Reporte

Alma Eliza Guerrero Sánchez.

1. Introducción

La Optimización se a convertido una parte muy importante en todas las disciplinas, o por lo menos en la mayoría de las disciplinas que se tienen. Existen varias razones del porqué, pero la más fuerte es la competitividad que existe en la actualidad en los servicios o productos.

El Doctor Aurelio en su conferencia “Aplicación de la teoría evolutiva de Darwin en ingeniería”, define la Optimización como “Encontrar los valores de las variables de diseño de decisión de manera que las funciones minimicen o maximicen, según sea el caso y dependiendo de una Metodología”.

Este problema en particular, del “viajero frecuente” es un problema de Minimización.

Para la optimización de este tipo de problemas se puede hacer uso de algoritmos como “Método del Descenso por Gradiente”.

El método del descenso por Gradiente es una de las primeras técnicas usadas para problemas de optimización de minimización. Este método es la base en la que se fundamentan varios otros algoritmos de optimización mas sofisticados. La desventaja es su lenta convergencia que tiene, pero se utiliza como base por su facilidad de programación y por simple.

El uso de técnicas de computo evolutivo para la optimización de problemas multivariables, es la mejor opción tanto para el tiempo de la ejecución de este y la solución final.

2. Marco teórico

2.1 Aplicación de algoritmos Evolutivos en problemas de Optimización

Los algoritmos genéticos han resuelto con efectividad problemas de Optimización tanto de Minimización como de Maximización de resultados.

Los operadores que conforman los algoritmos genéticos como selección, cruce o mutación, son los encargados de mejorar los resultados paulatinamente mientras va pasando el numero de generaciones, para poder llegar a la convergencia llegada.

La idea básica de los algoritmos evolutivos es mejorar la soluciones candidatas simulando mecanismos de la evolución natural, como lo son los operadores que mencione anteriormente.

2.2 Cruza

En los seres vivos la cruza consiste en un intercambio complejo de información a nivel de genotipos.

En los algoritmos este proceso se simula con el intercambio de información a nivel vectores.

En el Reporte anterior vimos que en los algoritmos genéticos básicos tenemos los tipos de cruza como :

- Cruza a un corte
- Cruza a dos cortes

Estos dos tipos de cruza son muy utilizados por la simplicidad y la facilidad de manejarse, pero una desventaja que se tiene es que en algunas partes del espacio de búsqueda pues no explora por completo todo el espacio mas en cadenas que son muy largas.

Es por eso que se propusieron métodos diversos de cruza para que se hicieran uso de ella dependiendo de el largo de las cadenas.

La cruza es el operador mas sobresaliente en los algoritmos genéticos porque produce descendencia mediante la recombinación del material genético parental. Se hace uso de dos padres, porque hasta donde se tiene ningún organismo de la tierra implica multi padres. En ciertas simulaciones de algoritmos genéticos no es necesario limitar el numero de padres para la etapa del cruce, pero esta técnica se usa en "crossover multiparental" y a los algoritmos que hacen uso de esas técnicas se les denomina algoritmos genéticos multiparentales. Pero este concepto para este trabajo no se hizo presente solo se hace mención.

Para propósitos de este reporte se hizo uso del método de cruce Partially mapped crossover.

Partially Mapped Crossover es uno de los métodos de cruza mas populares y utilizados en algoritmos genéticos para la resolución de problemas de optimización combinatoria.

PMX puede ser considerado como una modificación de cruce de dos puntos pero adicionalmente utiliza una relación de mapeo para legalizar descendientes que tienen números duplicados.

Goldberg y Lingle [] desarrollaron un cruce parcialmente emparejado (PMX) que preserva la posición absoluta utilizando dos puntos de corte en los padres. Este operador primero selecciona aleatoriamente dos puntos de corte en ambos padres. Para crear una descendencia, la subcadena entre los dos puntos de corte en el primer padre reemplaza la subcadena correspondiente en el segundo padre. Entonces, lo inverso

El reemplazo se aplica fuera de los puntos de corte para eliminar duplicados y recuperar todas las ciudades.

2.3 Problema del agente viajero(TSP)

Travelling Sales Problem, es la respuesta a una pregunta:

> **Dada una lista de ciudades y las distancias entre cada una de ellas ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?**

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.

Cruza PMX

Paso 1.- Seleccionar 2 posiciones a lo largo de la cadena.

- **Fija**
- **Aleatoria**

Paso 2.- Las subcadena definidas por las 2 posiciones que seleccione del paso 1, se convertirán en mi matriz de correspondencia o mapeo.

Paso 3.- Determinar la relación del mapeo, entre las 2 secciones de mapeo

Cuadro 2.Seudocódigo cruza PMX .

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==4)
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=pmx(pareja1,pareja2)
       generacionX=[generacionX;hijos]
 end
 gene=generacionX
 sumas=sumatoria(gene);
 bar(sumas,0.4,'r')
 end
end
```

Cuadro. Código con selección PMX Aleatoria

```
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)
hold on

while do
c=c+1 %%contador de generaciones
 if(c==4)
   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=pmx1(pareja1,pareja2)
    generacionX=[generacionX;hijos]
 end
 gene=generacionX
 sumas=sumatoria(gene);
 bar(sumas,0.4,'r')
 end
end
```

Cuadro. Código con selección PMX Fija

5. Resultados

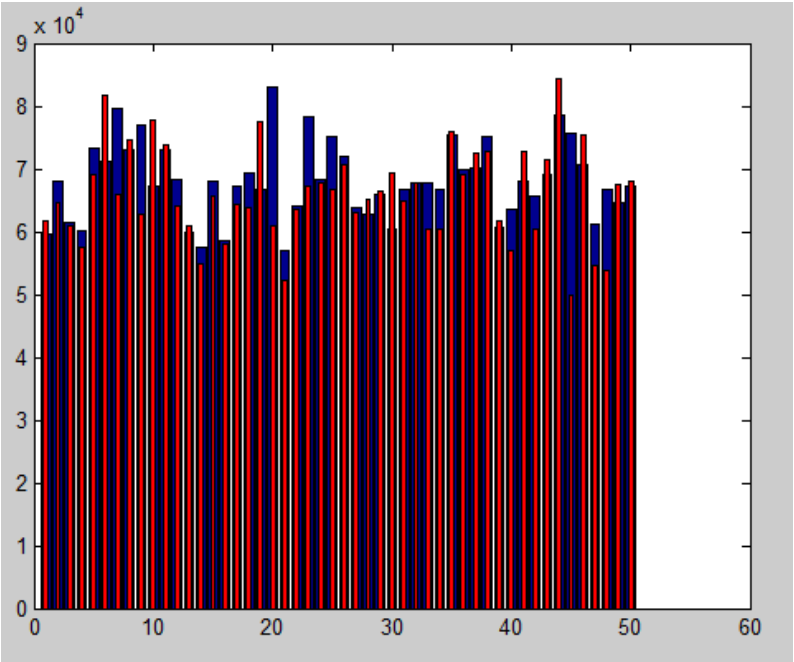

Figura 1. Cruza con PMX Aleatorio

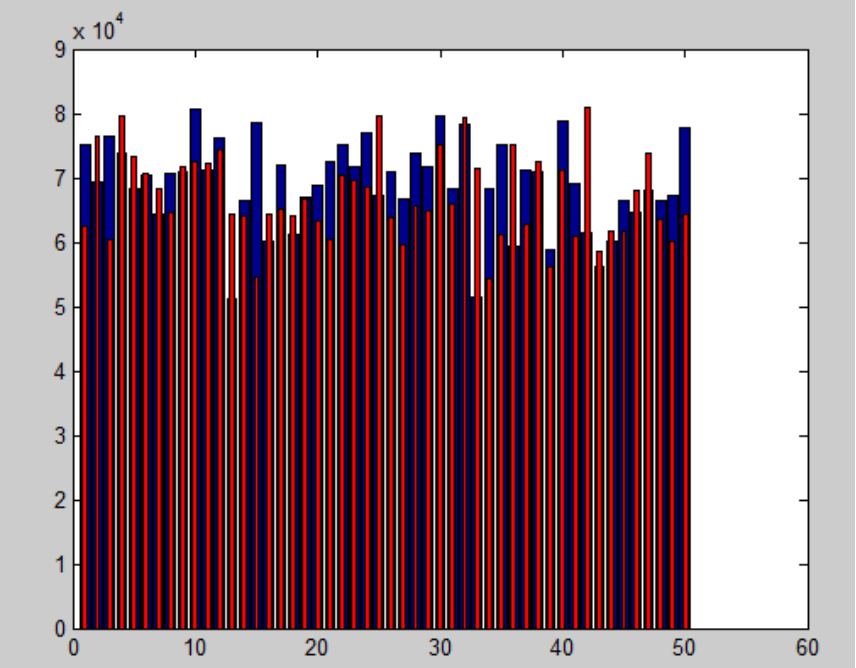

Figura 2. Cruza con PMX fija

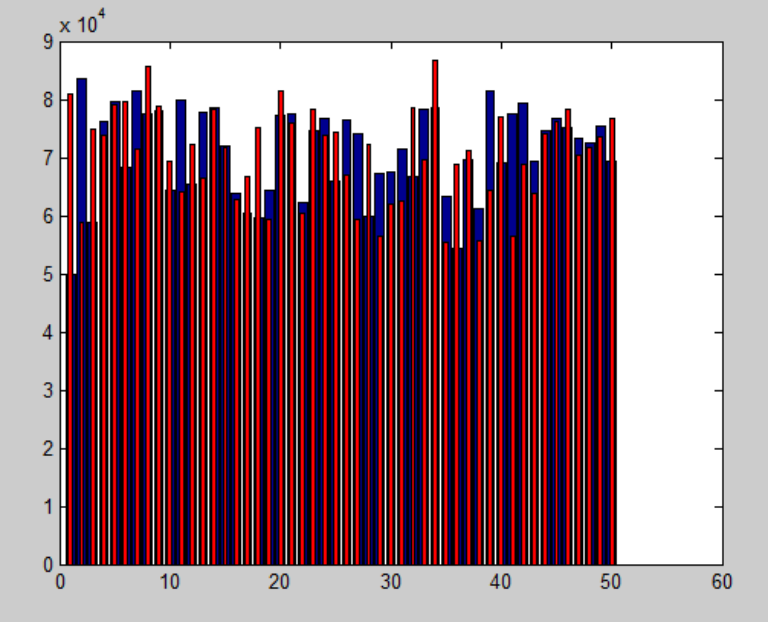

Figura 3. Cruza con PMX fija

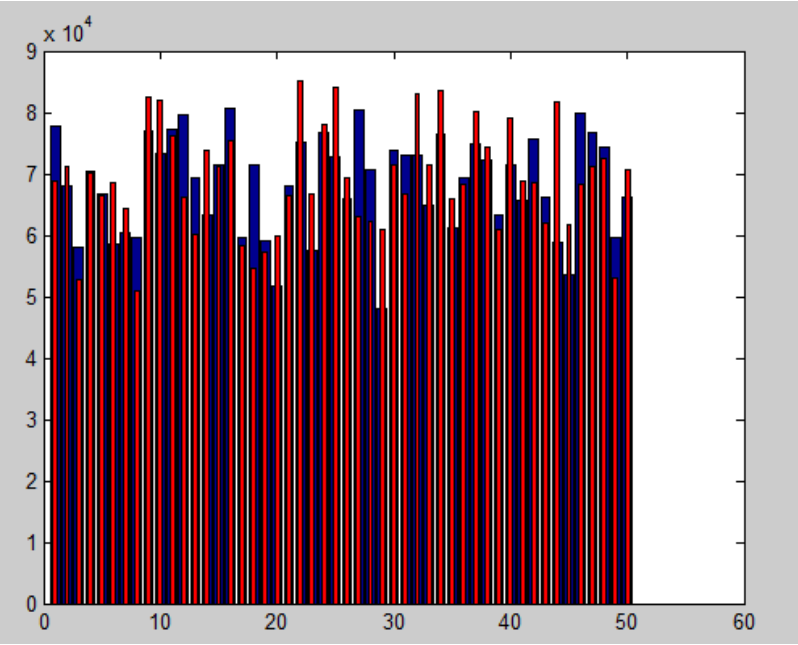

Figura 4. Cruza con PMX fija

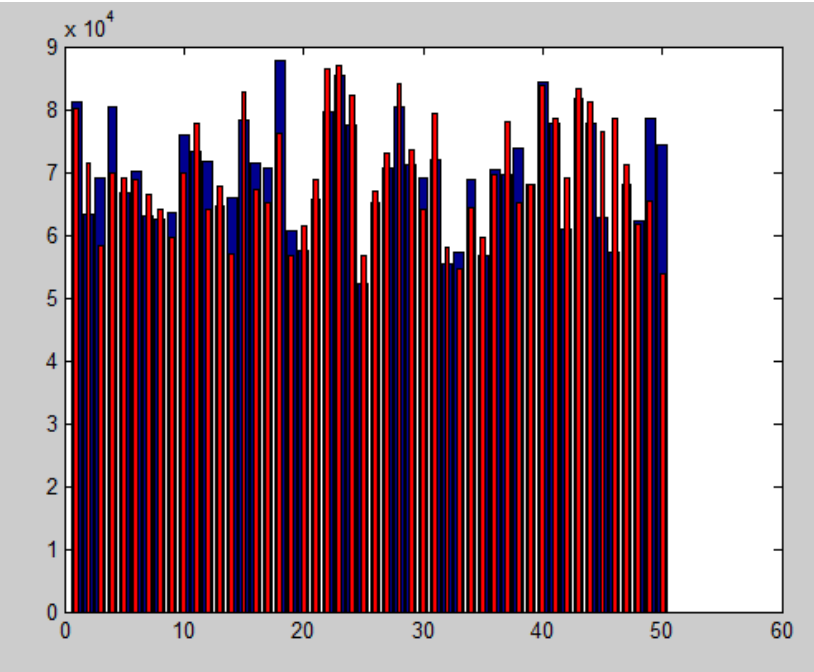

Figura 5. Cruza con PMX fija

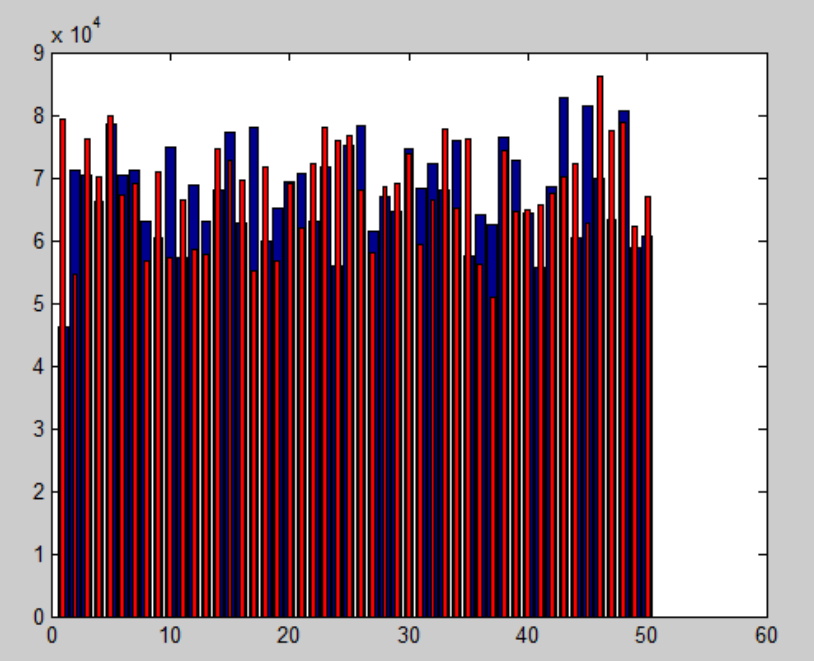

Figura 6. Cruza con PMX fija

Las barras azules representan la generación inicial y las barras rojas la generación final después de pasar por el proceso de cruza.

En la figura 1 como dice la leyenda de la parte inferior es resultado de la Cruza de manera aleatoria, esto quiere decir que las posiciones para realizar el corte y realizar la matriz de mapeo fueron de manera aleatoria.

Los resultados como se pueden observar, se tiene una mejoría notable de manera parcial. Porque no hubo una optimización de manera notoria como se hubiera esperado de esta parte.

En la figura 2 se muestran los resultados de la cruza PMX de manera fija. Con “fija” se refiere a que se hizo un corte desde una posición definida.

A partir de la figura 3 hasta la 6 se manipuló el tamaño de la matriz de mapeo, esto con el fin de ver como se afecta en el resultado.

Como se puede observa de la figura 3 a la 5 se usan matrices de mapeo de mayor tamaño, aumento el tamaño de la matriz de manera ascendente. Con este se quiere decir que la figura 5 tiene la matriz de mapeo más grande y así disminuyendo hasta la figura 3. Como se observan en estas figuras no se tuvo una optimización adecuada, no tuvo un cambio muy notable.

En la figura 6 se uso una matriz de mapeo mas pequeña, y como se puede observar se tiene una optimización más notable.

6. Discusión

Se sabe que la cruza es uno de los operadores de los algoritmos genéticos que influyen en cierto porcentaje en el resultado final, el tener una cruza un tanto dinámica en mi opinión existe una mejora notable en los datos finales.

El uso de la aleatoriedad para definir la matriz de mapeo, en mi opinión como es de manera aleatorio no se tiene un control del tamaño de la matriz de mapeo, y no se podría tener con certeza tanto el tamaño de la matriz de mapeo ni cual seria el resultado final y que tanto influyo esto.

7. Conclusiones

En el trabajo pasado una de las conclusiones que de manera parcial se llego fue que la cruza es uno de los aspectos que influye en los resultados finales. En este trabajo de alguna manera me lo reafirmo, a pesar de no aplicar otros operadores para la mejora de resultados se obtuvo una optimización de una manera sutil.

Con respecto a la pregunta realizada en clase ¿Qué efecto tiene el tamaño de la matriz y la posición donde se extrae la misma en el resultado?

Como se pudo notar en los resultados obtenidos el tamaño de la matriz de mapeo influye de sobremanera en los resultados finales.

Se debe tener una variedad o una diversidad en los genes, para poder tener hijos con genes fuertes y variados y así el resultado va a tener una variación de manera notable

Se tiene expectativa de que al aplicar operadores como mutación, elitismo, se obtenga mejores resultados óptimos para la resolución de este problema

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.