Operadores de Mutación en Algoritmos Genéticos Celulares Aplicados a Problemas Continuos

Sosa, H.[1] Villagra, S.[2] Villagra, A.[2]
[1] {Becario de Investigación}
[2] {Docente Investigador UNPA}
hassio_09@hotmail.com, { svillagra, avillagra } @*uaco.unpa.edu.ar*
UNPA UACO
Universidad Nacional de la Patagonia Austral - Unidad Académica Caleta Olivia
Departamento de Ciencias Exactas y Naturales
LabTEm- Laboratorio de Tecnologías Emergentes
Caleta Olivia, 2013

RESUMEN

El diseño de algoritmos eficientes para resolver problemas complejos es uno uno de los aspectos más importantes de investigación en el campo de la informática. El objetivo perseguido es fundamentalmente el desarrollo de nuevos métodos capaces de resolver problemas complejos con el menor esfuerzo computacional posible, mejorando así a los algoritmos existentes de una forma eficaz.
Los Algoritmos Genéticos Celulares (cGAs) forman parte de las herramientas de optimización más populares. Estos algoritmos se enfocan en encontrar soluciones óptimos en un tiempo reducido en comparación a métodos exactos. En este trabajo se propone un estudio comparativo de diferentes operadores de mutación en un cGa aplicado a problemas académicos clásicos de optimización continua.
Palabras clave: algoritmo genético celular, metaheurística, operadores de mutación, problema de optimización continua.

1. Introducción

La optimización es una disciplina fundamental en campos de las ciencias tales como Informática, Inteligencia Artificial, Logística, Biología, Tecnología de la Producción, Física, etc. El concepto de optimización puede verse como el proceso de encontrar y mejorar el rendimiento de un a aplicación o dispositivo a partir de determinados cambios lógicos o físicos. Un problema de optimización se formaliza como un par (S, f), donde $S \neq \emptyset$ representa el espacio de soluciones (o de búsqueda) del problema, mientras que f es un criterio de calidad conocido como función objetivo, definida como $f : S \rightarrow R$. Así, resolver un problema de optimización consiste en encontrar un conjunto de valores adecuados de forma que la solución representada por estos valores $i^* \in S$ satisfaga la siguiente desigualdad $f(i^*) \leq f(i), \forall i \in S$. Asumir el caso de maximización o minimización no restringe en ningún caso la generalidad de los resultados, puesto que se puede establecer una igualdad entre tipos de problemas de maximización y minimización [1], [2].
Sin embargo, los métodos exactos necesitan tiempos exponenciales de computación cuando se trata con instancias grandes de problemas complejos. Los problemas NP - completos no tienen un algoritmo en tiempo polinómico que los resuelva. También existe otro tipo de problemas al menos tan difíciles de resolver como los anteriores denominados NP-duros para los cuales tampoco existe un algoritmo polinómico que los resuelva, es decir que esta clase de problemas NP no puede abordarse de forma realista con técnicas exactas. En consecuencia, el uso de técnicas aproximadas está recibiendo en las últimas décadas cada vez más atención. En estos métodos aproximados se sacrifica la garantía de encontrar el óptimo global al problema (en muchos casos, aunque no siempre) con el fin de encontrar soluciones buenas en un tiempo significativamente reducido en comparación con los métodos exactos.
Los algoritmos evolutivos (EAs) son técnicas de optimización que trabajan sobre poblaciones de soluciones y que están diseñadas para buscar valores óptimos en espacios complejos. La mayoría de los EAs trabajan sobre una única población (panmixia) de individuos, aplicando los operadores a la población como un todo, por ejemplo los Algortimos Genéticos (GA). Por otro lado, existe también una cierta tradición en el uso de EAs estructurados (en los que la población se descentraliza de alguna forma). Entre los muchos tipos de EAs estructurados, los algoritmos distribuidos y los celulares son las herramientas de optimización más populares. En el marco de de los celulares encontramos los Algoritmos Genéticos Celulares (cGAs).
En este documento se describe el estudio realizado sobre diferentes operadores de mutación en un cGA aplicado a problemas académicos clásicos de optimización continua, con el objetivo de identificar el operador adecuado.

Este documento se organiza de la siguiente forma: en la Sección 2 se describe conceptualmente un Algoritmo Genético Celular. En la Sección 3 se mencionan los Operadores de Mutación utilizados, y en la Sección 4 se presentan los Problemas de Optimización Continua. Por su parte en la Sección 5 se describe el Diseño de los experimentos y algunos resultados. Finalmente en la Sección 6, se exponen las conclusiones arribadas y líneas de trabajos a futuro.

2. Algoritmos Genéticos Celulares

La metaheurística central de este trabajo está basada en los Algoritmos Genéticos Celulares (cGA). Un cGA es un tipo de Algoritmo Genético, basado en una clase de población descentralizado en el que las soluciones tentativas evolucionan en barrios superpuestos [3], [4] . En un cGA, conceptualmente los individuos son situados en una malla toroidal bidimensional (normalmente es bidimensional, aunque el número de dimensiones puede ser extendido fácilmente a tres o más), y se les permite recombinarse con individuos cercanos.
Estos barrios superpuestos ayudan en la exploración del espacio de búsqueda debido a que por medio de una lenta difusión de las soluciones a través de la población, proporciona una especie de exploración, mientras que la explotación tiene lugar dentro de cada barrio por los operadores genéticos.
El barrio más utilizado se llama L5 (también llamado vecindario de NEWS) integrado por los individuos Norte, Este, Oeste y Sur que se muestran en la Figura 1. Los individuos sólo pueden interactuar con sus vecinos en el bucle reproductor que aplica los operadores de variación. Este bucle reproductor es aplicado dentro del vecindario de cada individuo y consiste generalmente en seleccionar dos individuos del vecindario como padres de acuerdo a un cierto criterio, aplicarles los operadores de variación (recombinación y mutación), y reemplazar el individuo considerado por el descendiente recientemente creado siguiendo un determinado criterio, por ejemplo, si el descendiente representa una mejor solución que el individuo considerado [4].

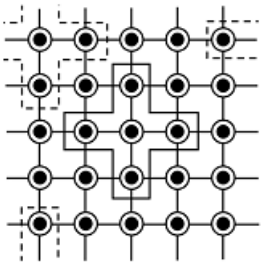

Figura 1: Disposición de la población en un cGA

El siguiente es un pseudocódigo de un cGA canónico (Figura 2). Se puede ver que tras la generación y evaluación (líneas 2 y 3 respectivamente) de la población inicial, los operadores genéticos tales como selección de los padres, recombinación, mutación y el reemplazo del individuo actual por un descendiente (de líneas 8 a 12), son aplicados a cada uno de los individuos dentro del entorno de sus vecindarios iterativamente hasta alcanzar una condición de finalización (en este caso MAX_PASOS).
Los individuos que formarán parte de la población de la siguiente generación (los nuevos descendientes generados o bien los individuos de la población actual, dependiendo del criterio de reemplazo declarado en la línea 12) van almacenándose en una población auxiliar que tras cada generación reemplaza a la población actual. Por tanto, en este modelo todos los individuos son actualizados simultáneamente en la población [5].

```
1. proc Evoluciona(cga)       // Parámetros del algoritmo en 'cga'
2. GeneraPoblaciónInicial(cga.pobl);
3. Evaluación(cga.pobl);
4. para s ← 1 hasta MAX_PASOS hacer
5.    para x ← 1 hasta cga.ANCHO hacer
6.       para y ← 1 hasta cga.ALTO hacer
7.          vecinos ← CalculaVecindario(cga,posición(x,y));
8.          padres ← Selección(vecinos);
9.          descendiente ← Recombinación(cga.Pc,padres);
10.         descendiente ← Mutación(cga.Pm,descendiente);
11.         Evaluación(descendiente);
12.         Remplazo(posición(x,y),pobl_auxiliar,descendiente);
13.      fin para
14.   fin para
15.   cga.pobl ← pobl_auxiliar;
16. fin para
17. fin proc Evoluciona
```

Figura 2: Pseudocódigo de un cGA canónico

En la siguiente sección se mencionan los Operadores de Mutación utilizados en este trabajo.

3. Operadores de Mutación

La mutación se basa en un operador básico, que brinda aleatoriedad a los individuos de una población. Si bien el operador de cruce se encarga de hacer una búsqueda en el espacio de posibles soluciones, es el operador de mutación el encargado de aumentar o reducir el espacio de búsqueda en un algoritmo genético y de proporcionar cierta variabilidad genética de los individuos.
Existen diversos operadores de mutación. A continuación se detallan los operadores de mutación para representaciones reales utilizados en este trabajo.

3.1. Mutación Gaussiana

La mutación Gaussiana, propuesta por John Holland en 1975 [6], funciona de la siguiente manera: dado un cromosoma x con un gen seleccionado para la mutación i, se le aplica una distribución normal N de media pi y desviación estándar s (parámetro). Dado un cromosoma p como j-ésimo de un gen seleccionado para mutación, se produce un cromosoma c de la siguiente forma:

$$C_i = N\delta_i, \theta \text{ si j=i; } P_i \text{ en caso contrario} \quad (1)$$

Donde $N\delta_i, \theta$ es una distribución normal con media P_i y desviación estándar θ (parámetro). Alternativamente se puede disminuir el valor de θ a medida que aumente el número de generaciones.

3.2. Mutación polinomial

Propuesta por Deb y Goyal [7], donde dada una solución que actúa como padre, este operador genera soluciones cercanas con una probabilidad más alta que soluciones lejanas a él, independientemente de la iteración. Utiliza una distribución de probabilidad polinomial de la siguiente manera:

$$x_i^{(t+1)} = x_i^{(t)} + (x_i^{(U)} - x_i^{(L)}).\delta_i, \quad (2)$$

donde siendo $u_i < 0,5$, la δ_i es igual a:

$$(2 \cdot u_i)^{\frac{1}{(n+1)}} - 1, \quad (3)$$

y si $u_i \geq 0,5$, entonces δ_i resulta:

$$1 - [2 \cdot (1 - u_i)]^{1/(n+1)} \quad (4)$$

$u_i \sim U[0,1]$ y n es un parámetro que controla la variabilidad de la perturbación.

3.3. Mutación uniforme

Este operador de mutación reemplaza el genoma, ya sea inferior o superior con destino al azar. Esto puede ser usado para los genes enteros y de coma flotante. Por ejemplo, dada una población $p = \{x_1, ..., x_m\}$, el individuo mutado será: $p' = \{x_1, ..., x_k^1, ..., x_m\}$ donde x_k^1 es el individuo alterado con un valor entre rangos mínimos y máximos, usando una distribución uniforme.

3.4. Mutación no uniforme.

Es una mutación propuesta por Michalewicz [8]. Este operador hace, en las generaciones iniciales, una búsqueda uniforme en el espacio (exploración), pero esta búsqueda es más local en las generaciones finales (explotación). La probabilidad de crear una solución cercana al padre es mayor que la de crearla alejado de él, pero según avanzan las iteraciones, esta probabilidad se hace más y más grande. Para el caso de un solo objetivo, este operador ha

proporcionado buenos resultados y ha sido incluso utilizado como elemento principal en el diseño de algoritmos nuevos.
En la versión de Deb [9], este operador tiene la siguiente forma:

$$x_i^{t+1} = x_i^t + \sigma \cdot \left(x_i^U - x_i^L\right) \cdot \left(1 - u_i^{1-t/t_{max}^b}\right), \tag{5}$$

donde $u_i \sim U\,[0,1]$, t_{max} es el número máximo de iteraciones permitido, σ vale +1 o -1, con probabilidad 0,5, y b es un parámetro de que determina el grado de dependencia de la iteración.

4. Problemas de Optimización Continua

Los problemas de optimización continua que presentamos en esta sección, son funciones académicas clásicas conocidas. Este conjunto de problemas incorpora diversas características que permiten una importante variedad de escenarios para analizar.

4.1. Problema de Ackley

El problema de Ackley [10] es un problema de minimización. Originalmente este problema fue definido para dos dimensiones, pero ha sido generalizado a N dimensiones [1].
Formalmente, este problema puede ser descrito como la búsqueda de una cadena $\vec{x} = \{x_1, x_2, ..., x_N\}$, con $x_i \in (-32,768; 32,768)$, que minimiza la siguiente ecuación:

$$f(x) = -20exp(-0{,}2\sqrt{\frac{1}{D}\sum_{i=t}^{D} x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\phi x_i)) + 20 + e \tag{6}$$

4.2. Problema de expansión de F10 (EF10)

F10 es una función que tiene interacciones no lineales entre dos variables. Su versión ampliada EF10, está construida de tal manera que se induce la interacción no lineal a través de múltiples variables. Es no separable [11]. Su función de fitness está dada por:

$$Zfef_{10}(\vec{x}) = 10(x_1, x_2) + ... + f_{10}(x_{i-1}, x_i) + ... + f10(x_n, x_1), \tag{7}$$

con un n=10 y valores de las variables: $-100{,}0 < x_i \leq 100{,}0$ [5].

4.3. Problema de Griewangk

El problema de Griewangk [12] define un problema de minimización, sobre un espacio multidimensional, para encontrar una cadena definida por $\vec{x} = \{x_1, x_2, ..., x_N\}$, con $x_i \in (-600; 600)$, lo cual minimiza la siguiente ecuación:

$$f_{Gri}(\vec{x}) = \frac{1}{d}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1 \tag{8}$$

4.4. Problema de Rastrigin

En la optimización matemática, la función Rastrigin es una función no-convexa utilizado como un problema de prueba de rendimiento de los algoritmos de optimización. Es un ejemplo típico de la función multimodal no lineal. Inicialmente fue propuesto por Rastrigin[12] como una función bidimensional, siendo posteriormente generalizado por Mühlenbein. Esta función supone un problema díficil de optimizar debido a su amplio espacio de búsqueda así como por el gran número de mínimos locales .
Está definida por:

$$f(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\phi x_i) + 10\right] \tag{9}$$

Donde D es el número de dimensiones.

4.5. Problema de Ecuaciones Lineales (Sle)

El problema de ecuaciones lineales (Sle) consiste en encontrar los valores del vector $\vec{x}$ tal que $A\vec{x} = \vec{b}$ con:

$$A = \begin{pmatrix} 5,4,5,2,9,5,4,2,3,1 \\ 9,7,1,1,7,2,2,6,6,9 \\ 3,1,8,6,9,7,4,2,1,6 \\ 8,3,7,3,7,5,3,9,9,5 \\ 9,5,1,6,3,4,2,3,3,9 \\ 1,2,3,1,7,6,6,3,3,3 \\ 1,5,7,8,1,4,7,8,4,8 \\ 9,3,8,6,3,4,7,1,8,1 \\ 8,2,8,5,3,8,7,2,7,5 \\ 2,1,2,2,9,8,7,4,4,1 \end{pmatrix}, \vec{b} = \begin{bmatrix} 40 \\ 50 \\ 47 \\ 59 \\ 45 \\ 35 \\ 53 \\ 50 \\ 55 \\ 40 \end{bmatrix} \tag{10}$$

La función de evaluación que minimizamos en nuestros experimentos se muestra en la ecuación [5]:

$$f_{Sle}(\vec{x}) = |\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}.x_j - b_i)| \tag{11}$$

Esta función tiene la solución óptima $f_{Sle}(\vec{x*}) = 0,0$, y los valores de las diez variables del problema están dentro del intervalo $[-9,0; 11,0]$.

4.6. Problema de Sphere

El problema Sphere es un problema de minimización para el que el mínimo global se encuentra en $x* = (0, ..., 0)$. La mayor dificultad en cuanto a la optimización de este problema reside en el amplio espacio de búsqueda en el que las variables pueden tomar valores [5]. Este es un buen problema para analizar cómo interaccionan las variables entre sí (interacciones de grado 2) cuando se aplica un modelo de regresión con penalización L1.
Su función de fitness se define por:

$$S_{Sph}(\vec{x}) = \sum_{i=1}^{n} x_i^2 \tag{12}$$

Con $n = 25$ y $-5{,}12 \leq x_i \leq 5{,}12$.

5. Diseño de experimentos y resultados

Para realizar el estudio de cGA con diferentes operadores de mutación se utilizan dos configuraciones de parámetros. La primera configuración establece la parametrización del algoritmo definida en [5]. El tamaño de la población se fija en 400 individuos (20x20). El mecanismo de selección de los padres es la selección por torneo. El operador de recombinación es el crossover de dos puntos (DPX) con una probabilidad de recombinación establecida en 1. La probabilidad de mutación para los operadores usados (Gaussiana, Polinomial, Uniforme y No Uniforme) es 1 y el número máximo de evaluaciones se fija en 1000000. La Tabla 1 resume los valores utilizados.
Para la segunda configuración de parámetros se modifica únicamente la probabilidad de mutación para algunas instancias de problema. El resto de los parámetros se mantiene igual a la primera configuración. Los valores de probabilidades de mutación se establecen en base a lo definido en [5], y se utiliza para cada probabilidad de mutación 1/2L donde L es la longitud del individuo.

Tamaño de población	400 individuos (20x20)
Selección de padres	Torneo
Operador de recombinación	DPX (dos puntos)
Probabilidad de mutación	1
Máximo de evaluaciones	1000000

Tabla 1: Parámetros de la primera configuración

Para cada uno de las configuraciones de parámetros y para cada operador de mutación se realizan 30 corridas independientes. Todos los algoritmos están programados en Java y se ejecutaron en una máquina tipo PC de 2.53 GHz Intel i5 procesador con 4 GB RAM bajo Windows 7.
En este trabajo los mejores valores obtenidos están resaltados en **negrita** y el valor de la solución óptima para todos los problemas es 0,0 en todos los casos.
En la Tabla 2 se muestra el resumen de las corridas para los seis problemas y las cuatro operadores de mutación aplicados en cGA utilizando la primera configuración de parámetros. La primer columna corresponde a la instancia utilizada (problema). Luego para cada uno de los operadores de mutación se muestra valor objetivo encontrado y el tiempo (en milisegundos).
Podemos observar que para cuatro de las seis instancias (Ackley, EF10, Rastrigin y Sphere) el menor valor objetivo es obtenido por cGA con la mutación Gaussiana.
En cuanto al tiempo, se distingue que en cuatro de las seis instancias (Ackley, EF10, Sle y Sphere) el menor tiempo de ejecución se obtuvo con la mutación Cauchy.

	Cauchy		Uniforme		No uniforme		Gaussiana	
Problema	Valor obj.	Tiempo	Valor obj.	Tiempo	Valor obj.	Tiempo	Valor obj.	Tiempo
Ackley	9,81E-04	**12748**	0,016	13237	0,013	13577	**6,04E-04**	12982
EF10	0,127	**13445**	0,821	14702	0,667	15104	**0,117**	13727
Griewangk	**9,16E-07**	18460	0,010	**17104**	0,014	17411	1,95E-06	17300
Rastrigin	1,78E-04	17157	0,001	**15444**	0,002	15899	**1,16E-04**	15714
Sle	6,47E-06	**4513**	2,53E-05	4540	**1,81E-06**	4827	6,09E-06	7336
Sphere	6,94E-07	**14008**	7,60E-06	14126	7,66E-06	14484	**5,30E-07**	14288

Tabla 2: Resultados obtenidos por cGA utilizando la primera configuración de parámetros

En la Tabla 3 se muestra, usando la segunda configuración, el resumen de las corridas. Los resultados obtenidos nos muestran que para cuatro instancias

(Ackley, EF10, Rastrigin y Sle) la mutación Gaussiana es la que obtuvo mejor rendimiento, con valores más cercanos al óptimo.
Respecto al tiempo de ejecución, no se observa una mutación con mejores resultados en la mayoría de los problemas. No obstante, el menor tiempo obtenido está distribuido en los distintos operadores utilizados.

	Cauchy		Uniforme		No uniforme		Gaussiana	
Problema	Valor obj.	Tiempo	Valor obj.	Tiempo	Valor obj.	Tiempo	Valor obj.	Tiempo
Ackley	0,025	11781	1,396	12967,5	1,064	9377	**0,019**	**9135**
EF10	0,483	13230	2,643	11666	3,439	**11570**	**0,444**	12076
Griewangk	**2,60E-05**	12984	0,564	13243	0,496	13471	0,047	**12981**
Rastrigin	0,126	**44531**	0,873	48703	0,945	57250	**0,078**	47140
Sle	2,79E-04	**3817**	2,79E-04	**3817**	0,001	3990	**2,29E-05**	3810
Sphere	**0,001**	11127	0,008	**10658**	0,006	11234	0,692	35758

Tabla 3: Resultados obtenidos por cGA utilizando la segunda configuración de parámetros

En ambos experimentos se observa que la utilización de cGA con la mutación Cauchy obtiene los mejores valores en cuanto al valor óptimo. En vista de que la primera configuración posee los valores más cercanos a este, se realizará el análisis estadistico con los resultados de la Tabla 2. Tomaremos como variable de análisis el valor objetivo a fin de determinar determinar si las diferencias entre los resultados obtenidos por cGA con los distintos operadores de mutación son estadísticamente significativos.
Para realizar este análisis lo primero que se debe hacer es determinar el tipo de test a utilizar (paramétrico o no paramétrico). Para ello se deben cumplir tres condiciones: independencia (ya cumplida por ser resultados de ejecuciones distintas), normalidad y homocedasticidad.
El test de Kolmogarov-Smirnov nos permite conocer la normalidad y el test de Levene la homocedasticidad. La herramienta usada para realizar estos test es el software SPSS.
Todos los test utilizados obtienen el respectivo p-valor asociado, que representa la disimilitud de los resultados con respecto a la forma normal. Por lo tanto, un p-valor bajo señala una distribución no normal. En este estudio, vamos a considerar un nivel de significancia $\alpha = 0,05$, por lo que un p-valor mayor que α indica que la condición de normalidad se cumple.
En la Tabla 4 se muestra la prueba de Kolmogorov-Smirnov aplicado a cGA con diferentes operadores de mutación, donde el símbolo "*" indica que los resultados obtenidos por el algoritmo no cumplen con la condición de normalidad. Podemos decir que cuando los resultados de al menos un algoritmo no cumple las condiciones de normalidad, debemos entonces aplicar test no

paramétricos (siempre que los datos también posean homocedasticidad).
Se observa entonces que los algoritmos Ackley, Rastrigin y Sphere poseen una distribución normal.

Problema	Cauchy	Gaussiana	Uniforme	No Uniforme
Ackley	0,492	0,888	0,994	0,997
EF10	(*)0,020	0,412	0,455	0,832
Griewangk	(*)4,13E-4	(*)0,004	0,641	0,771
Rastrigin	0,123	0,192	0,570	0,643
Sle	(*)0,084	0,192	0,340	0,643
Sphere	0,274	0,727	0,775	0,829

Tabla 4: Prueba de Kolmogorov-Smirnov aplicado a cGA con diferentes operadores de mutación

A continuación, se realiza el test de Levene (Tabla 5). Su resultado permite saber si el conjunto de datos posee homocedasticidad, cuyo caso es posible si el estadístico de Levene es inferior a 0,05.

Problema	Estadístico de Levene
Ackley	0,49
EF10	0,00
Griewangk	0,00
Rastrigin	0,00
Sle	0,00
Sphere	0,00

Tabla 5: Test de Levene aplicado a cGA con diferentes operadores de mutación

Luego de este análisis, comprobamos que los resultados obtenidos por los algoritmos en el problema Ackley poseen homocedasticidad. Como también tiene una distribución normal, se le debe aplicar tests paramétricos. Para el resto de los problemas analizados, la condición de homocedasticidad no se cumple por lo que se deben aplicar test no paramétricos.
Teniendo en cuenta lo anterior, se aplica entonces el test de ANOVA para los resultados de los algoritmos que resuelven el problema de Ackley y el test Kruskal-Wallis con los resultados de los algoritmos que resuelven el resto de los problemas, para determinar la existencia de diferencia estadísticamente significativa (Tabla 6). Se utiliza el signo (+) para especificar que existen

diferencias estadísticamente significativas entre los resultados obtenidos en los problemas y (-) en caso contrario.

Problema	Valor óptimo				Test ANOVA/ Kruskall-Wallis
	Cauchy	Uniforme	No Uniforme	Gaussiana	
Ackley	9,81E-04	0,016	0,013	**6,04E-04**	(+)
EF10	0,127	0,821	0,667	**0,117**	(+)
Griewangk	**9,16E-07**	0,010	0,014	1,95E-06	(+)
Rastrigin	1,78E-04	0,001	0,002	**1,16E-04**	(+)
Sle	6,47E-06	2,53E-05	**1,81E-06**	6,09E-06	(+)
Sphere	6,94E-07	7,60E-06	7,66E-06	**5,30E-07**	(+)

Tabla 6: Test de Kruskal-Wallis aplicado a cGA con diferentes operadores de mutación

Se puede observar que para todas las instancias ejecutadas existe diferencia estadísticamente significativa.
Se aplica entonces el test de Tukey, que nos permite determinar entre qué algoritmos las diferencias son estadísticamente significativas. Se muestran únicamente dos resultados representativos del conjunto analizado.
La Figura 3 muestra el test de Tukey aplicado a cGA, donde el cGA con mutación Cauchy posee diferencia estadísticamente significativa con las tres mutaciones restantes. Esto significa que podemos asegurar que el cGA con mutación Cauchy tiene mejor rendimiento promedio respecto a cGA con las otras tres mutaciones para el problema de Ackley.

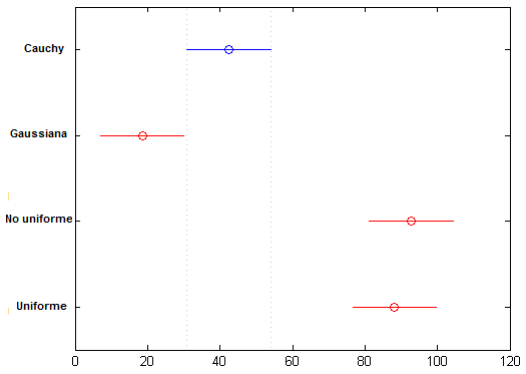

Figura 3: Test de Tukey aplicado a cGA con diferentes operadores de mutación para resolver el problema de Ackley

El test de Tukey reflejado en la Figura 4 para EF10, muestra que los resultados obtenidos por cGA con mutación Cauchy y Gaussiana tienen diferencia

estadísticamente significativa con respecto a los resultados obtenidos por cGA con la mutación Uniforme y No Uniforme.

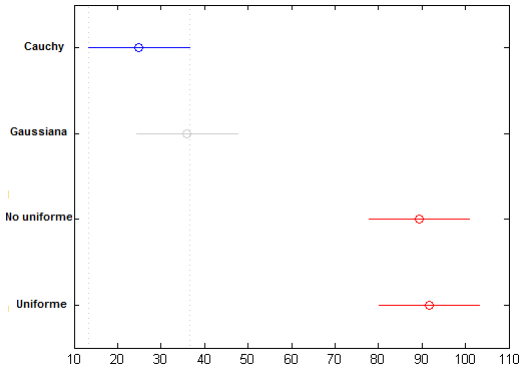

Figura 4: Test de Tukey aplicado a cGA con diferentes operadores de mutación para resolver el problema de EF10

En la Figura 5 se muestra en box-plot los resultados obtenidos por cGA con diferentes operadores de mutación, donde podemos ver como se distibuyen los valores a través de la mediana. Los resultados obtenidos por cGA con la mutación Cauchy y con mutación Gaussiana son más robustos (los valores están compactos y muy cercanos a la mediana) y bastante similares.

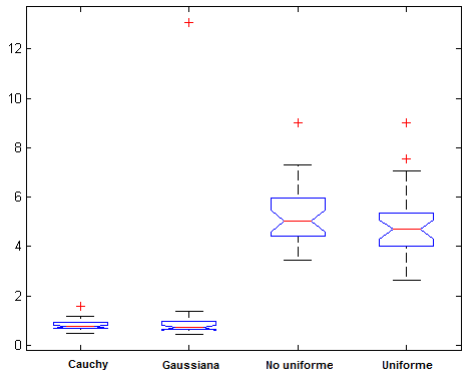

Figura 5: Boxplot de los resultados de cGA con diferentes operadores de mutación para resolver el problema de EF10

En el resto de los análisis realizados se observó un comportamiento similar, en el cual se obtienen diferencias estadísticamente significativas utilizando cGA con mutación Cauchy y Gaussiana con respecto a las otras mutaciones.

6. Conclusiones

En este trabajo se ha realizado el estudio del comportamiento de un cGA con cuatro operadores de mutación diferentes (Cauchy, Gaussiana, Uniforme y no Uniforme). Para ello se han seleccionado una serie de problemas de características diversas (Ackley, EF10, Griewangk, Rastrigin, Sle y Sphere). Si bien ninguno de los algoritmos ha llegado al óptimo, los valores obtenidos han sido muy cercanos al mismo. Debemos tener en cuenta que existen otros operadores y procesos dentro de un cGA que colaboran en el proceso de búsqueda. No obstante, realizando un profundo análisis estadístico sobre los algoritmos presentados, podemos confirmar con un 95 % de confianza que cGA con mutación Cauchy y cGA con mutación Gaussiana obtienen mejores resultados en los problemas analizados.
Como trabajo futuro se estudiarán otras componentes del cGA, en particular operadores de crossover, selección y reemplazo a fin de determinar la mejor configuración de los mismos.
Otro campo de investigación será también la hibridación de los operadores de mutación con el objeto de combinar las mejores características de cada uno y de esta manera fortalecer el proceso de búsqueda del cGA.

7. Agradecimientos

Se agradece la cooperación del equipo de proyecto del LabTEm y la Universidad Nacional de la Patagonia Austral, por el continuo apoyo y los valiosos aportes para el desarrollo de este trabajo.

Referencias

[1] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

[2] David Edward Goldberg et al. *Genetic Algorithms in Search, Optimization, and Machine Learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[3] Enrique Alba, Bernabé Dorronsoro, and Hugo Alfonso. Cellular memetic algorithms evaluated on sat. In *XI Congreso Argentino de Ciencias de la Computación (CACIC)*, 2005.

[4] Enrique Alba, Bernabé Dorronsoro, Francisco Luna, Antonio J Nebro, Pascal Bouvry, and Luc Hogie. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan manets. *Computer Communications*, 30(4):685–697, 2007.

[5] Enrique Alba and Bernabé Dorronsoro. *Cellular Genetic Algorithms*, volume 42. Springer, 2009.

[6] John H Holland. Adaption in natural and artificial systems. 1975.

[7] Kalyanmoy Deb and Mayank Goyal. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics*, 26:30–45, 1996.

[8] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[9] Kalyanmoy Deb and Others. *Multi-objective Optimization Using Evolutionary Algorithms*, volume 2012. John Wiley & Sons Chichester, 2001.

[10] David H Ackley. A connectionist machine for genetic hillclimbing. 1987.

[11] C. Graves K. Mathias D. Whitley, R. Beveridge. Test driving three 1995 genetic algorithms: New test functions and geometric matching. *Journal of Heuristics*, 1:77–104, 1995.

[12] Aimo Torn and Antanas Zilinskas. *Global Optimization*. Springer-Verlag New York, Inc., 1989.