UNIVERSIDAD DE CAMAGÜEY

FACULTAD DE INFORMÁTICA

Trabajo de diploma para optar por el Título de Master

Algoritmos Evolutivos Estimadores de Distribución Celulares para Problemas de Optimización Discretos.

Autor:

Demetrio Alejandro Rodríguez Fernández

Tutor:

MSc. Yoan Martínez López

Camagüey, Marzo de 2020

"Año 61 de la Revolución"

Dedicatoria y Agradecimientos

Dedico esta tesis y este momento tan especial de mi vida:

A Dios que me dio la vida y la capacidad de poder estudiar, y las fuerzas cada día para seguir adelante.

A mi esposa Denysse Guevara y mi niña Amanda Sofia, por compartir conmigo todos estos momentos, que han llenado mi corazón y me han dado tanta tranquilidad y alegría, por hacer que siempre me sienta bien, por cada momento que se mantuvieron ayudándome con esta tesis, por llenarme la vida de sueños y hacer que los ame tanto.

A mis padres por estar siempre a mi lado, por todo su amor, cariño, consejos y por estar pendiente de mí siempre.

A mi tutor Yoan Martínez por ayudarme en todo momento, y por dedicar gran parte de su tiempo en mí; para la realización de esta Tesis.

A mis amigos los que están lejos y cerca, por siempre darme ánimos para seguir adelante

Muchas gracias.

Declaración de Autoría

Por medio de esta declaración se hace constar que el presente trabajo pertenece al autor que firma esta declaratoria; la información publicada puede ser consultada o referenciada siempre y cuando se cite la fuente, la Universidad de Camagüey "Ignacio Agramonte Loynaz" está totalmente autorizada para hacer uso pertinente de esta investigación.

Y para que así conste se firma el presente a los _____ días del mes de _____ de 2020.

Autor _____ Tutor _____

Lic. Demetrio A. Rodríguez Fernández MS.c Yoan Martínez López

Resumen

En este documento, se presenta un nuevo algoritmo bayesiano de estimación celular para problemas de optimización discretos. Esta clase de algoritmo de optimización estocástica con aprendizaje de la estructura y los parámetros de las poblaciones locales se basa en la prueba de independencia y el esquema de poblaciones descentralizadas, lo que puede reducir el número de evaluaciones de funciones que resuelven problemas de optimización discretos. Los resultados experimentales mostraron que esta propuesta reduce el número de evaluaciones en la búsqueda del óptimo para funciones discretas de referencia con respecto a otros enfoques de la literatura. Además, logró mejores resultados estos algoritmos del estado del arte.

PALABRAS CLAVE: EDA celulares; Redes bayesianas; aprendizaje; algoritmo evolutivo.

Abstract

In this paper, a new Cellular Estimation Bayesian Algorithm for discrete optimization problems is presented. This class of stochastic optimization algorithm with learning from the structure and parameters of local populations are based on independence test and decentralized populations scheme, which can reduce the number of function evaluations solving for discrete optimization problems. The experimental results showed that this proposal reduces the number of evaluations in the search of the optimal for a benchmark discrete function with respect to other approaches of the literature. Also, it achieved better performance than them.

KEYWORDS: Cellular EDAs, Bayesian networks, learning, evolutionary algorithm

Índice

Contenido

Introducción

La computación evolutiva es una disciplina relativamente joven, el término fue adoptado en la década de los 90, esta puede catalogarse como un conjunto de métodos heurísticos que funcionan emulando mecanismos de evolución natural, los métodos heurísticos se basan en la reducción del espacio de búsqueda de la solución de un problema, basándose en el uso de alguna información específica, a partir de estos métodos surgen nuevas técnicas conocidas como metaheurísticas.

La idea fundamental de las metaheurísticas es utilizar distintos operadores de variación para explorar de forma eficiente y eficaz el espacio de soluciones, entre ellas se encuentran los algoritmos evolutivos, estos trabajan sobre una población de individuos o conjunto de soluciones, que evoluciona utilizando mecanismos de selección y construcción de nuevas soluciones candidatas mediante recombinación de características de las soluciones seleccionadas (Luke, 2010).

La computación evolutiva (EC) es una de las ramas de la Inteligencia Artificial que se aplica para la resolución de problemas de optimización combinatoria, la cual está inspirada en los mecanismos de evolución biológica propuestos por Darwin, Mendel y Lamark, donde Darwin propuso la “Selección natural de los más adaptados “, Mendel propuso la “Teoría corpuscular de la herencia” y Lamark propuso la “Herencia de caracteres adquiridos “.

Dentro de la computación evolutiva, se engloban los diferentes algoritmos evolutivos (EA) o estrategias para la resolución de problemas de optimización. Estas estrategias son las siguientes (Pino, 2013):

- Procesos de Búsqueda Evolutiva: Fue propuesta por Alan Turing en el año 1948
- Estrategias Evolutivas (EE): Propuesto por Rechenberg en 1964. Representan a los individuos con Vectores reales.
- Programación Evolutiva (PE): Propuesto por Fogel en 1965. Utilizan máquinas de estado finito.
- Algoritmos Genéticos (AG): Propuesto por Holland en 1975. Representan a los individuos como cadenas binarias.
- Programación Genética (PG): Propuesto por Koza en 1992. Utilizan Árboles LISP.

Los Algoritmos con Estimación de Distribuciones (EDA por sus siglas en inglés) son metaheurísticas que se basan principalmente en sustituir los operadores de cruce y mutación de individuos de los algoritmos genéticos por la estimación y posterior muestreo de una distribución de probabilidad aprendida a partir de los individuos seleccionados de una población.

En la última década los EDA se han aplicado a numerosos problemas de optimización que pudieran considerarse desafiantes. En muchos de estos estudios, los EDA fueron capaces de resolver problemas que eran intratables con otras técnicas o estas eran incapaces de lograr resultados comparables. (Larrañaga, Lozano et al., 2003) destacan la popularidad alcanzada por estos algoritmos en los últimos años. Los EDA son algoritmos eficientes que ahorran tiempo de ejecución y número de evaluaciones en el proceso de encontrar una solución lo suficientemente cercana al óptimo global de una función determinada.

En todo problema de optimización, existen dependencias entre las variables, las que no son inferidas por la mayoría de los métodos de optimización actuales (Algoritmos Genéticos, Particle Swarm Optimization, etc.). Para detectar las dependencias los EDA utilizan técnicas estadísticas las cuales en su mayoría no están implementadas en lenguajes de alto nivel, solo si se trata de bibliotecas especializadas.

Una debilidad de los EDAs en general es la eficiencia desde el punto de vista evaluativo en la resolución de problemas de optimización. Para afrontar esta debilidad se proponen los EDAs celulares, que permiten la descentralización de los individuos de la población.

Un EDA celular es una colección de EDAs colaborativos y descentralizados, también llamados algoritmos miembros que desarrollan poblaciones solapadas (Alba, 2006, pp.5-7) Un rasgo distintivo de esta clase de algoritmo es que se descentralizan a nivel de los algoritmos y la selección en otro algoritmo evolutivo usualmente ocurre a nivel de recombinación.

Por otro lado, es habitual aplicar diferentes clasificadores sin estudiar el comportamiento de los datos y en determinadas situaciones este no suele ser el deseado, por lo que es conveniente antes de implementar algoritmos que estiman distribuciones en lenguajes de alto nivel verificar su funcionalidad mediante la implementación de los mismos en lenguajes sencillos que permitan obtener resultados comparables con otros ya existentes.

Como resultado de lo planteado anteriormente el **problema a resolver** que se propone esta Tesis es ¿Cómo validar el funcionamiento de un EDAs Discretos antes de su implementación en

un lenguaje de alto nivel?, quedando definido como **objeto de estudio** los Algoritmos con Estimación de Distribución Celular.

Esta Tesis está motivada por la existencia de investigaciones anteriores que demuestran la efectividad de los EDAs Discretos al compararlos con otros algoritmos de optimización y obtener resultados positivos y por la existencia de bibliotecas implementadas anteriormente en lenguajes de alto nivel como Java y C++, por lo que la tesis pretende ampliar el desarrollo y utilización de los EDAs Discretos incorporando un lenguaje de programación de gran utilidad en el manejo de la computación estadística, en este caso el lenguaje R que por sus características y funcionalidades resulta idóneo para el trabajo con este tipo de algoritmos.

Una vez analizado al problema a resolver y teniendo en cuenta el objeto de estudio y el campo de acción propuesto, se plantea como **objetivo general** de esta tesis: Desarrollar una biblioteca de clases en lenguaje R para el trabajo con EDAs Celulares Discretos, la cual debe contar con un diseño que permita incorporar nuevas funcionalidades posteriormente.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas investigativas:**

1. Caracterización de los principales operadores que intervienen en los EDAs Celulares Discretos, así como de su estructura y funcionamiento.

2. Caracterización de las principales funcionalidades del lenguaje de programación R así como de las herramientas que brinda para el trabajo con métodos estadísticos y probabilísticas.

3. Implementación de algunos de los principales exponentes de la familia de los EDAs Celulares Discretos en R.

4. Validación mediante experimentos de los EDAs Celulares Discretos implementados y comparación con otros del estado del arte.

En el capítulo 1 se analiza el marco teórico de los EDAs Celulares Discretos, a su vez se definen los principales modelos gráficos probabilísticos que intervienen en la detección de dependencias y las clasificaciones de los EDAs Celulares Discretos teniendo en cuenta el dominio sobre el cual trabajan. Además se muestra una panorámica de las bibliotecas desarrolladas anteriormente así como de las principales características del lenguaje y entorno de desarrollo R que se utilizarán en el trabajo.

En el capítulo 2 se presenta el análisis y diseño de la biblioteca, teniendo en cuenta la estructura de las clases y funciones implementadas, así mismo se define el funcionamiento de la biblioteca y cómo interactúan los diferentes componentes de la misma.

En el capítulo 3 se presenta un resumen de los resultados obtenidos una vez realizados los experimentos que ilustran el comportamiento del algoritmo EDA celular en las funciones de pruebas con diferentes vecindades.

Capítulo 1.Algoritmode Estimación de distribución Celular

En un EDA celular el ciclo reproductivo se ejecuta dentro de cada número de individuos de la población local, el cuales usualmente llamado célula, tienen sus propias poblaciones locales definidas por subpoblaciones vecinas y al mismo tiempo una célula pertenece a muchas poblaciones locales.El conjunto de todas las células define una partición de la población global.

La característica principal de la mayoría de EDA celulares es la utilización de modelos gráficos probabilísticos para detector las dependencias entre las variables de problemas a resolver, Larrañaga y Puerta (2003) plantean que en la estimación de la distribución de probabilidad recae el paso más complejo dentro de esta nueva aproximación que constituyen los EDA, como se muestra en la figura 1.

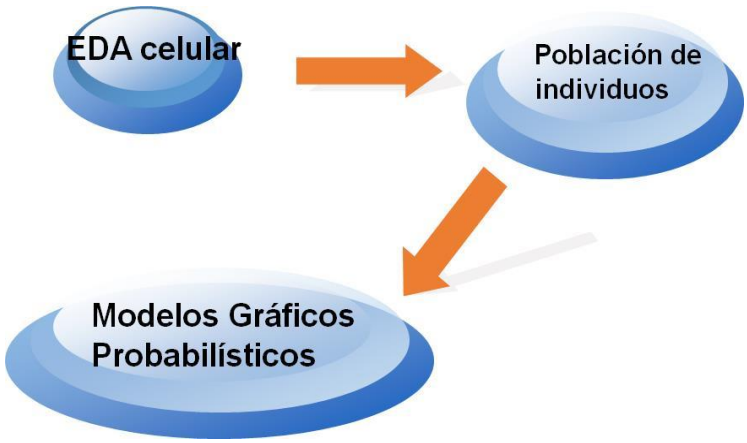

Figura 1. Representación de un ciclo reproductivo de EDA Celular

Modelos Gráficos Probabilísticos.

Los Modelos Gráficos (MG) según Madera (2008):

Son herramientas que permiten representar distribuciones de probabilidad conjunta. Los Modelos Gráficos Probabilísticos (MGP) constituyen grafos en los cuales los nodos representan

variables aleatorias y los arcos representan relaciones de dependencia condicional. Estos grafos proveen una forma compacta de representar la distribución de probabilidad.

Los MGP empleados por los algoritmos EDA varían en función del dominio de las variables del problema. Si estas variables son discretas se utilizan redes Bayesianas. Si por el contrario se trata de variables continuas se utilizan redes Gaussianas. Existe la posibilidad de generar modelos probabilísticos híbridos, adaptados para problemas convariables discretas y continuas, como se muestra en la figura 2.

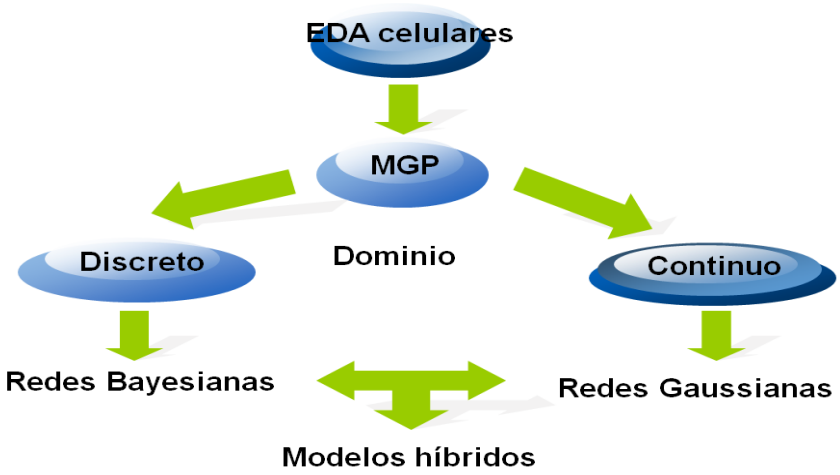

Figura 2. Representación de Modelos Gráficos Probabilísticos

1.1.1 Redes Bayesianas

Una red Bayesiana es un tipo de MG que utiliza Grafos Acíclicos Dirigidos (DAG), por lo que toma en consideración la dirección de los arcos. Una red Bayesiana se define mediante el par $\langle G, P \rangle$ donde G es un grafo que representa las relaciones de dependencia entre las variables y *P* es la factorización de la distribución de probabilidad representada por *G*.

Formalmente se define una red Bayesiana sobre un conjunto, V = {V_1,…, Vn}, de variables aleatorias.

La factorización de la probabilidad conjunta puede expresarse como:

$$P(V) = P(V_1, V_2, ..., V_n) = \prod_{i=1}^{n} P(V_i \mid Pa_i)$$

La expresión permite definir una red Bayesiana con la condición de Markov, cada variable (Vi) es independiente de cualquier subconjunto de las variables no descendiente de ella, condicionado en su conjunto de padres (Pa_i).

Para el aprendizaje de Redes Bayesianas existen dos técnicas fundamentales: el aprendizaje basado en restricciones (constraint based learning), o algoritmos que detectan independencias, y el aprendizaje basado en optimización de métricas (search-and-score based learning), conocidos como métodos de puntuación. También los autores Larrañaga et al. (1999) clasifican los modelos que aprenden Redes Bayesianas de acuerdo a dos puntos de vista, el primero de acuerdo a la complejidad del problema aprendido (árbol, poliárbol o múltiples conexiones). Estos modelos llevan a cabo dos tareas fundamentales: primero realizan un aprendizaje estructural para identificar la topología de la red y a partir de este estiman los parámetros (aprendizaje paramétrico) representados mediante probabilidades condicionales (Madera, 2008).

Los algoritmos que tratan de recuperar la estructura de una Red Bayesiana a partir de la detección de independencias tiene como entrada algunas relaciones de dependencias condicionales entre algunos subconjuntos de variables del modelo, y tienen como salida un grafo acíclico dirigido que representa un gran porcentaje de esas relaciones. Una vez que la estructura ha sido aprendida, las distribuciones de probabilidad condicionales requeridas para especificar el modelo son estimadas a partir de los datos usando algunas de las diferentes aproximaciones para el aprendizaje paramétrico. La información de entrada para los algoritmos de esta categoría puede ser una base de datos con pruebas estadísticas que posibilitan determinar la corrección de algunas relaciones de independencias condicionales, una distribución de probabilidad n-dimensional donde se posible probar la veracidad de las relaciones de independencias condicionales y una lista relaciones de dependencia e independencias condicionales entre tripletas de variables , en la práctica la diferencia entre estos tres tipos de información radica en aspectos relacionados con el costo de realizar las pruebas estadísticas, que incrementa con el número de variables que se tiene en cuenta a la hora de realizar las pruebas y por otro lado la fiabilidad de los resultados de las pruebas, la cual es menos robusta si el número de variables es muy grande(Larrañaga et al., 2003).

Por otro lado los algoritmos basados en optimización de métricas utilizan métodos de puntuación (probabilidad máxima con penalización, probabilidad marginal, basados en teoría de la información, etc.) y aproximaciones de búsqueda (tabu, greedy, etc.).

Kontos (2009) expone que el aprendizaje a partir de los datos, es típicamente proyectado como un problema de optimización en el cual la tarea computacional es encontrar la estructura que maximice el resultado, encontrar la Red Bayesiana óptima es posible solo para las redes que contienen unos pocos individuos, ya que las técnicas de aprendizaje dirigidas a la solución de este problema no garantizan conducir al óptimo global.

Pese a ello, el aprendizaje de Redes Bayesianas de grandes dimensiones se ha convertido en una tarea prioritaria en los últimos años, esto se debe a su aplicación en diferentes campos como es el de la Bioinformática. Ejemplo de ello son los trabajos de Friedman (pionero de esta rama) en la aplicación de Redes Bayesianas para el aprendizaje a gran escala de los grafos dirigidos de los datos de microarrays (pequeñas muestras de ADN para pruebas genéticas).

Estrategia de aprendizaje

Un asunto crítico en un EDA celulares el uso de una estrategia que aprendan del modelo probabilístico porque usualmente no son eficientes desde el punto de vista evaluativo, lo cual puede afectar el rendimiento del algoritmo, por lo que el aprendizaje de la estructura y los parámetros a partir de poblaciones locales, puede ser una de las alternativas para dar solución a este problema. El operador de aprendizaje de los EDA celulares se basa en el algoritmo estimador de Shrinkage a partir de la matriz de covarianza de un conjunto de datos.

Vecindades

Un vecindario es “un conjunto de individuos vecinos a uno dado, es decir, que están situados próximos a él en la población según una topología especialdad a de la rejilla”(Dorronsoro,2008).

El vecindario de 5 individuos, denominado comúnmente NEWS (North, East, West, South), considera el individuo central y los inmediatamente superior, inferior, izquierdo y derecho. Existen otras vecindades, como es el caso de One, L9,C9,C13 o compactos de C25 y C41,el usar un vecindario de menor radio hace que las soluciones se extiendan más lentamente por la población, induciendo una menor presión selectiva global y manteniendo mayor diversidad genética que al usar vecindarios mayores, como se muestra en la figura 3.

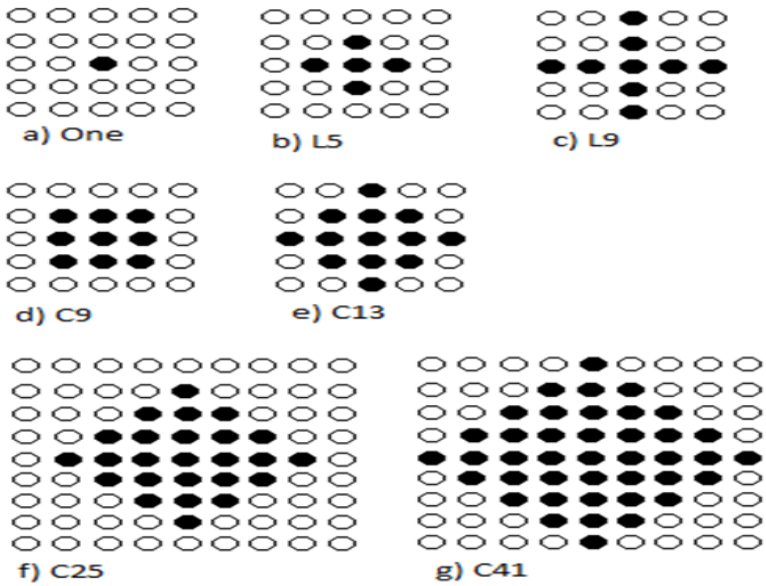

Figura3. Representación de diferentes vecindades

1.1.2 EDA discretos.

Dentro del grupo de algoritmos que trabajan sobre dominio discreto se encuentra el UMDA (Univariate Marginal Distribution Algorithm), este fue propuesto por Mühlenbein (1998) generalizando trabajos previos de Syswerda (1993), Eshelman y Schäffer (1993) y Mühlenbein y Voigt (1996). El UMDA es el algoritmo más simple de la familia de los EDA, en él cada distribución de probabilidad univariante se estima a partir de las frecuencias marginales. Dentro del grupo de algoritmos que asumen independencia entre las variables y que tienen un comportamiento similar al UMDA se encuentran el PBIL (Population Based Incremental Learning), introducido por Baluja (1994) y posteriormente mejorado por Baluja y Caruana (1995), este algoritmo ha recibido mucha atención por parte de los investigadores y se pueden encontrar trabajos donde se aplica el PBIL al problema de obtener los pesos óptimos en una red neuronal y a un problema relacionado con vehículos inteligentes, también en este grupo se encuentra el cGA (compact Genetic Algorithm) presentado por Harik et al.(1998).

El algoritmo voraz MIMIC (Mutual Information Maximization for Input Clustering), fue presentado por Bonet et al. (1997), este algoritmo busca en cada generación la mejor permutación

entre las variables con el objetivo de encontrar en cada generación la distribución de probabilidad p_l (x), que se encuentra más cercana en divergencia de Kullback-Leibler a la distribución empírica del conjunto de individuos seleccionados. Los autores proponen un algoritmo voraz hacia adelante que evita buscar en todo el espacio de permutaciones de dimensión n!.

En el algoritmo PADA (Polytree Approximation of Distribution Algorithms), propuesto por Soto, et al.(1999), la factorización se lleva a cabo por medio de una red Bayesiana con estructura de poliárbol (no existe más de un camino no dirigido entre cada par de variables). Este algoritmo puede considerarse un híbrido entre los métodos que se basan en detectar independencias condicionales y aquellos englobados dentro de los denominados métodos score + búsqueda.

El algoritmo BOA (Bayesian Optimization Algorithm) fue propuesto por Pelikan et al.(1999) y Pelikan et al.(2000), este algoritmo asume interacciones generales entre las variables y representa el modelo de probabilidad conjunta a través de las redes Bayesianas , el algoritmo BOA usa la métrica BDe (Bayesian Dirichlet equivalence) para determinar cuál es la red que mejor se ajusta a los datos. Esta métrica Bayesiana tiene la propiedad de asignar un mismo valor a estructuras que reflejan las mismas independencias condicionales. La búsqueda de la mejor estructura es voraz y comienza en cada generación en el grafo sin arcos. Para tratar de reducir la cardinalidad del espacio de búsqueda se asume la restricción de que cada nodo en la red Bayesiana tiene a los sumo k padres.

Otro algoritmo que propone la optimización Bayesiana es el EBNA(Estimation of Bayesian Networks Algorithm), este fue presentado en los trabajos de Etxeberria y Larrañaga (1999) y Larrañaga et al.(2000), el algoritmo la factorización de la distribución de probabilidad conjunta es codificada por medio de una red Bayesiana inducida en cada generación a partir de la base de datos conteniendo los individuos seleccionados. A partir de este se implementaron criterios para guiar la búsqueda de buenas estructuras, así surgieron instancias del EBNA: los basados en scores: EBNABIC, EBNAK2+pen y los basados en el testeo de independencias condicionales entre tripletas de variables EBNAPC. Similar al BOA y al EBNA es el algoritmo LFDA (Learning Factorized Distribution Algorithm) introducido por Mühlenbein y Mahnig (1999), en el LFDA la complejidad del modelo aprendido se controla por la métrica BIC en conjunción con una restricción acerca del máximo número de padres que cada variable puede llegar a tener en la red Bayesiana.

El algoritmo FDA-learning fue propuesto por Ochoa et al.(1999), este algoritmo inicial, aprende (por medio de pruebas de independencia condicional) un árbol de unión a partir de una base de datos. La idea subyacente es obtener el árbol de unión que mejor satisface las independencias condicionales detectadas por las pruebas de independencia condicional.

1.2 Bibliotecas de EDAs Celulares Discretos.

Como punto de partida para la creación de una nueva biblioteca de clases resulta conveniente realizar un análisis general de las bibliotecas implementadas anteriormente.

El trabajo de diploma de Duarte (2003) propone una biblioteca de clases para el desarrollo e implementación de EDA tanto secuenciales como paralelos. El objetivo fundamental de esta biblioteca (EDALib) es proveer un marco común para el desarrollo de algoritmos EDA tanto secuenciales como paralelos, fácil de utilizar y de extender. La Programación Orientada a Objetos (POO) se encamina hacia estos objetivos y por eso EDALib fue pensada siguiendo ese paradigma. Como lenguaje para la implementación se escogió C++ ya que este lenguaje cuenta con compiladores para las plataformas existentes en el mundo científico para el desarrollo de aplicaciones paralelas. En esta biblioteca se implementaron dos métodos de estimación de distribuciones: UMDA en el dominio discreto y UMDAc en el dominio continuo.

Por otra parte el trabajo de diploma de Aviles y Mahdi (2008) de la Facultad de Informática de la Universidad de Camagüey propone una biblioteca para el desarrollo de EDA secuenciales, la biblioteca (EDALib) se implementó en lenguaje Java y siguiendo el paradigma de la POO, EDALib posibilita la utilización de las operaciones comunes del trabajo con EDA, a su vez permite definir nuevas operaciones e incorporar nuevas funcionalidades a las ya existentes, lo que la define como una biblioteca de fácil usabilidad y con numerosas opciones de extensión. Los EDA fueron tratados como objetos que actúan sobre una población de individuos a la cual se le aplican los operadores comúnmente usados: selección, estimación (en este caso se implementó el UMDA para dominio continuo y discreto), muestreo, etc.

Aunque estas bibliotecas en lenguajes de alto nivel garantizan la incorporación de nuevas funcionalidades, solo implementan el operador con estimación UMDA, el cual constituye el modelo más simple ya que no considera relaciones de dependencia entre las variables, si se quisieran incorporar nuevos operadores de estimación o aprendizaje con modelos más complejos la

implementación de los mismos resultaría complicada por lo que se necesaria la utilización de nuevas herramientas que faciliten el trabajo con una gran variedad de modelos de estimación existentes.

La creación de una nueva biblioteca en R permite crear nuevos EDA con estimadores más complejos que tienen en cuenta las relaciones de dependencias entre las variables, a su vez el funcionamiento adecuado de estos algoritmos puedan ser verificado de manera sencilla.

1.3 El entorno R.

R es un sistema para análisis estadísticos y gráficos creado por Ihaka y Gentleman(1996), tiene una naturaleza doble de programa y lenguaje de programación y es considerado como un dialecto del lenguaje S creado por los Laboratorios AT&T Bell.

R puede definirse como un conjunto integrado de programas para manipulación de datos, cálculo y gráficos.

Entre otras características dispone de:

- Almacenamiento y manipulación efectiva de datos.
- Operadores para cálculo sobre variables indexadas (Arrays), en particular matrices.
- Una amplia, coherente e integrada colección de herramientas para análisis de datos.
- Posibilidades gráficas para el análisis de datos, que funcionan directamente sobre pantalla o impresora.
- Un lenguaje de programación bien desarrollado, simple y efectivo, que incluye condicionales, ciclos, funciones recursivas y posibilidad de entradas y salidas. (Debe destacarse que muchas de las funciones suministradas con el sistema están escritas en el lenguaje R).

El término “entorno" lo caracteriza como “un sistema completamente diseñado y coherente, antes que como una agregación incremental de herramientas muy específicas e inflexibles”, como ocurre frecuentemente con otros programas de análisis de datos (Ihakay Gentleman, 2000).

La forma más conveniente de usar R es en una estación de trabajo con un sistema de ventanas. Existen dos versiones de R para Microsoft Windows: Una basada en ventanas MDI, RGui.exe (utilizada en el trabajo), y otra en ventanas SDI, Rterm.exe, pensada especialmente para uso no interactivo.

1.3.1 R Orientado a Objetos

R es un lenguaje interpretado (lo cual significa que los comandos escritos en el teclado son ejecutados directamente sin necesidad de construir ejecutables) y Orientado a Objetos.

En R las variables, datos, funciones, resultados, etc., se guardan en la memoria activa del computador en forma de *objetos* con un nombre específico. El usuario puede modificar o manipular estos objetos con *operadores* (aritméticos, lógicos y comparativos) y *funciones* (que a su vez son objetos).

Los argumentos pueden ser objetos ("datos", fórmulas, expresiones, etc.), algunos de los cuales pueden ser definidos por defecto en la función; sin embargo estos argumentos pueden ser modificados por el usuario con opciones. Una función en R puede carecer totalmente de argumentos, ya sea porque todos están definidos por defecto (y sus valores modificados con opciones), o porque la función realmente no tiene argumentos, ver figura 4.

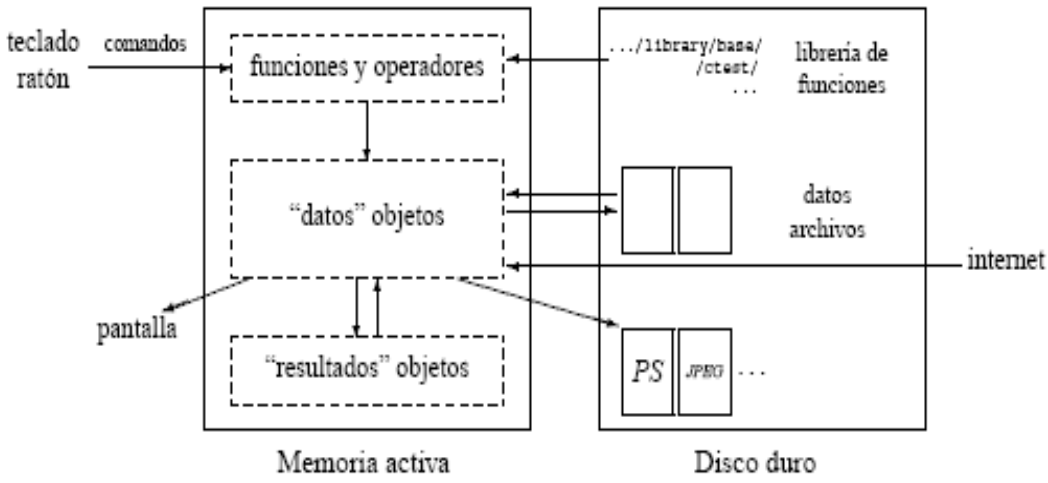

Figura 4 Funcionamiento de R.

Como se observa en la figura 4 todas las acciones en R se realizan con objetos que son guardados en la memoria activa del ordenador, sin usar archivos temporales. La lectura y escritura de archivos solo se realiza para la entrada y salida de datos y resultados (gráficas).

Paradis(2003) plantea que:

> El usuario ejecuta las funciones con la ayuda de comandos definidos. Los resultados se pueden visualizar directamente en la pantalla, guardar en un objeto o escribir directamente en el disco (particularmente para gráficos). Debido a que los resultados mismos son objetos, pueden ser considerados como datos y analizados como tal. Archivos que contengan datos pueden ser leídos directamente desde el disco local o en un servido remoto a través de la red.

Gracias a la programación orientada a objetos el usuario puede crear en R sus propias clases con diferentes atributos (en R denominados slots), los cuales a su vez pueden ser objetos de otra clase, asímismo R permite utilizar la herencia entre clases.

Cada objeto en R pertenece a una clase que determina de qué forma será tratado el mismo, esto se realiza a través de una función llamada genérica que realiza una tarea o acción específica sobre los argumentos del objeto de una clase determinada, este mecanismo ofrece la posibilidad de diseñar y escribir funciones genéricas independientes de las ya incorporadas por el lenguaje para propósitos específicos.

1.3.2 Paquetes en R.

Si nos enfocamos en el contexto de los EDAs Celulares Discretos como algoritmos que se basan en métodos probabilísticos y estadísticos, R constituye un lenguaje apropiado para el trabajo con los mismos, entre las funcionalidades que presenta y que son de gran interés a la hora de implementar los operadores que intervienen en los EDAs Celulares Discretos se encuentran la generación de grandes cantidades de datos ya sea de forma aleatoria o siguiendo una distribución y la capacidad para seleccionar, ordenar y muestrear datos a partir de variables indexadas, específicamente en matrices.

Estas ventajas que brinda R han sido explotadas por numerosos autores que se han dado a la tarea de complementar R con numerosas funcionalidades adicionales que han posibilitado extender el uso del lenguaje a diversos contextos incluyendo el de la computación evolutiva.

Díaz-Uriarte(2003) plantea que R consta de un"sistema base de paquetes" y por otro lado de "paquetes adicionales" que extienden la funcionalidad. Existen centenares de paquetes contribuidos para R creados por diferentes autores. Algunos de estos paquetes implementan métodos estadísticos especializados, entre ellos se encuentran los estadísticos de orden n, utilizados para estimar distribuciones de probabilidad en los EDAs Celulares Discretos. En este trabajo se utilizan los paquetes importados "bnlearn" y"corpcor".

Paquete"bnlearn".

El paquete "bnlearn" versión 2.8 fue creado por Scutari(2011), este paquete está diseñado:

> Para el aprendizaje estructurado de Redes Bayesianas mediante algoritmos basados en restricciones (con straintbase dlearning), en optimización de métricas (search-and-score based learning) y algoritmos híbridos. A su vez contiene funciones que permiten la comparación de modelos, la manipulación y generación aleatoria de datos, y la generación de gráficos.

Paquete"corpcor".

El paquete"corpcor"versión 1.6.4 fue creado por Schäfer et al.(2012) para:

La estimación eficiente de la covarianza y la correlación parcial. Este paquete implementa el estimador con contracción de James Stein para la matriz de covarianza con contracciones separadas para las varianzas y las correlaciones. Además el paquete provee estimadores para las correlaciones y varianzas parciales.

Ambos paquetes pueden ser instalados y cargados desde la consola principal de R para su uso en la biblioteca.

1.4 Visual Paradimg

Es una poderosa herramienta CASE, por excelencia para ser utilizada en un ambiente de software libre, debido a la posibilidad de ejecutarse sobre cualquier sistema operativo, lo que la convierte en una herramienta multiplataforma. Permite crear tipos diferentes de diagramas en un ambiente totalmente visual.

Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite graficar todos los tipos de diagramas

delases, código inverso, generar código desde diagramas y generar documentación. También proporciona tutoriales, demostraciones interactivas y proyectos UML.(Paradimg,2013).

1.5 Conclusiones parciales.

- Los EDAs Celulares Discretos constituyen una herramienta útil de gran explotación en el ámbito evolutivo, las numerosas investigaciones sobre el tema atraen la atención de estudiosos que se han enfrascado en la búsqueda de nuevas aproximaciones que permitan extender cada vez más esta familia de algoritmos con el propósito de ampliar su uso específicamente en la búsqueda de soluciones a problemas de optimización.
- El hecho de que la estimación de la distribución de probabilidad constituye un paso complejo en el desarrollo del algoritmo puede frenar de alguna manera la utilización de los mismos si no se cuenta con un lenguaje de programación que posibilite la adecuada implementación del modelo gráfico a utilizar.
- Las bibliotecas de clases facilitan el trabajo con EDAs Celulares Discretos y posibilitan la realización de experimentos que permiten obtener resultados concretos.
- Dado el carácter estadístico del entorno R, con la creación de una biblioteca de clases en este lenguaje se aprovechan las numerosas funcionalidades que brinda el entorno y de esta manera los algoritmos implementados pueden ser modificados de forma sencilla y validados a través de experimentos que los comparen con resultados ya existentes.

Capítulo 2. Diseño e implementación de los EDAs Celulares Discretos.

2.1 Introducción

Una vez analizado el marco teórico en el cual se concentra el objetivo del trabajo, en este capítulo se presenta el diseño de la biblioteca de clases, para ello en el epígrafe 2.2 se realiza un análisis detallado del funcionamiento de las diferentes clases y funciones de la biblioteca, en el epígrafe 2.3 se presentan los resultados experimentales que se obtuvieron de las corridas realizadas a los EDA implementados en la biblioteca. Por último, en el epígrafe 2.4 se presentan las conclusiones del capítulo.

2.2 Diseño de la biblioteca.

En este trabajo se desarrolla una biblioteca de clases en R que provee un marco común para el desarrollo de algoritmos EDA. Con esta biblioteca, de ahora en adelante REDA, se pretende mediante la implementación de EDA, realizar experimentos que permitan observar el comportamiento de estos algoritmos utilizando diversos criterios de estimación, muestreo, etc. Para ello REDA debe cumplir con las siguientes funcionalidades:

- Inicializar una población para dominio discreto.
- Seleccionar un conjunto N de individuos de acuerdo a un método de selección.
- Evaluar una población en una función de aptitud para dominio discreto.
- Aprender la distribución de probabilidad de una población utilizando un algoritmo de aprendizaje.
- Muestrear una nueva población a partir del aprendizaje realizado y la población seleccionada.
- Verificar si se cumple un criterio de parada.
- Ejecutar un EDA discreto.
- Ejecutar un EDA modificando los parámetros de configuración a través de la consola.
- Mostrar en consola los resultados de la corrida del algoritmo.

REDA está integrada por dos componentes fundamentales: clases y funciones genéricas. Un objeto de la clase EDA se comporta como la unión de los diferentes operadores implementados en las

clases correspondientes, los cuales interactúan con las poblaciones de individuos en las diferentes generaciones del algoritmo, a su vez, cada operador es completamente independiente de los demás y cada uno es responsable de la función que ejerce sobre el algoritmo, de esta manera si es necesario cambiar la configuración de alguno de ellos los demás no se verán afectados y el algoritmo EDA no sufrirá variación alguna en cuanto a su estructura general. A continuación se muestra el diagrama de clases que muestra como están relacionadas las clases existentes en la biblioteca, ver figura 5.

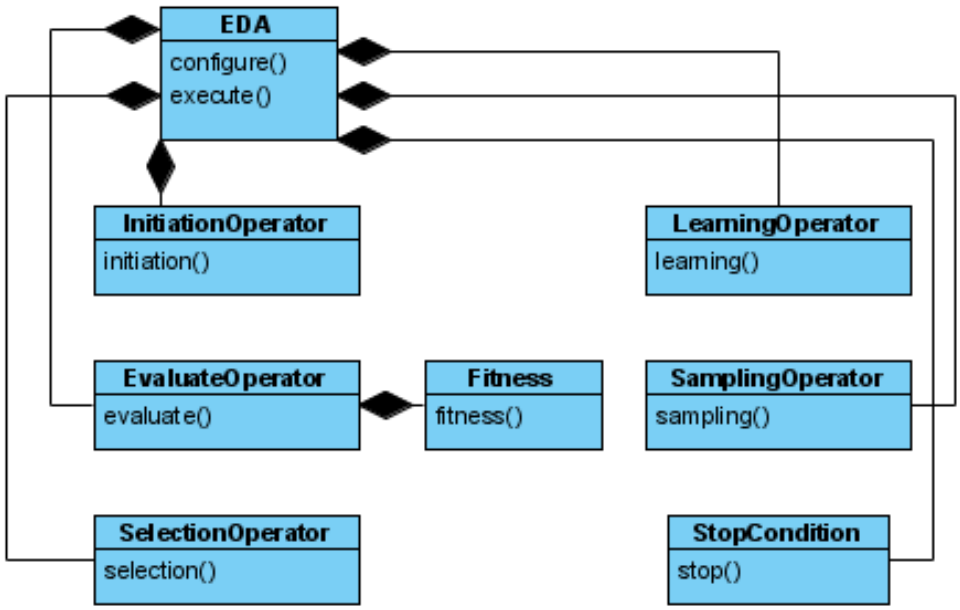

Figura 5. Diagrama de clases de la biblioteca RCEDA.

Para que la biblioteca logre una estructura que resulte fácil de usar y que permita incorporar nuevas funcionalidades, los métodos que utiliza REDA se implementaron utilizando las funciones genéricas de R, una función genérica determina a partir de la clase del objeto que se pasa por parámetro cuál será la acción que realizará, si tomamos como ejemplo el operador de selección, podemos crear varios operadores de este tipo ya que son numerosas las formas existentes de seleccionar individuos de una población, a cada clase que implementa un nuevo operador de selección se le asigna una acción de la función genérica selection() a través de la sentencia setMethod(), y de esta manera basta con llamar a selection() y pasarle el operador de selección como primer parámetro, esta función genérica se encarga de ejecutar la acción que previamente fue

asignada a ese tipo de operador de selección. Si se quieren adicionar nuevos operadores de selección solamente es necesario adicionar a la función selection() acciones que determinen el trabajo con estos operadores, lo cuál no representa una alteración de las funcionalidades implementadas con anterioridad.

2.2.1 Poblaciones

En la figura 6 se muestra el diagrama de clase que ilustra las principales características de una población:

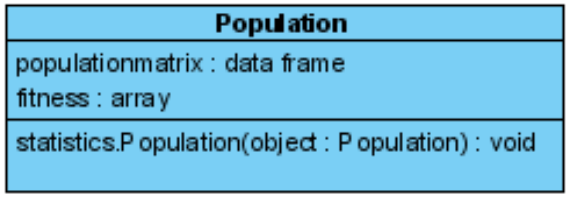

Figura 6. Diagrama de la clase Population.

Un objeto de la clase Population contiene una población de individuos (population matrix) y los valores de aptitud de los mismos (fitness), la población de individuos está representada por una matriz donde las variables de cada fila corresponden a los valores que puede tomar un individuo de la población, los valores de aptitud se almacenan en un arreglo donde cada valor es el resultado de evaluar un individuo en la función objetivo, de esta manera el primer valor de aptitud del arreglo fitness corresponde al primer individuo de la matriz de población y así sucesivamente. El almacenamiento de los individuos en una matriz de datos de R posibilita utilizar las diferentes funciones y operaciones para el trabajo con matrices que brinda este lenguaje.

El método statistic.Population se encarga de imprimir en la consola de R los valores de las variables del mejor individuo de cada generación así como su valor de aptitud.

2.2.2 Operadores de iniciación.

Los operadores de iniciación crean e inicializan un matriz de población a partir de los parámetros suministrados y en dependencia del tipo de dominio (continuo o discreto). Como se muestra en la figura 7.

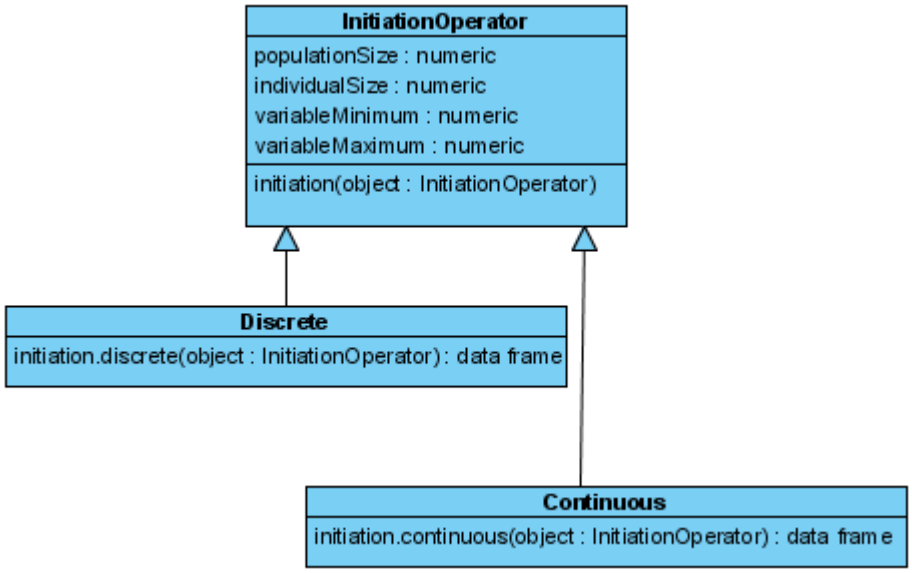

Figura 7. Diagrama de clases de los operadores de iniciación.

Como se observa en la figura 7 un operador de iniciación contiene el tamaño de la población a generar (population Size) y de los individuos de la misma(individual Size), así como la cota inferior(variable Minimun) y superior(variable Maximun) de los valores que pueden tomar los individuos de la población, las clases Discrete y Continuous heredan de la clase Initiation Operator, estas solo se diferencian en el dominio sobre el cual operan, para ello se definen los métodos que especifican de que forma se creará la nueva población, ambos métodos, initiation.discrete e initiation.continuous crean una matriz de m*n dimensiones, la cual es inicializada de forma eficiente recurriendo a las funcionalidades brindadas por R para el manejo de este tipo de datos, para llenar la matriz de población initiation.discrete genera valores enteros en un intervalo utilizando la función *sample* del paquete *base* de R que genera una muestra de tamaño específico a partir de los elementos de un vector dado. Por otro lado, initiation.continuous utiliza la función *runif* del paquete *base* de R que utiliza una distribución uniforme para generar valores aleatorios continuos dentro de un intervalo específico. Ambos métodos retornan una matriz de población con los individuos generados.

Es posible adaptar estos operadores si se desea modificar algún parámetro a la hora de llenar la matriz de población, por otro lado pueden incorporarse a la biblioteca operadores de iniciación que presenten características similares a los ya implementados.

2.2.3 Operadores de evaluación.

La clase Evaluate Operator tiene un único parámetro de clase Fitness que contiene el nombre de la función con la cual se va a correr el algoritmo y los parámetros adicionales de esta en caso de tener alguno.

A continuación se muestra el diagrama de clases relacionado con los operadores de evaluación:

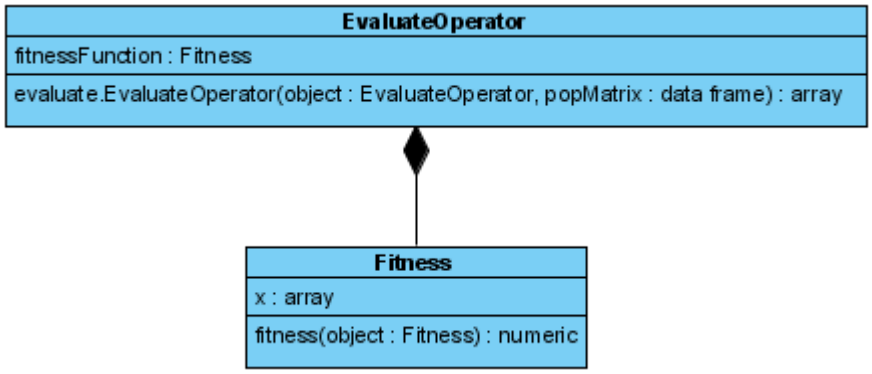

Figura 8. Diagrama de clases del operador de evaluación.

Como se observa en la figura 8 la clase Fitness contiene un arreglo que representa a un individuo de la población que será evaluado en una función de aptitud. De esta clase heredan las diferentes funciones continuas y discretas implementadas en la biblioteca.

Al método evaluate se le pasa por parámetro la matriz de población, de la cual se almacenan las filas(individuos) en el parámetro x del objeto Fitness de la clase EvaluateOperator, mediante un ciclo evaluate llama a la función fitness de la clase de dicho nombre que es la encargada de evaluar a x en la función de aptitud anteriormente especificada y devolver un único valor numérico correspondiente al fitness de cada individuo que se almacena en un arreglo que finalmente devuelve el método evaluate.

2.2.4 Operadores de selección

Estos operadores seleccionan individuos de una población a partir de un criterio específico. Como se observa en la figura 9 un operador de selección contiene un atributo *selSize* que representa la cantidad de individuos a seleccionar de la población, la clase Truncation hereda de SelectionOperator y contiene un método *selection.Truncation* que selecciona los mejores individuos de la población que analiza atendiendo a un problema de maximización, es decir,

escoge los individuos que poseen mayor valor de aptitud (fitness). Este método retorna una matriz con la cantidad de individuos que forman la población seleccionada.

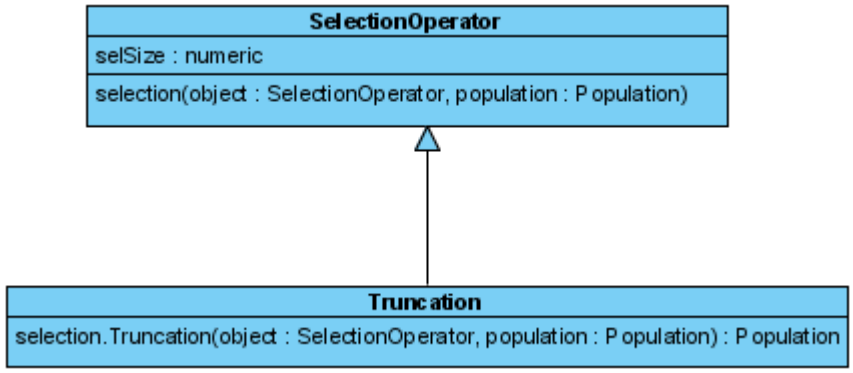

Figura 9. Diagrama de clases del operador de selección.

2.2.5 Operadores de aprendizaje

Figura 10. Diagrama de clases del operador de aprendizaje.

La clase LearningOperator como se muestra en la figura 10 no contiene atributos ya que los operadores de aprendizaje que implementa la biblioteca utilizan funciones del paquete *bnlearn* de R, estas funciones presentan diferentes parámetros de configuración por lo que a la hora de inicializar un operador de aprendizaje pueden definirse los parámetros que se van a utilizar, si en la corrida de un algoritmo no se va a utilizar un operador de aprendizaje este se inicializa por defecto.

A continuación se presentan los diferentes algoritmos del paquete *bnlearn* utilizados para implementar los operadores de aprendizaje:

Algoritmos con aprendizaje basado en restricciones o algoritmos que detectan independencias.

• Grow-Shrink (gs): Basado en el Grow-Shrink Markov Blanket, este es el primero y más simple de los algoritmos de detección de redes de Markov usado en un algoritmo de aprendizaje estructurado(Margaritis, 2003).

• Incremental Association (iamb): Basado en el algoritmo de detección de redes de Markov del mismo nombre, el cual se basa en un esquema de selección de dos fases: selección hacia delante seguida por un intento de remover los falsos positivos(Tsamardinos et al., 2003).

• Fast Incremental Association (fast.iamb): Variante del algoritmo IAMB que reduce el número de pruebas de independencias condicionales (Yaramakala y Margaritis, 2005).

• Interleaved Incremental Association (inter.iamb): Variante del algoritmo IAMB que evita falsos positivos en la fase de detección de redes de Markov(Tsamardinos et al., 2003).

Algoritmos con aprendizaje basado en optimización de métricas o métodos de puntuación.

• Hill-Climbing (hc): Búsqueda ávida con escalador de colinas en el espacio de los grafos dirigidos.

•Tabu Search (tabu): Modificación del escalador de colinas capaz de escapar al óptimo local seleccionando una red que disminuya de forma mínima la función de puntuación.

Algoritmos híbridos.

• Max-Min Hill-Climbing (mmhc): Algoritmo híbrido que combina los algoritmos Max-Min Parents and Children (restringe el espacio de búsqueda) y Hill-Climbing (para encontrar la estructura de la red óptima en el espacio restringido).

• Restricted Maximization (rsmax2): Implementación más general del algoritmo Max-Min Hill-Climbing que puede usar cualquier combinación de algoritmos basados en restricciones y en optimización de métricas.

Algoritmos del paquete *corpcor* utilizados para implementar los operadores de aprendizaje:

- Shrinkage Estimator (cov.shrink): Retorna la matriz de covarianza de un conjunto de datos.

2.2.6 Operadores de muestreo

Los operadores de muestreo se encargan de generar una nueva población a partir de la población de individuos seleccionados de la generación anterior y la información resultante del aprendizaje realizado a dicha población. A continuación se muestra el diagrama de clases relacionado con los operadores de muestreo.

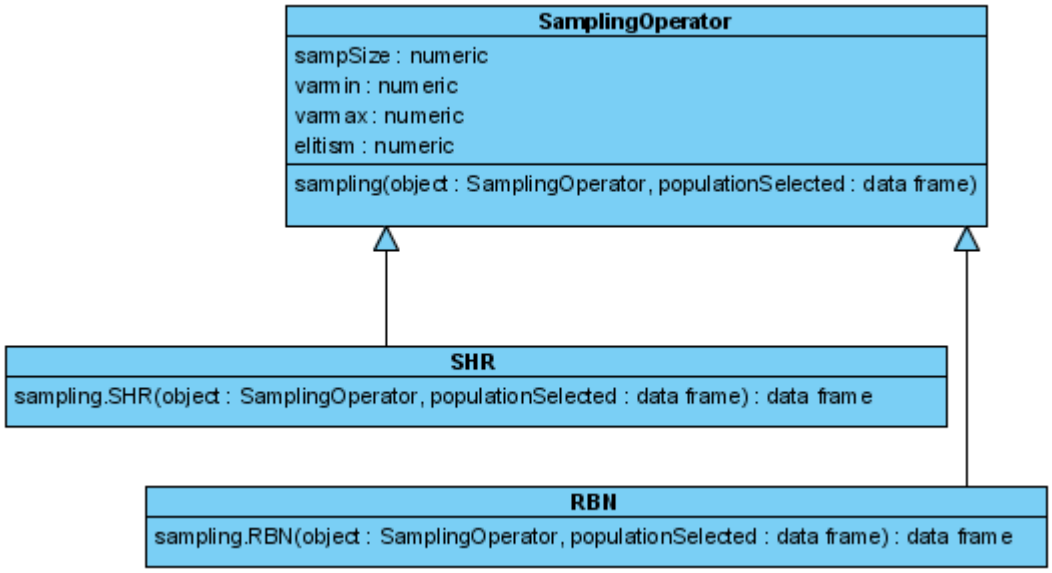

Figura 11. Diagrama de clases de los operadores de muestreo.

Como se observa en la figura 11 un operador de muestreo contiene el tamaño de la nueva población a generar (sampSize) y la cota inferior (varmin) y superior (varmax) de los valores que pueden tomar los individuos de la nueva población. El atributo elitismo (elitism) indica la cantidad de individuos de la población seleccionada que pasarán directamente a formar parte de la nueva población, si el valor del elitismo es igual a cero la nueva población será muestreada a partir de la población seleccionada, si el valor del elitismo es igual a 1, la nueva población estará compuesta por el mejor individuo de la población seleccionada anteriormente y la población resultante del muestreo de los restantes individuos.

La clase RBN implementa un operador para el muestreo de poblaciones con datos discretos y continuos, el método sampling.RBN retorna la nueva población, para ellos utiliza la función rbn del paquete bnlearn de R que genera datos aleatorios a partir de una Red Bayesiana y la información de los individuos a partir de los cuales fue aprendida la red.

La clase SHR implementa el operador para el muestreo de poblaciones con datos continuos que utilizan aprendizaje de la matriz de covarianza, el método sampling.SHR retorna una nueva población, para generar dicha población se auxilia de la función matrix() del paquete corpcor.

2.2.7 Condiciones de parada

A continuación se muestra el diagrama de clases relacionado con las condiciones de parada que implementa la biblioteca:

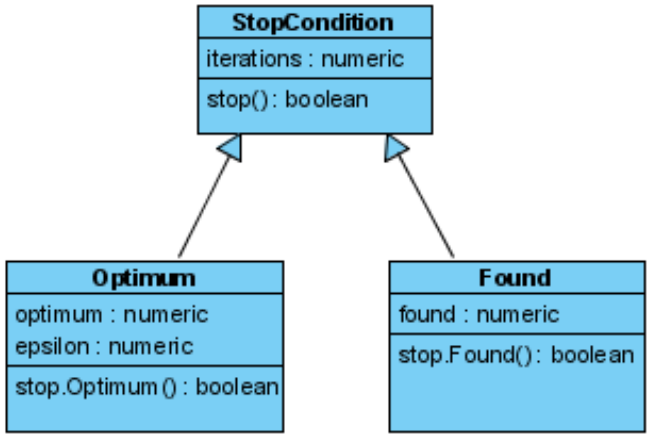

Figura 12. Diagrama de clases de las condiciones de parada.

La clase StopCondition contiene un atributo que representa la cantidad de generaciones que realizará el algoritmo (iterations), esta es la condición de parada que se ejecuta por defecto cuando no existe una condición de parada relacionada con algún parámetro específico de la población sobre la cuál se trabaja, de esta clase heredan las demás condiciones de parada que verifican criterios ampliamente utilizados en los algoritmos evolutivos.

Para el dominio continuo la biblioteca implementa la condición de parada Optimum, el método stop.Optimum de la clase Optimum verifica si el valor absoluto de la diferencia entre el valor de aptitud del mejor individuo de una generación y el óptimo de la función es menor que un valor epsilon determinado, si esta se cumple el método retorna verdadero.

Para el dominio discreto biblioteca implementa la condición de parada Found, para ello el método stop.Found de la clase Found verifica si el valor de aptitud del mejor individuo de una generación coincide con el óptimo de la función, si se cumple la condición el método retorna verdadero.

2.2.8 Algoritmos RCEDA.

En RCEDA un algoritmo EDA constituye un objeto de la clase del mismo nombre, la cual esta compuesta por los diferentes operadores que intervienen en el funcionamiento del algoritmo. De esta clase heredan las clases EDAContinuous y EDADiscrete, basándose en la clasificación de los EDA de acuerdo al tipo de dominio sobre el cual operan. En el diagrama de clases de la figura 13 se observa el comportamiento de las clases mencionadas anteriormente.

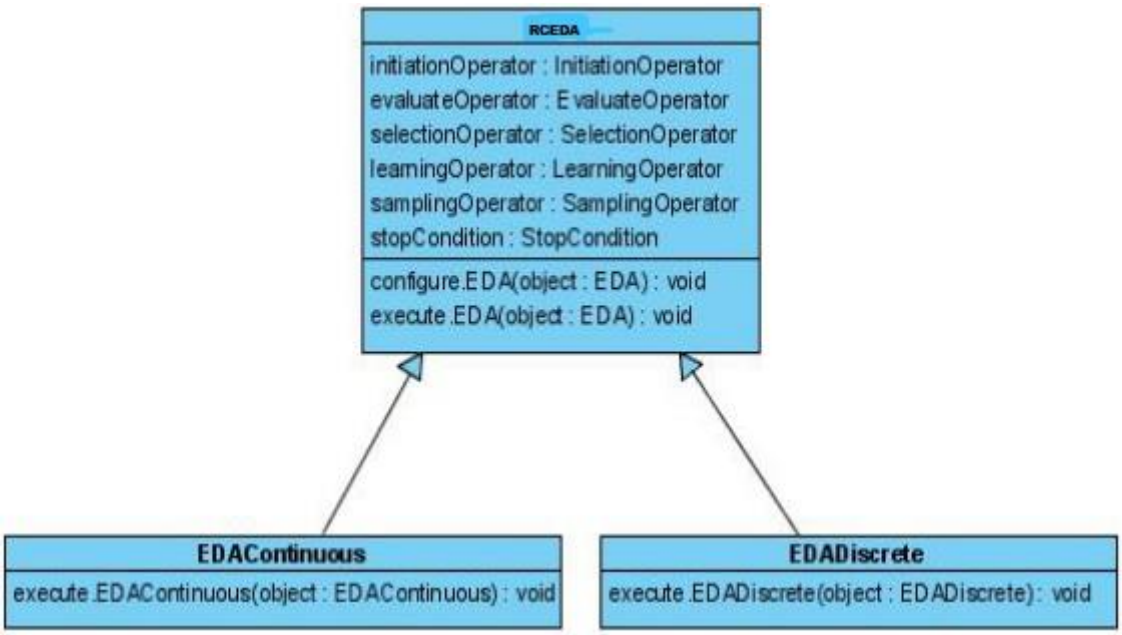

Figura 13. Diagrama de clases de los algoritmos RCEDA.

Para correr un algoritmo se crea un nuevo objeto de la clase EDA el cual es configurado a partir de la llamada al método configure.EDA, una vez ajustados los parámetros e inicializados los operadores que intervienen en la corrida del algoritmo se crea la población original y se evalúa en la función de aptitud, con la población creada y el objeto de clase EDA configurado se llama al método execute, a partir del nuevo objeto de la clase EDA que se pasa como parámetro este método determina cuál algoritmo se ejecutará, si el nuevo EDA es de tipo continuo se ejecutará el método execute.EDAContinuous, en caso de ser de tipo discreto se ejecuta el método execute.EDADiscrete. Ambos métodos operan de forma similar, siguiendo los pasos esenciales que presenta un algoritmo de este tipo, solo se diferencian en aspectos específicos relacionadas con el dominio de los datos con los cuales trabajan. En la figura 14 se observa el pseudo-código que muestra el comportamiento de estos dos métodos.

```
if no se cumple el número máximo de generaciones then
        if se cumple StopCondition then
        Terminar el algoritmo;
        end if
        Seleccionar M ≤ N individuos de acuerdo a SelectionOperator;
        Aprender la distribución de probabilidad de M según LearningOperator;
        Generar L ≤ N nuevos individuos de acuerdo a SamplingOperator;
        Evaluar los individuos generados a través de EvaluateOperator;
        if L < N  then
        Seleccionar de M los individuos que sobreviven;
        Integrar estos individuos a los L generados;
        end if
        Crear la nueva población;
        Imprimir los resultados de la nueva población;
        Incrementar el contador de generaciones;
else
Terminar el algoritmo;
end if
```

Figura 14. Pseudo-código de los algoritmos de RCEDA.

Como puede observarse estos métodos no tienen valor de retorno, en ellos solo se muestran a través de la consola los resultados de cada iteración realizada por el algoritmo, a medida que se ejecuta el algoritmo se muestra para cada iteración el mejor individuo de la población y su valor de aptitud para la función seleccionada, de esta manera se tiene una visión del funcionamiento del algoritmo y es posible detener su computación si el comportamiento del mismo no es el deseado, es decir, no se acerca al óptimo de la función. Un mal funcionamiento del algoritmo puede ser consecuencia de la configuración incorrecta de los operadores o de los parámetros concernientes a las poblaciones con las cuales se trabaja (tamaño de población, tamaño de los individuos, etc.). No obstante modificar estos parámetros no constituye dificultad alguna, a través de la consola es posible volver a ejecutar el algoritmo modificando cualquiera de las condiciones que previamente se habían definido. Esto hace posible utilizar la biblioteca de una manera práctica a la hora de realizar experimentos.

2.3 Conclusiones parciales

- En este capítulo se presentó el diseño de la biblioteca de clases para los RCEDA discretos.
- Se realizó un análisis detallado del funcionamiento de las diferentes clases y funciones de la biblioteca.
- Se presentaron los resultados experimentales que se obtuvieron de las corridas realizadas a los EDA implementados en la biblioteca.

Capítulo 3: Resultados experimentales

3.1 Funciones objetivo discretas.

Deceptive3

Las funciones deceptive de orden k están definidas como la suma de las más elementales funciones deceptive f_k de k variables:

$$f(x) = \sum_{j=1}^{l} f_k(s_j)$$

Donde s_j son cadenas de caracteres de x que contienen k elementos, en la biblioteca se implementó la función deceptive de orden 3(Deceptive3) que está definida de la siguiente forma:

$$f_{dec}^{3} = \begin{cases} 0.9 & \text{for } u = 0 \\ 0.8 & \text{for } u = 1 \\ 0.0 & \text{for } u = 2 \\ 1.0 & \text{for } u = 3 \end{cases}$$

El óptimo global se alcanza en el punto (1,1,…,1) donde $n = 3*l$.

FirstPolytree3

La función FirstPolytree3 (F3Poly en la biblioteca) es una función separable que se descompone aditivamente con bloques de longitud tres. En cada bloque se avalúa la función f_3^{Poly}. La función f_3^{Poly} tiene como propiedad que su distribución de Boltzmann con parámetro β=2 tiene una estructura de poliárbol con los siguientes arcos:

x1 $\rightarrow$ x2, y x3 $\rightarrow$ x2.

Luego de definida la función f_3^{Poly} podemos definir la función F3Poly como:

$$F_{FP3}(\vec{x}) = \sum_{i=1}^{l} f_3^{Poly}(x_{3*i-2}, x_{3*i-1}, x_{3*i})$$

El objetivo es maximizar F3Poly, teniendo el óptimo global localizado en el punto (0; 0; 1;…; 0; 0; 1) y la función toma el valor de $l*1.047$; donde $n = 3*l$.

FirstPolytree5

La función FirstPolytree5 (F5Poly en la biblioteca) es una función separable que se descompone aditivamente con bloques de longitud cinco. En cada bloque se avalúa la función f_5^{Poly}. La función f_5^{Poly} tiene como propiedad que su distribución de Boltzmann con parámetro $\beta = 2$ tiene una estructura de poliárbol con los siguientes arcos:

x1 $\rightarrow$ x3, x2 $\rightarrow$ x3, x3 $\rightarrow$ x5, y x4 $\rightarrow$ x5.

Luego de definida la función f_5^{Poly} podemos definir la función F5Poly como:

$$F_{FP5}(\overrightarrow{x}) = \sum_{i=1}^{l} f_5^{Poly}(x_{5*i-4}, x_{5*i-3}, x_{5*2}, x_{5*i-1}, x_{5*i})$$

El objetivo es maximizar F5Poly, teniendo el óptimo global localizado en el punto (0; 1; 0; 0; 1…; 0; 1; 0; 0; 1) y la función toma el valor de $l*1.723$; donde $n = 5*l.$

OneMax El OneMax el blanco discreto, es una función tiene (n + 1) los valores diferentes de aptitud que es polinomio distribuído dónde la meta es aumentar al máximo la función de OneMax.

$$\text{Onemax}(x) = \sum_{i}^{n} x$$

Donde n es que el número de variables y xi es el ith inconstante en el problema. El global óptimo se alcanza al punto (1,... ..1) Y su valor es n. Además, es una función que aditivamente descompuso donde el valor del global óptimo (1,... ..1) es igual al número de variables n.

3.2 EL ESTUDIO EXPERIMENTAL

Para comprobar el correcto funcionamiento de la biblioteca, se realizaron una serie de experimentos consistentes en realizar corridas a los diferentes algoritmos implementados en el dominio discreto, para ellos existe un conjunto de parámetros comunes, la población seleccionada siempre será el 30 % de la población global, el método de selección utilizado fue el truncamiento donde se seleccionan los mejores individuos de la población, en todos los experimentos se utilizó el elitismo igual a 1. Para todos los algoritmos se realizaron 30 corridas, el criterio de parada consiste en encontrar el óptimo o realizar un número fijo de generaciones, en el caso de las funciones discretas 20 .

neighborhoo	Obj.	n	τ	Generation	%Succes
One	F3Poly	30	0.3	7.39	29
L5				4	100
C13				5.29	93
C25				6.20	93
C41				9.94	76
C9				4.05	100
L9				4.86	99
One	F5Poly	30	0.3	28.4	13
L5				5.01	98
C13				6.46	90
C25				7.34	94
C41				9.42	79
C9				4.88	100
L9				5.00	100
One	IsoPeak	30	0.3	23.7	43
L5				3.85	100
C13				4.85	96
C25				6.30	93
C41				8.32	82
C9				3.96	100
L9				4.51	100

Tabla 1: Resultados Experimentales de F3Poly, F5Poly y funciones de IsoPeak

Tabla 1. muestra los resultados experimentales de ejecutar el algoritmo para los barrios diferentes, mientras teniendo en cuenta las funciones de objetivo: F3Poly, F5Poly e IsoPeak.

El algoritmo de EDA celular con L5, C9 y los barrios de L9 exigen a menos generaciones encontrar óptimo de éstos funcione que tenía el porcentaje mejor de éxito.

Como mostrado en Figura 15., los barrios que usan el EDA celular con las estructuras de aprendizaje locales y parámetros, aprendiendo el Bayesian Networks, el uno con el resultado mejor es el L5, C9 y L9 que realizan menos de las evaluaciones para cada número de variables. El barrio Uno es la peor eficacia, desde que para estas funciones tenía un número alto de evaluaciones, para encontrar el óptimo. Además, los barrios C13 y C25 tenían la eficacia del evaluative buena.

EDA celular que usa las Rejas diferentes

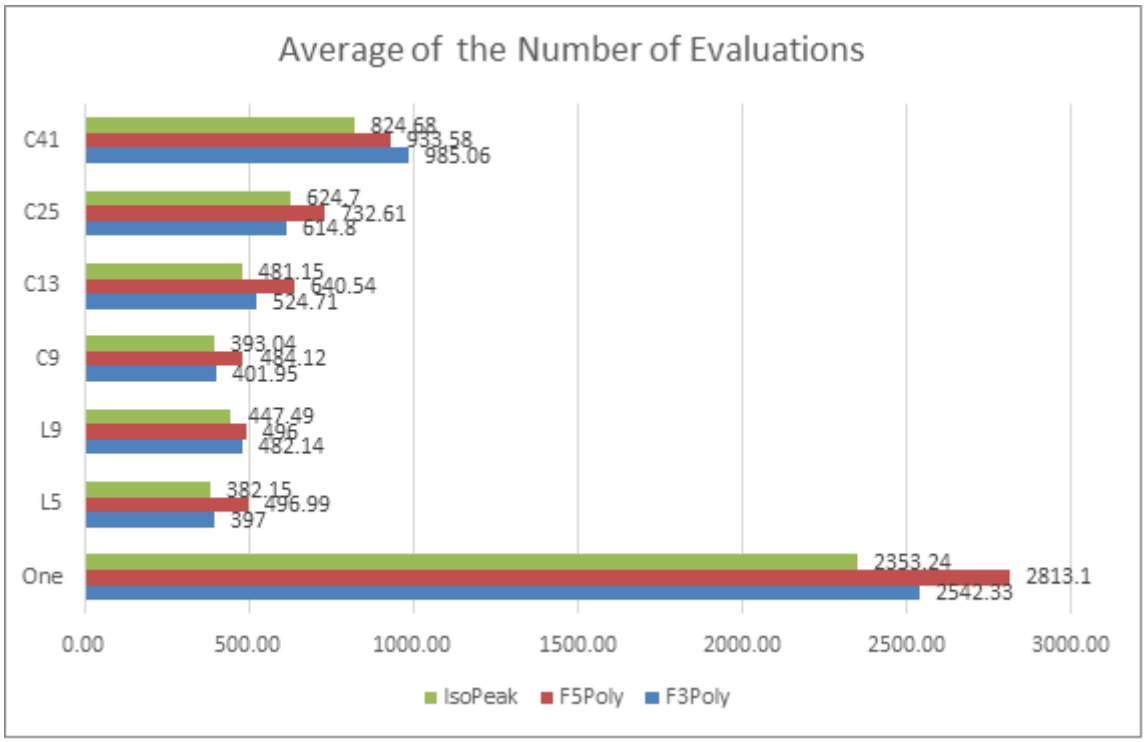

Figure 15: Resultados de número de evaluaciones para F3Poly, F5Poly y funciones de IsoPeak

neighborhoo	Obj.	n	Generation	%Succes
L5(5x5x2x2)	OneMax	100	7.71	95
L5(10x10x2x			2.08	100
L5(20x20x2x			2	100
L5(5x5x4x4)			9.04	77
L5(10x10x4x			3.12	96
L5(2x2x5x5)			17.2	67
L5(4x4x5x5)			9.27	77
L5(2x2x10x1			10.7	84
L5(4x4x10x1			5.83	89
L5(2x2x20x2			10.6	88

Tabla 2: Resultado Experimentalde las funciones OneMax para las diferentes rejas.

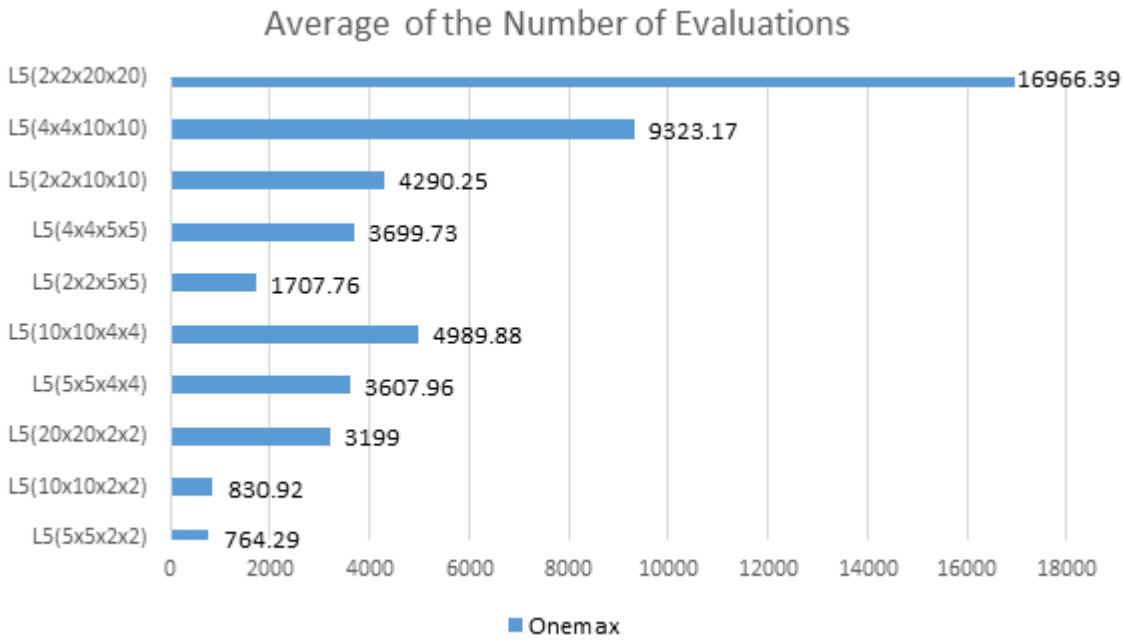

Figure 16: Resultados del número de evaluaciones para la función de Onemax

las evaluaciones para el barrio L5, es la configuración 10x10x2x2 y 5x5x2x2, y 2x2x20x20 eran las peores configuraciones, como se muestra en la Tabla 2

La comparación entre EDA celular y otro acercamiento de EDA

Estudiar la eficacia de este EDAs celular, varios acercamientos eran seleccionados, incluso PADA, SPADA y CUMDA. El algoritmo con Distribución de Factorización basado en Polytrees se publica con el nombre de PADA[10, 16] que se nombra como el Polytree Algoritmo de Distribución y Aproximación . PADA es un EDA que usa a los modelos simplemente conectados como modelo de sus distribuciones. Por otro lado, SPADA[12] se basaron en las pruebas de independencia, requiere las distribuciones de orden-superior de informática, como ocurre con las redes interconectadas. Sin embargo, como en el polytrees es sólo posible insertar los bordes y su costo es más bajo que las redes interconectadas. El algoritmo de la búsqueda local que usa SPADA es un procedimiento glotón que verifica para la existencia non-directed, en lugar de dirigido, ciclos. Tabla 3. muestra los resultados de los experimentos con estos acercamientos.

Estos resultados sugieren que EDA celular con los barrios L9, C9 y C41; y reja 20x20x2x2 tienen una actuación buena tenida a los acercamientos de EDA (EDA Simple, PADAp3, PADAp2, PADAt3, PADAt2, SPADA? =1, SPADA? =6), tomando el número de evaluación de cuenta y generaciones.

Otro estudio llevado a cabo con EDA celular y el Univariante celular de Distribución Marginal Algoritmo (CUMDA)[2], dónde la Tabla 3 muestra el resultado del EDA celular que usa el barrio diferente contra CUMDA, teniendo en cuenta Isopeak función para treinta variables.

Estos resultados sugieren que EDA celular con vecinos L9, C9, C13, C25 y L5; y rejilla

5x5x2x2 tuvo un mejor rendimiento que CUMDA con el vecindario C9 y la cuadrícula 10x10x2x2.

Algorithm	%Succes	NumEva	Generation
cEDA(L9-20x20x2x	100	639	4
cEDA(C9-20x20x2x	100	639	4
cEDA(C41-20x20x2	100	6412.99	4.01
Simple	100	10795.60	5.40
PADAp3(1000)[10,	77	18142.84	18.1
PADAp2(1000) [10,	72	17613.37	17.6
PADAt3(1000) [10,	98	10400.59	10.4
PADAt2(1000) [10,	96	10800.19	10.8
SPADA π=1(1000)	92	9071.92	9.08
SPADA π=6(1000)	96	10150.84	10.1

Tabla 3: Resultados experimentales de función Deceptive (n=30)

Algorithm	%Succes	Generation	NumEva
cEDA(L9-5x5x2x2)	100	4.51 2.33	447.49
cEDA(C9-5x5x2x2)	100	3.96 1.92	393.04
cEDA(C13-5x5x2x2)	96	4.85 5.31	481.15
cEDA(C25-5x5x2x2)	93	6.30 6.93	624.70
cEDA(L5-5x5x2x2)	100	3.86 1.20	3.86 1.20
CUMDA(C9-10x10-2x2	86.66	9.69 2.26	4085 908

Tabla 4: Resultados experimentales de función de Isopeak (el n=30)

Es más, los EDA celulares buenos son los cEDA con aprendizaje y estructura de población local (L5 y C9) con generación (3.86 1.20) y (3.96 1.92); y número de evaluación (382 118.88) y (393.04 190.31) respectivamente, como se muestra en la Tabla 4

Función IsoPeak:

Es una función no separable, y está compuesta por las funciones bivariadas Iso1 e Iso2. La solución para esta función es un vector n-dimensional tal que n = 2 × m (las variables se dividen en grupos de dos). Primero, las funciones auxiliares Iso1 e Iso2 se definen de la siguiente manera:

	00	01	10	11
Iso1	m	0	0	m-1
Iso2	0	0	0	m

El objetivo es maximizar la función IsoPeak y el óptimo global se encuentra en el punto (1, 1, 0, 0, ..., 0, 0).

Algoritmo propuesto.

En este algoritmo, el ciclo reproductivo sirve dentro de cada número de personas locales, que generalmente se llama célula, tienen sus propias poblaciones locales definidas por subpoblaciones vecinas, mientras que una célula pertenece a muchas poblaciones locales. El conjunto de todas las celdas define una partición de la población global.

Resultados experimentales

Para probar el algoritmo propuesto, consiste en llevar a cabo una serie de experimentos y ejecuciones en diferentes vecindarios, de modo que se eligió un conjunto de parámetros comunes. La población objetivo siempre será el 30% de la población mundial. Para todos los algoritmos se realizaron 100 ejecuciones. El criterio de detención fue encontrar el óptimo o realizar 30 iteraciones.

Algorithm	%Success	Evaluation numbers	Generations
cEDA(L9-20x20x2x2)	100	6397	4
cEDA(C9-20x20x2x2)	100	6397	4
cEDA(C41-20x20x2x2)	100	6412.99	4.01
Simple EDA(2000)[7]	100	10795.60	5.40
PADAp3(1000) [4, 5]	77	18142.84	18.16
PADAp2(1000) [4, 5]	72	17613.37	17.63
PADAt3(1000) [4, 5]	98	10400.59	10.41
PADAt2(1000) [4, 5]	96	10800.19	10.81
SPADA π=1(1000) [6]	92	9071.92	9.08
SPADA π=6(1000) [6]	96	10150.84	10.16

Tabla 5: Resultados experimentales de función de Isopeak (el n=30)

Estos resultados sugieren que EDA celular con vecindarios L9, C9 y C41; y la cuadrícula 20x20x2x2 tuvo un mejor rendimiento que los enfoques EDA (EDA simple, PADAp3, PADAp2, PADAt3, PADAt2, SPADA π = 1, SPADA π = 6), teniendo en cuenta el número de evaluación y las generaciones.

3.3 Conclusiones Parciales

- Se implementaron en lenguaje R los diferentes operadores que integran la biblioteca y posibilitan el funcionamiento de los EDA Celulares Discretos.
- Se implementaron funciones objetivo para dominio discreto.
- Utilizando las funciones y operadores implementados se crearon EDA con los cuales se realizaron experimentos para probar el funcionamiento de la biblioteca.

Conclusiones

1. Se creó una biblioteca de clases en R auxiliándose de las funcionalidades estadísticas que brinda este lenguaje para el manejo de los datos, en este caso, se utilizaron los paquetes importados “bnlearn” y “corpcor” para implementar las principales funciones de RCEDA.
2. A partir de los operadores y funciones objetivo implementados se crearon EDA celulares que trabajan sobre dominio discreto para comprobar el correcto funcionamiento de la biblioteca.
3. Se realizaron experimentos para el dominio discreto, a partir de estos fue posible validar los algoritmos implementados mediante la comparación con los del estado del arte, con los cuales mostraron resultados similares.

Recomendaciones

- Aplicar estas funciones a problemas prácticos de la vida.
- Probar las funciones con otras vecindades.
- Probar las funciones con otras estructuras celulares.

Referencias bibliográficas.

Aviles, Y. y Mahdi, G. (2008). *Biblioteca para el desarrollo de algoritmos evolutivos que estiman distribuciones*. Trabajo de grado, Ingeniería en Informática, Universidad de Camagüey, Camagüey, Cuba.

Baluja,S. (1994). *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. [s.l]: Carnegie Mellon University.

Baluja, S. y Caruana, R. (1995). *Removing the genetics from standard genetic algorithm*. Ponencia presentada en The International Conference on Machine Learning.

Bilmes, L. (2000). *Dynamic Bayesian Multinets*. Ponencia presentada en The 16th conference on Uncertainty in Artificial Intelligence.

Bonet, J. S. de, Isbell, C. L. y Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, (9), 13- 16.

Colectivo de autores. (2000). *Introducción a R. Notas sobre R: Un entorno de programación para Análisis de Datos y Gráficos*. [s.l]: [s.n]

Díaz-Uriarte, R. (2003). *Introducción al uso y programación del sistema estadístico R*. [s.l]: Centro Nacional de Investigaciones Oncológicas (CNIO).

Duarte, L. A. (2003*). Biblioteca para el desarrollo de algoritmos evolutivos que estiman distribuciones.*Trabajo de grado, Ingeniería en Informática, Universidad Central "Marta Abreu" de Las Villas, Santa Clara.

Echegoyen, C., Mendiburu, A., Santana, R. y Lozano, J. A. (2012). *Clases de Equivalencia en Algoritmos de Estimación de Distribuciones*. San Sebastián: Universidad del País Vasco.

Eshelman, L. J. y Schäffer, J. D. (1993). Productive recombination and propagating and preserving schemata. *Foundations of Genetic Algorithms*, (3), 10-14.

Etxeberria, R. y Larrañaga, P. (1999). *Global optimization with Bayesian networks*. Ponencia presentada en the II Symposium on Artificial Intelligence.CIMAF99. Special Session on Distributions and Evolutionary Optimization.

Grahl, J. y Rothlauf, F. (2004). *PolyEDA: Combining Estimation of Distribution Algorithms and linear inequality constraints.* Ponencia presentada en the Genetic and Evolutionary Computation Conference (GECCO-2004).

Harik, G., Lobo, F. G. y Golberg, D. E. (1998). *The compact genetic algorithm*. Ponencia presentada en The IEEE Conference on Evolutionary Computation.

Hermida, R. S. (2004). *Modelación probabilística basada en modelos gráficos no dirigidos en Algoritmos Evolutivos con Estimación de Distribuciones*. [s.l]: [s.n]

Ihaka, R. y Gentleman, R. (1996). R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, (3), 5-10.

Karshenas, H., Santana, R., Bielzaa, C. y Larrañaga, P. (2012). Regularized continuous estimation of distribution algorithms. *Applied Soft Computing*, *13*, 2412–2432.

Kontos, K. (2009). *Gaussian Graphical Model Selection for Gene Regulatory Network Reverse Engineering and Function Prediction*., Bruselas: Universidad Libre de Bruselas.

Larrañaga, P., Etxeberria, R., Lozano, J. A. y Peña, J. M. (1999). *Optimization by learning and simulation of Bayesian and Gaussian networks*. (No. EHU-KZAA-IK-4/99). País Vasco: University of the Basque Country.

Larrañaga, P., Etxeberria, R., Lozano, J. A. y Peña, J. M. (2000). *Combinatorial optimization by learning and simulation of Bayesian networks*. Ponencia presentada en The Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford.

Larrañaga, P. y Lozano, J. A. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. [s.l]: [s.n].

Larrañaga, P., Lozano, J. A. y Mühlenbein, H. (2003). Estimation of Distribution Algorithms Applied To Combinatorial Optimization Problems. *Revista Iberoamericana de Inteligencia Artificial*, *19*, 149-168.

Larrañaga, P. y Puerta, J. M. (2003). *Algoritmos de Estimación de Distribuciones*. [s.l]: [s.n].

Luke, S. (2010). *Essentials of Metaheuristics* (No. 978-0-557-14859-2). [s.l]: George Mason University.

Madera, J. (2008). *Algoritmos Evolutivos con Estimación de Distribuciones basados en Pruebas de Independencia*. Tesis de maestría no publicada, Universidad de Camagüey "Ignacio Agramonte Loynaz", Camagüey, Cuba.

Margaritis, D. (2003). *Learning Bayesian Network Model Structure from Data*. [s.l]: Carnegie-Mellon University.

Mühlenbein, H. (1998). The Equation for Response to Selection and its Use for Prediction. *Evolutionary Computation*, (5), 303–346.

Mühlenbein, H. y Mahnig, T. (1999). Convergence Theory and Applications of the Factorized Distribution Algorithm. *Journal of Computing and Information Technology*, (7), 19-32.

Mühlenbein, H. y Paaß, G. (1996). From Recombination of Genes to the Estimation of Distributions. *Computer Science 1411: Parallel Problem Solving from Nature – PPSN*, *IV*, 178–187.

Mühlenbein, H. y Voigt, H. M. (1996). Gene pool recombination in genetic algorithms. *Metaheuristics: Theory and applications*, (6), 53-62.

Ochoa, A. (2010). *Opportunities for Expensive Optimization with Estimation of Distribution Algorithms* [Electronic Version]. [s.l]: [s.n]

Ochoa, A., Soto, M., Santana, R., Madera, J. y Jorge, N. (1999). *The Factorized Distribution Algorithm and the junction tree: A learning perspective*. Ponencia presentada en the Second Symposium on Artificial Intelligence. Adaptive Systems.CIMAF 99, La Habana, Cuba.

Paradis, E. (2003). *R para Principiantes*. [s.l]: [s.n]

Pelikan, M., Goldberg, D. E. y Cantú-Paz, E. (1999). BOA*: The Bayesian optimization algorithm.* Ponencia presentada en The Genetic and Evolutionary Computation Conference GECCO-99, San Francisco.

Pelikan, M., Goldberg, D. E. y Cantú-Paz, E. (2000). *Bayesian optimization algorithm, population sizing, and time to convergence*. Ponencia presentada en The Genetic and Evolutionary Computation Conference.

Pelikan, M., Hauschild, M. W. y Lobo, F. G. (2012). *Introduction to Estimation of Distribution Algorithms* (No. MEDAL Report No. 2012003). St. Louis: University of Missouri.

Santana, R. (2004). *Modelación probabilística basada en modelos gráficos no dirigidos en Algoritmos Evolutivos con Estimación de Distribuciones.*[s.l]: [s.n]

Santana, R. (2006). *Advances in Probabilistic Graphical Models for Optimization and Learning Applications in Protein Modeling*. Saint Sebastian: Department of Computer Science and Artificial Intelligence of the University of the Basque Country.

Schäfer, J., Opgen-Rhein, R., Zuber, V., Ahdesmäki, M., Silva, P. D. y Strimmer, K. (2012). *Efficient Estimation of Covariance and (Partial) Correlation.* [s.l]: [s.n]

Schäfer, J. S. (2005). *Small-Sample Analysis and Inference of Networked Dependency Structures from Complex Genomic Data*. Alemania: Universität München.

Scutari, M. (2011). *Bayesian network structure learning, parameter learning and inference.*[s.l]: [s.n]

Soto, M., Ochoa, A., Acid, S. y Campos, L. M. (1999). *Introducing the polytree aproximation of distribution algorithm.* Ponencia presentada en The Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99, La Habana.

Syswerda, G. (1993). Simulated crossover in genetic algorithms. *Foundations of Genetic Algorithms*, (2), 239–255.

Tsamardinos, I., Aliferis, C. y Statnikov, A. (2003). *"Algorithms for Large Scale Markov Blanket Discovery"*. Ponencia presentada en The Sixteenth International Florida Artificial Intelligence Research Society Conference.

Yaramakala, S. y Margaritis, D. (2005). *"Speculative Markov Blanket Discovery for Optimal Feature Selection"*. Ponencia presentada en The Fifth IEEE International Conference on Data Mining.