UNIVERSIDAD DE CAMAGÜEY
FACULTAD DE INFORMÁTICA

Trabajo de diploma para optar por el Título de Ingeniero Informático

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

Autora:

Janet Richardson Ibañez

Tutor:

MSc. Yoan Martínez López

Camagüey, julio de 2017.

"Año 59 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 mamá Teresa, que aunque el destino me robó poder compartir con ella este maravilloso momento de mi vida; hizo por mí hasta lo imposible, por su sacrificio, compañía y comprensión en todo momento. Por regalarme el ejemplo de su vida, por cada regaño, por cada consejo, por todos los lindos momentos que pasamos juntas y sobre todo por haber sido la mejor madre que ha existido en el mundo.

A mi papá Alberto, que es una de las personas que más amo, por estar siempre a mi lado, por todo su amor, cariño, caricias, consejos y por estar pendiente de mí siempre.

A mi hermano Yonathan, para que esté orgulloso de mí, que es lo que más quiero en este mundo, por confiar en mí y amarme tanto como yo lo amo a él.

A mi esposo Oelvy, por compartir conmigo todos estos momentos durante este año, que ha llenado mi corazón y me ha dado tanta tranquilidad y alegría, por hacer que siempre me sienta protegida, por todo lo que he aprendido de él, por cada noche que se mantuvo despierto ayudándome con esta tesis, por llenarme la vida de sueños y hacer que lo ame tanto.

A mi madrastra Lianis, por recibirme y atenderme tan bien en su casa, por quererme como si fuera su hija, por haber acompañado a mi papá todos estos años.

A mi familia, por inspirarme a seguir adelante y cumplir con todas mis metas, a ellos por la confianza depositada en mí y por estar siempre a mi lado, brindándome el amor y la compresión necesaria para hacer realidad mis sueños.

A la familia de mi esposo y en especial a mi suegra Niurka y a mi abuela suegra Margarita por la preocupación y el apoyo brindado durante esta etapa.

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.

A todos los profes que tuve en el transcurso de mi vida escolar; por todas sus enseñanzas y dedicación.

A mis compañeros de estos 6 años aunque algunos ya no estén en el aula en especial a Raiza, Aimée, Idalni, Yari y Joha por todo lo que estudiamos, vivimos y compartimos juntos.

A todas mis compañeras de trabajo tanto de la Escuela Grandes Alamedas como del Pol. Finlay.

A todas las personas que quiero y que con tanto amor guardo en mi corazón, por todo lo que me han enseñado, por su apoyo incondicional y sobre todo por creer en mí.

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 2017.

Autor _____ Tutor _____

Janet Richardson Ibañez MS.c Yoan Martínez López

Resumen

Un tipo de Algoritmo de Estimación de Distribución (EDA) es propuesto: EDA de tipo celular en la resolución de problemas de optimización continuos. Esta nueva clase de algoritmo de optimización estocástico basado en poblaciones se obtiene a partir del aprendizaje basado en pruebas de independencias y esquemas descentralizados de poblaciones locales. Los resultados experimentales mostraron que esta propuesta reduce número de evaluaciones en la búsqueda del óptimo para la función Ackley, manteniendo su eficacia, en comparación a otros enfoques de la literatura.

Palabras claves: EDA celular, redes Gaussianas, aprendizaje, Modelo Gráfico Probabilístico

Abstract

One type of Distributed Estimation algorithm (EDA) is proposed: cellular-type EDA in solving continuous optimization problems. This new class of population-based stochastic optimization algorithm is derived from evidence-based learning of independence and decentralized schemes of local populations. The experimental results showed that this proposal reduces the number of evaluations in the search for optimum for the Ackley function, maintaining its effectiveness, in comparison to other approaches in the literature.

Keywords: cellular EDA, Gaussian networks, learning, Probabilistic Graph Model

Índice

Introducción

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 EDA (Estimation Distribution Algorithms) son un grupo de algoritmos de EA que permiten ajustar el modelo a la estructura de un problema determinado, realizados por una estimación de distribuciones de probabilidades a partir de soluciones seleccionadas (Larrañaga, 2003). El modelo reflejado por el sesgo es una distribución de probabilidades. Estos algoritmos son metaheurísticas que se basan en sustituir los operadores de cruce y mutación de los 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. Los algoritmos evolutivos celulares son un tipo de algoritmos evolutivos de grupos discretos basados en estructuras espaciales, donde cada individuo interactúa con su vecino adyacente. Una vecindad solapada ayuda en la exploración del espacio de búsqueda, mientras que la explotación toma lugar dentro de una vecindad por operadores estocásticos. Los algoritmos genéticos celulares son propuestos por Dorronsoro (2008), los cuales manejan sus vecinos de acorde a la calidad

de los individuos de una población un conjunto más razonable de estructura de la población basada en la velocidad de convergencia, entonces los parámetros de configuración para la población y las estructuras vecinas son menos necesarias. Otros investigadores (Alba, 2000,2008; Lu, 2013; Huang, 2015), propusieron una variante de algoritmos evolutivos que se basaban en la mejor capacidad de mantenimiento de la diversidad de mantenimiento de la población relativa al algoritmo genético canónico.

Los algoritmos de evaluación diferencial celular son algoritmos propuestos por Storn y Price, son poblaciones basadas en algoritmos de optimización iterativos paralelos que superan a muchos métodos en términos de velocidad de convergencia y robustez sobre funciones de puntos de referencias comunes y en problemas reales, los cuales son influenciados por los siguientes criterios: un factor de escala F, un tamaño de población NP y una razón de recombinación (Mühlenbein, 2003 y Ding, 2015).

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.

La organización de los EDAs celulares se basan en la tradicional estructura 2D de vecinos solapados, se conocen mejor en términos de dos rejillas, esto quiere decir que una rejilla contiene cadenas y otra contiene conjuntos disjuntos de cadenas (células), ver figura 1.

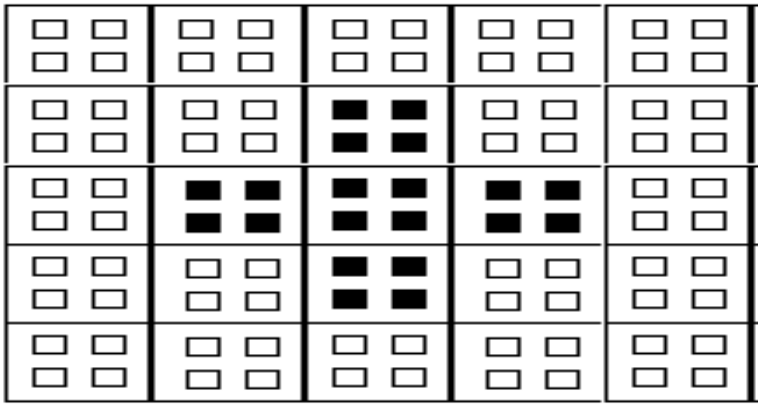


Figura 1. Representación de un EDA celular

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 EDAs Continuos utilizan técnicas estadísticas las cuales en su mayoría no están implementadas en lenguajes de alto nivel, solo si se trata de bibliotecas especializadas.

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 Continuos 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 Continuos 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 Continuos 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 Continuos, 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 Continuos, 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 Continuos en R.
4. Validación mediante experimentos de los EDAs Celulares Continuos implementados y comparación con otros del estado del arte.

En el capítulo 1 se analiza el marco teórico de los EDAs Celulares Continuos, a su vez se definen los principales modelos gráficos probabilísticos que intervienen en la detección de dependencias y las clasificaciones de los EDAs Celulares Continuos 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, asimismo 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 la función de prueba Ackley con diferentes vecindades.

Capítulo 1. Algoritmo de Estimación de Distribución Celular

En un EDA celular el ciclo reproductivo se ejecuta dentro de cada número de individuos de la población local, el cual es 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 detectar las dependencias entre las variables del problema 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.

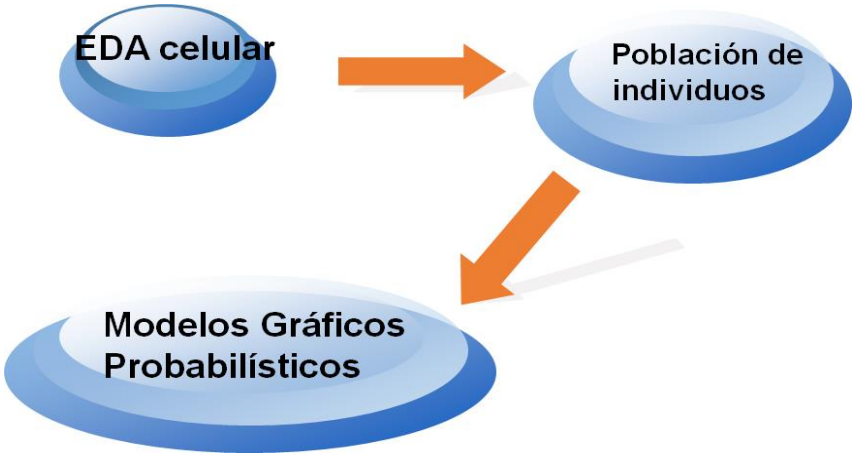


Figura 2. Representación ciclo reproductivo de EDA Celular

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 (pp. 4-9).

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 probabilísticas híbridos, adaptados para problemas con variables discretas y continuas.

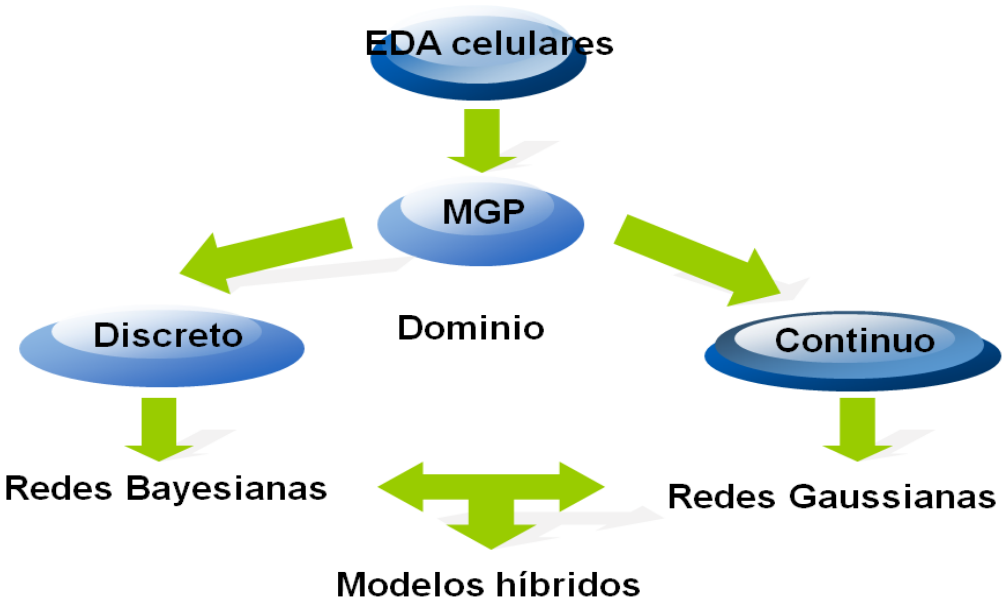


Figura 3. Representación Modelos Gráficos Probabilísticos

Redes Gaussianas.

Schäfer (2005) describe a las redes Gaussianas como:

> Modelos Gráficos de interacción para la distribución normal multivariada, estos modelos en lo adelante GGM son similares a las redes Bayesianas en el hecho de que su concepto subyacente es la independencia condicional, sin embargo a diferencia de las Redes Bayesianas solo contienen arcos no dirigidos, esto convierte al GGM en uno de los modelos más simples conceptualmente pero a su vez en uno de los más aplicados.(p.-4)

En esta aproximación que constituyen los GGM, los datos X se asumen como mutuamente independientes y con una distribución p-variada normal $N_p(\mu,\Sigma)$ con un vector de media $\mu=(\mu_1,...,\mu_p)^T$ y la matriz positiva definida de varianza-covarianza $\Sigma=(\sigma_{ij})$, donde $1\le i,j\le p$. A través de la fórmula $\sigma_{ij}=\rho_{ij}\sigma_i\sigma_j$ la matriz de covarianza puede descomponerse en componentes de varianza σ_i^2, con $i=1,...,p$, y la correlación de Pearson's $P=(\rho_{ij})$.

La densidad normal multivariada está dada por:

$$f(x) = (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp\left\{ -(x-\mu)^T \Sigma^{-1} (x-\mu)/2 \right\} \quad (1)$$

En términos exponenciales, la parametrización alternativa se da a través de parámetros canónicos que se definen como $\Omega = \Sigma^{-1}$ y $\beta = \Sigma^{-1} \mu$, así la densidad normal multivariada de la ecuación anterior puede redefinirse como:

$$f(x) = \exp\left\{ \alpha + \beta^T x - x^T \Omega x / 2 \right\} = \exp\left\{ \alpha + \sum_{i=1}^{p} \sum_{j=1}^{p} w_{ij} x_i x_j / 2 \right\} \quad (2)$$

Donde α es la constante de normalización y $\Omega = (w_{ij})$ es llamada matriz de precisión o concentración.

Aunque los GGM tienen mucho menos demanda computacional que las Redes Bayesianas su uso se ha extendido a muchas disciplinas científicas, esto conlleva a la búsqueda de nuevas alternativas que posibiliten un uso más eficiente de los mismos.

Regularización de parámetros probabilísticos.

Karshenas et al. (2012) describe que:

> Las técnicas de regularización son frecuentemente utilizadas en el aprendizaje estadístico para obtener una estimación más robusta de los modelos probabilísticos con un error de predicción pequeño. El modelo de estimación regularizado intenta disminuir el error de predicción general del modelo de estimación mediante la reducción de la gran varianza causada por la predicción de muestras nuevas, al costo de introducir un pequeño margen de error en el modelo.
>
> El modelo de estimación en los EDA presenta algunas características que motivan el uso de las técnicas de regularización. La falta de estadísticas adecuadas puede conllevar a que el modelo se haga muy parcial en regiones específicas del espacio de búsqueda,

esto reduce su capacidad de generalizar la cual es un factor importante cuando se muestrea el modelo. El uso de la regularización puede reducir el error general del modelo estimado en los EDA. Otro aspecto importante es la escalabilidad de los EDA con respecto al tamaño del problema ya que estimar el modelo de la distribución de probabilidad de grandes espacios de búsqueda requiere poblaciones de gran tamaño, dado que la estimación del modelo y el posterior muestreo en los EDA son grandes consumidores de tiempo, el rendimiento del algoritmo puede bajar de forma precipitada si el tamaño de la población es muy grande. Estimar un modelo de calidad comparable usando poblaciones mucho más pequeñas es un requerimiento importante en estos algoritmos. (pp. 2412–2432)

Ochoa (2010) propone:

La discusión de la técnica de estimación con contracción o restricción, esta brinda a los EDA la capacidad de construir mejores modelos de las distribuciones de búsqueda sobre poblaciones pequeñas. El mérito de esta técnica de contracción radica en que esta mejora la eficiencia y la precisión de la estimación y proporciona una matriz de covarianza bien condicionada y definida positivamente, lo cual constituye un aspecto importante para calcular su inversa.

La idea de la estimación con contracción es simple, suponiendo que se tiene un modelo no restringido de grandes dimensiones y un submodelo reducido con dimensiones restringidas, mediante el ajuste de cada uno de los dos modelos diferentes a los datos observados correspondientes se obtienen estimaciones. La estimación no restringida exhibirá una variación relativamente alta debido a la mayor cantidad de parámetros que necesitan ser ajustados, mientras que su homólogo de pocas dimensiones tendrá menor varianza, pero potencialmente también será considerable como un estimador del modelo no restringido verdadero (pp. 115-120).

Estrategia de aprendizaje

Un asunto crítico en un EDA celular es 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 unas de las alternativas para dar solución este problema. El operador dede 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 espacial dada de la rejilla” (Dorronsoro, 2008, p.21).

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 4.

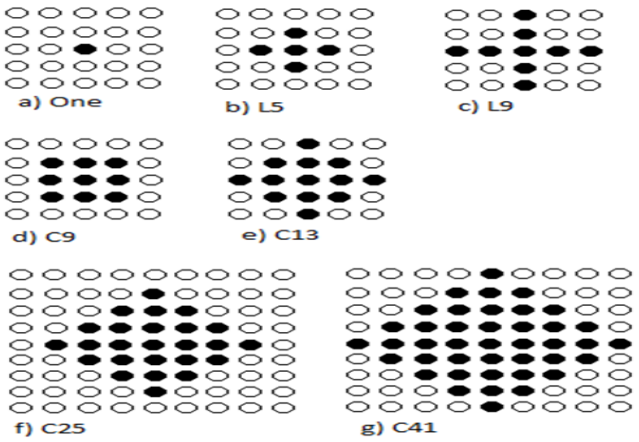


Figura 4. Representación de diferentes vecindades

1.1EDAs continuos

El UMDAc (Univariate Marginal Distribution Algorithm for Continuous domain) fue propuesto por Larrañaga et al.(1999), este algoritmo asume que:

En cada generación las variables son independientes siguiendo una distribución normal (pueden tenerse en cuenta otras distribuciones), el UMDAc estima dos parámetros fundamentales para las variables en cada generación t, ellos son: la media $\mu_i(t)$ y la desviación estándar $\sigma_i(t)$. El PBILc (Population Based Incremental Learning for Continuous domain) sigue un esquema similar al UMDAc. (pp. 225-227)

El algoritmo MIMICc (Mutual Information Maximization for Input Clustering for Continuous domain) constituye una extensión del algoritmo MIMIC, es su caso la diferencia radica en que las variables toman valores reales. La idea es similar a la que sigue el algoritmo discreto, fijando el modelo a los datos empíricos y analizando solo las relaciones que existen entre pares de variables de la misma forma que UMDA. (pp. 228-230).

El algoritmo EMNA (Estimation of Multivariate Normal Algorithm) propuesto por Larrañaga y Lozano (2003) utiliza:

Una función de densidad normal multivariada para aprender la factorización de los individuos seleccionados. Alternativas a este algoritmo (conocido $EMNA_{global}$) son el EMNAa y el EMNAi, ambos generan un solo individuo. El primero es adaptativo, incorpora el individuo si es mejor que el peor de la población. El segundo es incremental, adiciona el individuo a la población.

El algoritmo EGNA (Estimation of Gaussian Network Algorithm) utiliza aprendizaje y simulación de redes Gaussianas, una de las variantes propuestas es el EGNABGe, en esta variante la inducción del modelo se realiza mediante un método de puntuación que utiliza una medida bayesiana. El algoritmo EGNABIC sigue un esquema similar pero utiliza la métrica BIC para dominio continuo y el EGNAEE (Algoritmo con Estimación de Redes Gaussianas con Exclusión de Arcos) utiliza detección de independencias para construir la red Gaussiana (pp. 348-352)

Otra propuesta para dominio continuo es el algoritmo Poly EDA propuesto Grahl y Rothlauf (2004), este no es más que “la combinación de algoritmos de estimación de distribuciones y restricciones con desigualdades lineales”. Este algoritmo fue propuesto para la solución de problemas con restricciones.

1.2 Bibliotecas de EDAs Celulares Continuos.

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 implementó el método de estimación de distribuciones: 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 UMDAc para dominio continuo), 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(Ihaka y 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.

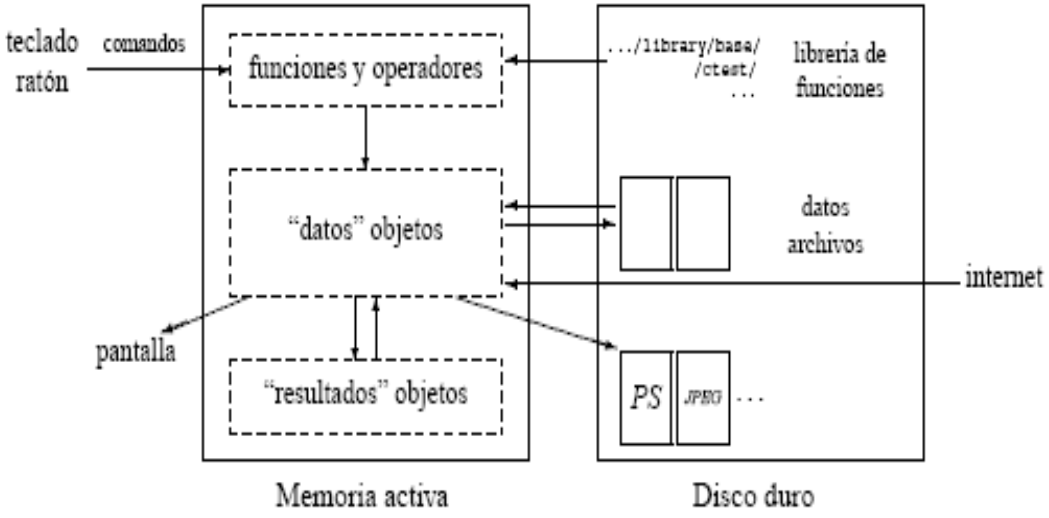


Figura 1.3.1 Funcionamiento de R.

Como se observa en la figura 1.3.1 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. (p.85)

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, asimismo R permite utilizar la herencia entre clases.

Cada objeto en R pertenece a una clase que determina de que 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 Continuos como algoritmos que se basan en métodos probabilísticas 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 Continuos 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 Continuos. 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 (constraint based learning), 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. (p. 96)

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. (p. 98)

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 de

clases, 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 Continuos 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 uso de Modelos Gráficos Probabilísticas para estimar la distribución de probabilidad ha dado paso al surgimiento de una gran variedad de este tipo de algoritmos evolutivos.
- 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 Continuos 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.

2.1Diseño de la biblioteca.

Para la implementación de la Biblioteca se partió de una ya existente implementada en otra tesis (Pino, 2013), solo se reutilizó lo que se necesitó para los Problemas de Optimización Continuos. La biblioteca de clases en R, de ahora en adelante será RCEDA, que provee un marco común para el desarrollo de algoritmos EDAs Celulares Continuos.

RCEDA está integrada por dos componentes fundamentales: clases y funciones genéricas. Un objeto de la clase EDAs Celulares Continuos 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.

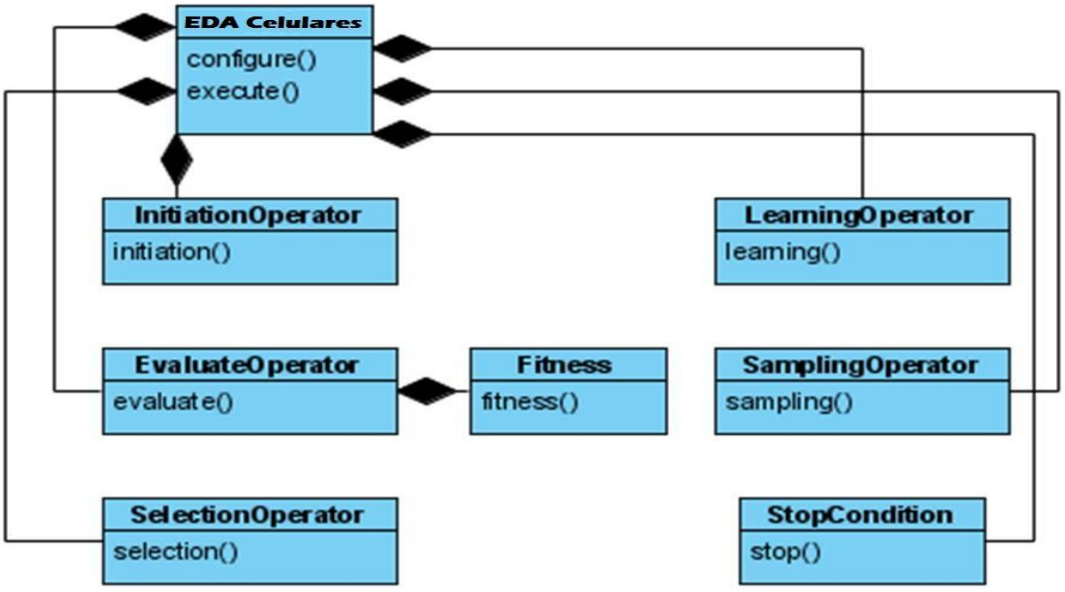


Figura 2.1. Diagrama de clases de la biblioteca RCEDA.

Para la biblioteca se logró una estructura que resulte fácil de usar y que permita incorporar nuevas funcionalidades, los métodos que utiliza RCEDA 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 cual no representa una alteración de las funcionalidades implementadas con anterioridad.

2.1.1 Operadores de iniciación.

Los operadores de iniciación crean e inicializan una matriz de población a partir de los parámetros suministrados y en dependencia del tipo de dominio (continuo).

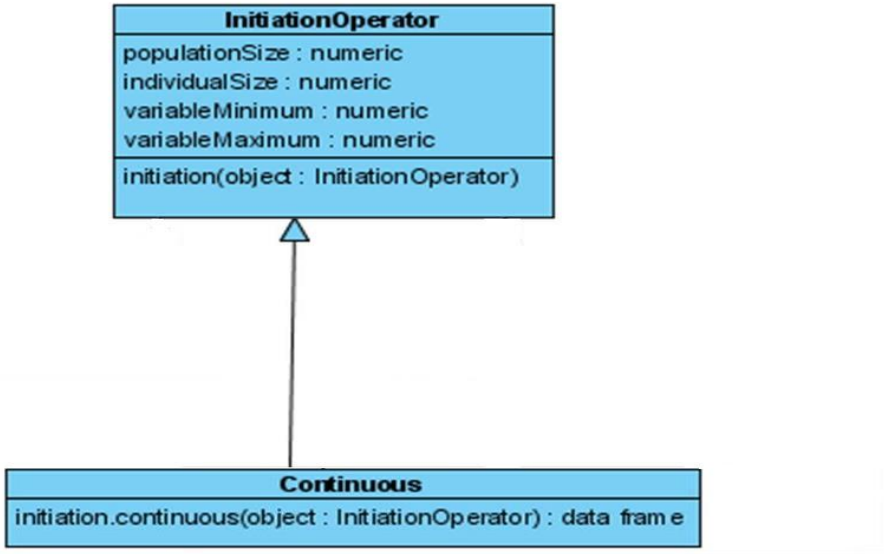


Figura 2.1.1. Diagrama de clases de los operadores de iniciación.

Como se observa en la figura 2.1.1 un operador de iniciación contiene el tamaño de la población a generar (populationSize) 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 clase Continuous hereda de la clase InitiationOperator, estas solo se diferencian en el dominio sobre el cual operan, para ello se definen los métodos que

especifican de qué forma se creará la nueva población, el método initiation.continuous crea 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.

2.1.2 Operadores de aprendizaje

La clase Learning Operator 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 (Margaritis, 2003) usado en un algoritmo de aprendizaje estructurado.(pp. 245-247)

• Incremental Association (iamb): Basado en el algoritmo de detección de redes de Markov del mismo nombre (Tsamardinos, Aliferis et al., 2003), el cual se basa en un esquema de selección de dos fases: selección hacia delante seguida por un intento de remover los falsos positivos.(pp. 247-248)

• Fast Incremental Association (fast.iamb): Variante del algoritmo IAMB que reduce el número de pruebas de independencias condicionales (Yaramakala and Margaritis, 2005).(pp.248-249)

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

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

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

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.

2.1.3 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.

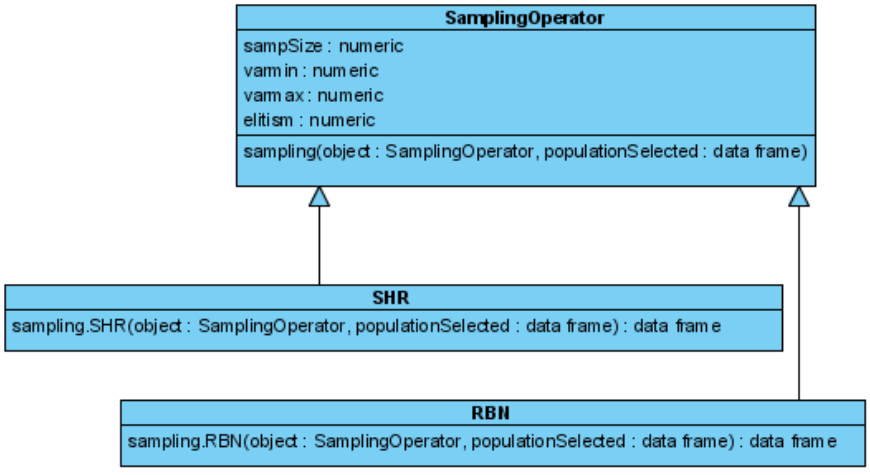


Figura 2.1.3 Diagrama de clases de los operadores de muestreo.

Como se observa en la figura 2.1.3 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 generando datos aleatorios.

La clase SHR implementa el operador para el muestreo de poblaciones con datos continuos que utilizan aprendizaje de la matriz de covarianza.

2.1.4 EDAs Celulares Continuos

En RCEDA un algoritmo EDAs Celular constituye un objeto de la clase del mismo nombre, la cual está compuesta por los diferentes operadores que intervienen en el funcionamiento del algoritmo. De esta clase hereda la clase EDAsCelularesContinuos, basándose en la clasificación de los EDAs Celulares de acuerdo al tipo de dominio sobre el cual operan.

Para correr un algoritmo se crea un nuevo objeto de la clase EDAsCelular el cual es configurado a partir de la llamada al método configure.EDAsCelular, 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 EDAs Celular configurado se llama al método execute, a partir del nuevo objeto de la clase EDAsCelular que se pasa como parámetro, este método determina cuál algoritmo se ejecutará; si el nuevo EDAs Celular es de tipo continuo se ejecutará el método execute.EDAContinuos.

2.1.5 Algoritmo de la propuesta

En el algoritmo siguiente se presenta el pseudocódigo del modelo de EDA celular propuesto. Cada iteración del EDA celular consiste de exactamente una iteración de todos los algoritmos miembros. Cada uno de estos algoritmos es responsable de actualizar exactamente una sub-población, y esto se realiza aplicando un modelo de EDA clásico local a la población a los individuos de las subpoblaciones vecinas (según el vecindario definido).

```
#Create a New Population
#Create Grid from the New Population
#For i in 1 to Maximum of iterations do
  #If StopCondition is True
    End Algorithm;
  #For each cell of the Grid
   #Exploration the neighborhood of Grid
      #For each cell of the neighborhood
        #Obtain the cell and add population of the neighborhood
    #Run EDA
     #Select m individuals according to operator selection
     #Learning from the probability distribution according to the operator learning
     #Estimate probabilities of the individuals obtained from learning
    #Evaluate the obtained individuals
   #Generate the new points (Size of cell) from the estimation for the population of
individuals
   #Create new cell and save into new Grid
```

2.1.6 Conclusiones parciales

En este capítulo se fundamentó sobre el análisis y diseño de la biblioteca y su funcionamiento, así como el comportamiento de sus clases y operadores.

Capítulo 3: Resultados experimentales

Problemas de optimización

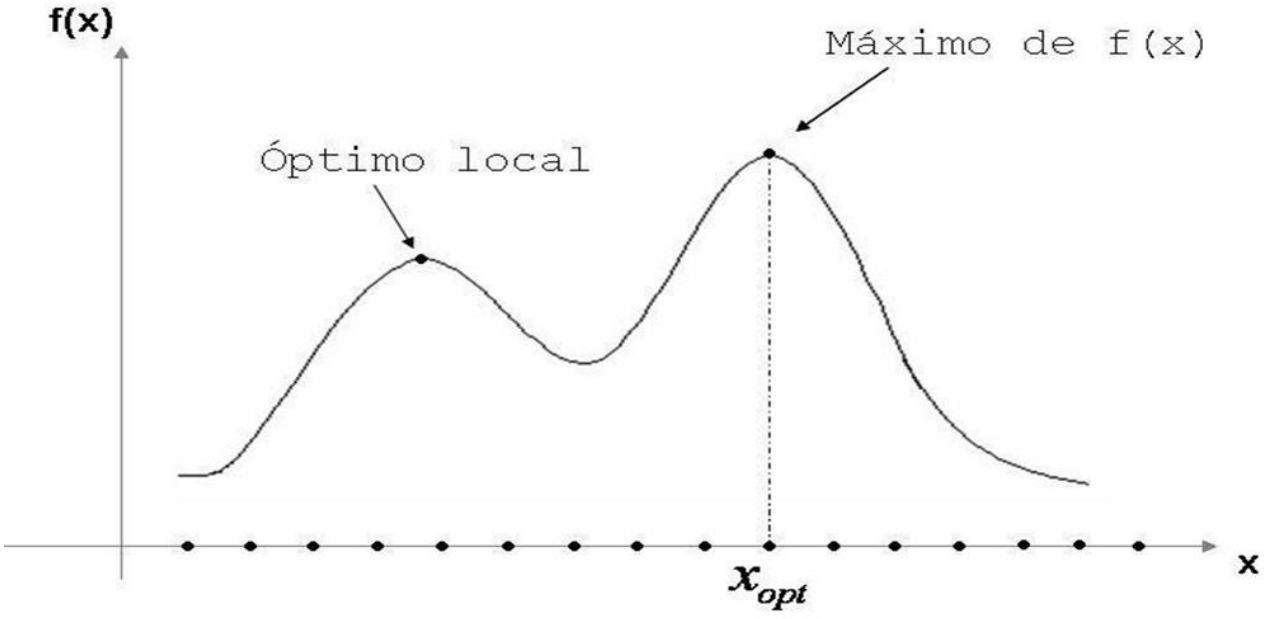
Los autores (Pelikan, 2012) plantean que:

> Un problema de optimización puede definirse como un conjunto de soluciones potenciales al problema y un procedimiento para evaluar la calidad de estas soluciones. La tarea es encontrar una solución a partir del conjunto de soluciones potenciales que aumenta al máximo la calidad definida por el procedimiento de evaluación. (p. 115)

Un problema de optimización puede definirse de la siguiente manera:

$$X_{opt} = \arg\max_{x \in D^n} f(x)$$

En la función anterior x i= (x1, x2,…, xn) denota un vector de dominio continuo de variables aleatorias. Para el caso continuo cada xi toma valores en un intervalo, la variable xi toma, de forma continua, todos los posibles valores comprendidos en el intervalo. La solución a este problema consiste en encontrar el punto máximo (óptimo) de la función f (x) □□ R. A partir de este momento las soluciones candidatas serán tratadas como individuos pertenecientes a la población con la cual se trabaja.



3.1Funciones objetivo continuas.

En la biblioteca se implementaron las siguientes funciones que operan sobre dominio continuo.

Sphere

Este problema de optimización se utiliza para crear una línea base en la comparación con otros problemas o algoritmos. Las variables x_i se definen en el intervalo $-600 \le x_i \le 600$, i = 1, 2,. . ., n, y el valor de actitud de cada individuo es computado de la forma:

$$Fsphere\left(\vec{x}\right) = \sum_{i=1}^{n} x_i^2$$

El valor óptimo $F^* sphere\left(\vec{x}\right)$ = 0 se alcanza cuando todas las variables toman valor 0.

Griewangk

Este es un problema de minimización. Las variables x_i están definidas en el intervalo $-600 \le x_i \le 600$, i = 1, 2,. . ., n, y el valor de aptitud de cada individuo es computado de la siguiente manera:

$$F_{Griewangk}\left(\vec{x}\right) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

El valor óptimo se alcanza cuando todas las variables toman valor 0.

Ackley

Este problema de minimización tiene su valor óptimo en $F_{Ackley}(\vec{x}) = 0$. Este óptimo se alcanza cuando todas las variables toman valor cero. Las variables x_i están definidas en el intervalo -$-6.0 \le x_i \le 6.0$, i = 1, 2,. . ., n. La definición de la función objetivo para dimensión n es la siguiente:

$$F_{Ackley}\left(\vec{x}\right) = -20.\exp\left(-0{,}2.\sqrt{\frac{1}{n}.\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}.\sum_{i=1}^{n}\cos\left(2.\pi.x_i\right)\right)$$

Rosenbrock

Este es un problema de minimización. Las variables x_i están definidas en el intervalo $-30 \le x_i \le 30$, i = 1, 2,. . ., n, y el valor de aptitud de cada individuo es computado de la siguiente manera:

$$F_{Rosenbtock}(\vec{x}) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$$

El valor óptimo se alcanza cuando todas las variables toman valor 1 y es igual a 0.

Rastrigin

Este problema de minimización tiene su valor óptimo en $F_{Rastrigin}(\vec{x}) = 0$. Este óptimo se alcanza cuando todas las variables toman valor cero. Las variables x_i están definidas en el intervalo $-5.12 \le x_i \le 5.12$, i = 1, 2,. . ., n. La definición de la función objetivo para dimensión n es la siguiente:

$$F_{Rastrigin}\left(\vec{x}\right) = \sum_{i=1}^{n}\left[x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right]$$

Para comprobar el correcto funcionamiento de la biblioteca, se realizaron una serie de experimentos consistentes en realizar corridas a los diferentes algoritmos implementados en ambos dominios, 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, en este caso el mejor individuo de la población seleccionada pasa directamente a formar parte de la nueva población.

3.2 Resultados y discusión

En esta sección se presenta un conjunto de experimentos que ilustran el comportamiento del algoritmo EDA celular en la función de prueba Ackley con diferentes vecindades.

Función objetiva continua Ackley

Esta función de optimización definida por n variables de entrada tiene propiedades que hacen posible examinar el rendimiento de diferentes algoritmos de optimización a probar, dada sus características, por lo que es una de las más usada para el estudio de estos algoritmos, para medir su eficacia.

Este problema de minimización tiene su valor óptimo en 0. Este óptimo se alcanza cuando todas las variables toman valor cero. Las variables están definidas en el intervalo -, i = 1, 2,. . ., n. La definición de la función objetivo para dimensión n es la siguiente:

$$F_{Ackley}\left(\vec{x}\right) = -20.\exp\left(-0{,}2.\sqrt{\frac{1}{n}.\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}.\sum_{i=1}^{n}\cos\left(2.\pi.x_i\right)\right)$$

La figura 3 muestra la representación de esta función en un espacio de tres dimensiones:

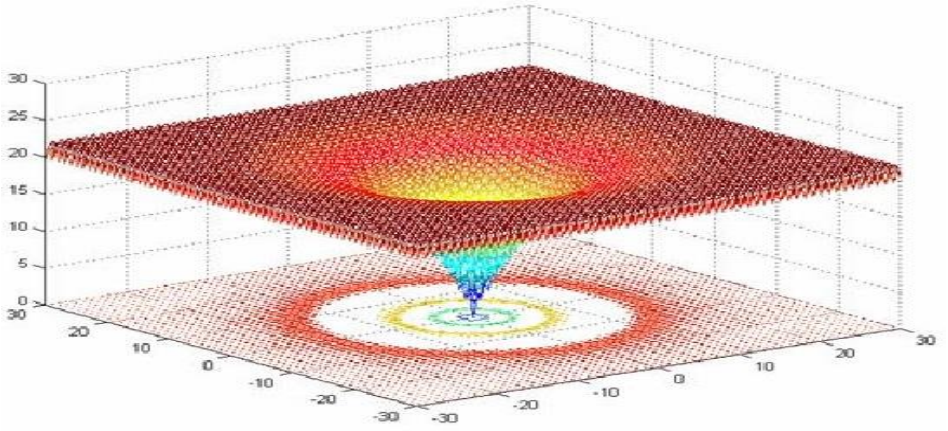


Figura 3. Representación de la función Ackley.

Resultados experimentales

Para el domino continuo se seleccionó la función Ackley, con el valor óptimo que se localiza en 0 y el margen de error utilizado para encontrarlo fue de 1e-8, con umbral de truncamiento 0.3 y

elitismo de selección de un individuo. La configuración de la rejilla fue de 4 filas por 4 columnas, con 5 x 5 individuos por cada célula.

En las tablas se muestran los resultados obtenidos para la función utilizando regularización de la covarianza, para un tamaño(n) de 10 y 30 variables, para cada vecindad para encontrar el óptimo y el valor de la desviación estándar(Desv), como medida de aproximación de esos resultados. El algoritmo se ejecutó 25 veces, marcándose en negrita los mejores resultados entre los EDAs celulares, siempre que tengan un comportamiento aceptable.

Tabla 1 Resultados del EDA celular para la función ***Ackley(n=10)***

Vecindad	Óptimo Encontrado	Desv	Iteraciones	Desv	Eval	Desv
one	-4.66E-11	1.28E-10	40.20	0.50	16439.80	0.00
l5	-9.34E-10	1.74E-09	25.28	0.46	10486.72	182.84
l9	-1.56E-09	2.47E-09	19.76	0.52	8284.24	208.60
c9	-2.39E-09	2.89E-09	19.88	0.53	8332.12	209.87
c13	-1.84E-09	2.91E-09	12.44	0.51	5363.56	202.14
c25	-1.81E-09	2.82E-09	11.00	0.71	4789.00	282.14
c41	**-4.44E-16**	0.00	**2.68**	1.07	1469.32	3.01923E-31

Tabla 2 Resultados del EDA celular para la función ***Ackley(n=30)***

Vecindad	Óptimo Encontrado	Desv	Iteraciones	Desv
one	-1.07E-09	2.1346E-09	73.12	0.60
l5	-4.44E-09	3.1307E-09	34.64	0.57
l9	-5.45E-09	3.1118E-09	24.28	0.84
c9	-4.04E-09	3.1563E-09	24.92	0.49
c13	-3.82E-09	3.2604E-09	13.92	0.70
c25	-3.57E-09	2.7936E-09	13.12	0.88
c41	**-4.44E-16**	3.0192E-31	**2.80**	1.26

Todos los experimentos muestran resultados semejantes para el EDA celular con vecindad c41, donde le valor del óptimo encontrado -4.44E-16 para 10 y 30 variables es el más cercano al valor 0, teniendo el número de iteraciones más bajos para ambos casos.

En los gráficos de bigote se muestran el número de evaluaciones de los EDAs celulares por cada vecindad, tanto para 10 variables, como para 30 variables.

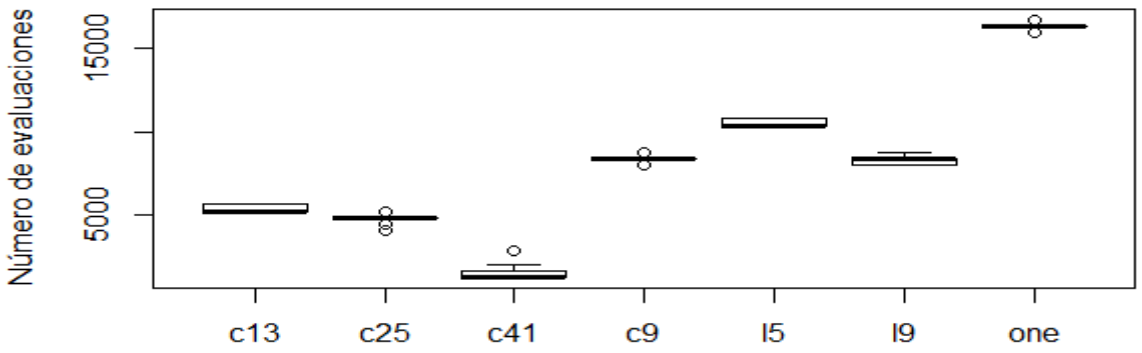


Figura *4.* Números de evaluaciones del EDA celular para la función Ackley(n=10)

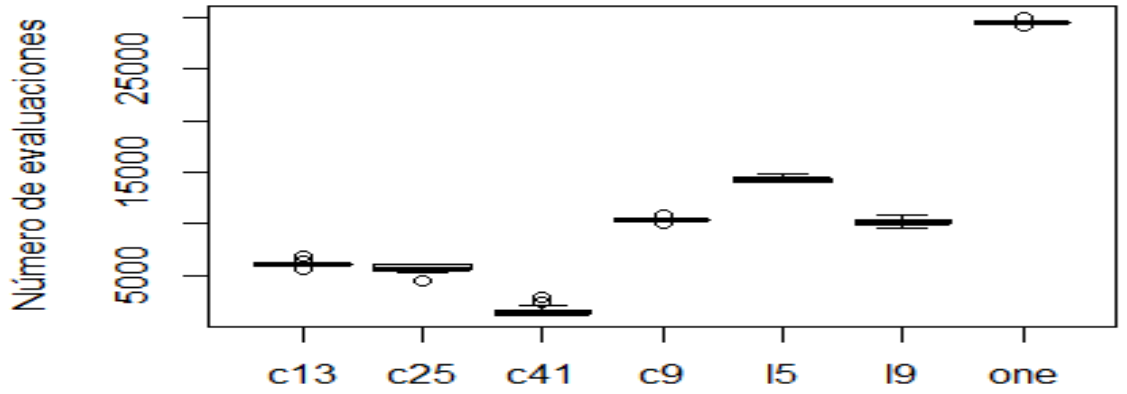


Figura 5. Números de evaluaciones del EDA celular para la función Ackley(n=30).

Como se visualizan en las figura anteriores, el algoritmo EDA celular con vecindad C41 tiene mayor eficiencia evaluativa para las diferentes cantidades de variables que con el resto de las vecindades, al tener el menor número de evaluaciones, tanto para 10 variables, como para 30 variables.

La de vecindad One es la de peor eficiencia, ya que para 10 y 30 variables el número de evaluciones superan las 25000 evaluaciones para encontrar el óptimo. Además, las vecindades C13 y C25 tuvieron una buena eficiencia evaluativa.

En la figura 5 se visualizan los valores de Mayor reducción del número de evaluaciones de la función objetivo que en el caso continuo.

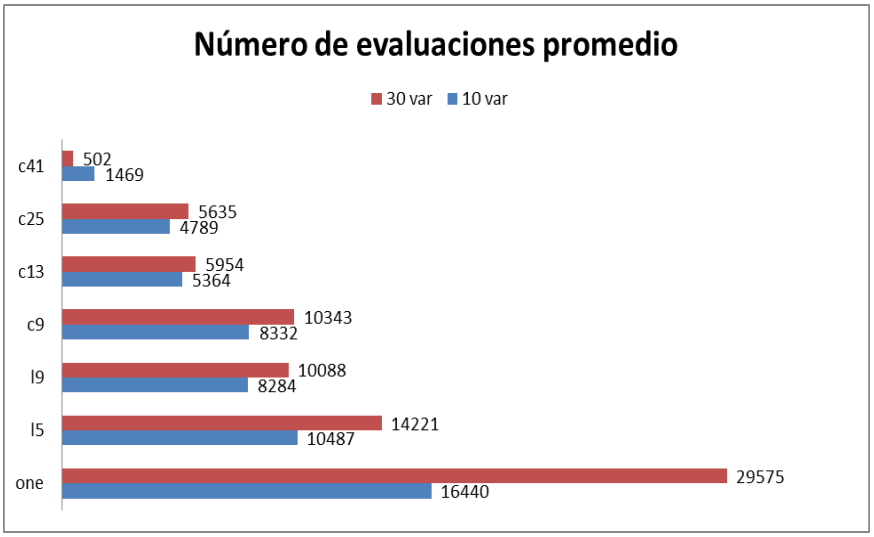


Figura 5.Números de evaluaciones del EDA celular para la función Ackley(n=10, n=30).

Como se muestra en la figura anterior, de las vecindades que usa el EDA celular con aprendizaje local de las estructuras y los parámetros,que aprenden las Redes Gaussianas, el de mejor resultado es la C41 que realiza de 502 y 1429 evaluaciones para 10 y 30 variables respectivamente.

Los resultados de los experimentos de los EDAs celulares para la función Ackley(n=10) fueron comparados con algunos algoritmos de la literatura, ver tabla 3.

Tabla 3 Resultados de diferentes algoritmos para la función Ackley(n=10)

Algoritmo	**Valor obtenido**	**Evaluaciones**
$EBCOA_{NB}$	7.7E-6	18116
$EBCOA_{SNB}$	6.1E-6	11891
$EBCOA_{TAN}$	6.7E-6	11333
UMDAc	8.8E-6	23063
MIMICc	7.8E-6	23382
EGNAee	7.9E-6	22983
EGNABGe	8.5E-6	22904
CMA-ES	1.9E-7	23962
cEDA-one	-4.66E-11	16440
cEDA -l5	-9.34E-10	10487
cEDA -l9	-1.56E-09	8284
cEDA -c9	-2.39E-09	8332
cEDA -c13	-1.84E-09	5364
cEDA -c25	-1.81E-09	4789
cEDA-c41	**-4.44E-16**	**1469**

Como se puede apreciar, la mayoría de los EDAs celulares tiene un valor obtenido más pequeño que los de la literatura, siendo el cEDA-c41 con el valor mínimo del óptimo encontrado y el de menor número de evaluaciones.

Grienwankg(n=10)

Función	Optimo Encontrado	Desv	Iteraciones	Desv	Eval	Desv
one	**-4.83E+05**	2.56E+05	**29.80**	11.05	**12290.20**	4408.60
l5	-2.27E-09	2.05E-09	8.40	0.82	3751.60	325.78
l9	**-1.08E-13**	1.61E-13	**2.00**	0.00	**1198.00**	0.00
c9	-6.33E-14	1.23E-13	2.32	0.63	1325.68	250.24
c13	-1.23E-13	1.94E-13	2.04	0.20	1213.96	79.80
c25	-4.95E-14	8.75E-14	2.00	0.00	1198.00	0.00
c41	-8.56E+00	13.72	2.04	0.20	1213.96	13.71514641

Los experimentos muestran resultados para el EDA celular donde se obtuvo la vecindad **l9** con mejor resultado, donde le valor del óptimo encontrado es **-1.08E-13** para n=10 siendo este el más cercano al valor 0 y teniendo el menor número de iteraciones y de evaluaciones y como peor resultado la vecindad **one** con un óptimo encontrado **-4.83E+05** y teniendo el mayor número de iteraciones y de evaluaciones.

Rastrigin(n=10)

Función	Optimo Encontrado	Desv	Iteraciones	Desv	Eval	Desv
one	**-3.75E+05**	2.53E+05	**30.36**	3.75	**12513.64**	1496.83
l5	-1.79E+05	1.80E+05	8.32	0.85	3719.68	340.13
l9	-9.09E-02	2.28E-01	2.08	0.28	1229.92	110.48
c9	-2.84E-02	1.16E-01	2.40	0.50	1357.60	199.50
c13	-5.23E-01	2.56E+00	2.04	0.20	1213.96	79.80
c25	**-3.41E-02**	1.25E-01	**2.00**	0.00	**1198.00**	0.00
c41	-2.84E-02	0.08	2.12	0.44	1245.88	0.082046408

Los experimentos muestran resultados para el EDA celular donde se obtuvo la vecindad **c25** con mejor resultado, donde le valor del óptimo encontrado es **-3.41E-02** para n=10 siendo este el más cercano al valor 0 y teniendo el menor número de iteraciones y de evaluaciones y como

peor resultado la vecindad **one** con un óptimo encontrado **-3.75E+05** y teniendo el mayor número de iteraciones y de evaluaciones.

Rosenbrock(n=10)

Función	Optimo Encontrado	Desv	Iteraciones	Desv	Eval	Desv
one	-5.12E+07	2.98E+07	18.92	1.15	7949.08	459.57
l5	-4.42E+05	2.71E+05	19.56	1.96	8204.44	781.88
l9	-1.02E+05	2.24E+05	13.28	1.06	5698.72	423.52
c9	**-6.89E-09**	1.69E-09	**42.16**	3.79	**17221.84**	1513.57
c13	-2.22E-09	2.70E-09	7.88	0.44	3544.12	175.44
c25	-1.51E-09	2.10E-09	7.04	0.84	3208.96	335.41
c41	**-5.59E-11**	0.00	**2.48**	1.05	**1389.52**	2.79463E-10

Los experimentos muestran resultados para el EDA celular donde se obtuvo la vecindad **c41** con mejor resultado, donde le valor del óptimo encontrado es **-5.59E-11** para n=10 siendo este el más cercano al valor 0 y teniendo el menor número de iteraciones y de evaluaciones y como peor resultado la vecindad **c9** con un óptimo encontrado **-6.89E-09** y teniendo el mayor número de iteraciones y de evaluaciones.

Sphere(n=10)

Función	Optimo Encontrado	Desv	Iteraciones	Desv	Eval	Desv
one	**-5.08E+05**	2.78E+05	**26.00**	0.41	**10774.00**	162.89
l5	-5.32E+05	2.63E+05	16.20	0.41	6863.80	162.89
l9	-2.96E+05	2.19E+05	12.48	0.51	5379.52	203.45
c9	-3.66E+05	2.33E+05	12.72	0.46	5475.28	182.84
c13	-1.45E+05	1.72E+05	8.28	0.46	3703.72	182.84
c25	-2.74E+05	2.79E+05	7.12	0.73	3240.88	289.56
c41	**-3.82E-02**	0.14	**2.44**	1.00	**1373.56**	0.136496644

Los experimentos muestran resultados para el EDA celular donde se obtuvo la vecindad **c41** con mejor resultado, donde le valor del óptimo encontrado es **-3.82E-02** para n=10 siendo este el más cercano al valor 0 y teniendo el menor número de iteraciones y de evaluaciones y como

peor resultado la vecindad **one** con un óptimo encontrado **-5.08E+05** y teniendo el mayor número de iteraciones y de evaluaciones.

Conclusiones

Los EDAs Celulares Continuos descentralizados con aprendizaje local de las estructuras y los parámetros, que utilizan las Redes Gaussianas para el aprendizaje basados en pruebas de independencias pueden reducir el número de evaluaciones en la resolución de problemas de optimización continuos. Esto se evidencia a través de los resultados obtenidos al estudiar las diferentes vecindades de poblaciones locales. Además, se mostró que este tipo de EDAs, disminuye el número de evaluaciones del óptimo de la función continua Ackley, en comparación con otros enfoques de la literatura que resuelven el mismo problema.

Recomendaciones

- ✖ Extender el uso de la biblioteca a un mayor número de funciones de aptitud para dominio continuo y discreto.
- ✖ Implementar nuevos EDA que utilicen núcleos y cópulas para estimar la distribución de probabilidad.

Bibliografía

1. Alba, E., Madera, J., Dorronsoro, B., Ochoa, A. y Soto , M. (2006). *Theory and practice of cellular UMDA for discrete optimization.* Berlin: Springer-Verlag.

2. Bibliography qingfeng Ding, G. Z. (2015). *The Cellular Differential Evolution Based on Chaotic Local Search. Mathematical Problems in Engineering.* Recuperado el 5 de febreo de 2017, de http://dx.doi.org/10.1155/2015/128902.

3. Bonet, J. S. De, Isbell, C. L. , and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities.En ”*Advances in Neural Information Processing Systems*.(vol.9, pp. 424-430). University of California, California:The MIT Press.

4. Bruegge, B. &. (2002). Ingeniería de Software Orientado a Objetoss. *Pearson Educación,* 3 (12)7.

5. Carbó, Y. M. (2014). *Impacto de las TIC en Cuba*. Recuperado el 9 de junio de 2017, de http://www.art.jovenclub.cu/?p=2713

6. Díaz,T. (2003). *Las funcionalidades del paquete R*. Reino Unido: Universidad Oxftord

7. Ding, A. (2015). *Algorithms of Estimation Different.* Institute of Cybernetics, Mathematics and Physics. La Habana, Cuba

8. Dorronsoro, B., Alba, E., Luque, G. y Bouvry, P. (2008). A Self-Adaptive Cellular Memetic Algorithm for the DNA Fragment Assembly Problem. *IEEE Congress on Evolutionary Computation (CEC2008).* Donostia - San Sebastián, Spain

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

10. 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). Berlin: Springer.

11. Huang, L. (2015). *Algoritmos de Estimación de Distribuciones Evolutivas.* San Sebastián: Universidad del País Vasco

12. Ivar Jacobson, G. B. (2004). *El Proceso Unificado de Desarrollo de Software*. Villa Clara,Cuba: Universidad Felix Varela.

13. K. Lu, D. M. (2013). "Intelligent test paper construction method based on cellular genetic algorithm", *Computer Engineering and Applications*. 49, 16, 57-60.

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

15. Larman. (1999). *UML y patrones y la Modelación de datos: Prentice Hall Iberoamericana UML*. Murcia,España: Whitepaper de Rational Rose.

16. Larrañaga, P. y Puerta, J. M. (2003). *Algoritmos de Estimación de Distribuciones*. México: Universidad Autónoma del estado de Mexico.

17. 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.

18. 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.

19. Madera, J., Ochoa, A. (2006 (Actualizado en 2008). *Una versión paralela del algoritmo MMHCEDA. Instituto de Cibernética, Matemática y Física (ICIMAF).* La Habana,Cuba

20. Martínez-López, Y., Madera, J. & de Varona, I. L. (Noviembre, 2016). Algoritmos Evolutivos con Estimación de Distribución Celulares. *Revista Cubana de Ciencias Informáticas*. 10, No. Especial UCIENCIA. Recuperado el 12 de enero de 2017, de http://rcci.uci.cu

21. Ochoa, A. . (2010). *Opportunities for Expensive Optimization with Estimation of Distribution Algorithms.* Institute of Cybernetics, Mathematics and Physics. La Habana, Cuba

22. Paradigm,V.(2013) Visual Paradigm for uml.*Visual Paradigm for UML-UML tool for software application development,*71

23. Paradis,Y.(2003) *Ingeniería Del Software con Objetos y Componentes*. Salamanca España: Universidad de Salamanca.

24. Partido Comunista de Cuba. (2011). *Lineamientos de la Politica Economica y Social del VI Congreso del PCC.* La Habana, Cuba: Partido Comunista de Cuba.

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

26. Pino, O. S. (2013). *Biblioteca de clases en r para el trabajo con algoritmos que estiman distribuciones.* Trabajo de Tesis para optar por el título de Ingeniera Informática.Universidad de Camagüey, Cuba.

27. Pressman, R. S. (s.f.). *Ingenieria de Software Un Enfoque Práctico*. Villa Clara, Cuba: Félix Varela.

28. Rumbaugh, J. J. (2000). *El lenguaje unificado de modelado. Manual de referencia*. Madrid,España:Universidad de Salamanca

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

30. Schäfer, J., Opgen-Rhein, R., Zuber, V., Ahdesmäki, M., Silva, P. D. y Strimmer, K. (2012*). Efficient Estimation of Covariance and (Partial) Correlation.* Reino Unido:Oxford

31. Troya, E. A. (2000). “Cellular evolutionary algorithms: Evaluating the influence of ratio,” in *Parallel problem solving from nature PPSN-6, ser.* (pp.29–38). Berlin, Alemania H. Springer-Verlag, Ed.

Anexos

Este es el código de la aplicación de consola:

```
init.all()
run <- function(fitFunction = "ShiftedSchwefelProblem1.2Noise",indsize
= 10,popSize = 25,select =
30,min=(-100),max=100,elitism=1,sample="GaussianSHR") {

  newcEDA <- new("cEDA")

  fitFun = new(fitFunction)
  init = "Continuous"
  popSize = popSize
  indSize = indsize
  selSize = select
  varMin = c(rep(min,indsize))
  varMax = c(rep(max,indsize))
  elit=elitism
  select = "Truncation"
  learn = new("COVSHRINK")
  samp = sample
  stop <- new("Optimum",optimum=(-450),epsilon=0.00000001,iterations=100000)

  #Config the cellular EDA
  lattice <-
    new(
      "LatticeConfiguration", totalRows = 5, totalCols = 5, totalRowsPerCell =
        2,
      totalColsPerCell = 2
    )

  neighbor = 1;
  if(neighbor==0){
    #one
    xradio <- array(c(0))
    yradio <- array(c(0))
  }
  else if(neighbor==1){
    #l5
    xradio <- array(c(0,0,0,-1,1))
    yradio <- array(c(0,-1,1,0,0))
  }
  else if(neighbor==2){
    #c13
```

```
    xradio <- array(c(0,0,0,0,0,-1,-1,-2,1,2,1,-1,1))
    yradio <- array(c(0,1,2,-1,-2,1,0,0,0,0,1,-1,-1))
  }
  else if(neighbor==3){
    #c25

    xradio <- array(c(0,0,0,0,0,0,0,-1,-1,-1,-2,-3,1,2,3,-2,1,-1,-1,-2,1,2,1,1,2))
    yradio <- array(c(0,1,2,3,-1,-2,-3,1,2,0,0,0,0,0,0,1,1,-1,-2,-1,-1,-1,-2,2,1))
  }
  else if(neighbor==4){
    #c41
    xradio <-
array(c(0,0,0,0,0,0,0,-1,-1,-1,-2,-3,1,2,3,-2,1,-1,-1,-2,1,2,1,1,2,0,0,1,1,2,2,3,3,4,-1,-1,-2,-2,-3,-3,-
4))
    yradio <-
array(c(0,1,2,3,-1,-2,-3,1,2,0,0,0,0,0,0,1,1,-1,-2,-1,-1,-1,-2,2,1,4,-4,3,-3,2,-2,1,-1,0,3,-3,-2,2,-
1,1,0))
  }else if(neighbor==5){
    #l9
    xradio <- array(c(0,0,0,0,0,-2,-1,1,2))
    yradio <- array(c(0,-1,-2,1,2,0,0,0,0))
  }else if(neighbor==6){
    #c9
    xradio <- array(c(0,0,0,-1,-1,1,1,-1,1))
    yradio <- array(c(0,1,-1,1,0,0,1,-1,-1))
  }

  neighborhood <- new("Neighborhood", xradio = xradio,yradio = yradio)
  config(
    newcEDA,fitFun,init,lattice,
neighborhood,indSize,selSize,elit,varMin,varMax,select,learn,samp,
stop
  )

}

cat(paste("Ejecucion del experimento Vecindad ONE"),"\n", file =
"console.txt", sep = " ", fill = FALSE, labels = NULL,
    append = FALSE)
for (i in 1:25) {
  run();
  cat(paste("FIN DE LA CORRIDA ",i),"\n", file = "console.txt", sep =
" ", fill = FALSE, labels = NULL,
      append = TRUE)
  print(paste("FIN DE LA CORRIDA ",i))
}
```

Este el código de implementación del EDA continuo:

```
executeSync.cEDAContinuous <- function(ceda,totalpopulation) {
  #Print the statistic for the grid
  neighborhood<-ceda@neighborhood
  points <- length(neighborhood@xradio)
  popLength
<-
points*ceda@latticeConfiguration@totalRowsPerCell*ceda@latticeConfiguration@totalColsPerC
ell
  totalpopLength <-
ceda@latticeConfiguration@totalRows*ceda@latticeConfiguration@totalCols*ceda@latticeConfi
guration@totalRowsPerCell*ceda@latticeConfiguration@totalColsPerCell
  sizeCell <- popLength/points
  individualsize <-ceda@initiationOperator@individualSize
  indSize <- ncol(totalpopulation@populationmatrix);
  popSize <- nrow(totalpopulation@populationmatrix);
  syncrone = TRUE
  globaling = FALSE

  statistics(totalpopulation,1)

  if(!globaling){
    sizeL <- popLength
  }
  else
  {
    sizeL <- totalpopLength
  }

  if(ceda@selectionOperator@selSize > sizeL){
    ceda@selectionOperator@selSize <- sizeL
  }


  for (iter in 2:ceda@stopCondition@iterations) {
    if (stop(ceda@stopCondition, totalpopulation))
      break;
    if(!globaling){
      #Local Strategy Learning
      if(syncrone){
        #Create a New Population
        totalpopMatrix<- as.data.frame(matrix(nrow = totalpopLength,
ncol = indSize))
```

```
        totalfitness <- array(dim = totalpopLength)
        countPop <- 1
        #For each cell of the Grid
        #Explorate the Grid Syncroned
        for (nRow in 1: ceda@latticeConfiguration@totalRows){
          for (nCol in 1: ceda@latticeConfiguration@totalCols){
            m<-1
            popMatrix <- as.data.frame(matrix(nrow=popLength, ncol=indSize))
            fit <- array(dim = popLength)

            #For each neigborhood
            for(n in 1:points){
              #Obtain the cell
              nRow <- nRow + neighborhood@yradio[n]
              if(nRow == 0 || ((nRow)%%ceda@latticeConfiguration@totalRows)==0){
                nRow <- nRow - 1
              }


              nCol<- nCol + neighborhood@xradio[n]
              if(nCol == 0 || ((nCol)%%ceda@latticeConfiguration@totalCols)==0){
                nCol <- nCol - 1
              }


              cell <- ((nRow)%%ceda@latticeConfiguration@totalRows) *
((nCol)%%ceda@latticeConfiguration@totalCols)
              for(cn in 0:(sizeCell-1)){
                j<-1
                while(j <= indSize){
                  popMatrix[m,j]=totalpopulation@populationmatrix[(cell+cn),j]
                  j<-j+1
                }
                fit[m]=totalpopulation@fitness[(cell+cn)]
                m<-m+1
              }

            }

            #Run EDA
            population <- new("Population",populationmatrix=
popMatrix, fitness=array(fit))

            #Select

            selPop <- selection(ceda@selectionOperator, population);
```

```
        selPop@populationmatrix <- as.data.frame(checkCont(selPop));

        learning <- learning(ceda@learningOperator,
convert.numeric(selPop@populationmatrix));

        sampMatrix <- sampling(ceda@samplingOperator,learning,
convert.numeric(selPop@populationmatrix));

        fitness <- evaluate(ceda@evaluateOperator, sampMatrix);

        if(ceda@samplingOperator@elitism==0){
          newMatrixPop=sampMatrix;
          newFitness=fitness;
        }else{

newMatrixPop=data.frame(rbind(selPop@populationmatrix[(1):(ceda@samplingOperator@elitis
m),],sampMatrix),row.names
= NULL,check.names=TRUE,stringsAsFactors = FALSE );
          newFitness=c(selPop@fitness[1:ceda@samplingOperator@elitism],fitness)
        }

        #generate the new points (Size of cell)
        index <- order(newFitness, decreasing = TRUE);
        #Create new cell and save into new Grid
        for(np in 1:sizeCell){
          j<-1
          while(j <= indSize){
            totalpopMatrix[countPop,j] = newMatrixPop[index[np],j]
            j<-j+1
          }

          totalfitness[countPop] = newFitness[index[np]]
          countPop <- countPop + 1
        }

      }
    }
    #Print the statistic for the grid
    newPopulation <-
new("Population",populationmatrix=totalpopMatrix,fitness=array(totalfitness));
    totalpopulation <- newPopulation;
    statistics(newPopulation,iter);
  }
  else{
    #Asyncronized
    for (m in 1: ceda@latticeConfiguration@totalRows){
      nRow = floor(runif(1,min = 1, max =
```

```
ceda@latticeConfiguration@totalRows))
        for (n in 1: ceda@latticeConfiguration@totalCols){
          nCol = floor(runif(1,min = 1, max =
ceda@latticeConfiguration@totalCols))

          popMatrix <- as.data.frame(matrix(nrow=popLength, ncol=indSize))
          fit <- array(dim = popLength)

          #For each neigborhood
          for(n in 1:points){
            #Obtain the cell
            nRow <- nRow + neighborhood@yradio[n]
            if(nRow == 0 || ((nRow)%%ceda@latticeConfiguration@totalRows)==0){
              nRow <- nRow - 1
            }


            nCol<- nCol + neighborhood@xradio[n]
            if(nCol == 0 || ((nCol)%%ceda@latticeConfiguration@totalCols)==0){
              nCol <- nCol - 1
            }


            cell <- ((nRow)%%ceda@latticeConfiguration@totalRows) *
((nCol)%%ceda@latticeConfiguration@totalCols)

            for(cn in 0:(sizeCell-1)){
              j<-1
              while(j <= indSize){
                popMatrix[m,j]=totalpopulation@populationmatrix[(cell+cn),j]
                j<-j+1
              }
              fit[m]=totalpopulation@fitness[(cell+cn)]
              m<-m+1
            }

          }

          #Run EDA
          population <- new("Population",populationmatrix=popMatrix,
fitness=array(fit))

          #Select

          selPop <- selection(ceda@selectionOperator,population);
          selPop@populationmatrix <- as.data.frame(checkCont(selPop));
```

```
          #Learning
          learning <- learning(ceda@learningOperator,
convert.numeric(selPop@populationmatrix));
          sampMatrix <- sampling(ceda@samplingOperator,learning,
convert.numeric(selPop@populationmatrix));

          #Evaluate
          fitness <-
evaluate(ceda@evaluateOperator,convert.numeric(sampMatrix));

          if(ceda@samplingOperator@elitism==0){
            newMatrixPop=sampMatrix;
            newFitness=fitness;
          }else{

newMatrixPop=data.frame(rbind(selPop@populationmatrix[(1):(ceda@samplingOperator@elitis
m),],sampMatrix),row.names
= NULL,check.names=TRUE,stringsAsFactors = FALSE );
            newFitness=c(selPop@fitness[1:ceda@samplingOperator@elitism],fitness)
          }

          #generate the new points (Size of cell)
          index <- order(newFitness, decreasing = TRUE);

          cell <- ((nRow)%%ceda@latticeConfiguration@totalRows) *
((nCol)%%ceda@latticeConfiguration@totalCols)
          for(cn in 0:(sizeCell-1)){
            j<-1
            while(j <= indSize){
              totalpopulation@populationmatrix[(cell+cn),j] =
newMatrixPop[index[cn+1],j]
              j<-j+1
            }
            totalpopulation@fitness[(cell+cn)] = newFitness[index[cn+1]]
          }
        }
      }
      statistics(totalpopulation,iter);
    }
  }
  else{
    #Global Strategy Learning
    #Explorate the Grid Syncroned
    for (nRow in 1: ceda@latticeConfiguration@totalRows){
      if(!syncrone)
      {
        nRow = floor(runif(1,min = 1, max =
```

```
ceda@latticeConfiguration@totalRows))
      }

      for (nCol in 1: ceda@latticeConfiguration@totalCols){

        if(!syncrone){
          nCol = floor(runif(1,min = 1, max =
ceda@latticeConfiguration@totalCols))
        }

        #Run EDA
        #Select
        selPop <- selection(ceda@selectionOperator,totalpopulation);
        #Select

        selPop <- selection(ceda@selectionOperator,population);
        selPop@populationmatrix <- as.data.frame(checkCont(selPop));

        #Learning
        learning <- learning(ceda@learningOperator,
convert.numeric(selPop@populationmatrix));
        sampMatrix <- sampling(ceda@samplingOperator,learning,
convert.numeric(selPop@populationmatrix));

        #Evaluate
        fitness <-
evaluate(ceda@evaluateOperator,convert.numeric(sampMatrix));

        if(ceda@samplingOperator@elitism==0){
          newMatrixPop=sampMatrix;
          newFitness=fitness;
        }else{

newMatrixPop=data.frame(rbind(selPop@populationmatrix[(1):(ceda@samplingOperator@elitis
m),],sampMatrix),row.names
= NULL,check.names=TRUE,stringsAsFactors = FALSE );
          newFitness=c(selPop@fitness[1:ceda@samplingOperator@elitism],fitness)
        }

        #generate the new points (Size of cell)
        index <- order(newFitness, decreasing = TRUE);

        cell <- ((nRow)%%ceda@latticeConfiguration@totalRows) *
((nCol)%%ceda@latticeConfiguration@totalCols)
        for(cn in 0:(sizeCell-1)){
          j<-1
          while(j <= indSize){
```

```
            totalpopulation@populationmatrix[(cell+cn),j] =
newMatrixPop[index[cn+1],j]
            j<-j+1
          }
          totalpopulation@fitness[(cell+cn)] = newFitness[index[cn+1]]
        }
      }
    }
    statistics(totalpopulation,iter);
  }
 }
}
```

Este es la aplicación principal q es a consola:

```
init.all()
run <- function(fitFunction = "ShiftedSchwefelProblem1.2Noise",indsize
= 10,popSize = 25,select =
30,min=(-100),max=100,elitism=1,sample="GaussianSHR") {

 newcEDA <- new("cEDA")

 fitFun = new(fitFunction)
 init = "Continuous"
 popSize = popSize
 indSize = indsize
 selSize = select
 varMin = c(rep(min,indsize))
 varMax = c(rep(max,indsize))
 elit=elitism
 select = "Truncation"
 learn = new("COVSHRINK")
 samp = sample
 stop <- new("Optimum",optimum=(-450),epsilon=0.00000001,iterations=100000)

 #Config the cellular EDA
 lattice <-
  new(
   "LatticeConfiguration", totalRows = 5, totalCols = 5, totalRowsPerCell =
    2,
   totalColsPerCell = 2
  )

 neighbor = 1;
 if(neighbor==0){
  #one
  xradio <- array(c(0))
  yradio <- array(c(0))
```

```
  }
  else if(neighbor==1){
    #l5
    xradio <- array(c(0,0,0,-1,1))
    yradio <- array(c(0,-1,1,0,0))
  }
  else if(neighbor==2){
    #c13
    xradio <- array(c(0,0,0,0,0,-1,-1,-2,1,2,1,-1,1))
    yradio <- array(c(0,1,2,-1,-2,1,0,0,0,0,1,-1,-1))
  }
  else if(neighbor==3){
    #c25

    xradio <- array(c(0,0,0,0,0,0,0,-1,-1,-1,-2,-3,1,2,3,-2,1,-1,-1,-2,1,2,1,1,2))
    yradio <- array(c(0,1,2,3,-1,-2,-3,1,2,0,0,0,0,0,0,1,1,-1,-2,-1,-1,-1,-2,2,1))
  }
  else if(neighbor==4){
    #c41
    xradio <-
array(c(0,0,0,0,0,0,0,-1,-1,-1,-2,-3,1,2,3,-2,1,-1,-1,-2,1,2,1,1,2,0,0,1,1,2,2,3,3,4,-1,-1,-2,-2,-3,-3,-
4))
    yradio <-
array(c(0,1,2,3,-1,-2,-3,1,2,0,0,0,0,0,0,1,1,-1,-2,-1,-1,-1,-2,2,1,4,-4,3,-3,2,-2,1,-1,0,3,-3,-2,2,-
1,1,0))
  }else if(neighbor==5){
    #l9
    xradio <- array(c(0,0,0,0,0,-2,-1,1,2))
    yradio <- array(c(0,-1,-2,1,2,0,0,0,0))
  }else if(neighbor==6){
    #c9
    xradio <- array(c(0,0,0,-1,-1,1,1,-1,1))
    yradio <- array(c(0,1,-1,1,0,0,1,-1,-1))
  }

  neighborhood <- new("Neighborhood", xradio = xradio,yradio = yradio)
  config(
    newcEDA,fitFun,init,lattice,
neighborhood,indSize,selSize,elit,varMin,varMax,select,learn,samp,
stop
  )

}

cat(paste("Ejecucion del experimento Vecindad ONE"),"\n", file =
"console.txt", sep = " ", fill = FALSE, labels = NULL,
    append = FALSE)
```

```
for (i in 1:25) {
  run();
  cat(paste("FIN DE LA CORRIDA ",i),"\n", file = "console.txt", sep =
" ", fill = FALSE, labels = NULL,
      append = TRUE)
  print(paste("FIN DE LA CORRIDA ",i))
}
```