Reporte

Alma Eliza Guerrero Sánchez.

1. Introducción

Los algoritmos genéticos como se saben son modelos computacionales inspirados por la evolución. Estos Modelos o algoritmos de alguna manera son la solución en potencia a algún problema con una estructura de datos similar a un cromosoma y aplicando operadores de recombinación para ampliar el espacio de búsqueda.

En la Optimización los algoritmos genéticos tienen un papel sobresaliente, dado a la rapidez de convergencia que se logra al realizar la aplicación de estos.

En la entrega anterior en la parte introductoria, se hizo mención de la definición de Optimización y así mismo se mencionó uno de los primeros algoritmos que se utilizó para optimización que fue el de Descanso Por Gradiente. Pero cuando el gradiente de la función objetivo no esta presente, se hace uso de otros algoritmos de Optimización como lo son Búsqueda Multidireccional, Nelder-Mead solo por hacer mención de algunos el uso de estos algoritmos en problemas donde el espacio de búsqueda no es tan amplio.

La mayoría de los problemas de optimización con implicaciones practicas en ciencias o ingeniería, tienden a ser complejos y difíciles de resolver y no pueden ser resueltos de manera exacta usando métodos de optimización clásicos como los antes mencionados. Por estas circunstancias los métodos de Algoritmos evolutivos se han ganado terreno como una solución alternativa para este tipo de problemas.

2. Marco teórico

El computo evolutivo es considerado una de las herramientas genéricas de optimización que pueden resolver problemas muy complejos caracterizados por tener un espacio de búsqueda demasiado grande, el computo evolutivo reduce el tamaño del espacio de búsqueda a uno mas efectivo y de esta manera resuelve los problemas de una manera más efectiva y rápida.

Como se ha hecho mención varias veces en entregas pasadas los algoritmos genéticos toman como inspiración la evolución de las especies para tratar de resolver problemas de diversa índole.

Los Algoritmos genéticos propuestos Por Holland y sus colegas[], se basaron en la teoría de autómatas como en los conceptos de adaptación de poblaciones artificiales de individuos[]. Holland y su equipo fueron los primeros en proponer operadores que permitieran la evolución o la mejora de cada una de las soluciones candidatas, estos operadores son la selección, cruza y mutación. Por motivos de este reporte haremos énfasis en la cruza.

La cruza de los seres vivos consiste en un intercambio complejo de información genética. Como se hace mención en el trabajo anterior. La cruza es un proceso que simula con el intercambio de información a nivel vectores, la información puede ser binarios, reales o de otro tipo.

Hasta ahora vimos cruza a un corte, a dos cortes y Partially Mapping Cross. Los primeros tipos de cruza son muy sencillos de aplicar pero cuando se refiere a grandes espacios de búsqueda no tienen esa diversidad genética que se requiere.

Por Otro lado tenemos Partially Mapping Crossover un operador de cruza que se usa en cadenas donde el Cromosoma (Individuo) contiene una cadena de genes muy largos y su espacio de búsqueda es muy grande por lo tanto debe de tener una diversidad genética muy variada para explorar todas las posibles soluciones.

El funcionamiento de PMX básicamente se divide en 3 pasos:

1.-Realizar de manera aleatoria dos cortes en el cromosoma tanto del padre 1 como del padre 2 estos formaran nuestra matriz de correspondencia o de mappeo.
2.-Seleccione las posiciones que no se copiaron del padre 1 al hijo 2 y las del padre 2 al hijo 1.
Para cada uno de estos valores:
- **i) Tener en cuenta el índice de este valor en Padre 2. Localizar el valor, V, de padre 1 en esta misma posición.**
- **ii) Localice este mismo valor en el padre 2.**
- **iii) Si el índice de este valor en el Padre 2 es parte de la franja original, vaya al paso i, usando este valor.**
- **iv) Si la posición no es parte de la franja original, inserte el valor del Paso A en el hijo en esta posición.**

3.- Copie cualquier posición restante del padre 2 al hijo.

Cycle Crossover es otro tipo de cruza, es un algoritmo que identifica una serie de ciclos llamados entre dos cromosomas parentales. Luego, para formar el Hijo 1, el ciclo uno se copia del padre 1, el ciclo 2 del padre 2, el ciclo

El principio básico de Cycle Crossover es que cada alelo viene de uno de los padres junto con su posición.

Divide los elementos en ciclos. Un ciclo es un subconjunto de elementos que tiene la propiedad de que cada elemento siempre aparece emparejado con otro Elemento del mismo ciclo cuando los dos padres están alineados. Cycle Crossover ocurre por seleccionar algunos ciclos de uno de los padres y los ciclos restantes del padre alternativo.

Todos los elementos en la descendencia ocupan las mismas posiciones en uno de los dos padres. Primero un Se crea el ciclo de alelos del padre 1. Luego los alelos del ciclo se ponen en el niño 1. Otros el ciclo se toma del padre 2 y el proceso se repite [Goldberg 1989, Sivanandam et al.2007].}

El procedimiento para crear ciclos es el siguiente:

- Comience con la primera posición no utilizada y el alelo en el primer padre (P1).
- Mire el alelo en la misma posición en el segundo padre (P2).
- Ir a la posición con el mismo alelo en P1.
- Añadir este alelo al ciclo.
- Repita estos pasos hasta que se alcance el primer alelo de P1.

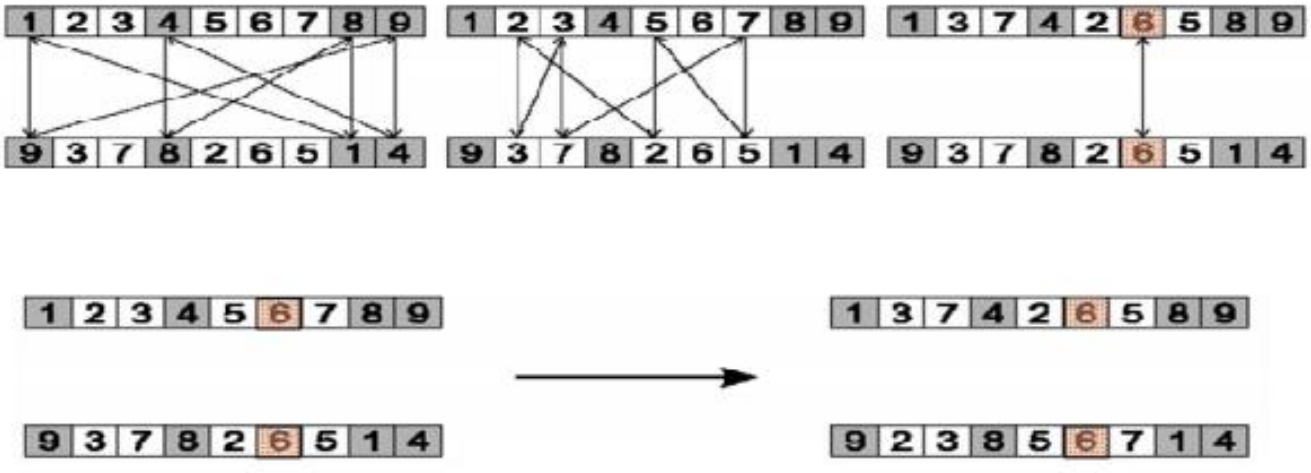

Figura 1.Procedimiento grafico de Cruza Cycle Cross

En resumen construye la descendencia de tal manera que cada ciudad (y su posición) proviene de uno de los padres.

3. Metodología

```
generacion=[números del 1 al 18, aleatorios]
sumatoria(generacion)
plot(genaracion)
  numeroDeGeneraciones=100 %50

selección=selecciónRanking(generacion)
        generacion= cruza(seleccion)
        plot(generacion)
 end
```

Cuadro 1.Seudocódigo Algoritmo TSP

Seudocódigo Cruza PMX.

Input=**Padres** $x1=[X_{1,1},X_{1,2},...X_{1,n}]$, $x2=[X_{2,1},X_{2,2},...X_{2,n}]$
Output= **Hijos** $y1=[Y_{1,1},Y_{1,2},...Y_{1,n}]$, $Y2=[Y_{2,1},Y_{2,2},...Y_{2,n}]$

Inicialización

- **$Y_1 = X_1$ y $Y_2=X_2$**
- **Inicialice p1 y p2 la posición de cada índice en y1 y y2;**
- **Elija dos puntos de cruce a y b tales que 1 ≤ a ≤ b ≤ n;**

for each i entre a y b do
t1 = y1,i and t2 = y2,i ;
y1,i = t2 and y1,p1,t1 = t1 ;
y2,i = t1 and y2,p2,t2 = t2 ;
p1,t1= p1,t2 and p1,t2 = p1,t1 ;
p2,t1 = p2,t2 and p2,t2 = p2,t1 ;
endfor

Cuadro 2.Seudocódigo cruza PMX .

Seudocódigo Cruza Cycle Crossover.

Input=**Padres** $x1=[X_{1,1},X_{1,2},...X_{1,n}]$, $x2=[X_{2,1},X_{2,2},...X_{2,n}]$
Output= **Hijos** $y1=[Y_{1,1},Y_{1,2},...Y_{1,n}]$, $Y2=[Y_{2,1},Y_{2,2},...Y_{2,n}]$
Inicialización

- **Inicializa Y1 y Y2 como genotipos vacíos**

$Y_{1,1=} X_{1,1}$
$X_{2,1=} Y_{2,1}$
i=1
Repeat
j → Índice donde encontramos $X_{2,i,}$ en X_1
$y_{1,j}=x_{1,j;}$
$y_{2,j}=x_{2,j;}$
i=j;

Until $X_{2,i} \notin y_1$

For cada gen no inicializado do

$y_{1,i}=x_{1,i}$;

$y_{2,i}=x_{2,j}$;

Endfor

Cuadro 3.Seudocódigo cruza Cycle Crossover.

4. Código Documentado

```
   do = true;
c=1
ruta_r  = randperm(18)
gene=[]
for i=1:50 %%generando una poblacion de
ruta_r  = randperm(18)
gene=[gene;ruta_r]
end
%%sumatoria de cuidades
sumas=sumatoria(gene);
bar(sumas)
hold on

while do
c=c+1 %%contador de generaciones
 if(c==20)
   do=false;

%%%%Seleccion
par1=Seleccion(gene)
par2=Iseleccion(gene)
%%%%
b=size(par1)
n=b(1)
hijos=[]
generacionX=[]
%Cruza pmx
 for(i=1:size(par1))
    pareja1=par1(i,:)
    pareja2=par2(i,:)
    hijos=pmx(pareja1,pareja2)
    generacionX=[generacionX;hijos]
 end
 gene=generacionX
 sumas=sumatoria(gene);

 bar(sumas,0.4,'r')
 end
end
```

Cuadro 4. Código con selección PMX

```
do = true;
c=1
ruta_r  = randperm(18)
gene=[]
for i=1:50 %%generando una poblacion de 100 pobladores con 18 genes
```

```
ruta_r  = randperm(18)
gene=[gene;ruta_r]
end
seguridad=gene
%%sumatoria de cuidades
sumas=sumatoria(gene);
bar(sumas)
%plot(sumas)
hold on

while do
c=c+1 %%contador de generaciones
 if(c==0)
   do=false;

%%%%Seleccion
par1=Seleccion(gene)
par2=Iseleccion(gene)
%%%%
b=size(par1)
n=b(1)
hijos=[]
generacionX=[]
 for(i=1:size(par1))
    pareja1=par1(i,:)
    pareja2=par2(i,:)
    hijos=ciclo1(pareja1,pareja2)
    generacionX=[generacionX;hijos]
 end
 gene=generacionX
 sumas=sumatoria(gene);
bar(sumas,0.4,'r')
 end
end
```

Cuadro 5. Código con selección Cycle Cross

5. Resultados

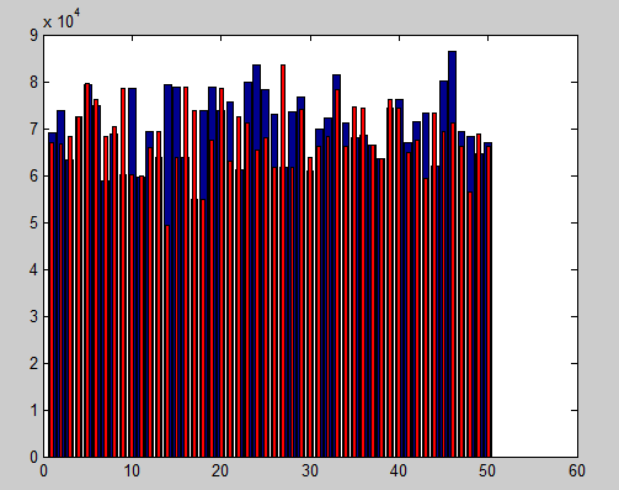

Figura 2.PMX 50 generaciones Sin elitismo

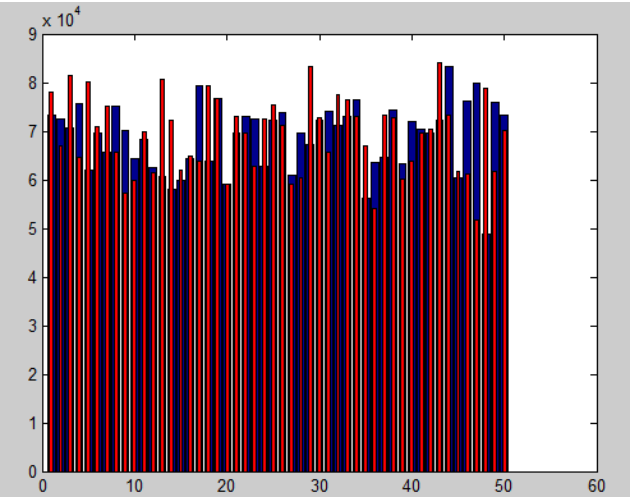

Figura 3. Cruza Cycle Cross 50 Generaciones.

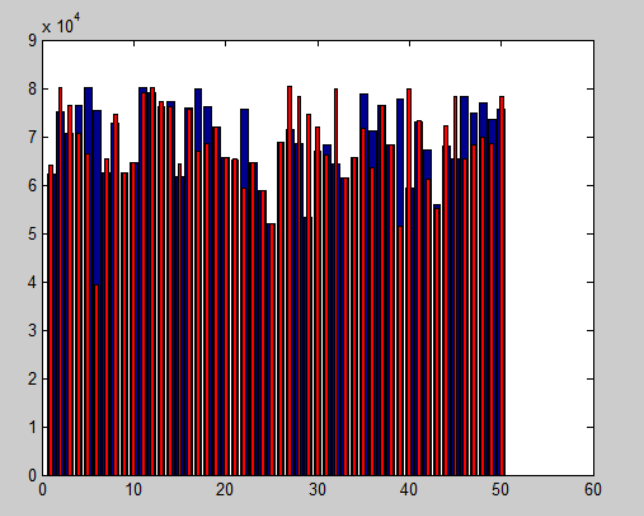

Figura 4. Cruza Cycle Cross 50 Generaciones.

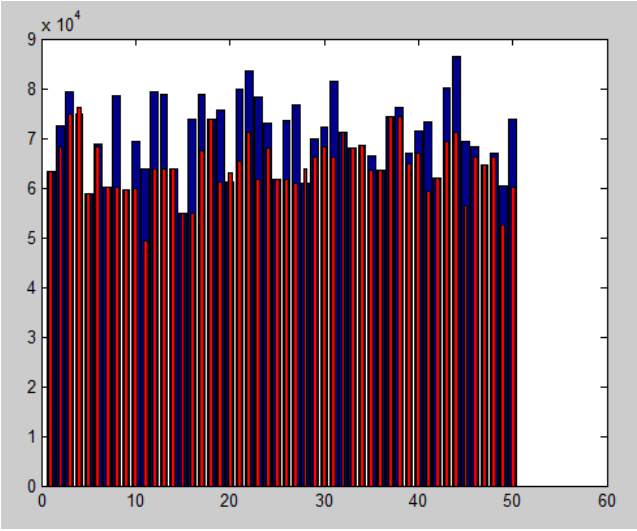

Figura 5 Cruza Cycle Cross 50 Generaciones con Elitismo

Características	PMX con Elitismo		PMX sin Elitismo		CX con Elitismo		CX Sin Elitismo	
N	50	100	50	100	50	100	50	100
Tiempo de Computo	0.500237 seconds	0.556237 seconds	0.456237 seconds	0.486237 seconds	0.379732 seconds	0.432215 seconds	0.456237 seconds	0.476237 seconds
Mejor valor	4.0010e+004	3.0300e+004	5.1499e+004	5.0978e+004	4.9910e+004	3.3038e+004	5.0516e+004	4.1016e+004
Peor Valor	7.1357e+004	6.0107e+004	8.4377e+004	7.1357e+004	5.4431e+004	4.4431e+004	8.4382e+004	7.0056e+004

Tabla 1. Resultados

Los Datos arrojados por las diferentes pruebas que se realizaron se muestran en forma de resumen en la tabla 1.N es el número de generaciones al que el algoritmo se sometió. En el segundo renglón tenemos el tiempo de computo de cada algoritmo con y sin elitismo.

Y en los últimos dos renglones tenemos el mejor valor y el peor valor que el algoritmo arrojo de todas las generaciones en que los algoritmos fueron sometidos.

Podemos observar que CX con Elitismo nos da el valor mas optimo, para la resolución de este problema.

6. Discusión

La recombinación genética es uno de los aspectos mas importantes en los algoritmos genéticos, dado que se necesita tener un espacio de búsqueda amplio y para esto es necesario la diversidad genética.

7. Conclusiones

En el reporte, se analiza la recombinación de la solución, es decir, los operadores de cruce en el contexto del problema del viajero frecuente. Se sabe bien que este operador genético juega un papel importante en el desarrollo de algoritmos genéticos.

Se implementaron 2 tipos de cruza para ver la influencia de este operador en el proceso de búsqueda genética. Fueron Partially Mapping Crossover(PMX) y Cycle Cross (CX) a cada uno fue con y sin elitismo.

El mejor de los resultados fue Cycle Cross con elitismo.

El peor fue PMX sin Elitismo.

Ahora estos resultados pueden mejorar en aspectos como representación de los datos, por la manera grafica en que se muestran y de igual forma mediante operaciones de estadística básica para obtener más información de resultados mostrados. Y ciertos detalles que faltan de afinar.

8.-Biografía

Macias, R. G. (2019). Optimización-Algoritmos programados con Matlab. *Ingenio y Conciencia Boletín Científico de la Escuela Superior de Cd. Sahagún*, *6*(11).

TING, Chuan-Kang; SU, Chien-Hao; LEE, Chung-Nan. Multi-parent extension of partially mapped crossover for combinatorial optimization problems. *Expert Systems with Applications*, 2010, vol. 37, no 3, p. 1879-1886.

DEEP, Kusum; MEBRAHTU, Hadush. Variant of partially mapped crossover for the Travelling Salesman problems. *International Journal of Combinatorial Optimization Problems and Informatics*, 2012, vol. 3, no 1.

Oliver, I. M., Smith, D. J., & Holland, J. R. C. (1987). A study of permutation crossover operators on the traveling salesman problem. In Proceedings of the second international conference. on genetic algorithms (ICGA'87) (pp. 224–230). Cambridge, MA:Massachusetts Institute of Technology.

Alireza Arab Asadi, Ali Naserasadi and Zeinab Arab Asadi. Article: A New Hybrid Algorithm for Traveler Salesman Problem based on Genetic Algorithms and Artificial Neural Networks. International Journal of Computer Applications 24(5):6–9, June 2011. Published by Foundation of Computer Science

Dr. Nitin S Choubey. A Novel Encoding Scheme for Traveling Tournament Problem using Genetic Algorithm. IJCA Special Issue on Evolutionary Computation (2):79– 82, 2010. Published by Foundation of Computer Science.