UNA COMPARACIÓN DE ALGORITMOS EVOLUTIVOS PARA LA OPTIMIZACIÓN DE FUNCIONES MULTIMODALES

Alejandro C. Brero
Proyecto UNRC-188A[1]
*Area de Computación Facultad de Ciencias Exactas
Universidad Nacional de Río Cuarto
Ruta Nacional 8 Km. 601 (5800) Río Cuarto - Argentina
e-mail : abrero@dc.exa.unrc.edu.ar
T.E. 54 358 4676235

Raúl Gallard
Proyecto UNSL-338403[2]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 – Local 106 - Argentina
e-mail : rgallard@unsl.edu.ar
T.E. 54 358 4676235

RESUMEN

La Computación Evolutiva (CE) ha sido reconocida recientemente como un campo de investigación que estudia un nuevo tipo de algoritmos: los algoritmos evolutivos (AEs).

Opuestamente a los enfoques tradicionales que mejoran una única solución, estos algoritmos procesan poblaciones de soluciones y poseen como características comunes la reproducción, la variación aleatoria, la competición y la selección de individuos. Los algoritmos genéticos (AGs) y las estrategias evolutivas (EEs) constituyen hoy los algoritmos evolutivos de mayor uso en el campo de la optimización.

En este trabajo se comparan diferentes técnicas evolutivas para tratar la optimización de dos funciones altamente multimodales. Por un lado se utilizan dos tipos de Algoritmos Genéticos, uno con representación binaria y otro con representación en punto flotante; por otro lado se aplica el concepto de las Estrategias Evolutivas.

PALABRAS CLAVES: Algoritmos Evolutivos, Algoritmos Genéticos, representación, Estrategias Evolutivas, mutación delta.

[1] El grupo de investigación es apoyado por la Universidad Nacional de Río Cuarto.
[2] El grupo de investigación es apoyado por la Universidad Nacional de San Luis y la ANPCYT.

1 - INTRODUCCION

Durante las últimas décadas ha habido un interés creciente en los sistemas de resolución de problemas basados en el principio de la evolución de las especies. La selección natural (supervivencia del más apto) y la reproducción juegan un papel fundamental en la adaptación evolutiva del mundo biológico en que vivimos [11]. Los hijos heredan las características de sus progenitores, no obstante, ningún hijo es una copia perfecta de ellos. Así, pequeñas modificaciones genéticas producen variantes que permiten que las especies evolucionen adaptándose al mundo que las rodea. Se utiliza el término *fitness* para indicar el grado de habilidad/adaptación de un individuo para sobrevivir y reproducirse en un determinado ambiente [8].

En *computación evolutiva* se simula la evolución de una población en una computadora aplicando los conceptos de selección y reproducción a un conjunto de estructuras que representa el espacio de búsqueda de algún problema a resolver. Se pueden obtener resultados exitosos en una gran variedad de problemas altamente combinatorios donde fallan o no son prácticas otras alternativas, como por ejemplo: búsqueda random, hillclimbing, búsqueda exhaustiva, o cualquier otro algoritmo "razonable".

El término *computación evolutiva* se utiliza para abarcar a las clases de algoritmos basados en la adaptación evolutiva. Actualmente, estos algoritmos pueden clasificarse en: Algoritmos Genéticos (AGs), Estrategias Evolutivas (EEs), Programación Evolutiva (PE) y Programación Genética (PG) [6]. En nuestro trabajo nos ocuparemos de los dos primeros mencionados.

Algoritmos Genéticos: Cada punto del espacio de búsqueda es representado mediante un string, frecuentemente binario, de longitud fija. Utilizan dos operadores genéticos: *crossover* (cruzamiento) y mutación [4]. Usualmente se aplica un crossover a dos padres para combinar su información y obtener dos hijos. Ambos hijos son afectados también por el operador de mutación.

Una variante a la representación binaria es la representación en punto flotante, esta alternativa es muy conveniente cuando se desea optimizar funciones reales. En este caso, los individuos son representados, al igual que en las estrategias evolutivas, como vectores *n*-dimensionales de valores reales.

Estrategias Evolutivas: los algoritmos de este estilo fueron concebidos específicamente para tratar problemas de optimización de parámetros de funciones reales [1,10]. Funcionan muy bien cuando se desea encontrar los parámetros para los cuales una función de *n* variables reales adquiere su máximo/mínimo valor. Para optimizar una función de *n* variables, los individuos son representados mediante vectores *n*-dimensionales. En cada ciclo se seleccionan algunos padres para la etapa de reproducción. De cada padre se produce un hijo, donde los elementos del vector hijo son obtenidos a partir del vector padre mediante la aplicación de una variable con distribución Normal (mutación). Este tipo de procreación se basa en la reproducción biológica, en la cual se observa que los hijos son similares a los padres y que los cambios pequeños ocurren mas a menudo que los grandes.

En lo que sigue entenderemos por *optimización global* de una función a la tarea de encontrar una solución, dentro de un conjunto de soluciones denominado *espacio de búsqueda*, para la cual la función obtiene su mejor valor (máximo o mínimo, dependiendo del problema en cuestión). Un óptimo local será aquel para el cual la función obtiene su mejor valor en cierto entorno del espacio de búsqueda. Las funciones multimodales son aquellas que poseen numerosos óptimos (locales o globales). En general, este tipo de funciones son *difíciles* en el sentido en que no se puede garantizar una solución óptima global en un número finito de pasos [11]. Muchos métodos de búsqueda fallan

al momento de tratar de optimizar esta clase de funciones. Una de las alternativas que existe para tratar la *dificultad* es relajar el requisito de encontrar el máximo global exacto, en cambio se intenta obtener una solución aproximada. Al tomar éste tipo de decisión es bueno proveer de resultados que informen sobre la bondad de las soluciones obtenidas.

En este trabajo se trata la optimización de dos funciones altamente multimodales mediante Algoritmos Genéticos con representación binaria y en punto flotante como así también mediante el enfoque de Estrategias Evolutivas.

2 - TIPOS DE AES ESTUDIADOS EN ESTE TRABAJO

Existe una variedad potencialmente grande de estos tipos de algoritmos, dependiendo de la representación escogida, métodos de selección, tipos de crossover y mutación, elitismo/no-elitismo, selección generacional o "al vuelo", entre otras consideraciones [7]. En esta sección se explican los dos tipos de AGs usados y las EEs implementadas. También se describen los operadores genéticos, los métodos de selección y los parámetros de cada tipo de algoritmo.

2.1 - AGS BINARIOS

Estos algoritmos también son denominados "puros" cuando se utilizan operadores genéticos tradicionales, por haber sido originalmente concebidos por [11].

Representación del espacio de búsqueda: utilizan representación binaria. Los operadores genéticos (crossover y mutación) actúan sobre secuencias, de longitud fija, de ceros y unos llamadas *cromosomas*. Cada elemento de la secuencia es un *gen* del cromosoma. A su vez, cada cromosoma representa un *individuo* o elemento del espacio de búsqueda.

Operadores genéticos: En este trabajo se utilizaron los operadores tradicionales: crossover con un punto de corte por pareja de padres y mutación binaria (*bit swap*) aplicada a cada par de hijos inmediatamente después del crossover.

Métodos de selección: se utilizaron tres métodos clásicos: *Proporcional* (o Ruleta) [4], y los propuestos por [2] *Ranking Lineal* y *Ranking no-Lineal* . Son aplicados de manera *generacional*, es decir: el conjunto de padres permanece fijo hasta haberse generado todos los hijos que conformarán la siguiente generación. El algoritmo es *elitista* [3], esto es: el mejor individuo de cada generación es introducido automáticamente en la población de la siguiente generación. De esta manera se preserva la mejor solución encontrada hasta el momento.

Parámetros: las variables *pcross* y *pmut*, probabilidad de aplicar crossover y mutación respectivamente, fueron seteadas con diferentes valores en cada corrida. Al utilizar los métodos de selección ranking lineal y no-lineal se introduce un nuevo parámetro: *qrank*. La variable *popsize* indica el tamaño de la población, en todos los casos el algoritmo se ejecutó con una población de 50 individuos.

En principio, cualquier problema de búsqueda podría ser resuelto con un AG-Binario, basta con codificar cada elemento del espacio como un string de ceros y unos. Así, los AG puros pueden ser concebidos como herramientas genéricas capaces de resolver una variada y extensa gama de problemas. No obstante, esta gran independencia de dominio hace que los AG-Binarios no sean muy eficientes para un problema determinado.

2.2 - AGs Punto Flotante

La necesidad de incorporar conocimientos específicos de un problema dentro del algoritmo ha llevado a obtener variantes que se especializan en problemas particulares, en algunas bibliografías estos algoritmos se denominan *híbridos*. Principalmente se trata de utilizar, para la representación de los individuos, una estructura de datos y operadores genéticos mas *naturales*, acordes al problema a resolver. Es claro que mientras más conocimiento del problema es introducido dentro del algoritmo, más eficiente será su funcionamiento, pero como contrapartida, se verá limitada la aplicabilidad del algoritmo a otros problemas. Esta observación, evidencia las ventajas y desventajas de un AG puro respecto de uno híbrido y viceversa.

Representación del espacio de búsqueda: En particular, cuando se trata de optimizar una función de n variables reales, es conveniente representar a cada individuo x como un vector de n números reales, $x=(v_1,...,v_n)$. El vector en sí es llamado *cromosoma* y cada elemento del vector se denomina *gen*. En lo sucesivo y debido a esta representación, estos algoritmos serán referenciados como AG-Flotantes.

Operadores genéticos: Los operadores han de ser modificados a fin de adaptarlos a la representación escogida (espacio real) [7]. Se experimenta aquí con dos tipos de crossover y dos tipos de mutación distintos a los usados en el AG-Binario:

-*Crossover Aritmético:* este operador es definido como una combinación lineal de dos vectores: v^t y w^t dan origen a dos hijos $v^{t+1}=a\cdot w^t + (1-a)\cdot v^t$ y $w^{t+1}=a\cdot v^t + (1-a)\cdot w^t$. El valor a es escogido al azar en el rango [0,1) en cada aplicación del operador.

-*Crossover Simple:* definido igual que el crossover binario pero los puntos de corte permitidos son entre las variables v'_s para un cromosoma $x=(v_1,...,v_n)$.

-*Mutación Uniforme:* en un cromosoma $x^t=(v_1,...,v_n)$ cada v_k puede ser mutado por un valor v'_k escogido al azar dentro del dominio del k-ésimo parámetro.

-*Mutación No-Uniforme:* si en un cromosoma $x^t=(v_1,...,v_n)$ el elemento v_k con dominio $[l_k,u_k]$ fué elegido para ser mutado, entonces se obtiene un nuevo vector $x^{t+1}=(v_1,...,v'_k,...,v_n)$ donde:

$$v'_k = v_k+\Delta(t,u_k-v_k) \quad \text{si } r=0$$
$$v'_k = v_k-\Delta(t,v_k-l_k) \quad \text{si } r=1$$

siendo $r\in\{0,1\}$ un valor obtenido al azar

donde $\Delta(t,y)=y\cdot(1-r^{(1-t/T)b})$, siendo T el número máximo de generaciones y b un parámetro del algoritmo que determina el grado de no-uniformidad. Notar que $\Delta(t,y)\in[0,1]$.

Métodos de selección y Parámetros: Se utilizaron los mismos que se aplican en el AG-Binario antes descripto. También aquí se considera elitismo.

2.3 - Estrategias Evolutivas

Existen diferentes variantes de EEs. Particularmente, al considerar las variables *tamaño de la población*, *cantidad de hijos engendrados*, *selección tipo generacional* o *al vuelo*, surge la siguiente clasificación:

-($\mu+\lambda$)-EE: Una población de μ individuos engendra λ hijos, se obtiene una nueva generación reduciendo la población de $\mu+\lambda$ individuos a μ individuos mediante algún método de selección. Los hijos compiten con sus padres por la supervivencia, esto es una forma de selección *al vuelo*.

-(μ,λ)-EE: Una población de μ individuos engendra λ hijos, con $\lambda>\mu$. La nueva generación se obtiene al reducir la población de λ hijos a una de tamaño μ. Los padres sólo subsisten durante una generación, los hijos compiten entre si, esto es selección *generacional*.

Los casos particulares (1+1)-EE y (μ+1)-EE son llamados: *two-membered* ES y *multimembered ES* respectivamente. En este trabajo se empleó únicamente la variante (μ+λ)-EE con diferentes valores para μ y λ.

Representación del espacio de búsqueda: Un individuo está representado por un par (x,σ), donde $x=(x_1,...,x_n)$ es un punto del espacio de búsqueda y $\sigma=(\sigma_1,...,\sigma_n)$ son las desviaciones estándares usadas por el operador de mutación.

Operadores genéticos: Como se menciona en la sección introductoria, las estrategias evolutivas utilizan mutación como el principal operador genético. Nosotros consideramos aquí aquellas EEs que usan la mutación como único operador genético. Las dos componentes del par son mutadas por separado, el vector x por un lado y el vector de desviaciones por otro.

Mutación del vector x: Este operador se utiliza para obtener un hijo v^{t+1} a partir de un padre v^t. Dado $v^t=(\sigma^t,x^t)$ se genera un hijo $v^{t+1}=(\sigma^t, x^t+ N(0,\sigma^t))$
siendo N un vector de números aleatorios independientes Gausianos con media cero y una desviación estándar σ^t.

Mutación del vector σ: En este trabajo se realizaron experimentos mutando el vector σ con dos métodos diferentes:

-*Variante de la Regla 1/5:* los vectores σ de todos los individuos son actualizados cada cinco generaciones según la siguiente función obtenida de la "regla del 1/5" propuesta por [9]:

$$\sigma^{t+1} = \begin{cases} c_d \cdot \sigma^t \text{ , si } \varphi \leq 1 \\ c_i \cdot \sigma^t \text{ , si } \varphi > 1 \end{cases}$$

donde φ cuenta la cantidad de mutaciones exitosas ocurridas durante las últimas cinco generaciones. Los valores c_d y c_i son parámetros del sistema.

-*Mutación Delta:* Nuestro trabajo introduce este nuevo tipo de mutación. Aquí los vectores σ son actualizados en cada generación mediante la siguiente fórmula:

$$\sigma^{t+1} = \Delta(t,\sigma_I)$$ donde σ_I es un parámetro del algoritmo

la función Δ es la misma que la usada por la mutación no uniforme en el AG-Flotante antes descripto.

Observaciones: Los parámetros c_d, c_i para el método *Variante de la Regla 1/5* y el parámetro b para la mutación delta controlan de alguna manera la velocidad de reducción del vector σ. Esto es muy importante puesto que al inicio se requiere un σ lo suficientemente grande como para que el algoritmo explore bien el espacio de búsqueda, mientras que en las últimas generaciones σ debe ser lo suficientemente pequeño como para poder explotar al máximo algunos individuos y así satisfacer las exigencias de precisión impuestas.

Método de selección: Una vez que los μ padres generan λ hijos, se seleccionan los μ mejores individuos de la población μ+λ. Un individuo es mejor que otro si su *fitness* es mayor.

Parámetros: Tal como se muestra en las tablas, se utilizaron distintos valores para μ y λ. Los valores para las variables c_d, c_i, b y σ_I fueron establecidos especialmente para cada una de las funciones a optimizar luego de algunos testeos y análisis previos.

3 - EXPERIMENTOS Y RESULTADOS

Para cada función a optimizar, los tres algoritmos fueron ejecutados en series de 100 corridas cambiando adecuadamente sus parámetros. A continuación se muestran las dos funciones utilizadas:

a) Función de Schaffer

$$f_a(x,y) = \frac{sin^2(\sqrt{(x+\alpha)^2+(y+\beta)^2})-0.5}{[1+0.001*((x+\alpha)^2+(y+\beta)^2)]^2}$$

$-100 \le x \le 100$, $-100 \le y \le 100$,

α= -12.000000149 , β=6.9999996834

f_a ($-\alpha,-\beta$)=1 es el máximo de f_a

b) Función de Michalewicz

f_b (x,y)= $21.5+x*sin(4*\pi*x)+ y*sin(20*\pi*y)$ $-3 \le x \le 12.1$, $4.1 \le y \le 5.8$,

El máximo de f_b es 38.8502944795

En lo sucesivo se utilizará la siguiente terminología:

-*Corrida:* a la ejecución completa del algoritmo con determinados parámetros sobre una función dada.

-*Caso de Testeo:* Dado un algoritmo y sus parámetros, se denomina *Caso de Testeo* al resultado obtenido luego de reiterar 100 veces la ejecución del algoritmo manteniendo fijos sus parámetros.

-*Solución q-óptima:* Es aquella solución que difiere del óptimo conocido en a lo sumo un valor ε.

-*Hit:* Se dice que una corrida tuvo un *hit* si el algoritmo obtuvo una solución óptima, esto es hallar un individuo que se encuentre a una distancia menor o igual a un cierto valor ε del óptimo. El valor ε indica el grado de precisión que se exige al algoritmo.

Algunos parámetros fueron fijados con idénticos valores para los tres algoritmos**.**

-*Población inicial:* los individuos de la primera generación son escogidos al azar con una distribución uniforme.

-*Criterio de parada:* la ejecución culmina cuando se encuentra una solución q-óptima o bien cuando se alcanza un máximo de 600 generaciones (iteraciones).

-*Precisión exigida:* se fijó un valor $\varepsilon=10^{-6}$ para la función a) y $\varepsilon=10^{-9}$ para la función b)

-*Popsize:* En los AGs se opera con una población fija de 50 individuos. En las EEs se realizaron testeos con distintos valores para μ (1 y 5) y λ (1, 20 y 40).

Se definen a continuación una serie de variables de performance que permiten realizar una comparación más objetiva.

-*Hits:* cantidad de veces que el algoritmo encontró una solución q-óptima en 100 corridas. Este dato señala el porcentaje de veces que el algoritmo encuentra una solución satisfactoria.

-*Gbest:* para una corrida indica en qué número de generación se encontró la solución q-óptima. Si el algoritmo no encontró ninguna se considera Gbest = ∞.

-*Ebest:* indica el porcentaje de error entre el *fitness* del mejor individuo (F_{best}) encontrado en una corrida y el *fitness* del óptimo global (F_{opt}).

$$Ebest = (F_{opt} - F_{best})*100/ F_{opt}$$

-*EPop:* indica el porcentaje de error entre el *fitness* medio de los individuos de la última generación (F_{pop})y el *fitness* del óptimo global.

$$Epop = (F_{opt} - F_{pop})*100/ F_{opt}$$

-*MGbest*, *MEbest* y *MEpop*, indican los valores medios de las variables de performance arriba definidas obtenidos a lo largo de 100 corridas.

Dada la similitud de comportamiento de los algoritmos evidenciada sobre ambas funciones se mostrarán aquí sólo aquellos obtenidos respecto de la función de Michalewicz antes definida.

3.1 - EXPERIMENTOS CON LOS AGS

A continuación se comparan los resultados de las variables de performance obtenidos por el AG-Binario y el AG-Flotante. Inicialmente se realizaron pruebas con selección proporcional, tanto en el AG-Binario como en el AG-Flotante que arrojaron resultados poco exitosos. En el AG-Flotante, se realizaron varias pruebas combinando los dos tipos de crossover y los dos tipos de mutación antes descriptos. La combinación Crossover Aritmético con Mutación No-Uniforme resultó ser significativamente mejor que las otras. Estos resultados, por razones de espacio, no son aquí discutidos en detalle. Posteriormente en cada testeo se variaron los valores de *pcross* y *pmut* tratando de optimizar estos parámetros a fin de obtener los mejores resultados. Se realizaron 25 experimentos, explicitados en tabla 1, cuyos resultados en términos de las variables de performance están resumidos en las figuras 1 a 4. De la observación de las mismas se pueden extraer las siguientes conclusiones:

En general, el Ranking No-Lineal fue mejor que el Ranking Lineal según los valores de Hits obtenidos en ambos enfoques (fig. 1). Para ambas funciones, en la mayoría de los testeos, la cantidad de Hits obtenidos con el AG-Flotante es notablemente mayor que con el Binario. Consecuentemente MEbest (fig. 2) es también es mejor en AG-Flotante. Analizando la variable MEpop (fig. 3), se observa que el AG-Flotante logra evolucionar a la población de manera tal que los individuos de la ultima generación se encuentran, en promedio, muy cerca del optimo. El AG-Binario es menos eficiente en este sentido. La figura 4, muestra que el AG-Binario obtiene su solución q-óptma en menos generaciones que el AG-Flotante pero, en general, con peor valor de *fitness*.

Análisis más detallados indican que el AG-Binario tarda entre 2 y 5 veces más en computarse que el Flotante. En caso de requerir mas precisión en la optimización, en el AG-Binario se debe aumentar adecuadamente la longitud del cromosoma. Al hacer esto, los operadores genéticos tienen que manejar estructuras mas grandes, esto conduce a una demora en la ejecución del algoritmo. En este sentido, una ventaja que posee un AG-Flotante es que la precisión exigida puede ser aumentada sin necesidad de modificar el algoritmo y, por otro lado, el costo computacional, en cuanto a tiempo, no se ve afectado. La máxima precisión posible dependerá del hardware sobre el cual es ejecutado el algoritmo.

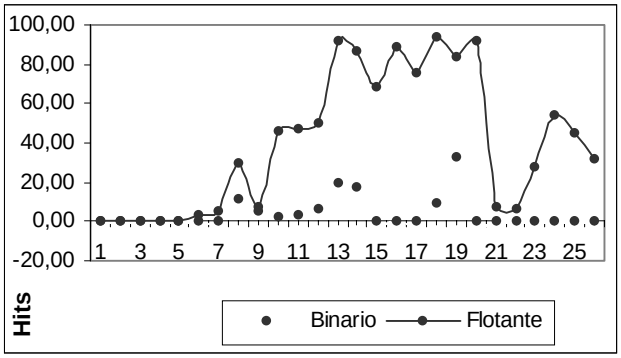

Fig. 1. Valores de Hits en cada experimento

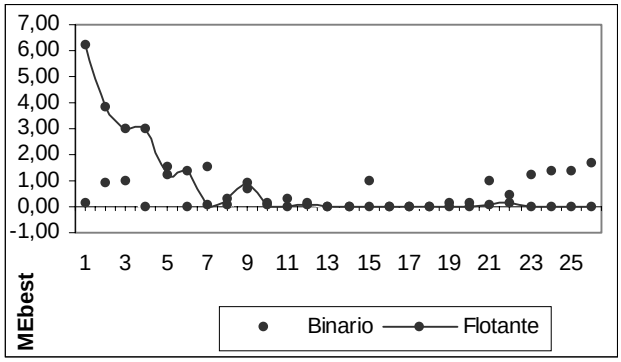

Fig. 2. Valores de MEbest en cada experimento

#Test	Selecc.	*qRank*	*Pcross*	*Pmut*
1	Rank	1,5	0,65	0,1
2	Rank	1,5	0,65	0,5
3	Rank	2	0,65	0,5
4	Rank	2	0,65	0,1
5	Rank	1,5	0,25	0,5
6	Rank	1,5	0,25	0,1
7	Rank	2	0,25	0,5
8	Rank	2	0,25	0,1
9	Rank	2	0,25	0,05
10	RankNL	0,1	0,25	0,1
11	RankNL	0,2	0,25	0,1
12	RankNL	0,07	0,25	0,1
13	RankNL	0,1	0,65	0,1
14	RankNL	0,1	0,45	0,1
15	RankNL	0,1	0,65	0,5
16	RankNL	0,1	0,8	0,1
17	RankNL	0,1	0,9	0,1
18	RankNL	0,1	0,725	0,1
19	RankNL	0,1	0,725	0,05
20	RankNL	0,1	0,725	0,3
21	RankNL	0,1	0,65	0,8
22	RankNL	0,1	0,65	0,9
23	RankNL	0,15	0,65	0,65
24	RankNL	0,12	0,65	0,65
25	RankNL	0,1	0,5	0,65

Tabla 1. Listado de experimentos en AGs

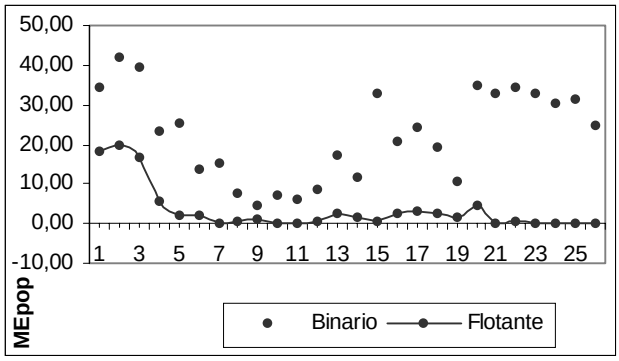

Fig. 3. Valores de MEpop en cada experimento

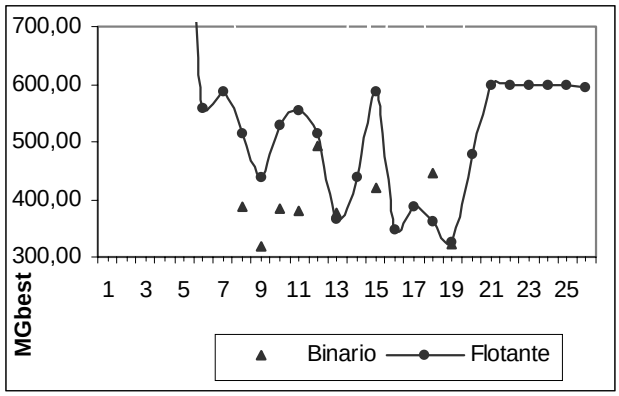

Fig. 4. Valores de MGbest en cada experimento

3.2 - EXPERIMENTOS CON LAS EES

Aquí se comparan los dos tipos de mutación antes propuestos: Regla del 1/5 y Mutación Delta. Después de algunos testeos previos se observó que los mejores valores para los parámetros c_d, c_i, σ_I y b eran aquellos con los cuales se obtenía un valor de MGbest cercano a 540. Se pudo comprobar que con las parametrizaciones en que MGbest arrojaba valores menores que 500, el algoritmo convergía demasiado rápido, impidiendo así aprovechar el máximo de generaciones disponibles.

Así mismo, si MGbest era mayor que 540, entonces muchos individuos resultaban cercanos a la solución q-óptima pero no podían llegar a ésta con la precisión requerida. Teniendo en cuenta estos hechos, después de algunas pruebas se fijaron los valores adecuados para estas variables. Estos valores fueron obtenidos en forma individual para cada función y se detallan en tabla 2:

	Regla del 1/5		**Mutación Delta**	
	cd	***ci***	σ_I	***b***
función f_a	1.07	.93	(20,20)	2.5
función f_b	1.125	.889	(10,10)	5

Tabla 2. Valores paramétricos en las EEs

#Test	μ	λ
1	1	1
2	1	20
3	1	40
4	5	40

Tabla 3. Listado de experimentos en EEs

Para cada tipo de mutación se realizaron cuatro experimentos, con diferentes valores para las variables μ y λ, según se especifica en la tabla 3.

Los resultados, en términos de las variables de performance, están resumidos en las figuras 5 a 8. De la observación de las mismas se pueden extraer las siguientes conclusiones:

Según la variable Hits, el método de mutación delta es superior a la regla de 1/5 en la mayoría de los testeos para ambas funciones. Más aún, en el test No. 4 obtiene un 100% de éxitos (fig.5).

Para el test 1 los valores de MEbest y MEpop son altos, entre el 10 y el 12%. Para los testeos 1,2 y 3 MEBest y MEpop son mejores en el método de la regla del 1/5 que en el de la mutación delta. En el test 4 ambos valores son muy bajos pero mejores en el caso de mutación delta. La figura 8, muestra que la EE con la mutación de la regla de 1/5 obtiene su solución q-óptma en menos generaciones que la que utiliza la mutación delta pero, en general, con peor valor de *fitness*.

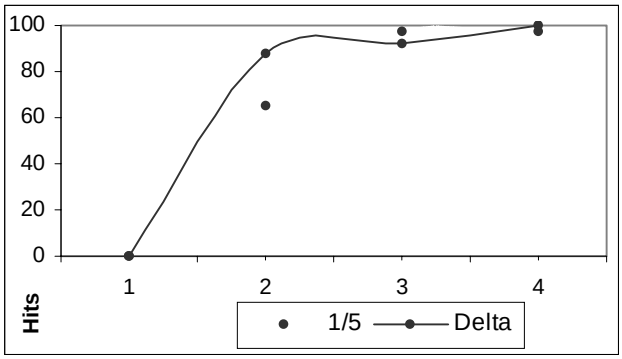

Fig. 5. Valores de Hits en cada experimento de EEs

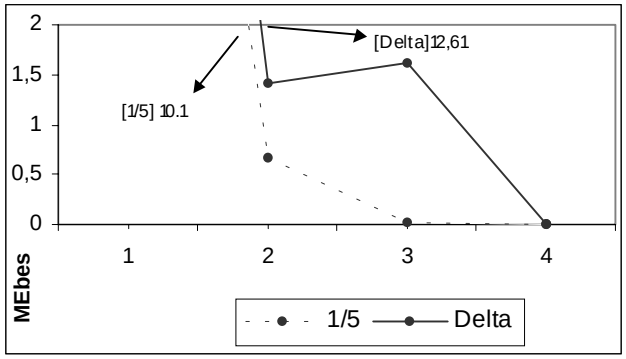

Fig. 6. Valores de MEbest en cada experimento de EEs

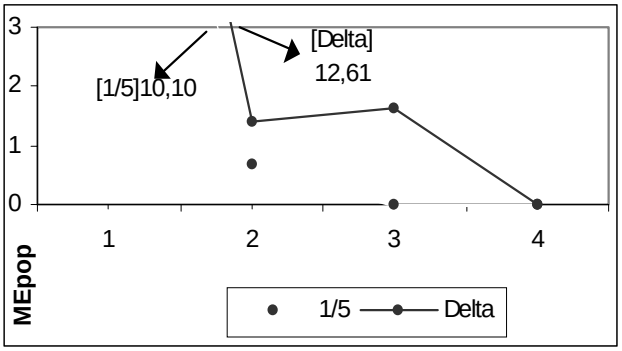

Fig. 7. Valores de MEpop en cada experimento de EEs

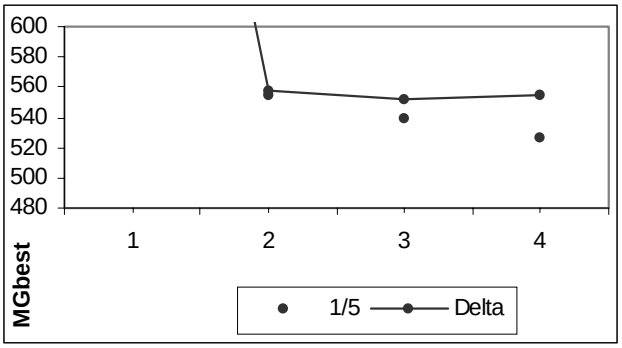

Fig. 8. Valores de MGbest en cada experimento de EEs

4 - CONCLUSIONES FINALES

El AG-Binario resulta ser una herramienta demasiado genérica, muy independiente de dominio. Otros métodos como AG-Flotante y las EE son muchos mas eficaces al momento de resolver problemas de optimización de funciones reales. Los AG-Flotantes son, en realidad, una especialización de los AG-Binarios y se comportan mejor que aquellos en los problemas abordados aquí.

Las Estrategias Evolutivas fueron ideadas desde su origen para aplicarlas específicamente a problemas de optimización. Las EEs mostraron ser, en las pruebas realizadas en este trabajo, mejores que los AG-Flotantes, logrando incluso un 100% de éxito utilizando una parametrización adecuada. Además, requieren menos tiempo de ejecución y los parámetros del algoritmo son más fáciles de configurar para obtener los mejores resultados.

El trabajo futuro incluye el estudio de nuevos algoritmos evolutivos que utilicen conjuntamente características propias de los dos enfoques aquí mostrados.

5 – AGRADECIMIENTOS

Agradecemos la cooperación del grupo de proyecto por la provisión de ideas nuevas y críticas constructivas. Asimismo a la Universidad Nacional de Río cuarto, la Universidad Nacional de San Luis y la ANPCYT de quienes recibimos continuo apoyo.

6 - REFERENCIAS

[1] Bäck, T., Hoffmeister, F., and Schwefel, H.–P., - *A Survey of Evolution Strategies* – pp. 2-9.

[2] Baker J. E. - *Adaptive selection methods for genetic algorithms* - . Proc. of the 1st International Conf. On Genetic Algorithms and Their Applications, Lawrence Erlbaum Associates, 1985.

[3] De Jong K. A. - *Analysis of the Behavior of a Class of Genetic Adaptive Systems* - PhD Dissertation, University of Michigan, 1975.

[4] Goldberg, D.E., - *Genetic Algorithms in Search, Optimization and Machine Learning* - , Addison-Wesley, Reading, MA, 1989.

[5] Holland J. – *Adaptation in Natural and Artificial Systems* – Ann Arbor, MI: University of Michigan Press. 1975.
[6] Kinnear, Kenneth E. Jr. - *Advances in Genetic Programming* – Bradford Book, MIT Press, Cambridge, Massachusetts, 1994.
[7] Michalewicz Z. - *Genetic Algorithms + Data Structures = Evolutions Programs* -Springer-Verlag, third, revised edition, 1996.
[8] Mitchell, Melanie – *An Introduction to Genetic Algorithms* - Bradford Book, MIT Press, Cambridge, Massachusetts, 1997.
[9] Rechenberg, I., - *Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* - Fromman – Holzboog Verlag, Stuttgart, 1973.
[10] Schwefel H.-P: *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.
[11] Törn, Aimo A. - *Global Optimization* - Akademi University DataCity [FIN-20520 Åbo Finland].