Tarea 9 - Optimización estocástica

Erick Salvador Alvarez Valencia

CIMAT A.C.,
erick.alvarez@cimat.mx

Resumen En el presente reporte se hablará sobre los resultados obtenidos en la tarea 9 de optimización estocástica, en donde se analiza el comportamiento del algoritmo de evolución diferencial en algunos aspectos de su ejecución y de la misma forma se hacen algunas comparaciones con el algoritmo genético en donde se quiere ver el cómo trabajan estos dos algoritmos con el conjunto de funciones que se han venido probando desde hace algunas tareas.
Para finalizar se darán algunas conclusiones sobre los ejercicios trabajados.

Keywords: Algoritmo genético, Evolución diferencial, cruce, mutación.

1. Ejercicio 1 - Implementación de la evolución diferencial y realización de pruebas en contra del algoritmo genético

Para esta primera parte de la tarea se hizo la implementación de la evolución diferencial en su forma más estándar y se usó como operador de cruce el *DE/rand/1* el cual genera a los hijos en base a individuos que toma de manera aleatoria en la población. A continuación se muestra el pseudocódigo del algoritmo DE implementado.

Algorithm 1 Evolución diferencial.

```
1: procedure DE(popSize, iterMax, dim, CR, F)
2:     pop ← generatePop(popSize).
3:     for i = 1 → iterMax do
4:         newPop ← {}.
5:         for j = 1 → popSize do
6:             r1 ← rand(1, popSize, exclude = {j}).
7:             r2 ← rand(1, popSize, exclude = {j, r1}).
8:             r3 ← rand(1, popSize, exclude = {j, r1, r2}).
9:             r ← rand(0, 1).
10:            kRand ← floor(r * dim) + 1.
11:            newInd ← {}.
12:            for k = 1 → dim do
13:                p ← rand(0, 1).
14:                if p ≤ CR or k = kRand then
15:                    newInd[k] ← pop[r1].ind + F * (pop[r2].ind - pop[r3].ind).
16:                else
17:                    newInd[k] ← pop[j].ind.
18:            fEval ← f(newInd).
19:            if fEval ¡pop[j].fitness then
20:                newPop.append({newId, fEval}).
21:        pop = newPop.
22:    return son1.
```

En el algoritmo anterior se muestra el método de evolución diferencial usando cruza aleatoria con una diferencia. Podemos ver que en cada iteración principal se hace un ciclo que recorre toda la población, en donde se eligen tres indices aleatorios mutuamente excluyentes que servirán para generar el nuevo individuo, la generación de dicho individuo se basa en recorrer todas las variables y bajo una probabilidad de cruza se determina si generar dicha variable con el operador de cruza o heredarla directamente de su padre. La cruza se hace con la siguiente fórmula: $newInd_k = pop_{r_1} + F(pop_{r_2} - pop_{r_3})$ donde F es un parámetro que está entre cero y uno.
En la fórmula se involucran tres individuos de la población anterior, los cuales son elegidos mediante los tres índices mencionados anteriormente. Finalmente este nuevo de individuo se somete a un torneo con su padre para ver si se queda en la nueva población o es el padre el que se conserva. Al finalizar el ciclo interno tendremos una nueva población que sustituirá completamente a la anterior.

2. Ejecuciones y resultados del AG

En esta parte se mostrarán los resultados obtenidos al implementar la evolución diferencial para minimizar un conjunto de funciones, en general se usaron las mismas funciones que en la tarea anterior, hay que mencionar que la representación de los individuos se hace mediante números reales, a diferencia del

algoritmo genético que usa una codificación binaria, y por lo tanto la inicialización de los mismos se hace con números aleatorios dentro de un cierto intervalo. Las funciones a optimizar son las siguientes.

1. **Sphere** : $\sum_{i=1}^{d} x_i^2$
2. **Ellipsoid** : $\sum_{i=1}^{d} 10^6 \frac{i-1}{d-1} x_i^2$
3. **Zakharov** : $\sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0{,}5 i x_i^2)^2 + (\sum_{i=1}^{d} 0{,}5 i x_i^2)^4$
4. **Rosenbrock** : $\sum_{i=1}^{d-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2]$
5. **Ackley** : $-20 \exp\left(-0{,}2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)\right) + 20 + e$
6. **Griewangk** : $\sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos(\frac{x_i}{\sqrt{i}}) + 1$
7. **Rastrigin** : $10d + \sum_{i=1}^{d} [x_i^2 - 10cos(2\pi x_i)]$

Para cada una de estas funciones se hicieron pruebas tanto con la evolución diferencial como con el algoritmo genético usando las mismas condiciones para que estas pruebas no estuvieran tan desequilibradas. Al igual que la tarea anterior el número de evaluaciones se fijó en 150000, además, la probabilidad de cruza en ambos algoritmos se fijó en 0.9, la probabilidad de mutación en el algoritmo genético fue de $\frac{1}{180}$ donde 180 es el tamaño de la cadena de cada individuo, se usaron 150 individuos por población, además que la dimensión del problema continuó en 10 y el número de bits de cada variable en los individuos del algoritmo genético fue de 19, finalmente la cruza usada en el algoritmo genético fue la de dos puntos.
A continuación se muestran los resultados obtenidos de realizar 30 ejecuciones de la DE con los parámetros anteriormente descritos. En dichos resultados se muestran algunos valores estadísticos con respecto al error (mejor aptitud como V.A.) tales como media, mediana, desviación estándar, entre otros.

Cuadro 1: Resultados de la ED con cruza *ED/rand/1*

Función	Promedio del error	Desviación estándar del error	Mediana del error	Mejor aptitud	Peor aptitud
Sphere	0	0	0	0	2.22507e-308
Ellipsoid	0	0	0	0	2.22507e-308
Zakharov	0	0	0	0	2.22507e-308
Rosenbrock	0	0	0	0	2.22507e-308
Ackley	7.99361e-16	1.06581e-15	4.44089e-16	4.44089e-16	3.9968e-15
Griewangk	0.0168023	0.0118436	0.0147761	0	0.0517034
Rastrigin	0	0	0	0	2.22507e-308

En la tabla anterior podemos ver que en general arrojó resultados bastante buenos tras dejarse ejecutando 30 veces con 150 mil evaluaciones cada una. Para todas las funciones excepto la 5 y la 6 se lograron resultados casi perfectos, en donde en promedio el mejor valor de aptitud encontrado fue de cero, lo cual es el óptimo para todas estas funciones, aunque también se puede pensar que en general no es así, esto por el parámetro encontrado en la columna de peor aptitud que este no es cero pero es un número bastante bajo y por lo tanto al

promediar y sacar la desviación estándar el resultado se truncó a cero debido a unos pequeños errores numéricos, aunque ignorando esto los resultados fueron excelentes. En el caso de las funciones 5 y 6 no se logró esto pero de igual manera se llegó a resultados buenos, en la función Ackley en promedio el valor de aptitud rondaba en el orden de 10^{-16}, además podemos ver que la desviación estándar se encuentra de igual manera en este mismo orden de magnitud, por lo que los valores obtenidos siempre estaban muy cerca del promedio. En el caso de la función Griewangk se obtuvieron los peores resultados, ya que en promedio el mejor valor de aptitud encontrado estaba en el orden de 10^{-2}, esto se debe a que la función en general es muy dependiente de las variables y por lo tanto hubiera funcionado mejor si bajáramos el parámetro de probabilidad de cruza.

Cuadro 2: Resultados del AG con cruza de dos puntos

Función	Promedio del error	Desviación estándar del error	Mediana del error	Mejor aptitud	Peor aptitud
Sphere	5.23873e-05	7.53957e-16	5.23873e-05	5.23873e-05	5.23873e-05
Ellipsoid	1.95259e-09	6.70166e-20	1.95259e-09	1.95259e-09	1.95259e-09
Zakharov	4.47455	23.6173	2.75033e-07	5.96633e-08	131.637
Rosenbrock	18.254	31.1287	8.73756	7.0959	151.386
Ackley	0.000305487	2.18684e-15	0.000305487	0.000305487	0.000305487
Griewangk	0.00796986	0.00810645	0.0086336	7.68511e-06	0.0270588
Rastrigin	1.1548e-05	0	1.1548e-05	1.1548e-05	1.1548e-05

En la tabla anterior se aprecian los resultados obtenidos en las 30 ejecuciones del algoritmo genético para todas las funciones, en general podemos ver que lo obtenido es bueno pero no supera a los resultados arrojados por la ED, primeramente en la funcion de Rosenbrock no se pudo llegar el óptimo de la misma, y en la función Zakharov pese a que si se logró llegar a este valor, en promedio los valores de aptitud rondaban por el 4.5 y el peor valor en estas dos funciones está bastante lejano al óptimo. Hablando del resto de funciones podemos ver que se obtuvieron resultados buenos aunque no tanto como los obtenidos por la ED, podemos ver que funciones como la Sphere y Rastrigin encontraron sus mejores valores de aptitud con una magnitud de 10^{-5}, en cambio la Ackley y la Griewangk en promedio tuvieron valores de aptitud medianamente buenos, rondando valores de magnitud 10^{-2} y 10^{-3}.
Para concluir podemos decir que la evolución diferencial obtuvo resultados mucho mejores que el algoritmo genético al correrlos con casi las mismas condiciones.

3. Ejercicio 2

La reproducción clásica de ED, la que vimos en clase se le conoce como "$DE/rand/1$" pero se han propuesto varias variantes como "$DE/current\_to\_best/1$". Documentar las diferentes formas de reproducción más importantes (son 4 o 5) y dar un comentario sobre ella. Por ejemplo, qué se pretende que haga o cual ventaja presume tener cada variante.

Para la evolución diferencial, las diferentes estrategias de reproducción son elistadas de la forma *DE/x/y/z* usando x como el método de seleccionar el vector objetivo, y indica el número de diferencias vectoriales usadas y z indica el método de cruza usado. Como se menciona en la descripción del ejercicio, el método *DE/rand/1* ya se analizó y usó en las pruebas del ejercicio pasado, por lo cual se verán otros enfoques de reproducción. En las fórmulas consiguientes se usará la variable u para hacer referencia a los hijos y la x para los padres, además de que estas variables se definirán en función del tiempo (iteración actual).

1. **DE/rand/2** : la fórmula usada es $u_i(t) = x_{r_1}(t) + F(x_{r_2}(t) - x_{r_3}(t)) + F(x_{r_4}(t) - x_{r_5}(t))$. Este operador es muy parecido al *DE/rand/1* pero en este caso se añade otra diferencia de vectores pesada por el parámetro F, la idea es hacer más exploración metiendo la contribución de otros dos vectores diferentes.

2. **DE/best/1** : La fórmula usada es $u_i(t) = x_{Best}(t) + F(x_{r_1}(t) - x_{r_2}(t))$. Este operador es también muy parecido al *DE/rand/1* pero en lugar de tomar un vector aleatorio del conjunto de individuos, se toma como referencia al mejor de la población actual. Y la idea principal de esto es generar a los hijos en la dirección que este vector valla tomando, dicha estrategia puede resultar en un enfoque muy greedy ya que no se da oportunidad a otros miembros de la población para generar hijos.

3. **DE/current-to-best/1** : La fórmula usada es $u_i(t) = x_i(t) + F(x_{Best}(t) - x_i(t)) + F(x_{r_1}(t) - x_{r_2}(t))$. Con esta estrategia el padre es mutado usando al menos dos vectores de diferencia ya que esta puede ser extendida a más valores aleatorios. Nota que una diferencia es calculada con el mejor vector y con el padre, la otra es calculada con dos vectores elegidos de manera aleatoria. Esto anterior implica que el vector base para la mutación denota un punto en la línea que une el vector objetivo y el mejor miembro de la población, esto es una recombinación aritmética entre $x_i(t)$ y $x_{Best}(t)$.

4. **DE/rand-to-best/1** : La fórmula usada es $u_i(t) = \gamma x_{Best}(t) + (1-\gamma)x_{r_1}(t) + F(x_{r_2} - x_{r_3})$. Con esta estrategia el vector resultante es calculado usando el mejor vector de la población actual y al menos dos vectores aleatorios (pueden ser mas). El parámetro $\gamma \in [0, 1]$ controla el nivel de *voracidad* del algoritmo, mientras más cercano esté γ de 1 más voraz este será ya que se le da más peso al mejor vector de la población, por el contrario, si este se acerca a cero, se favorecerá la explotación del operador. Una estrategia usada es trabajar con un γ adaptativo, empezando con $\gamma(0) = 0$ e ir aumentando de manera lineal en cada iteración hasta llegar a 1. De esta forma se tiene un operador exploratorio al principio y uno que realiza más búsquedas intensivas al final. Nota que si $\gamma = 0$ tenemos la reproducción *DE/rand/y/z* y por el contrario si $\gamma = 1$ tendremos la reproducción *DE/best/y/z*.

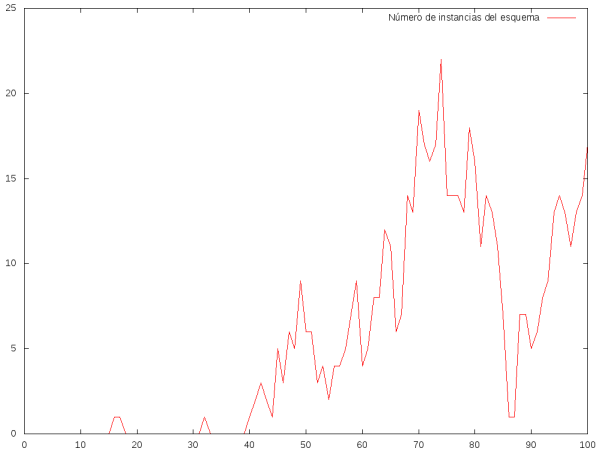
4. Ejercicio 3

Para este ejercicio se hizo un análisis en la parte de los esquemas que se van preservando a lo largo de las generaciones de individuos en el algoritmo genético. La idea fue trabajar con la función de la esfera en dos dimensiones y limitarnos a sólo 100 generaciones. Se hicieron 5 propuestas de esquemas que se pensó iban a tener una evolución interesante a lo largo de las generaciones del algoritmo genético y por cada generación se analizó a los individuos para verificar cuántas instancias de dichos esquemas estaban presentes, finalmente, si el esquema estaba cerca de la solución que debía generar el algoritmo genético, este iba a sobrevivir, en caso contrario iba a desaparecer. Para ilustrar esto se generó una gráfica por cada evolución del esquema que se iba monitoreando en esa corrida, la cual muestra en el eje X el número de generación actual, y en el eje Y el número de instancias contadas del esquema. A continuación se muestran los resultados obtenidos y para ello las siguientes configuraciones en el AG fueron usadas: *Población de 150 individuos. *Probabilidad de cruza de 0.8. *Probabilidad de mutación de $\frac{1}{36}$. *18 bits por variable. *Cruza de dos puntos.

1. **Esquema 1** : El primer esquema monitoreado es el siguiente:

   $$0 * * * *1111111111111$$

   Hay que notar que dicho esquema puede generar una de las dos soluciones a las que debe llegar el AG, la cual está formada por el primer bit como cero y los demás como uno, eso mapeado a los números reales representa el número cero, por lo cual se espera que tenga un alto grado de presencia a lo largo de las generaciones de individuos. Para poder generar los resultados con este esquema se tuvo la certeza de que el AG tomara el camino de la solución anteriormente descrita al menos para la primer variable. A continuación se muestra la gráfica generada por este esquema.



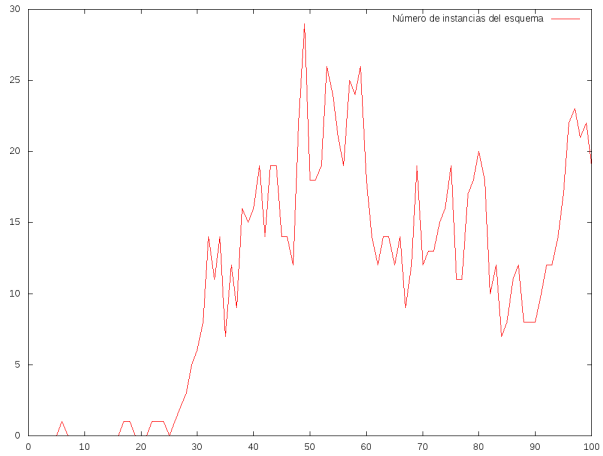
(a) Figura 1. Grafica generada por el primer esquema.

Se puede ver en la Figura 1. que dentro de las primeras 39 generaciones el esquema casi no tuvo presencia en la población del individuo pero posteriormente a esto la presencia del mismo fue creciendo poco a poco hasta llegar a un máximo de 23 instancias, luego hubo una caida bastante importante pero finalmente el esquema no murió y volvió a tener mayor presencia en las poblaciones hasta terminar con las 100 generaciones.

2. **Esquema 2** : El segundo esquema monitoreado es el siguiente:

$$100000000 * * * *000000$$

Al igual que el caso anterior, este esquema genera otra de las soluciones a las que debe llegar el AG la cual consiste en el primer bit en uno y los demás en cero, la diferencia aquí es el hecho de que los cuatro comodines se posicionaron en medio de la cadena. Para poder generar los resultados con este esquema se tuvo la certeza de que el AG tomara el camino de la solución anteriormente descrita al menos para la primer variable. A continuación se muestra la gráfica generada por este esquema.



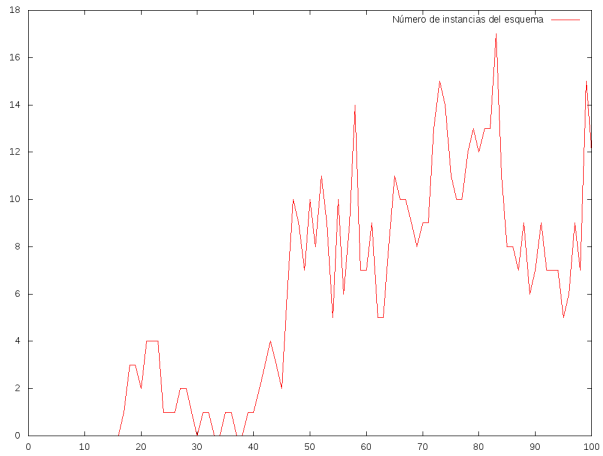
(b) Figura 2. Grafica generada por el segundo esquema.

Para este esquema podemos apreciar resultados algo parecidos al caso anterior, donde el mismo tuvo muy poca presencia en las primeras 25 generaciones pero después empezó a aumentar el número de instancias, llegando a un pico de casi 30 instancias en una sola generación, eso equivale al 20 % de los individuos. Luego se tuvo un pequeño descenso en el número de instancias entre las iteraciones 60 y 90 hasta que se llegó a un nuevo aumento de repeticiones y el esquema tomó valores relativamente altos en las últimas instancias, como se esperaba tuvo sobrevivencia.

3. **Esquema 3** : El tercer esquema monitoreado es el siguiente:

$$100000000 * * * *000001$$

Ahora se tomó un caso más interesante, el esquema estudiado genera individuos que están muy cerca de la solución pero no forman parte de ella, se propuso un esquema casi idéntico al anterior pero con un uno al final de la cadena y para ello se esperaba que tuviera presencia en la población aunque no tanta como los casos anteriores. A continuación se muestra la gráfica generada por este esquema.



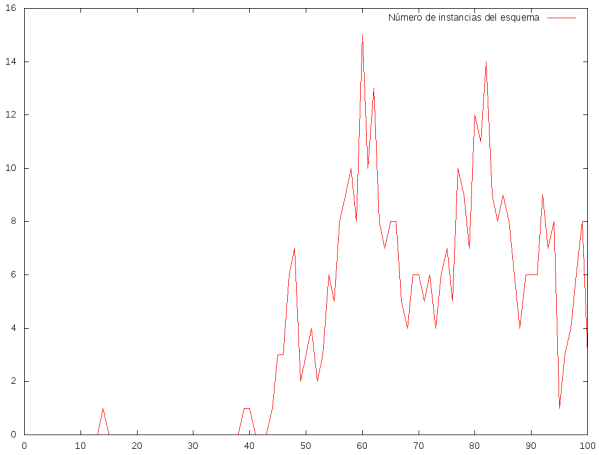
(c) Figura 3. Grafica generada por el tercer esquema.

Antes de analizar la gráfica hay que mencionar que la escala es diferente en comparación a las otras dos, ahora el eje Y va de 0 a 18. Posteriormente se puede analizar una presencia creciente de dicho esquema en la población, llegando a un pico de 17 instancias en la generación 80, posteriormente vemos que esta baja y luego vuelve a subir en las últimas generaciones. Aquí se puede pensar que hasta el punto de la generación 100 el AG no lograba encontrar a la serie de individuos que lograba la convergencia de la función y por ello este esquema seguía sobreviviendo.

4. **Esquema 4** : El cuarto esquema monitoreado es el siguiente:

$$0111 * * * *11111111110$$

Igual que el esquema anterior, la elección del presente se hizo con la misma idea, el esquema seleccionado es muy parecido al del caso uno pero el último dígito se cambió y por lo cual este esquema no puede generar individuos que sean solución a la función. A continuación se muestra la gráfica generada por este esquema.



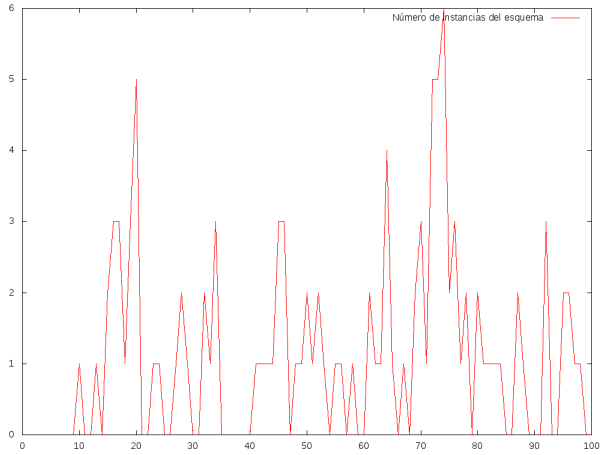
(d) Figura 4. Grafica generada por el cuarto esquema.

Se hace el mismo comentario que en el caso anterior de tener énfasis en la escala de la gráfica para no confundir los resultados con lo obtenido en los dos primeros casos. Para esta gráfica podemos ver resultados muy parecidos al caso anterior, como el esquema es muy parecido a una de las soluciones a las que debe llegar el AG este sobrevive a lo largo de las generaciones y como el algoritmo no logra llegar a la solución dentro de las 100 generaciones este no muere.

5. **Esquema 5** : El quinto y último esquema monitoreado es el siguiente:

$$0 * *111101111101 * **$$

Para este caso se eligió un esquema algo parecido a una de las soluciones pero ahora con mas errores que los dos anteriores y por ello se agregaron más comodines. Lo que se esperaría de este esquema es que tenga muy poca presencia en la población y poco a poco vaya desapareciendo. A continuación se muestra la gráfica generada por este esquema.



(e) Figura 5. Grafica generada por el quinto esquema.

Podemos ver en la Figura 5. que lo mencionado anteriormente es lo que se obtuvo con el experimento, el esquema tuvo muy poca presencia en las poblaciones llegando a un pico de 6 instancias en la generación 75 aprox. Lo interesante del caso es que este desaparece en algunas iteraciones y posteriormente vuelve a aparecer en las consiguientes. Esto anterior podría deberse al hecho de que está cerca de la solución a la que debe llegar el AG, además de que se colocaron más comodines que en los casos anteriores, por lo cual es viable suponer que el esquema tenga algunas instancias en ciertas generaciones.

5. Ejercicio 4

Teorema de los esquemas : Primeramente se hablará del concepto de esquema. Un esquema es un conjunto de símbolos que denotaremos como s_i los cuales están dentro de un alfabeto A el cual contiene los siguientes valores $A = \{1, 0, *\}$. Los dos primeros símbolos corresponden a los valores 1 y 0 que toman los individuos de la codificación del genético, el último caracter es un comodín que puede tomar cualquiera de los dos valores anteriores. Un ejemplo de esquema es el siguiente: *1100**0101*, lo que nos dice ese esquema es que pueden generar individuos que tengan la misma forma que el esquema y en donde hay comodines se pueden tomar ya sea 0 o 1. Algunos de las cadenas que puede generar el esquema anterior son *1100000101*, *1100110101*, *1100010101*, etc.
El teorema de los esquemas nos da una desigualdad la cual sirve como cota para ver cuál es la esperanza de supervivencia de un esquema, dicha cota es la que sigue:

$$E(m(H, t+1)) \geq \frac{\hat{u}(H,t)}{\hat{f}(t)} m(H,t)(1 - P_c \frac{d(H)}{L-1})(1-p_m)^{o(H)} \qquad (1)$$

Donde H es el esquema, $m(H,t)$ es el número de instancias del esquema H en la población de la generación t, $f(t)$ es el valor de aptitud promedio en la generación t, P_c y P_m son las probabilidades de cruza y mutación.
En general esta esperanza anterior lo que nos propone es una cota relacionada con la supervivencia de un esquema en una determinada generación, y para ello se consideran factores que podrían destruir un esquema como la probabilidad de cruza y la probabilidad de mutación.

Hipótesis de los bloques constructores : Ya se explicó brevemente qué es un esquema y el teorema de los esquemas, ahora para lo consiguiente se definirán dos conceptos relacionados con los esquemas:

1. **Orden de un esquema** : El orden es el número de caracteres en el esquema que no son comodines.

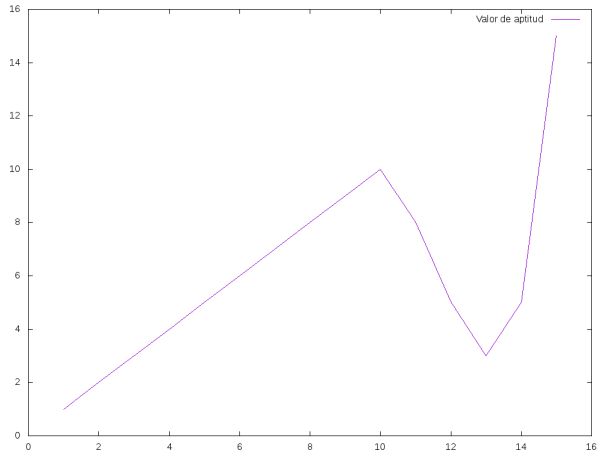
2. **Distancia de un esquema** : La distancia es el número de caracteres en el esquema que que hay entre el primer y el último comodín.

La hipótesis de los bloques constructores nos dice en primera instancia que los esquemas que cumplan con tener un bajo orden, una distancia muy corta y que tengan buen valor de aptitud son los que sobrevivirán a lo largo de las generaciones, y a estos esquemas los llaman bloques constructores debido a que son los que formarán a la solución final del AG.

El problema de la carretera real : Una vez que se formuló la hipótesis de los bloques constructores un conjunto de personas realizó un experimento para verificar la validez de esta hipótesis. El experimento consistía en generar una población inicial en el AG que viniera de los siguientes esquemas: $1111 * * * ... * *$, $* * * * 1111 * *... * *$, $* * *... * * 1111$ y como función a optimizar se propuso contar el número de unos en la cadena binaria, por lo tanto el óptimo de dicha función era la cadena que contuviera sólamente unos. La suposición en este problema era que como la población inicial era instancia de esquemas que cumplían lo que propone la hipótesis de los bloques constructores (bajo orden, distancia corta y buen valor de aptitud) entonces el AG iba a funcionar de una buena manera e iba a encontrar la solución óptima en muy poco tiempo, y para comprobar esto se puso a competir en contra de otros algoritmos como algunas búsquedas locales y los resultaron indicaron que el AG tuvo el peor desempeño en esta competencia, contradiciendo a la hipótesis de los bloques constructores. Esto anterior se debió a que en estos problemas el algoritmo le cuesta mucho converger ya que debe voltear algún caracter específicamente y esto anterior le cuesta varias generaciones.

Función engañosa (deceptive function) : Una función engañosa es una función fabricada artificialmente para hacer que el AG tenga una convergencia muy lenta y que en algunos casos no llegue al óptimo de la misma. Un ejemplo de esta función sería la que cuenta el número de unos en la cadena binaria y

en cierto punto de valores bajos de aptitud a números altos de bits, esto va a confundir al AG ya que este iba formando cadenas con muchos unos y en cierto momento tiene que empezar a cambiarlos por cero debido a que la función va a penalizar estos valores altos de unos, el óptimo de esta función será la cadena con todos los valores en uno. Dicha función se vería de la siguiente forma:



(f) Figura 6. Ejemplo de función engañosa.

6. Ejercicio 5

Para este ejercicio se hicieron pruebas con la evolución diferencial en donde se varió el parámetro de probabilidad de cruza para ver el efecto que este tenía con relación a las dependencias de las variables.
Las pruebas consistieron en hacer 10 ejecuciones de la ED fijando el número de generaciones a 150000, el intervalo de las variables a $\{-20, 20\}$, 10 y 30 como dimensión del problema y el valor de $F = 0{,}75$, lo único que varió fue la probabilidad de cruza de 0.1 a 0.9 variando en 0.1. Sólamente se probó esto anterior en 4 de las 7 funciones del conjunto. A continuación se detallan ciertas características de cada función y posteriormente se mostrarán los resultados obtenidos.

Función Sphere :

1. Multimodal: No.
2. Separable: Si.

Función Griewank :

1. Multimodal: Si.
2. Separable: No.

Función Rastrigin :

1. Multimodal: Si.
2. Separable: No.

Función Ackley :

1. Multimodal: Si.
2. Separable: No.

Primero se mostrarán los resultados obtenidos en dimensión 30.

Cuadro 3: Resultados de las ejecuciones en dimensión 30

	Sphere		Griewank		Rastrigin		Ackley	
CR	**Mejor aptitud**	**Peor aptitud**	**Mejor aptitud**	**Peor aptitud**	**Mejor aptitud**	**Peor aptitud**	**Mejor aptitud**	**Peor aptitud**
0.1	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	3.9968e-15	7.54952e-15
0.2	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	3.9968e-15	7.54952e-15
0.3	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	3.9968e-15	3.9968e-15
0.4	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	3.9968e-15	3.9968e-15
0.5	0	2.22507e-308	0	2.22507e-308	56.361	65.8819	3.9968e-15	3.9968e-15
0.6	0	2.22507e-308	0	2.22507e-308	81.4134	96.1929	3.9968e-15	3.9968e-15
0.7	1.15942e-216	1.77671e-215	0	2.22507e-308	93.2172	131.591	3.9968e-15	3.9968e-15
0.8	4.92061e-134	6.27763e-133	0	2.22507e-308	51.1937	100.576	3.9968e-15	3.9968e-15
0.9	1.14729e-88	4.42124e-87	0	2.22507e-308	2.98488	7.29193	3.9968e-15	3.9968e-15

En la Tabla anterior podemos ver en las columnas el mejor y peor valor aptitud obtenido por cada función con un cierto CR, lo primero que podemos notar es que para las funciones Sphere y Rastrigin se obtuvo mejores resultados con CR bajo, de hecho en las Rastrigin se empezaron a obtener resultados muy malos con un CR de 0.5 hacia arriba. En cambio las funciones Griewank y Ackley se mantuvieron igual para todas las corridas realizadas. Ahora se mostrarán los resultados obtenidos con pruebas de dimensión 10.

Cuadro 4: Resultados de las ejecuciones en dimensión 10

	Sphere		Griewank		Rastrigin		Ackley	
CR	**Mejor aptitud**	**Peor aptitud**	**Mejor aptitud**	**Peor aptitud**	**Mejor aptitud**	**Peor aptitud**	**Mejor aptitud**	**Peor aptitud**
0.1	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	3.9968e-15
0.2	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	3.9968e-15
0.3	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	4.44089e-16
0.4	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	4.44089e-16
0.5	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	4.44089e-16
0.6	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	4.44089e-16
0.7	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	4.44089e-16
0.8	0	2.22507e-308	0	2.22507e-308	0	2.22507e-308	4.44089e-16	4.44089e-16
0.9	0	2.22507e-308	0	0.0147724	0	2.22507e-308	4.44089e-16	4.44089e-16

En la Tabla 4 podemos ver los resultados obtenidos con las mismas funciones sólo cambiando la dimensión a 10, y en general vemos que para casi todos los valores de CR el mejor y peor valor de aptitud se conservó, esto podría justificarse por la menor dimensión en la que es tratado el problema es más fácil encontrar el óptimo.

7. Comentarios finales

En la presente tarea se analizó la implementación de la evolución diferencial estándar con cruza tipo *DE/rand/1* y se hicieron algunas pruebas usando el conjunto de funciones que se ha trabajado desde hace dos tareas. Se puede concluir que para este conjunto de funciones la evolución diferencial funcionó mucho mejor que el algoritmo genético, tanto en tiempo de ejecución como en resultados obtenidos.
Otro aspecto que se analizó en la tarea fue los esquemas que trabaja el AG a lo largo de sus generaciones. En base a las pruebas realizadas podemos concluir que los esquemas que generan a las cadenas de solución o cadenas que se parescan mucho a la solución tienden a sobrevivir mucho más que esquemas que no generen cadenas cercanas al óptimo de la función.

Referencias

1. John Wiley & Sons. *Computational Intelligence: An Introduction.* A.P. Engelbrecht, 2007.

2. Swagatam Das, Sankha Subhra Mullick, P. N. Suganthan. *Recent Advances in Differential Evolution - An Updated Survey.* Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata-700 108, India.

3. Swagatam Das, Ponnuthurai Nagaratnam Suganthan. *Differential Evolution: A Survey of the State-of-the-Art.* IEEE transactions on evolutionary computation, vol. 15, No. 1, February 2011.