Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Tipo de artículo: Artículo original
Temática: Ingeniería y Gestión de Software
Recibido: 10/05/2017 | Aceptado: 04/09/2017

Evaluación de proyectos usando sistemas basados en algoritmos genéticos de aprendizaje de reglas

Evaluation of Projects by using Genetic Rules Based Systems

Alain Guerrero Enamorado[1*], Iliana Pérez Pupo[1], Sebastián Ventura[2], Carlos Morell[3], Pedro Yobanis Piñero Pérez[1]

[1] Universidad de las Ciencias Informáticas (UCI). alaing@uci.cu, iperez@uci.cu, ppp@uci.cu, La Habana, Cuba.
[2] Universidad de Córdoba (UCO). sventura@uco.es, Córdoba, España.
[3] Universidad Central "Marta Abreu" de las Villas (UCLV). morell@uclv.cu, Santa Clara, Cuba.

[*] Autor para correspondencia: alaing@uci.cu

Resumen

En el presente trabajo se evalúa el comportamiento del algoritmo evolutivo MCGEP en diferentes versiones de una base de datos con información sobre la evaluación de proyectos informáticos. La idea fundamental del trabajo es evaluar la posibilidad de aplicación de un algoritmo evolutivo que utiliza programación de expresiones genéticas frente a otros siete algoritmos muy utilizados del estado del arte. Los algoritmos utilizados en la evaluación son capaces de generar modelos de clasificación interpretables utilizando técnicas evolutivas para obtenerlos. Los experimentos se realizaron en cinco versiones creadas a partir de un repositorio de datos con información sobre la evaluación de proyectos. Se logró mostrar como el algoritmo MCGEP queda en primer lugar entre los algoritmos comparados para la métrica de exactitud predictiva, además mejora significativamente a la mayoría de estos algoritmos en esta métrica. Por otro lado, la complejidad de los modelos que genera para lograr estos resultados no es demasiado elevada por lo cual MCGEP sobresale junto al algoritmo GASSIST como los más balanceados si se tienen en cuenta ambas métricas al mismo tiempo. Como valor añadido se aprovecha la capacidad de selección de atributos implícita que tiene este tipo de técnicas para sacar algunas conclusiones sobre cuáles son los indicadores de medición que más influyen en la evaluación de un proyecto y cuál o cuáles indicadores permiten detectar a tiempo que un proyecto no logrará una buena evaluación al finalizar.

Palabras clave: Algoritmos genéticos, Programación de Expresiones Genéticas, Algoritmo MCGEP, Evaluación de Proyectos, Aprendizaje de reglas.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Abstract

In the present work is assessed the behavior of the evolutionary algorithm MCGEP in different versions of a project management database which contains information for projects evaluation. The main idea of this work is to confirm the possibility of applying an evolutionary algorithm that uses genetic expression programming in front of seven other widely used algorithms in the state of the art. The algorithms used in the assess are able to generate interpretable classification models, using evolutionary techniques to obtain them. The experiments were carried out in five versions created from the database with information of the projects evaluation. The MCGEP algorithm achieves the first among the algorithms compared for the predictive accuracy metric; also, it significantly improves the majority of these algorithms for this metric. On the other hand, the complexity of the generated models was acceptable to achieve these results, so MCGEP and GASSIST algorithms excel as the most balanced if both metrics are taken into account at the same time. As value added, we take advantage of the implicit process of feature selection capability that have these kinds of techniques. With this, we draw some conclusions about which are the measurement indicators that most influence the evaluation of a project and which indicators can detect in time which project will not achieve a good evaluation when finalized.

Keywords: *Genetic Algorithms, Gene Expression Programming, Algorithm MCGEP, Project Evaluation, Rules´ learning*.

Introducción

El proceso de gestión de proyectos es de suma importancia para las organizaciones basadas en proyectos, en los últimos años se han realizado esfuerzos en la identificación de riesgos (Cordero-Morales y otros, 2013), la planificación de proyectos (García-Vacacela y otros, 2016) y en la planificación y control (Marín-Sánchez y otros, 2014). En particular, el proceso de evaluación de proyectos resulta de vital importancia para la toma de decisiones en estas organizaciones. En el caso particular de los proyectos de tecnologías de la información, la media histórica de proyectos satisfactorios es 30,7 % mientras que los proyectos renegociados son el 47,3 % y los cancelados el 22 % (The Standish Group International, 2014). Estas cifras implican que enormes presupuestos son afectados cada año por concepto de errores en la planificación o el control y seguimiento de los proyectos, con un impacto económico y social. Evaluar los proyectos en cada corte de forma rápida y eficaz, ayuda a mitigar estas pérdidas y a detectar de forma temprana los factores que afectan la ejecución de los mismos. El proceso de evaluación puede ser considerado un problema de clasificación, donde los proyectos son clasificados en alguna de las siguientes categorías “Bien”, “Regular” y “Mal”.

En este trabajo los autores presentan una alternativa de solución para este problema, a partir de proponer algoritmos que aprenden reglas para la clasificación de proyectos.

Por otra parte, en las organizaciones orientadas a proyectos, generalmente se gestionan múltiples proyectos simultáneamente, y se registra en sistemas de gestión de proyectos un gran volumen de información (Castro Aguilar y otros, 2016). De esta forma, cada vez se abre más la brecha entre la generación de los datos y la capacidad de los expertos para comprenderlos o extraer conocimiento útil de los mismos Witten y otros (2011). En este contexto, es preciso desarrollar técnicas que permitan analizar los datos e identificar que indicadores son más relevantes y cuáles son los factores o causas que mayor incidencia tienen en el éxito o fracaso de los proyectos.

Para resolver esta problemática, diferentes autores han propuesto diversos algoritmos entre los que se destacan: las Máquinas de Soporte Vectorial (SVM) (Vapnik, 1995), las Redes Neuronales Artificiales (ANNs) (McClelland, y otros, 1986) y el Bayesiano Ingenuo (NB) (Domingos, y otros, 1997). Sin embargo, todos ellos tienen el problema de que generan modelos con buen nivel de exactitud pero que son poco interpretables. Otros enfoques, potencian la clasificación pero manteniendo altos niveles de interpretabilidad, algunos de ellos son UCS (Bernadó-Mansilla, 2003), GASSIST (Bacardit, 2004), SLAVE (González, 1999), LOGIT-BOOST (Otero, 2006), CORE (Tan, 2006), Tan (2002), Bojarczuk (2004). Por otro lado, en Cano y otros (2013) se expone que este tipo de algoritmos interpretables pueden ser aprovechados para hacer selección de atributos.

La principal contribución del presente trabajo radica en validar la posibilidad de aplicar un algoritmo evolutivo de clasificación utilizando programación de expresiones genéticas (MCGEP) introducido en Guerrero-Enamorado y otros (2016) para la evaluación de proyectos. Como valor añadido se pretende también aprovechar la capacidad de selección de atributos implícita que tienen este tipo de algoritmos para conocer cuáles son los indicadores de medición que más influyen en la evaluación de un proyecto.

El trabajo se encuentra dividido en tres sesiones fundamentales. La siguiente sesión “Materiales y métodos” describe el algoritmo MCGEP propuesto en la investigación. Luego se presentan los resultados de experimentación y su análisis y finalmente se presentan las conclusiones del trabajo.

Materiales y métodos

En esta sección se explica el principio de funcionamiento del algoritmo MCGEP, así como algunos de sus elementos de diseño fundamentales. Para mayor profundidad puede revisarse el trabajo donde se introdujo el mismo (Guerrero-

Enamorado y otros, 2016). En MCGEP se utilizan dos fases, en la primera se sigue el esquema clásico de los algoritmos genéticos y en la segunda se ordenan las reglas obtenidas para crear el clasificador.

- Fase 1: Descubrimiento de reglas.
    1. Generación de la población inicial
    2. Evaluación de la población inicial.
    3. Selección, Cruzamiento, Mutación.
    4. Evaluación de la descendencia.
    5. Reemplazo y competición para generar un vector-solución de individuos no redundantes.
    6. Se repiten los pasos 3 al 5 hasta que se cumpla la condición de parada (cantidad de generaciones).
    7. Se guarda el último vector-solución de individuos no redundantes obtenido.
    8. Se repiten los pasos 2 al 7 para cada clase de la colección de datos.
    9. Se crea una lista con la unión de los vectores-solución no redundantes por cada clase.
- Fase 2: Ordenamiento de reglas.
    1. Crear un clasificador vacío.
    2. Ordenar la lista obtenida en la Fase 1 (paso 9).
    3. Adicionar al clasificador la mejor regla y eliminarla de la lista inicial.
    4. Eliminar de la colección de datos las instancias cubiertas por la mejor regla.
    5. Reevaluar y ordenar las reglas en las instancias que quedaron en la colección de datos.
    6. Repetir desde el paso 3, hasta que no queden reglas o instancias en la colección de datos.
    7. Devolver el clasificador para que se utilice en la etapa de evaluación.

Codificación de los individuos con Programación de Expresiones Genéticas (GEP)

En el diseño del algoritmo MCGEP se utilizó la tecnología de programación de expresiones genéticas de Ferreira (2006) para codificar los individuos. Esta innovadora tecnología aprovecha la eficiencia de los Algoritmos Genéticos (GA) para evolucionar y la potencia de los árboles de la Programación Genética (GP) para construir reglas en forma de Funciones Discriminantes (DF). De esta forma, en la "Programación de Expresiones Genéticas", los individuos son representados genotípicamente en forma de cadenas de caracteres y fenotípicamente en forma de árboles de expresión (ET) que al final se traducen en una función discriminante.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Como elemento distintivo, la tecnología GEP separa el genotipo en dos partes: la "cabeza" y la "cola". La cabeza puede generarse con funciones, atributos predictores o constantes, su tamaño es uno de los parámetros del algoritmo. Una vez definida la cabeza, el tamaño de la cola queda determinado puesto que está diseñada para garantizar que existan suficientes elementos terminales en caso de que la cabeza esté compuesta solo de funciones. Esto garantiza la completitud de los ET y es una de las ventajas que tiene, pues en GP se necesita verificar a nivel micro del proceso evolutivo la validez de los árboles que se van generando. En los casos donde la cabeza no tenga solo funciones, la información presente en el genotipo (cabeza + cola) no se expresa del todo a nivel fenotípico al igual que en los seres vivos donde existe información genética que no se expresa fenotípicamente, pero que permanece latente hasta que un cruzamiento o una mutación permite su activación.

La generación de la población inicial se realiza de manera muy simple, puesto que solo es necesario garantizar que la cabeza se genere aleatoriamente con funciones, atributos predictores o constantes mientras que la cola solo con constantes y atributos predictores. Otro elemento importante del MCGEP es que solo utiliza las funciones aritméticas básicas (suma, resta, multiplicación y división) para codificar los individuos, los elementos terminales pueden ser cualquiera de los atributos predictores o una constante escogida aleatoriamente de una lista que se provee como parámetro. Cada individuo codifica una DF, además se asigna como consecuente la clase que él predice. Para guiar el proceso evolutivo se utilizaron los siguientes operadores genéticos:

- Mutador simple (*GEPSimpleMutt*), puede ocurrir con una probabilidad definida como parámetro en cualquier lugar del genotipo, solo debe garantizarse que en la cola los atributos y constantes sean modificados por atributos y constantes, en la cabeza todo tipo de cambio es permitido entre los atributos, las constantes y las funciones.
- Mutador de transposición con inserción de secuencia (*GEPISTranspMutt*), la probabilidad de ocurrencia también se define como parámetro. Se selecciona entonces aleatoriamente una secuencia de elementos a transponer, esta secuencia es insertada en una posición aleatoria de la cabeza exceptuando el inicio para evitar que se formen individuos de un solo elemento (Ferreira, 2006).
- Mutador de transposición con inserción de secuencia en la raíz (*GEPISTranspMutt*), a grueso modo este operador selecciona aleatoriamente una secuencia de elementos a transponer, pero con la restricción de que inicie con una función. Esta secuencia es insertada en el inicio de la cabeza del genotipo.
- Recombinador por un punto (*GEPOnePointRecomb*), son seleccionados dos individuos, en ellos se selecciona aleatoriamente un mismo punto de corte, se intercambian entonces las partes resultantes para generar dos nuevos individuos. Por ejemplo, los individuos Aa y Bb generan la descendencia Ab y aB.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

- Recombinador por dos puntos (*GEPTwoPointRecomb*), similar al anterior, pero ahora se seleccionan dos puntos de corte en cada uno de los dos progenitores, estos puntos determinan una secuencia de genes en cada uno de ellos. Las secuencias de los progenitores son intercambiadas para generar dos nuevos individuos. Por ejemplo, AaA y BbB generarían la descendencia AbA y BaB. Ferreira (2006) recomienda no utilizar los operadores de recombinación (de un punto y de dos puntos) solos puesto que pueden generar convergencia prematura, pero combinados con los de mutación logran una mejor exploración del espacio característico.

Descubrimiento de Funciones Discriminantes (DF) para la clasificación de información

Los individuos del tipo GEP que involucran a los atributos predictores dentro de una función discriminante pueden utilizarse para extraer información de los datos como se expone en el trabajo de Espejo y otros (2009). Este tipo de funciones pueden ser evaluadas en el conjunto de los atributos predictores de una base de datos devolviendo un número real como salida, esta salida puede ser interpretada utilizando un umbral (normalmente cero) para definir una salida de clasificación binaria. Con este tipo de funciones (ver Ecuación 1) puede crearse una base de conocimientos.

$$\boldsymbol{if}\ (f(\vec{x}) > 0)\ \boldsymbol{then}\ \vec{x} \in Class1\ \boldsymbol{else}\ \vec{x} \in Class2 \qquad (1)$$

En MCGEP, el problema multi-clase se resuelve utilizando un enfoque uno-contra-todos, donde la solución del problema de *n*-clases es transformada en la solución de *n* problemas de clasificación binaria. Por tanto, dada una clase en cuestión, las instancias que pertenecen a dicha clase son tomadas como positivas y el resto como negativas en la matriz de confusión binaria de cada clase. Partiendo de esta matriz de confusión se puede calcular la función de aptitud que se detalla próximamente. Por otro lado, es poco probable que en los problemas del mundo real se logren encontrar reglas capaces de cubrir todas las instancias de una clase, por lo cual deben descubrirse varias reglas en cada clase. En la Fase 2 del algoritmo la base de reglas es ordenada utilizando para ello la precisión de cada regla conformando así el clasificador definitivo en forma de lista de decisión, al cual además se le adiciona una clase por defecto que sería la clase más numerosa de las instancias que no sean cubiertas. La Ecuación 2 representa un ejemplo de clasificador según este esquema.

$$\begin{aligned} \boldsymbol{if}\ (f_{11}(\vec{x}) > 0)\ \boldsymbol{then}\ \vec{x} &\in Class_1 \\ \boldsymbol{elseif}\ (f_{12}(\vec{x}) > 0)\ \boldsymbol{then}\ \vec{x} &\in Class_1 \\ \vdots & \\ \boldsymbol{elseif}\ (f_{n1}(\vec{x}) > 0)\ \boldsymbol{then}\ \vec{x} &\in Class_n \\ \boldsymbol{elseif}\ (f_{n2}(\vec{x}) > 0)\ \boldsymbol{then}\ \vec{x} &\in Class_n \end{aligned} \qquad (2)$$

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Función de aptitud

Para definir la función de aptitud se parte de la que se propone en Bojarczuk (2004) realizando algunas modificaciones se obtuvo la siguiente Ecuación 3.

$$aptitud = \frac{t_p}{t_p + w_1 * f_n} * \frac{t_n}{t_n + w_2 * f_p} * \frac{tamañoMáximo - 0.2 * tamañoRegla - 0.8}{tamañoMáximo - 1} \quad (3)$$

En la ecuación anterior, t_p, t_n, f_n y f_p representan los verdaderos positivos, verdaderos negativos, los falsos negativos y los falsos positivos respectivamente, extraídos de la matriz de confusión obtenida de evaluar un individuo en la colección de entrenamiento; w_1 y w_2 son parámetros que permiten variar el nivel de importancia que se les da a los f_n y f_p respectivamente. A los dos primeros factores también se les conoce como sensibilidad y especificidad. El tercer factor incluido en la Ecuación 3, permite tener en cuenta que los individuos más simples son preferibles pues tienden a tener mayor capacidad de generalización.

Estrategia de reemplazo

Se utiliza una estrategia de reemplazo de truncamiento con elitismo, es decir, un pequeño porcentaje de la población es preservado a toda costa (definido como parámetro), mientras que se mantiene una población constante reemplazando los menos aptos de la misma con la descendencia obtenida durante el proceso de aplicación de los operadores genéticos. Por otro lado, como parte de esta estrategia MCGEP es capaz de generar una lista de individuos no redundantes respecto a las instancias que cubren de la colección de entrenamiento, esto se logra utilizando una competición por las instancias (*TokenCompetition*) similar a la utilizada en Olmo (2013). Dicha competición hace que individuos redundantes tiendan a desaparecer de la población. Para esto, primero se ordena la población atendiendo a la función de aptitud de cada individuo, después durante la competición, se calcula una nueva función de aptitud re-ajustada por la razón entre las instancias no cubiertas y el total de instancias cubiertas. Los individuos redundantes obtienen un valor en la función de aptitud más bajo, pudiendo ser incluso igual a cero cuando estos no son capaces de cubrir ninguna instancia que no haya sido cubierta por otros individuos anteriormente. Todo el proceso evolutivo se repite durante un número de generaciones definido previamente, evolucionando de esta manera dicho vector solución el cual es reincorporado nuevamente a la población (algoritmo elitista). Al concluir el proceso, se guarda el último vector solución obtenido y se reinicia el proceso evolutivo tantas veces como clases tenga la colección de datos. Finalmente, en la fase 2 la base de reglas obtenida con la unión del conjunto de vectores no redundantes de cada clase es ordenada y utilizada como clasificador.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Resultados y discusión

En la evaluación de la propuesta se emplean técnicas de triangulación de datos espacial y se prueban diferentes algoritmos con cinco bases de datos asociadas a la evaluación de proyectos del "Repositorio para Investigaciones de Gestión de Proyectos" (Piñero Pérez y otros, 2016) que provee el "Laboratorio de Investigaciones en Gestión de Proyectos" de la "Universidad de las Ciencias Informáticas". En particular, estas bases de datos recogen información de 8430 cortes de 667 proyectos entre septiembre del 2012 y febrero del 2013 considerando 84 atributos predictores y tres clases como atributo decisión (Bien, Regular, Mal). Las bases de datos se describen en la Tabla 1.

Tabla 1. Versiones de la base de datos de evaluación de proyectos utilizadas en la experimentación.

Base de datos	Cantidad de instancias	Cantidad de atributos	Cantidad de clases	Descripción
PE0	8430	84	3	Datos originales que contiene algunos atributos con valores nulos
PE1	8430	76	3	Base de datos que no contiene atributos con valores nulos.
PE2	8430	75	3	Bases de datos donde se elimina el atributo "*project_id*".
PE3	8396	75	3	Base de datos que no contiene atributos con valores nulos y tampoco el atributo "*project_id*".
PE4	8396	74	3	Base de datos que no contiene atributos con valores nulos, ni el atributo "*project_id*" ni el atributo *Fecha*.

*Tabla de creación propia.

Para la validación del algoritmo MCGEP se comparan los resultados contra siete algoritmos similares descritos en (Bernadó-Mansilla, 2008):

- UCS (Bernadó-Mansilla, 2003): es un algoritmo genético basado en reglas, utiliza un enfoque Michigan. Es una mejora derivada del conocido algoritmo XCS, pero funciona utilizando aprendizaje supervisado. Evoluciona los mapas de acción que introduce XCS de manera más eficiente.
- GASSIST (Bacardit, 2004): algoritmo basado en GABIL. Se reporta en la bibliografía como uno de los algoritmos estilo Pittsburg más competitivos. Es un algoritmo genético que evoluciona individuos en forma de conjuntos de reglas ordenadas de longitud variable.
- SLAVE (González, 1999): algoritmo genético basado en reglas borrosas que utiliza el enfoque de aprendizaje iterativo con el objetivo de reducir el espacio de búsqueda.
- LOGIT-BOOST (Otero, 2006): algoritmo similar al AdaBoost que utiliza una "versión codiciosa del ajuste posterior" (*greedy version of generalized backfitting*) para construir un modelo aditivo. El algoritmo LOGIT-BOOST es una extensión del anterior que permite inducir clasificadores borrosos.

- CORE (Tan, 2006): algoritmo co-evolutivo para el descubrimiento de reglas, cada individuo codifica una regla y un conjunto completo de reglas es evolucionado simultáneamente. Las reglas cooperan unas con otras para producir un conjunto óptimo de reglas y al mismo tiempo el conjunto es reducido por medio de una competencia por las instancias similar a la utilizada en MCGEP.
- Tan (2002): algoritmo de programación genética que evoluciona reglas de clasificación del tipo *IF-THEN*, utilizando un enfoque Michigan, como se dijo anteriormente utiliza una competencia por las instancias para reducir redundancia. De este trabajo se utilizaron algunas ideas para implementar el MCGEP.
- Bojarczuk (2004): utiliza un enfoque hibrido entre Michigan y Pittsburgh, un individuo contiene múltiples reglas de clasificación, pero todas con el mismo consecuente. Utiliza una disyunción de los antecedentes de las reglas, donde cada antecedente es entonces una forma conjuntiva de condiciones lógicas de la forma atributo-valor.

Para los algoritmos MCGEP, Tan y Bojarczuk se utilizaron las implementaciones del JCLEC (http://jclec.sourceforge.net/). Este marco de trabajo se describe en (Ventura, y otros, 2008), es un software de código abierto que permite implementar algoritmos evolutivos desarrollado sobre tecnología Java. En la Tabla 2 se muestran las configuraciones utilizadas para el algoritmo MCGEP con vistas a garantizar la posibilidad de replicación de los experimentos.

Tabla 2. Parámetros de configuración del algoritmo MCGEP utilizados durante la experimentación

Parámetro	Valor
Tamaño de la población (population-size)	500
Máximo número de generaciones (max-of-generations)	100
Tamaño de la cabeza (head-size)	80
Probabilidad de elitismo (elitist-prob)	0.1
Soporte mínimo (support)	0.01
Parámetros (w1 ; w2)	(1.0 ; 1.0)
GEPSimpleMutator (mut-prob)	0.10
GEPISTranspositionMutator (mut-prob)	0.10
GEPRISTranspositionMutator (mut-prob)	0.10
GEPOnePointRecombinator (rec-prob)	0.40
GEPTwoPointsRecombinator (rec-prob)	0.40

Para el resto de los algoritmos se utilizaron las implementaciones del KEEL (http://www.keel.es/), este último es también una aplicación de código abierto en tecnología Java que permite realizar evaluaciones de técnicas de Aprendizaje Evolutivo y *Soft Computing* (Alcalá-Fdez y otros, 2011).

Las configuraciones utilizadas para UCS, GASSIST, HIDER, SLAVE, LOGIT-BOOST y CORE fueron las recomendadas por sus autores y se encuentran condensadas en el trabajo de (Orriols-Puig y otros, 2008). En los casos

de los algoritmos de Tan (2002) y Bojarczuk (2004) se utilizaron las configuraciones que sus autores recomiendan. Los experimentos se realizaron en una estación de trabajo con un Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 8GiB Memoria DDR3 y un sistema operativo Gnu/Linux con un núcleo 4.4.0-66-generic-Ubuntu-SMP.

Para evaluar el desempeño de los algoritmos se utilizaron las métricas: Exactitud predictiva (Acc) y el Número de reglas de los modelos obtenidos (NR). Estas métricas se promedian utilizando la técnica de validación cruzada con 10 particiones cada una con cinco semillas aleatorias. Estos resultados se muestran en la Tabla 3 y la Tabla 4 respectivamente. Cada fila representa los resultados para una colección de las cinco utilizadas. La última fila representa el ranking promedio del algoritmo en todas las colecciones.

Tabla 3. Resultados de la exactitud predictiva de cada algoritmo.

Colección/Acc	MCGEP	UCS	Gassist	SLAVE	CORE	LogitBoost	Tan	Bojarczuk
PE0	0,8819	0,7817	0,8654	0,8607	0,5046	0,5046	0,8174	0,7859
PE1	0,8760	0,8118	0,8728	0,8699	0,5046	0,5046	0,8110	0,8207
PE2	0,8735	0,7846	0,8690	0,8616	0,5046	0,5046	0,8128	0,8170
PE3	0,8751	0,8140	0,8732	0,8698	0,5027	0,5027	0,8091	0,8183
PE4	0,8792	0,8253	0,8729	0,8711	0,5027	0,5027	0,8185	0,8061
Ranking	**1.0**	**5.2**	**2.0**	**3.0**	**7.5**	**7.5**	**5.2**	**4.6**

Entre los algoritmos estudiados el MCGEP logra aproximar bases de reglas con una mayor exactitud predictiva en todos los casos. El algoritmo Gassist, cuya robustez ha sido mostrada en varios trabajos (Bernadó-Mansilla, 2008; Bacardit, 2004; Guerrero-Enamorado, y otros, 2016) queda en segundo lugar en esta métrica. Las implementaciones de los algoritmos CORE y LogitBoost utilizadas en esta experimentación colapsan y predicen siempre la evaluación de Mal para todos los casos, por eso el bajo valor de exactitud que devuelven.

Tabla 4. Tamaño de las bases de reglas obtenidas

Colección/NR	MCGEP	UCS	Gassist	SLAVE	CORE	LogitBoost	Tan	Bojarczuk
PE0	8,56	6400	8,24	35,28	1,00	50,00	18,00	4,00
PE1	8,52	6400	8,44	35,84	1,00	50,00	17,54	4,00
PE2	8,76	6400	8,28	36,96	1,00	50,00	17,74	4,00
PE3	9,18	6400	8,66	35,52	1,00	50,00	17,64	4,00
PE4	8,88	6400	8,52	36,30	1,00	50,00	16,94	4,00
Ranking	**4.0**	**8.0**	**3.0**	**6.0**	**1.0**	**7.0**	**5.0**	**2.0**

En el caso del algoritmo CORE logra quedar en primer lugar a expensas de un gran deterioro en la exactitud predictiva, puesto que tiende a generar solo una regla que siempre da la misma respuesta de clasificación.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Análisis estadístico de los resultados

Para realizar el análisis estadístico de los resultados se siguieron las recomendaciones de Derrac y otros (2011). Por lo cual se aplicaron varios test no-paramétricos. En todos los casos primero se verificó el cumplimiento o no de la hipótesis nula por medio del test de Friedman y en los casos donde fue rechazada se aplicaron entonces pruebas *posthoc*.

Para el caso de la exactitud predictiva la hipótesis nula fue rechazada por el test de Friedman con un p-valor de 0.000025 por lo cual se aplicaron las pruebas de Holm, Finner y Li (Derrac, y otros, 2011). Los resultados de estas pruebas se muestran en la Tabla 5, dicha tabla muestra que todas las pruebas estadísticas concuerdan en que el algoritmo MCGEP tomado como control, mejora significativamente a los algoritmos CORE, LogitBoost, UCS y Tan. Los test de Finner y Li que tienen un mayor poder discriminante también encuentran que MCGEP mejora a Bojarczuk. Finalmente, ninguno de los test utilizados logra encontrar diferencia significativa de MCGEP frente a los algoritmos SLAVE y Gassist.

Tabla 5. Pruebas estadísticas comparando MCGEP con el resto de los algoritmos respecto a variable exactitud predictiva

Orden	Algoritmo	p-valor sin ajuste	p-valor Holm	p-valor Finner	p-valor Li
1	CORE	0.000027	0.00019	0.00019	0.000056
2	LogitBoost	0.000027	0.00019	0.00019	0.000056
3	UCS	0.006706	0.033531	0.015578	0.01374
4	Tan	0.006706	0.033531	0.015578	0.01374
5	Bojarczuk	0.020137	0.06041	0.028077	0.040151
6	SLAVE	0.196706	0.393411	0.225502	0.290083
7	Gassist	0.518605	0.518605	0.518605	0.518605

Para el caso del tamaño de los modelos obtenidos la hipótesis nula fue rechazada por el test de Friedman con un p-valor de 0.000011 por lo cual también se aplicaron las pruebas de Holm, Finner y Li. Los resultados de estas pruebas se muestran en la Tabla 6, en ella se observa que todas las pruebas estadísticas concuerdan en que CORE, tomado como algoritmo de control, mejora significativamente a los algoritmos UCS, LogitBoost, SLAVE y Tan. Finalmente, ninguno de los test utilizados logra encontrar diferencia significativa de CORE frente a los algoritmos MCGEP, Gassist, Bojarczuk respecto al tamaño de los modelos obtenidos.

Tabla 6. Pruebas estadísticas no paramétricas para la exactitud predictiva.

Orden	Algoritmo	p-valor sin ajuste	p-valor Holm	p-valor Finner	p-valor Li
1	UCS	0.000006	0.000044	0.000044	0.000013
2	LogitBoost	0.000108	0.000645	0.000376	0.000223
3	SLAVE	0.001249	0.006244	0.002912	0.002587
4	Tan	0.009823	0.039293	0.017127	0.019998
5	MCGEP	0.052808	0.158423	0.073141	0.098853

6	Gassist	0.196706	0.393411	0.225502	0.290083
7	Bojarczuk	0.518605	0.518605	0.518605	0.518605

Análisis del poder discriminante de los atributos

Para poder realizar este análisis se tomaron las bases de reglas obtenidas en cada uno de los 50 experimentos (validación cruzada en 10 partes y 5 semillas aleatorias en cada una) realizados en cada base de datos, con estos se construyeron los histogramas de los atributos que aparecen en cada base de reglas, por razones de espacio no se muestra en este trabajo el histograma completo de los atributos obtenido para cada base de datos, en su lugar se muestra en la

Tabla 7 un resumen del *ranking* de los atributos más interesantes para analizar en cada base de datos. Entiéndase como atributos más interesantes a los primeros 10 en los *rankings* de cada base de datos y algunos más que permitieron arribar a conclusiones. La primera columna de esta tabla no es más que el *ranking* de dicho atributo aplicando el método de la Ganancia de Información (GI) en la base de datos PE0. La segunda columna es el nombre del atributo tal cual aparece en cada base de datos. Después viene el *ranking* de dicho atributo en la base de reglas (RB) obtenida para cada base de datos, finalmente se calculan los promedios (Prom.) y la desviación estándar (Desv. Std.) de estos valores. El promedio brinda una idea del poder discriminante de dicho atributo en todas las versiones de la base de datos y la desviación estándar de la consistencia del valor anterior.

Analizando los resultados de la Tabla 7, se puede llegar a la conclusión de que los siguientes atributos son imprescindibles para predecir la evaluación final de un proyecto:

- *ie*, índice de ejecución, representa el avance real de un proyecto.
- *ire*, índice de rendimiento de la ejecución, muestra el estado de avance o progreso del proyecto respecto a sus hitos de ejecución considerando todos los hitos.
- *real_ire*, impacto de los hitos de ejecución cerrados hasta la fecha de corte en el proyecto.
- *impacto_total_hitos_ejec*, impacto total esperado por los hitos de ejecución del proyecto.
- *plan_ire*, plan de la ejecución esperada de los hitos de ejecución hasta la fecha de corte.
- *irhf*, índice de rendimiento del recurso humano con respecto a la eficiencia, muestra el estado de avance o progreso del recurso humano en la realización de las tareas que tiene asignadas.

Una de las funciones discriminantes que se obtuvo con mucha fuerza muestra que “si el ire < 0.5 entonces es muy probable que el proyecto sea evaluado al final de Mal”. Esto independientemente del tiempo, del proyecto que sea, de los recursos humanos asignados y de todos los demás atributos.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Tabla 7. Ranking de los histogramas de cada atributo en las cinco bases de datos evaluadas.

Orden GI	Atributos	RB-PE0	RB-PE1	RB-PE2	RB-PE3	RB-PE4	Prom.	Desv. Std.
1	ire	1	1	1	1	1	1,0	0,00
2	project_id	21	67	-	-	-	44,0	32,53
3	ie	2	2	2	2	2	2,0	0,00
4	impacto_total_hitos_ejec	4	4	4	4	4	4,0	0,00
5	real_ire	3	3	3	3	3	3,0	0,00
6	plan_ire	9	8	6	5	5	6,6	1,82
7	iref	7	9	37	38	9	20,0	16,00
8	irp	8	47	68	22	11	31,2	25,67
9	real_irp	36	32	39	24	60	38,2	13,42
10	trtr	41	55	57	71	70	58,8	12,34
14	irhf	5	6	5	13	7	7,2	3,35
34	fecha	17	17	39	37	-	27,5	12,15
35	cant_esp_comp	6	14	11	34	27	18,4	11,67
38	cant_rga_recien_graduado_adiestramiento	43	28	49	40	71	46,2	15,83
39	cant_otros_trabajadores	71	63	62	68	23	57,4	19,58
49	cant_est_level4	59	53	71	75	61	63,8	9,01
51	cant_est_level5	50	69	74	57	60	62,0	9,57
53	cant_especialista	61	39	51	60	44	51,0	9,67
58	cant_est_level3	75	73	75	70	73	73,2	2,05

Un atributo que se analizó fue el identificador de proyecto *project_id*, dicho atributo solo estuvo presente en las versiones de las bases de datos PE0 y PE1, sin embargo, en ellas no logró aparecer como un atributo imprescindible para lograr clasificar correctamente la evaluación de los proyectos. No obstante, desde el punto de vista de la Ganancia de Información se evidencia que existe correlación entre dicho atributo y la salida de clasificación (evaluación) de un proyecto motivado porque la base de datos es de "cortes de un proyecto" y en los proyectos evaluados de "*Bien"* este atributo coincide en diversos registros al igual que en los evaluados de "*Mal"*. Algo interesante también ocurre con el atributo *iref* (índice de rendimiento de la eficacia), pues el mismo como puede observarse en la fila con valor 7 de GI de la Tabla 7 evidencia que existe una relación entre este y los atributos *project_id* y *fecha*. Para llegar a esta conclusión se puede notar como el *iref* es muy importante cuando están presentes *project_id* y *fecha* al mismo tiempo en las bases de datos (PE0 y PE1), también es muy importante cuando no están ambos atributos presentes al mismo tiempo (PE4). Sin embargo, cuando no está presente el atributo *project_id* y si lo está el atributo *fecha* el atributo *iref* pasa a ser un atributo poco relevante. Esto ocurre porque este atributo habla de la calidad del proyecto, y se tiene información relevante de este atributo solo en etapas avanzadas del proyecto. Finalmente se puede notar como se vuelve a evidenciar

la relación entre los atributos *project_id* y *fecha* en la fila con valor 34 de GI de la Tabla 7, puesto que a partir de la base de datos PE2 desde la cual se eliminó el atributo *project_id* la relevancia del atributo *fecha* decae bastante. Esto ocurre porque el atributo *iref* representa la calidad del proyecto y esta tiene valores concretos en etapas avanzadas del proyecto de allí su dependencia con la fecha.

Las filas con valores de GI 38, 39, 49, 51, 53 y 58 de la Tabla 7 evidencian que los atributos relativos a las cantidades de: recién graduados en adiestramiento, otros trabajadores, estudiantes de 4to año, estudiantes de 5to año, especialistas y estudiantes de 3er año influyen muy poco en la evaluación de los proyectos estudiados, esto puede deberse a una de dos causas posibles; la primera es que en todos los proyectos estudiados se asignaron las cantidades requeridas de estos trabajadores del proyecto, la segunda es que estas cantidades no influyen en la evaluación de un proyecto y cuando lo hacen se determinó que es más probable que lo hagan para predecir las evaluaciones de Mal o Regular. Cabe destacar que en todos los casos al menos existió como promedio un trabajador de cada una de las categorías anteriores entre todas las instancias de la colección, por tanto, para poder determinar la causa debe profundizarse aún más el estudio.

En contraposición a lo anterior observando la fila con valor de GI 35 el atributo de *cant_esp_comp* si fue determinante en la mayoría de las bases de datos (PE0, PE1 y PE2) no así cuando se eliminaron las filas nulas de la base de datos (PE3 y PE4). En todos los casos como promedio se disponía de 5 “especialistas con competencias elevadas” en los proyectos, y todo apunta a que cuando la cantidad de “especialistas con competencias elevadas” en el proyecto fue de 0 dicho atributo logró un mayor poder discriminante. Al mismo tiempo se evidencia que este influye bastante en la evaluación del proyecto, sobre todo cuando fue cercano a cero, además esto se pudo constatar revisando que dicho atributo aparece más en las reglas que predicen la evaluación de Mal y Regular en los proyectos. Por tanto, parece ser que la cantidad de “especialistas con competencias elevadas” solo es discriminante cuando estos no están presentes en los proyectos pues esto trae consigo una mala evaluación de los proyectos, sin embargo, cuando estos están presentes no determinan la evaluación de Bien, pues esta es determinada más por los atributos analizados con anterioridad.

Otras relaciones pueden estudiarse a partir de los resultados obtenidos, pero el presente trabajo solo se concentró en las más significativas para la toma de decisiones. Finalmente, en todas las bases de datos se encontraron reglas con un poder predictivo por encima del 90% para las evaluaciones de Mal y Bien, en cambio para la clase de Regular el poder predictivo de las reglas no fue bueno pues estuvo entre el 40% y el 50% de efectividad.

Revista Cubana de Ciencias Informáticas
Vol. 11, No. 4, Octubre-Diciembre, 2017
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 39- 56

Conclusiones

Con el presente trabajo se validó la aplicación del algoritmo evolutivo de clasificación con programación de expresiones genéticas: MCGEP para predecir la evaluación de bases de datos de gestión de proyectos. Dicho algoritmo genera una base de reglas para la clasificación, descubre las reglas con mayor nivel de sensibilidad, especificidad y menor nivel de complejidad en el dominio de aplicación. Se compararon los resultados del algoritmo MCGEP frente a otros siete algoritmos reportados en la literatura para generar reglas de clasificación.

Con respecto a la variable "exactitud predictiva" el MCGEP obtuvo mejores resultados en las cinco bases de datos analizadas, aunque sin diferencias significativas con los algoritmos SLAVE y Gassist; la robustez del algoritmo Gassist, ha sido mostrada en varios trabajos de revistas indexadas en la *Web* de la Ciencia. En cambio, MCGEP si fue significativamente mejor que los algoritmos CORE, LogitBoost, UCS, Tan y Bojarczuk. En cuanto a la variable "complejidad de los modelos de predicción" se encontró que el algoritmo CORE es el que obtiene los sistemas de reglas más simples, aunque no se encontraron diferencias significativas con MCGEP, Gassist y Bojarczuk. Tomando en consideración las variables "complejidad de los modelos de predicción" y "exactitud predictiva" simultáneamente los algoritmos con mejores resultados fueron MCGEP y Gassist.

Por otro lado, al analizar las bases de reglas obtenidas, se logró determinar que, de los 84 atributos descriptores, los atributos *ire*, *ie*, *real_ire*, *impacto_total_hitos_ejec*, *plan_ire* e *irhf*, son los que en conjunto más aportan, para predecir una evaluación de un proyecto en cualquiera de las categorías Bien, Regular o Mal. Se encontró que si el atributo *ire* es menor que 0.5 permite predecir con bastante exactitud (de más de un 90%) que el proyecto será evaluado de Mal. Se evidenció también que existe una fuerte relación entre los atributos *project_id* y *fecha*, este efecto se produce porque el *id* del proyecto implícitamente implica un orden cronológico de creación de este.

Las bases de reglas obtenidas mostraron que la presencia de "especialistas con competencias elevadas" influye directamente en la evaluación de los proyectos sobre todo cuando esta es cercana a cero pues en ese caso permite definir que un proyecto será evaluado de Mal con una alta probabilidad; el empleo de especialistas con menos competencias tiene menor influencia en la evaluación de los proyectos.

En las pruebas de sensibilidad respecto a los atributos el algoritmo MCGEP mostró robustez y mantuvo resultados estables en la evaluación de los proyectos. Finalmente, como trabajo futuro se recomienda continuar avanzando en el análisis de datos para la toma de decisiones de gestión de proyectos, por el impacto económico y social de esta área de la actividad humana. También se recomienda, analizar el impacto de la cantidad de recursos humanos y no humanos, así como las competencias de los recursos humanos en la evaluación de los proyectos. En la base de datos empleada se

debe profundizar en el análisis de los casos evaluados de Regular pues en ellos los resultados de los algoritmos no fueron buenos.

Referencias

ALCALÁ-FDEZ JESÚS [y otros] KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework [Publicación periódica] // Multiple-Valued Logic and Soft Computing. - 2011. - 2-3. - págs. 255-287.

BACARDIT JAUME Pittsburgh genetic-based machine learning in the data mining era: representations and generalization and run-time [Informe]. - Barcelona : Department of Computer Science, University Ramon LLull, 2004.

BERNADÓ-MANSILLA ALBERT ORRIOLS-PUIG AND JORGE CASILLAS AND ESTER Genetic-based machine learning systems are competitive for pattern recognition [Publicación periódica] // Evolutionary Intelligence. - 2008. - 3 : Vol. 1.

BERNADÓ-MANSILLA ESTER AND GARRELL-GUIU, JOSEP MARIA Accuracy-based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks [Publicación periódica] // Evol. Comput.. - Cambridge, MA, USA : MIT Press, 2003. - 3 : Vol. 11.

BOJARCZUK CELIA CRISTINA AND LOPES, HEITOR SILVÉRIO AND FREITAS, ALEX ALVES AND MICHALKIEWICZ, EDSON LUIZ A. Constrained-syntax Genetic Programming System for Discovering Classification Rules: Application to Medical Data Sets [Publicación periódica] // Artif. Intell. Med.. - Essex, UK : Elsevier Science Publishers Ltd., 2004. - 1 : Vol. 30.

CANO ALBERTO, ZAFRA AMELIA Y VENTURA SEBASTIÁN An interpretable classification rule mining algorithm [Publicación periódica] // Information Sciences. 2013. - Vol. 240. - págs. 1-20.

CASTRO AGUILAR GILBERTO FERNANDO [y otros] Método para el aseguramiento de ingresos en entornos de desarrollo de software [Publicación periódica] // Revista Cubana de Ciencias Informáticas. - Noviembre de 2016. - Especial UCIENCIA : Vol. 10. - págs. 43-57.

CORDERO MORALES DASIEL [y otros] Sistema de Razonamiento Basado en Casos para la identificación de riesgos de software [Publicación periódica] // Revista Cubana de Ciencias Informáticas. – 2013. Vol. 7. No. 2. - págs. 95-107.

DERRAC JOAQUÍN [y otros] A practical tutorial on the use of nonparametric statistical test as a methodology for comparing evolutionary and swarm intelligence algorithms [Publicación periódica] // Swarm and Evolutionary Computation. - [s.l.] : Elsevier B.V, 2011. - 1 : Vol. 1.

DOMINGOS PEDRO Y PAZZANI MICHAEL On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss [Publicación periódica] // Machine Learning. - Hingham, MA, USA : Kluwer Academic Publishers, 1997. - 2-3. - págs. 103-130.

ESPEJO PEDRO G., VENTURA SEBASTIÁN Y HERRERA FRANCISCO. A Survey on the Application of Genetic Programming to Classification [Publicación periódica] // IEEE Transactions on Systems, Man, and Cybernetics. - 2009. - Vol. Part C: Applications and Reviews.

FERREIRA CÂNDIDA. Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Revised and extended edition [Libro]. - [s.l.] : Springer, 2006. - Vol. 2nd Edition : pág. 478. - ISBN 3-540-32796-7.

GARCÍA-VACACELA ROBERTO [y otros] Experiencias usando algoritmos genéticos en la planificación de proyectos [Publicación periódica] // Revista Cubana de Ciencias Informáticas. – 2016. Vol. 10. No. Especial UCIENCIA, Noviembre- págs. 71-86.

GONZÁLEZ ANTONIO AND PÉREZ, RAÚL. SLAVE: A Genetic Learning System Based on an Iterative Approach [Publicación periódica] // Trans. Fuz Sys.. - Piscataway, NJ, USA : IEEE Press, 1999. - 2 : Vol. 7.

GUERRERO-ENAMORADO ALAIN [y otros] An algorithm evaluation for discovering classification rules with gene expression programming [Publicación periódica] // International Journal of Computational Intelligence Systems. - 2016. - 2 : Vol. 9.

MARÍN-SÁNCHEZ JACQUELINE [y otros] Proceso para la planificación y control de proyectos de software utilizando Xedro-GESPRO [Publicación periódica] // Revista Cubana de Ciencias Informáticas. – 2014. Vol. 8. No. 2. - págs. 144-161.

MCCLELLAND JAMES L., RUMELHART DAVID E. Y PDP. Research Group Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations [Libro]. - Cambridge, MA, USA : MIT Press, 1986.

OLMO ORTIZ, JUAN LUIS. Minería de Datos mediante Programación Automática con Colonias de Hormigas [Tesis]. - Córdoba, España : Universidad de Córdoba, Servicio de Publicaciones, 2013.

ORRIOLS-PUIG ALBERT, CASILLAS JORGE Y BERNADÓ-MANSILLA ESTER Genetic-based machine learning systems are competitive for pattern recognition [Publicación periódica] // Evolutionary Intelligence. - [s.l.] : Springer, 2008. - 3 : Vol. 1. - págs. 209-232.

OTERO JOSÉ AND SÁNCHEZ, LUCIANO Induction of descriptive fuzzy classifiers with the Logitboost algorithm [Publicación periódica] // Soft Computing. - [s.l.] : Springer-Verlag, 2006. - 9 : Vol. 10.

PIÑERO PÉREZ PEDRO Y. [y otros] Repositorio para el desarrollo de investigaciones en Gestión de Proyectos [Publicación periódica]. - La Habana : [s.n.], 2016.

TAN K. C. AND YU, Q. AND ANG, J. H. A coevolutionary algorithm for rules discovery in data mining [Publicación periódica] // International Journal of Systems Science. - 2006. - 37 : Vol. 12.

TAN KAY CHEN AND TAY, ARTHUR AND LEE, TONG HENG AND HENG, C. M. Mining multiple comprehensible classification rules using genetic programming [Conferencia] // Proceedings of the 2002 Congress on Evolutionary Computation CEC2002. - [s.l.] : IEEE Press, 2002.

THE STANDISH GROUP INTERNATIONAL CHAOS REPORT [Informe]. - New York : The Standish Group International, Inc, 2014.

VAPNIK VLADIMIR NAUMOVICH The nature of Statistical Learning Theory [Libro]. - New York, USA : Springer-Verlag, 1995.

VENTURA SEBASTIÁN AND ROMERO, CRISTÓBAL AND ZAFRA, AMELIA AND DELGADO, JOSÉ Y A. AND HERVÁS CÉSAR JCLEC: A Java Framework for Evolutionary Computation [Publicación periódica] // Soft Computing. - Berlin, Heidelberg : Springer-Verlag, 2008. - 4 : Vol. 12.

WITTEN IAN H., FRANK EIBE Y HALL MARK A. Data Mining. Practical Machine Learning Tools and Techniques [Libro]. - MA 01803, USA : Elsevier Inc., 2011. - ISBN 978-0-12-374856-0.