar en una estructura lineal simple que solo más tarde se convertirá en un árbol de expresión. La idea central de la GEP consistió en la invención de cromosomas capaces de representar cualquier árbol de análisis sintácti-

co, creándose para ello el lenguaje Karva, para leer y expresar la información codificada en los cromosomas. Además, como se plantea en (Ferreira, 2006) la estructura cromosómica fue diseñada para permitir la creación de múltiples genes, cada uno de los cuales codifica un árbol de programa o sub-expresión más pequeño. Vale la pena enfatizar que la programación de expresiones genéticas es el único algoritmo genético con múltiples genes. De hecho, la creación de individuos más complejos compuestos por múltiples genes está extremadamente simplificada en sistemas de genotipo/fenotipo verdaderamente funcionales.

La base de toda esta novedad reside en la estructura simple pero revolucionaria de los genes GEP. Esta estructura no solo permite la codificación de cualquier programa concebible sino que también permite una evolución eficiente. Esta organización estructural versátil también permite la implementación de un conjunto muy poderoso de operadores genéticos que luego pueden buscar de manera muy eficiente el espacio de la solución. Como en la naturaleza, los operadores de búsqueda de programación de expresiones genéticas siempre generan estructuras válidas (ya sean expresiones matemáticas complejas o redes neuronales artificiales complejas o árboles de decisión sofisticados) y, por lo tanto, son notablemente adecuadas para crear diversidad genética. (Ferreira, 2006).

En la GEP, el genoma o cromosoma consiste en una cadena lineal, simbólica de longitud fija compuesta por uno o más genes de igual longitud. Contrario a lo que pudiera pensarse dada su longitud fija, los cromosomas de GEP codifican árboles de expresión con diferentes tamaños y formas. En GEP, el sitio de inicio de un gen es siempre la primera posición del mismo, mientras que el punto de terminación no siempre coincide con la última posición de un gen. Es común que los genes GEP tengan regiones no codificadoras a continuación del punto de terminación. En la Figura 1 se muestran dos genotipos, compuestos por un único gen de longitud diez, así como los fenotipos correspondientes a cada uno de ellos, utilizando la notación Karva (Ferreira, 2006):

```
0123456789
Q*-aabcdbb
```

```
0123456789
Q*-+abcdbb
```

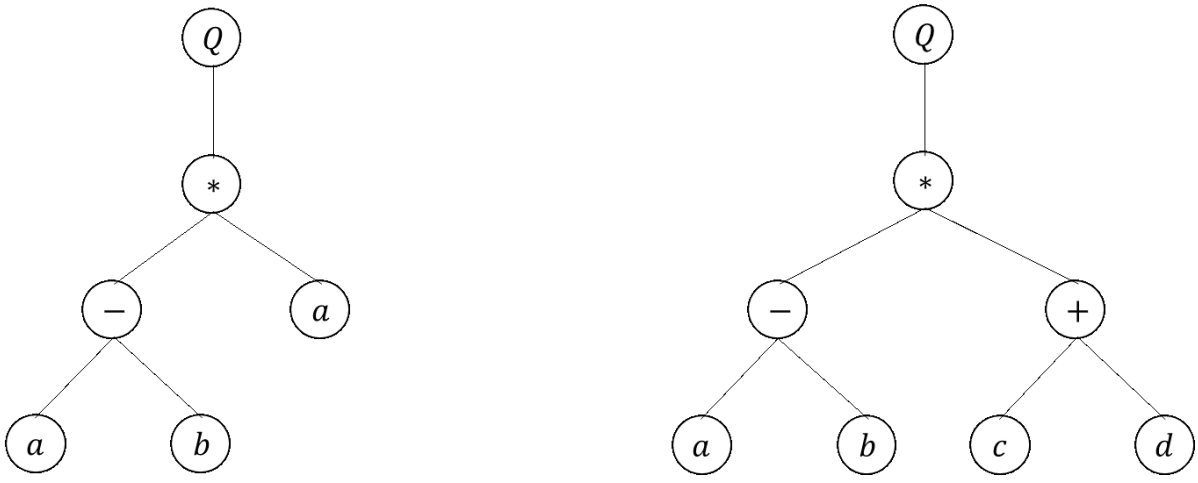

Figura 1 Genotipos GEP de longitud diez (superior) y fenotipos GEP correspondientes (inferior).

Los genes en GEP, están compuestos por dos dominios diferentes: un dominio de cabecera y un dominio de cola, cada uno con diferentes propiedades y funciones. El dominio de la cabeza se utiliza principalmente para codificar las funciones elegidas para el problema en cuestión, mientras que la cola funciona como un búfer o depósito de terminales con el fin de garantizar la formación de estructuras únicas válidas. Por lo tanto, el dominio principal contiene símbolos que representan tanto funciones como terminales, mientras que la cola se compone de solo terminales. (Ferreira, 2006).

Para cada problema, se elige la longitud de la cabeza h, mientras que la longitud de la cola t se determina a partir de la expresión $h \cdot (n_{max} - 1) + 1$, donde n_{max} representa la cantidad de argumentos de la función con más argumentos (denominado aridad máxima).

En (Ferreira, 2006) se pueden encontrar los pasos fundamentales del algoritmo GEP. El proceso comienza con la generación aleatoria de los cromosomas de un cierto número de individuos (la población inicial). Luego se expresan estos cromosomas y se evalúa la aptitud de cada individuo frente a un conjunto de casos de aptitud física (también llamado entorno de selección que, de hecho, es la entrada de un problema). A continuación, los individuos se seleccionan según su estado físico (su rendimiento en ese entorno particular) para reproducirse con modificaciones, dejando a los descendientes con nuevos rasgos. Estos nuevos individuos están, a su vez, sujetos al mismo proceso de desarrollo: expresión de los genomas, confrontación del entorno de selección, selección y reproducción con modificación. El proceso se repite durante un cierto número de generaciones o hasta que se encuentre una buena solución. (Ferreira, 2006).

Excepto por la replicación, donde los genomas de todos los individuos seleccionados se copian rigurosamente, todos los operadores restantes escogen aleatoriamente cromosomas para someterse a una determinada modificación. Sin embargo, a excepción de la mutación, cada operador no puede modificar un cromosoma más de una vez. Por ejemplo, para una tasa de transposición de 0.7, siete de 10 cromosomas diferentes se eligen al azar. (Ferreira, 2001) Además, en GEP, un cromosoma podría ser elegido por uno o varios operadores genéticos para introducir variación en la población. Esta característica también distingue GEP de GP donde una entidad nunca es modificada por más de un operador a la vez. Por lo tanto, en GEP, las modificaciones de varios operadores genéticos se acumulan durante la reproducción, produciendo descendientes muy diferentes a los padres.

En GEP se dispone de un total de ocho operadores genéticos, agrupados en cuatro categorías: Mutación, Inversión, Transposición y Recombinación. A continuación, se detalla el operador "Inversión", así como aquellos operadores en los cuales se detectaron inconsistencias.

Inversión

Las modificaciones destinadas a causar un gran impacto suelen producirse en las cabezas de los genes, así el operador inversión está restringido a estas regiones. Aquí cualquier secuencia puede ser seleccionada e invertida al azar. Vale la pena señalar que, como el operador de inversión fue restringido a las cabezas de los genes, no hay peligro de que una función termine en las colas y, en consecuencia, todos los nuevos individuos creados por inversión son programas sintácticamente correctos. Por lo general, se utiliza una pequeña tasa de inversión p_i de 0.1 ya que este operador rara vez se utiliza como fuente única de variación genética. (Ferreira, 2006).

En este operador se eligen aleatoriamente el cromosoma, el gen dentro del cromosoma que será modificado, así como los puntos de inicio y terminación de la secuencia que se va a invertir, los cuales tiene como única restricción que deben pertenecer a la cabeza del gen elegido al azar. De esta manera pudiera suceder que la raíz de un gen donde hubiese una función fuera cambiada por otra función con un número diferente de argumentos o por un terminal, representando esto último un cambio bastante drástico en su ET. De hecho, solo la mutación y la inversión son capaces de permitir este tipo de modificación en la raíz de los genes.

Considerando el cromosoma:

012345678012345678012345678

-+-+abaaa/bb/ababb*Q*+aaaba

Suponga que se elige el gen 2, con puntos de inicio y terminación dentro de dicho gen 0 y 2 respectivamente, obteniéndose:

012345678012345678012345678

-+-+abaaabb//ababb*Q*+aaaba

Observe como la raíz del gen 2 que estaba constituida por la función "/" fue sustituida por el terminal "b", propiciando así una macromutación (las cuales no siempre son malas, de hecho, son esenciales para llevar la evolución a otros picos muy distantes) en el ET correspondiente a dicho gen. En muchos casos, la inversión provee individuos menos aptos o incluso inviables. Pero, como en la naturaleza, la evolución ocurre debido a estos eventos extremadamente raros y altamente improbables.

Transposición de elementos IS

Cualquier secuencia en el genoma puede convertirse en un elemento IS y, por lo tanto, estos elementos son elegidos aleatoriamente en todo el cromosoma. El transposón se copia en el lugar de origen y luego se inserta la copia en un punto elegido al azar en la cabeza de un gen también seleccionado de forma aleatoria, exceptuando la posición de inicio (raíz del gen). (Ferreira, 2006) Por lo tanto, el operador de transposición elige al azar el cromosoma, los puntos de inicio y terminación del elemento IS, y el sitio de inserción (a partir de la selección aleatoria de un gen dentro del cromosoma). Típicamente, se usa una pequeña tasa de transposición IS p_{is} de 0.1, ya que este operador rara vez se utiliza como la única fuente de variación genética. Considere el cromosoma conformado por tres genes de longitud nueve:

012345678012345678012345678

-+-+abaaa/bb/ababb*Q*+aaaba

Suponga que se eligen como puntos de inicio y terminación del elemento IS las posiciones 3 y 5 respectivamente, en el gen 3, y como sitio de inserción la posición 1 en el gen 2 (esta debe ser elegida de acuerdo a la longitud del elemento IS), obteniéndose:

012345678012345678012345678

-+-+abaaab+aaababb*Q*+aaaba

Durante la transposición, la secuencia desde el inicio hasta la posición anterior al sitio de inserción permanece sin cambios, mientras que al final de la cabeza del gen en la cual se encuentra el punto de inserción se pierden tantos símbolos como longitud del elemento IS (en este caso la secuencia "bb/" fue eliminada). Asimismo, la cola del gen sometido a transposición y todos los genes cercanos permanecen sin cambios. Este operador también puede causar grandes cambios el ET correspondiente al gen modificado.

Transposición de genes

En la transposición de genes, un gen completo funciona como un transposón y se transpone al comienzo del cromosoma. A diferencia de las otras formas de transposición, en la transposición de genes, el transposón (el gen) se elimina en el lugar de origen. De esta manera, se mantiene la longitud del cromosoma. El cromosoma que se somete a la transposición de genes se elige al azar, y uno de sus genes (excepto el primero, obviamente) también es elegido aleatoriamente para transponer. Considerando el cromosoma compuesto por tres genes de longitud nueve:

012345678012345678012345678

-+-+abaaa/bb/ababb*Q*+aaaba

Suponga que el gen 2 fue elegido para someterse a la transposición de genes. Luego se obtiene el siguiente cromosoma:

012345678012345678012345678

/bb/ababb-+-+abaaa*Q*+aaaba

Observe como el gen 2 pasa a ser el gen 1 en el cromosoma, mientras el resto de los genes sufren un desplazamiento a la derecha.

Aparentemente, la transposición de genes solo es capaz de mezclar genes y, para los subárboles de expresión vinculados por funciones conmutativas, esto no contribuye en nada a la adaptación en el corto plazo. Tenga en cuenta, sin embargo, que cuando estos están vinculados por una función no conmutativa o forman parte de un sistema celular, el orden de los genes es importante y, en esos casos, la transposición de genes se convierte en un macromutador, generando la mayor parte del tiempo menos individuos más aptos o incluso inviables. Sin embargo, la transposición de genes se vuelve particularmente interesante cuando se usa junto con la recombinación, ya que permite no solo la duplicación de genes sino también una recombinación más generalizada de genes y bloques de construcción más pequeños. (Ferreira, 2006).

Recombinación de dos puntos

En la recombinación de dos puntos, los cromosomas padres se emparejan uno al lado del otro y se cortan en exactamente dos puntos (seleccionado aleatoriamente en todo el cromosoma). Luego para el primer descendiente se copia el material del primer padre desde el inicio hasta el punto de cruce y el resto se copia a partir de dicho punto, pero tomado del segundo progenitor; mientras el segundo descendiente se obtiene utilizando el mapeo inverso. Considere los siguientes cromosomas padres, formados por dos genes de longitud diez:

01234567890123456789

-b+Qbbabba/aQbbbaaba

/-a/ababba-ba-abaaab

Eligiendo como primer punto de cruce 3 en el gen 1 (entre posiciones 2 y 3) y como segundo punto de cruce 6 en el gen 2 (entre las posiciones 5 y 6). Luego, los pares de cromosomas se cortan en este punto formándose los descendientes a continuación:

01234567890123456789

-b+/ababba-ba-abaaba

/-aQbbabba/aQbbbaaab

El poder de transformación de la recombinación de dos puntos es mayor que la recombinación de un punto, y es más útil para desarrollar soluciones para problemas más complejos, especialmente cuando se usan cromosomas compuestos por varios genes. (Ferreira, 2001).

Resultados y discusión

Inconsistencias detectadas en la implementación de los operadores en JCLEC

A continuación, se exponen las inconsistencias detectadas en la implementación de los operadores: Transposición de elementos IS, Transposición de genes y Recombinación de dos puntos en el framework JCLEC.

En la transposición de elementos IS se debe tomar un fragmento aleatorio del genotipo y copiarlo en un lugar aleatorio de la cabeza de un gen cualquiera salvo en la raíz del gen (primer elemento de la cabeza). Para ello se debe seleccionar de forma aleatoria el lugar de copia (targetHeadGene) en la cabeza de un gen, escogido también de forma aleatoria, exceptuando la raíz, posteriormente elegir la longitud (lengthISElement) y origen de la secuencia a copiar (origin), para posteriormente realizar la copia en el lugar del genotipo seleccionado (target), desplazando el resto de los elementos hacia la derecha eliminándose los elementos al final de la cabeza. En la implementación de este operador se detectó que el targetHeadGene

en todos los casos coincidía con la raíz del gen seleccionado de forma arbitraria, la longitud de la secuencia de elementos de inserción (lengthISElement) no se realizaba correctamente, debido al problema mencionado con anterioridad, así como la selección del origen de dicha secuencia (origin) no contemplaba todas las posibilidades.

En el caso de la transposición de genes se debe tomar un gen aleatorio (gene), exceptuando el primero, de un cromosoma aleatorio y colocarlo en la primera posición de dicho cromosoma. Para ello se debe seleccionar de forma aleatoria el gen a transponer al inicio del cromosoma (gene), lo transpone y coloca a continuación de él, el resto de los genes eliminándolo del lugar de origen. En la implementación de este operador se detectó que siempre se seleccionaba el primer gen para transponer, cuando debía ser exactamente lo contario, además de ello el gen transpuesto quedaba sin información genética alguna.

Por último, en la recombinación de dos puntos es necesario seleccionar de forma aleatoria dos cromosomas padres (p0_genome, p1_genome), elegir arbitrariamente dos puntos de cruce a lo largo del todo el genotipo (cp1, cp2) por los cuales ambos cromosomas serán cortados, intercambiando su material genético entre dichos puntos, y dar como resultados dos nuevos hijos o descendientes. El problema con la implementación de este operador radicaba en la selección de forma incorrecta del segundo punto de cruce, este podía quedar fuera del cromosoma, así como podía suceder que ambos puntos coincidieran, aplicándose en dicho caso una recombinación de un punto.

JCLEC

Java Class Library for Evolutionary Computation (Librería de clases Java para Computación Evolutiva, JCLEC) constituye un sistema de software para la investigación de Computación Evolutiva (EC). El mismo fue desarrollado en el lenguaje de programación Java y proporciona un marco de software de alto nivel para hacer cualquier tipo de Algoritmo Evolutivo (EA), proporcionando soporte para algoritmos genéticos (codificación binaria, entera y real), programación genética (estilo Koza, fuertemente tipado y basado en la gramática) y programación evolutiva.

La arquitectura JCLEC sigue sólidos principios de programación orientada a objetos, donde las abstracciones están representadas por objetos débilmente acoplados y donde es común y fácil reutilizar el código. JCLEC proporciona un entorno EC con las siguientes características principales (JCLEC):

- Genericidad. Cualquier tipo de algoritmo EC se puede ejecutar utilizando JCLEC, ya que se cumplen algunos requisitos mínimos. La única condición necesaria es tener una población de individuos a la cual se aplica iterativamente una secuencia de operaciones evolutivas. Hasta ahora, JCLEC es compatible con la programación genética, la cadena de bits, el vector de valores enteros y los algoritmos genéticos de vectores de valor real, y la estrategia de evolución, entre otros. También es compatible con técnicas avanzadas de EC como la optimización multi-objetivo. El usuario puede hacer uso de cualquiera de estos marcos especializados, o incluso modificarlos para crear su propio algoritmo evolutivo personalizado.

- Fácil de usar. Se hicieron considerables esfuerzos para construir JCLEC de una forma bonita y agradable. JCLEC proporciona diversos mecanismos que ofrecen una interfaz de programación

fácil de usar. Sigue un estilo de programación de alto nivel, que permite el prototipado rápido de aplicaciones.

- Portabilidad. El sistema JCLEC ha sido codificado en el lenguaje de programación Java que garantiza su portabilidad entre todas las plataformas que implementan una Máquina Virtual Java.
- Eficiencia. Para garantizar una ejecución eficiente, se prestó especial atención a la optimización de secciones de códigos críticos. Se realizaron perfiles detallados de ejecución de estas secciones.
- Robustez. Las declaraciones de verificación y validación están incorporadas en el código para garantizar que las operaciones sean válidas y para informar problemas al usuario.
- Elegancia. La interfaz de JCLEC se desarrolló con cuidado. Se invirtió una gran energía en el diseño de un paquete de software coherente que sigue los buenos principios de programación orientada a objetos y genéricos. Además, se aplicaron estrictas reglas de programación para que el código sea fácil de leer, comprender y, eventualmente, modificar. El uso de XML como formato de archivo también es un aspecto central de JCLEC, que proporciona un terreno común para el desarrollo de herramientas para analizar y generar archivos, y para integrar el marco con otros sistemas.
- Fuente abierta. El código fuente de JCLEC es gratuito y está disponible bajo la Licencia Pública General de GNU (GPL). Por lo tanto, puede ser distribuido y modificado sin ningún costo.

A continuación, en la Figura 2, se muestra la arquitectura en capas de este sistema:

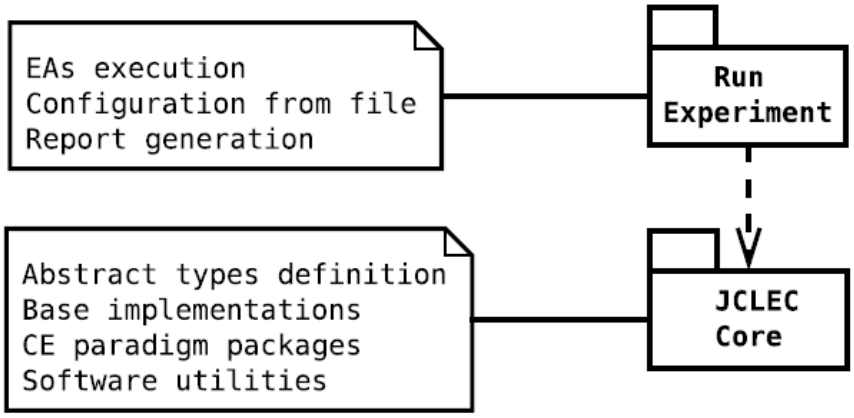

Figura 2 Arquitectura de capas JCLEC. (JCLEC4)

El sistema principal es la capa inferior. Tiene la definición de los tipos abstractos, su implementaciones base y algunos módulos de software que proporcionan toda la funcionalidad del sistema. Sobre la capa principal se encuentra el sistema de ejecución de experimentos en el que un trabajo es una secuencia de ejecuciones de algoritmos evolutivos definidos mediante un archivo de configuración. Recibe como entrada este archivo y devuelve como resultado uno o varios informes sobre las ejecuciones de los algoritmos. Permite resolver un problema más fácilmente utilizando los EAs disponibles de una interfaz específica. Configura el algoritmo, lo ejecuta de manera interactiva y también genera la informa-

ción en línea sobre el proceso evolutivo. Y el usuario puede incluir su propio código siempre que el código desarrollado cumpla con la jerarquía definida en el núcleo del sistema. (JCLEC4).

Este framework consta de un módulo de clasificación JCLEC que alberga implementaciones de métodos basados en reglas para la clasificación basada en GP, que admite múltiples representaciones de modelos y proporciona a los usuarios las herramientas para implementar fácilmente cualquier clasificador. Así como un módulo clasificación JCLEC-GEP que contiene implementaciones de métodos basados en reglas para la clasificación basada en GEP. Además, contiene la implementación del algoritmo Multiclases con Programación de Expresiones Genéticas (MCGEP), en el cual se hace uso de algunos de los operadores genéticos GEP (Guerrero-Enamorado, y otros, 2016).

Solución al problema

Se provee una implementación para el operador de inversión en el framework JCLEC, la cual permite invertir una secuencia cualquiera en la cabeza de un gen. Para ello se selecciona aleatoriamente un gen de un cromosoma (gene), así como los puntos de inicio (start) y terminación (end) de la secuencia a invertir en la cabeza de dicho gen. Posteriormente la secuencia es invertida sin importar el elemento que se encuentre en el start o en el end de dicha secuencia. Asimismo, cada una de las inconsistencias descritas con anterioridad fueron solventadas acorde con lo planteado en GEP.

Las implementaciones están contenidas en el módulo de clasificación JCLEC-GEP, específicamente los operadores de inversión y transposición en el paquete net.sf.jclec.gep.mut y el operador de recombinación en el paquete net.sf.jclec.gep.rec.

Conclusiones

GEP constituye un sistema de vida artificial, bien establecido más allá del umbral del replicador, capaz de adaptación y evolución, considerado un poderoso método evolutivo para la clasificación de datos. Con el desarrollo de este trabajo se obtuvo una versión mejorada del módulo de clasificación JCLEC-GEP, que contiene la implementación de los ocho operadores GEP propuestos. Con lo cual es posible implementar algoritmos evolutivos sobre el framework JCLEC utilizando la Programación de Expresiones Genéticas adecuadamente.

Referencias

CANNON, Walter B. 1932. The Wisdom of the Body. Norton and Company. New York, 1932.

TURING, Alan Mathison. 1950. Computing Machinery and Intelligence. Mind, 59. 1950. págs. 94-101.

FERREIRA, Cândida . 2001. Gene Expression Programming: a New Adaptive Algorithm for Solving Problems. Complex Systems. 2001.

VENTURA, Sebastián, y otros. 2007. JCLEC: A Java Framework for Evolutionary Computation. Soft Computing, vol. 12. 2007. págs. 381–392.

FERREIRA, Cândida. 2006. Gene Expression Programming:Mathematical Modeling by an Artificial Intelligence. s.l. : Revised and extended edition. Springer, 2nd edition, 2006. 3-540-32796-7.

JCLEC. JCLEC. Java Class Library for Evolutionary Computation. [En línea] http://jclec.sourceforge.net/.

JCLEC4. JCLEC 4 Tutorial.

GUERRERO-ENAMORADO, Alain , y otros. 2016. An algorithm evaluation for discovering classification rules with gene. s.l. : International Journal of Computational Intelligence Systems, 2016. 1875-6883.