Implementación de un Algoritmo Genético para el problema del Árbol de Costo Mínimo Capacitado

Armando Isaac Hernández Muñiz

Cinvestav Unidad Tamaulipas
Inteligencia Computacional - Dr. Gregorio Toscano

Abstract. En este documento se trabaja con el problema del árbol de costo minimo capacitado (CMST). Dado un árbol con un nodo concentrador el cual tiene un conjunto de nodos terminales cada uno con un valor de una demanda, a excepción del nodo central, todas las terminales están conectados al nodo concentrador ya que toda la información debe llegar hacia él. El objetivo del problema es encontrar una red entre los nodos de tal forma que el costo de sus flujos sea el más mínimo posible. Cada enlace entre las terminales consta de un costo, por lo que la restricción de este problema requiere que la suma de los costos de los subredes conectadas al nodo central no supere una capacidad de Q. Los algoritmos genéticos son métodos adaptativos utilizados para problemas de búsqueda y optimización de parámetros basados en la reproducción sexual y principio de supervivencia del mas apto. En este documento se describe una implementación de un algoritmo de genético para el problema del *Árbol de costo mínimo capacitado*, el cual es interpretado con una representación de codificación de arboles que se vera detalladamente. También se describe la función de evaluación y métodos de generación de soluciones aleatorias. Para obtener los resultados mas deseados de acuerdo a la implementación del algoritmo se hará uso de una sincronización de parámetros con la ayuda de covering arrays.

1 Introducción

El problema del árbol de expansión mínimo capacitado (CMST) se puede definir como el diseño de un árbol de costo mínimo que se extiende sobre todos los vértices en un grafo G no dirigido [1], para cada nodo se presenta una demanda, y la suma de estas demandas en cada subárbol de G no debe exceder un límite de capacidad Q [2]. Este problema representa cada subárbol T de G, y su definición es obtener el árbol de costo mínimo. El problema de CMST es de gran interés porque, debido a sus limitaciones de capacidad, hace que el problema del árbol de expansión mínima (MST) sea un problema NP-Hard [3]. El CMST se puede dividir en dos categorías. El *Homogéneo* es cuando las demandas de todos sus nodos que contiene son iguales y el *Heterogéneo* es cuando las demandas de sus nodos varían entre ellos [1]. Cuando todos los vértices del árbol son de costo 1, entonces el problema cambia para obtener un árbol de costo mínimo

enraizado que agrega los costos de los nodos hasta alcanzar los nodos Q, completando la capacidad máxima. El *árbol de costo mínimo capacitado (CMST)* es un problema fundamental para diseñar redes de comunicaciones. Esto consiste en encontrar una topología diseñada para obtener el costo mínimo en una red de procesamiento donde un nodo concentrador debe ser enlazado por un conjunto de subárboles de la topología llamados terminales, y cada una de estas contiene demandas.
El CMST es tambien importante en el diseño de redes de telecomunicaciones troncales, así como en la distribución, el transporte y la logística. También proporciona una relajación del clásico problema de enrutamiento de vehículos capacitados, debido a la influencia que MST tiene en la heurística constructiva, que es fundamental para muchos otros problemas complejos, incluido el diseño de redes de comunicaciones con estructuras de anillo topológicas [4]. Debido a que los problemas de enrutamiento de vehículos generalmente consideran limitaciones de capacidad, el diseño de algoritmos más efectivos o eficientes para el CMST también puede desempeñar un papel en el diseño de métodos eficientes para obtener soluciones viables para el problema de enrutamiento de vehículos capacitados [3].

2 Objetivo del Problema

En el problema de CMST se obtienen una seria de nodos cada uno es una posición; dentro de ese conjunto existe un nodo denominado nodo central y el resto se le denominan como nodos terminales. Cada nodo terminal se le asigna una demanda de tráfico la cual debe dirigirse al nodo central. Tambien, existe un peso o distancia fija entre cada nodo. El objetivo es encontrar una red de costo mínimo en con estructura de árbol que transporte el tráfico desde los nodos terminales al nodo central [5]. El caso de demanda unitaria indica que las demandas de todos los nodos terminales tienen el valor de uno. Por otro lado, en el caso de demanda no unitaria, la demanda de cada nodo terminal varía. El problema CMST se ha demostrado que es NP-completo, incluso en el caso de la demanda unitaria, por lo que la solución del CMST con métodos exactos consta de mucho tiempo lo que hace que sea imposible incluso en instancias de tamaño moderado. Se han propuesto varias formulaciones matemáticas y algoritmos exactos para el CMST. Los algoritmos exactos se basan en métodos de programación dinámica y de ramificación y solo resuelven CMSTP a pequeña escala o encuentran límites más bajos en la solución óptima [1].
El problema CMST se define dado un grafo no dirigido $G = (V_0, A)$ en donde $V_0 = \{0,1,..,n\}$ que representa el conjunto de vértices, y $A = \{(i,j) \| i, j \in V; i \neq j\}$ representando a las aristas o enlaces del árbol. El nodo 0 es llamado también *nodo raíz* y el valor de su demanda es $d_0 = 0$ y cualquier otro nodo su demanda es $d_i > 0$. Puede ser representado con una matriz $C = (c_{ij})$ el cual está asociado con A (aristas), en donde c_{ij} es la distancia o costo no negativo entre cada vértice i y j del árbol. El problema consiste en encontrar el árbol T de expansión mínima dentro del grafo original G, donde la suma de las demandas de cada uno de los

subárboles enraizados al *nodo 0* (centro) no debe exceder el valor de capacidad limite Q [4].
En este documento se propone trabajar el problema del *Árbol de costo mínimo capacitado (CMST)* para realizar una comparación de resultados generados entre las implementaciones de Búsqueda Ciega, un Algoritmo Genético y un Recocido Simulado. En la seccion 2 se describe algo sobre el trabajo relacionado sobre los algoritmos que se implementan. La sección 3 explica a cerca de la representación con arboles con secuencias de Prufer que sera utilizada para los tres algoritmos. En la sección 4 se describe detalladamente los parámetros y el funcionamiento del metodo de evaluación. En la sección 5 se describe la implementacion del algoritmo secuencial del problema el cual trabaja con un problema pequeño para evaluar exhaustivamente todo el espacio de busqueda para encontrar el óptimo. La seccion 6 describe la version aleatoria del algoritmo. En las secciones 7, 8 y 9 se describen los algoritmos de Busqueda Ciega, Recocido Simulado y Genético respectivamente. Y por ultimo un análisis de los resultados obtenidos de acuerdo a la sintonización de los parámetros mas aptos para cada uno de los algoritmos aplicados y realizar una comparación para visualizar cual algoritmo da los resultados mas favorables.

3 Tecnica de Inteligencia Computacional

3.1 Algoritmo Genetico

Los algoritmos genéticos simulan un proceso de evolucion natural [6], son metodos adaptativos utilizados para problemas de búsqueda y optimización de parametros basados en la reproduccion sexual y principio de supervivencia mas apto [7]. Más formalmente, y siguiendo la definición dada por Goldberg, *"los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural. Combinan la supervivencia del mas apto entre estructuras de secuencias son un intercambio de información estructurado, aunque aleatorizado, para construir asi un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas"* [8] .
John Holland, quien fue un investigador de la universidad de Michigan, fue quien desarrollo a gran medida los Algoritmos Genéticos. En la década de los 60 implemento una técnica de selección natural [9]. En la naturaleza, los individuos de una población compiten constantemente con otros por recursos tales como comida, agua y refugio. Los individuos que tienen más éxito en la lucha por los recursos tienen mayores probabilidades de sobrevivir y generalmente una descendencia mayor. Al contrario, los individuos peor adaptados tienen un menor n umero de descendientes, o incluso ninguno.Esto implica que los genes de los individuos mejor adaptados se propagar an aun n umero cada vez mayor de individuos de las sucesivas generaciones [9].
Para poder aplicar el algoritmo genético al area de resolución de problemas se deben seguir algunos pasos [10]:

- Evaluar la aptitud de cada uno de los individuos generados.

- Permitir la reproducción de de individuos que tengan mas probabilidad de reproducción.
- Aplicar operadores de cruce y mutación para los individuos generados.
- Organizar la nueva población.

Los pasos anteriores se repiten hasta cumplir con un criterio de terminación, ya sea por un numero máximo de iteraciones del algoritmo, o hasta que ya no sea capaz de generar una población distinta, dando por hecho que el algoritmo converge [10]. En un algoritmo genético primero se determina el espacio de las soluciones del problema que se requiere resolver. De acuerdo al dominio del problema, el algoritmo opera sobre *códigos genéticos*, sobre genotipos que deberán mapearse al espacio de las soluciones. Por eso es necesario *codificar* el dominio del problema para obtener las estructuras que el algoritmo genético pueda manejar. Estas estructuras representan el genotipo del individuo en términos biológicos [6].

Codificación
Dada una solución a un problema en específico, esta puede representarse con valores en una serie de parámetros. Éstos parámetros (denominados genes en la terminología de Algoritmos Genéticos) se codifican en una cadena de valores denominada cromosoma. El conjunto de los parámetros representado por un cromosoma particular recibe el nombre de genotipo. El genotipo contiene la información necesaria para la construcción del organismo, es decir, la solución real al problema, denominada fenotipo. Por ejemplo, en la terminología biológica, el genotipo sería la información genética del ADN del individuo, mientras que la forma o expresión del ADN o el individuo en sí seria el fenotipo [9].

- *Codificación Binaria*. Este tipo de codificación es la más extendida debido a que fueron utilizadas en los primeros algoritmos genéticos. En esta codificación, cada cromosoma es una cadena de bits que contiene dos valores 0 y 1. A su favor tiene que puede abarcar muchos cromosomas incluso con un número reducido de genes. Sin embargo por otro lado esta opción no es la idónea para muchos problemas y en algunas ocasiones es necesario realizar correcciones después de la reproducción y/o mutación [11].
- *Codificación Numérica*. En esta codificación se utilizan cadenas de números que representan un número en una secuencia. Es útil en problemas donde el objetivo es ordenar , el cual es de mucho uso. En algunos casos también es necesario como en el caso anterior realizar correcciones tras relaciones o mutaciones [11].

Evaluación de la Población
Con el objeto de ser capaces de comparar dos o más soluciones, debe introducirse una forma de evaluar las soluciones, basándose en la función objetivo y en las restricciones del problema. En caso de no tener la función objetivo definida en forma explícita, la evaluación puede llevarse a cabo con alguna subrutina de cálculo representativa [12]. Cada individuo de la población del algoritmo genético

debe ser calificado, que en términos biológicos seria un grado de adaptación *fitness*. Éste es un numero real no negativo que tanto mas grande o mas pequeño sea, dependiendo si se trata de maximizar o minimizar, mejor sera la solución del propuesta por el individuo. El objetivo de calificar cada individuo es distinguir cuales tienen mejor propuesta de solución de las que no lo tienen [6]

Selección

El operador de selección genera a partir de una población inicial otra población intermedia del mismo tamaño, reproduciendo con un mayor número de copias a los individuos más aptos y eliminando o asignando un menor número de copias a los individuos menos aptos. El operador de selección no produce puntos nuevos en el espacio de búsqueda, sino que determina qué individuos dejarán descendencia y en qué cantidad en la próxima generación [11]. Puesto que se trata de imitar lo que ocurre en la naturaleza, se ha de otorgar un mayor numero de oportunidades de reproducción a los individuos mas aptos. Por lo tanto la selección de un individuo estar a relacionada con su valor de ajuste. No se debe sin embargo eliminar por completo las opciones de reproducción de los individuos menos aptos, pues en pocas generaciones la población se volvería homogénea [12].

- *Selección por Ruleta.* En esta selección se crea una ruleta con los individuos presentes en la generación actual. Cada individuo tendrá una parte de esa ruleta mayor o menor en función a la puntuación que tenga cada uno. Después a manera de simulación, se gira la ruleta y se selecciona el individuo en el que se para. Cabe mencionar que por lógica, el individuo con mayor puntuación o fitness tendrá la mayor probabilidad de ser escogido. En caso de que las probabilidades sean muy diferentes una de la otra, este método dará problemas puesto que si un individuo tiene un 90% de posibilidades de ser seleccionado, el resto apenas saldrá lo que reduciría la diversidad genética [10].
- *Selección por Torneo.* La idea principal de este método consiste en realizar la selección en base a comparaciones directas entre individuos. Existen dos maneras de selección mediante torneo las cuales son la *determinística* y la *probabilística*. La deterministica selecciona p individuos al azar de la población actual, generalmente se escoge el valor de $p = 2$. Ya que fueron seleccionados los individuos, se debe seleccionar el más apto o mejor puntuación para pasar a la siguiente generación. En el caso de la versión probabilistica solo difiere en la selección del mas apto. En lugar de escoger siempre al mejor, se genera un numero aleatorio entre 0 y 1 y es comparado con un valor p (que es fijo para todo el proceso), si es mayor se escoge al mejor individuo, de lo contrario se selecciona al menos apto [9].

Cruzamiento

Ya que se realiza la selección de los individuos, procede a realizar la cruce entre dos individuos. El operador de recombinación (crossover) es el operador de búsqueda más importante en los algoritmos genéticos. Este es un operador sexuado que intercambia el material genético de un par de padres produciendo

descendientes que normalmente difieren de sus padres [11]. Más concretamente, el operador de cruce consiste en intercambiar de material genético entre dos individuos. El objetivo de éste operador es conseguir que el descendiente mejore la aptitud de sus padres. Para aplicar el cruce habrá se tiene que seleccionar antes a los dos individuos de la población con el método de selección ya antes mencionado. Además esta selección puede elegir el mismo padre para un descendiente. Esto no es problema ya que se asegura la mutación del individuo más dominante pero si este cruce se produjese con mucha frecuencia podría acarrear consecuencias adversas en caso de que ese individuo dominante presente algunos genes no deseados [10].

- *Crossover 1 Punto*. Los dos individuos seleccionados representan a los padres y se cortan por un punto. Se copia la información genética de uno de los padres desde el inicio hasta el punto de cruce y el resto se copia del otro progenitor [10].
- *Crossover 2 Puntos*. Se trata de generalizar el el método de cruce de 1 punto. En lugar de cortar en un solo punto los individuos de los padres se realizan dos cortes. Se debe tenerse en cuenta que ninguno de estos puntos de corte coincida con el extremo de los individuos para garantizar que se originen tres segmentos [9].
- *Crossover Uniforme*. El método de cruce uniforme es una técnica que difiere en su totalidad de los métodos antes vistos. Cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre [9].

Mutación

La mutación a un individuo hacen que sus genes varíen su valor de forma aleatoria. Aunque se pueden seleccionar los individuos directo de la población inicial y realizar la mutación antes de agregarlos en la nueva población, la mutación se suele utilizar de manera conjunta con el operador de cruce. Primero se seleccionan dos individuos de la población para realizar el cruce. Si el cruce tiene éxito entonces uno de los descendientes, o ambos, se muta con cierta probabilidad Pm. Se imita de esta manera el comportamiento que se da en la naturaleza, pues cuando se genera la descendencia siempre se produce algún tipo de error, por lo general sin mayor trascendencia, en el paso de la carga genética de padres a hijos. Desde otro punto de vista, la motivación para usar mutación es prevenir la pérdida permanente de un bit determinado y así evitar la convergencia prematura permitiendo escapar de óptimos locales [12].

4 Representación con estructura de árbol.

Para la implementación del problema CMST, se utilizará la técnica de representación del árbol Neville 3. Esta forma de representación consiste en generar una secuencia de números de modo que representen la conversión de los enlaces de un árbol. El número de posibles árboles que se pueden formar está dado por N^{N-2}, donde N es el número de nodos del árbol. Para este problema, se dará como entrada una secuencia generada aleatoriamente para un árbol inicial. La secuencia se forma a partir de N, donde su longitud es N-2 y cada uno de sus dígitos puede tener valores en un rango de 0 a N-1.

4.1 Generación de secuencia aleatoria

La Figura 1 muestra cómo se define la estructura de una secuencia de árbol. En el ejemplo, se construye una secuencia dados seis nodos (N = 6). Muestra que la longitud de la secuencia es 4 y los valores posibles que puede tomar cada posición. Los dígitos que se seleccionaron al azar se marcan en rojo, el número total de dígitos en el rango es igual al número de nodos N. La figura 2 muestra la representación del árbol con la matriz de costos.

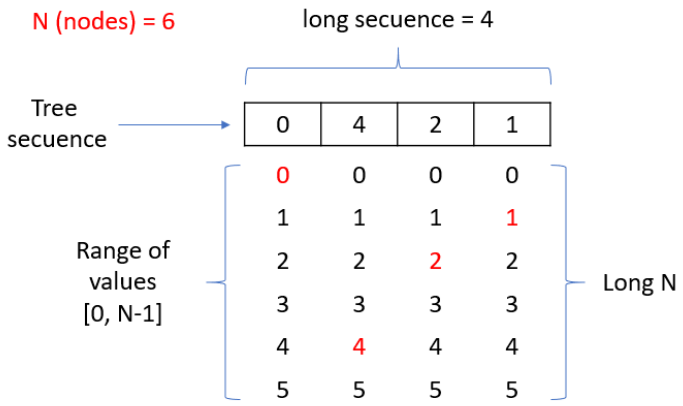


Fig. 1. Tree secuence structure

4.2 Conversión de secuencia a árbol

Para la conversión de secuencia a árbol, en la representación de neville 3 se utilizan dos vectores auxiliares para la codificación de los enlaces de árbol.

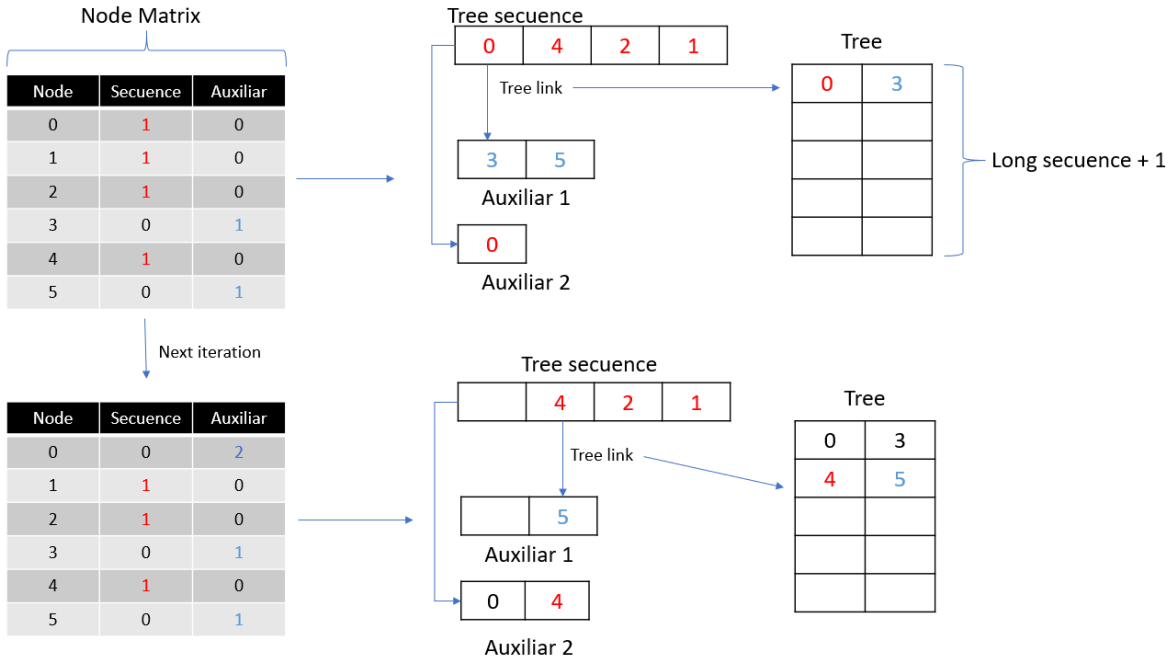


Fig. 2. Secuence to Tree Conversion Algorithm

La Figura 2 muestra el algoritmo implementado para la conversión de secuencia a árbol. Para esto, se creó una matriz llamada *NodeMatrix*, que tiene N número de filas y 3 columnas. Cada fila representa datos de cada nodo; la primera columna representa la etiqueta de todos los nodos (de 0 a N-1), la segunda representa el número de veces que el nodo se repite en la secuencia, y la tercera columna identifica en qué auxiliar cada nodo de la secuencia pasará una vez El enlace del árbol está codificado. El proceso se repite hasta que todos los enlaces en el árbol se obtienen en la matriz *Tree*, cuyas dimensiones son la longitud de la secuencia + 1.

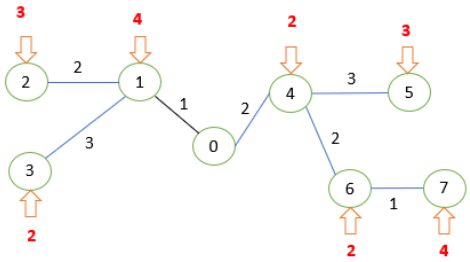
5 Función de Evaluación

La función de evaluación para el CMST requiere tres valores esenciales: el costo del árbol, la violación y el costo del árbol máximo encontrado:

$$f(sequence) = cost + (max\_tree * violation)$$

Esta función requiere la secuencia del árbol como parámetro para su evaluación. Una vez que se tiene la secuencia lo primero en hacer es realizar la conversión a árbol con el método de codificación de Neville 3, es decir, se deben

obtener los enlaces a partir de la secuencia, como muestra el siguiente ejemplo:



	0	1	2	3	4	5	6	7
0	0	1	3	2	2	1	2	4
1		0	2	3	3	1	4	5
2			0	3	2	3	1	1
3				0	3	4	1	1
4					0	3	2	3
5						0	4	3
6							0	1
7								0

Fig. 3. Tree representation

La Figura 3 muestra la representación del árbol en forma de una matriz de costos. La instancia de la matriz contiene valores para todas las posibles conexiones de nodos. Con la secuencia se generan los enlaces de un árbol y de acuerdo a los enlaces obtenidos son los costos que se tomaran en cuenta en la matriz. La matriz de costos ayuda a obtener el costo del árbol obtenido y al mismo tiempo cumplir con la restricción de capacidad.
La función de evaluación pide los valores del costo del árbol generado con la secuencia el cual consta en identificar los costos el la matriz de acuerdo a los enlaces obtenidos; también se debe obtener el árbol de costo máximo, éste árbol se calcula obteniendo el enlace de cada nodo fila con el nodo columna de mayor costo hasta obtener un total de N-1 enlaces donde N es el numero de nodos. Finalmente se debe obtener la violación del árbol la cual consta en obtener todas las subárboles enlazados al nodo concentrador (en el ejemplo es el nodo cero) obteniendo la suma de las demandas de cada uno, en el ejemplo de la Figura 3 las flechas que apuntan a los nodos son los valores de sus demandas, a excepción del nodo concentrador que no contiene. Se dice que el árbol viola la restricción de capacidad Q cuando al menos uno de los subárboles obtiene una suma total de demandas superior a Q, entonces *violación* = Q - *sumDemandas* En el caso de que mas de un subárbol sobrepase el limite de capacidad, las violaciones de cada subárbol se suman y seria la violación total del árbol original.

6 Implementacion de Algoritmo Genético

Para la implementación del AG se requieren de varios parámetros de entrada. Éste toma como inicio una población inicial, la cual es generada con individuos de forma aleatoria para después ser evaluada. El individuo que cuente con la mejor aptitud o evaluación dentro de la población inicial también es llamado el mejor global. Para llevar a cabo las generaciones del AG se requiere un ciclo con un determinado numero de iteraciones, siendo éste uno de los parámetros del algoritmo, también se pueden tomar en cuenta otros criterios de terminación como parar el algoritmo cuando ya no sea capaz de mejorar el mejor global o cuando la población de individuos ya no sufra mayores cambios durante las nuevas generaciones. Otra de las partes importantes es la aplicación de los operadores de cruce y mutación, que para este problema fueron usados tres tipos por cada operador; en el caso de los operadores de cruce se aplica 1) cruce de un punto, 2) cruce de dos puntos y 3) cruce uniforme, y para la mutación se aplica 1) cambio de una posición, 2) cambio de dos posiciones y 3) probar todos los posibles valores para una posición y obtener la mejor evaluada.

El **Algoritmo 1** muestra la implementación del algoritmo genético. Primero se genera una población inicial P de forma aleatoria con una cantidad de NP individuos. Cada uno de los individuos de P es un árbol válido dentro del dominio de N^{N-2} usando la representación con secuencias donde cada uno de los valores de ésta es generado de forma aleatoria en el rango de *N-1* tomando como base el cero. Una vez que se genera la población debe ser evaluada. El metodo *evaluar(P)* evalúa cada uno de los individuos de P mandando llamar a la función de evaluación. Ya que cada individuo es evaluado entonces se ordenan de acuerdo a su aptitud dejando en la primera posición de P al mejor evaluado. En cada generación del ciclo controlado por $GMAX$ se crea un nuevo vector P' que va almacenando individuos para generar una población nueva. Para ir llenando **P'** se hace una selección aleatoria de dos elementos de P mediante una selección por torneo, es decir, se seleccionan t cantidad de individuos y se selecciona el mejor de ellos, esta selección se aplica dos veces ya que se requieren dos individuos hijos. Ya que se tienen los hijos seleccionados se aplican los operadores.

Algorithm 1 Algoritmo Genetico

```
Require: NP .. Tamaño de la poblacion
Require: crossover_p .. Probabilidad de cruce
Require: mutation_p .. Probabilidad de mutacion
Require: GMAX .. Total de generaciones
Require: elitista .. Indica si hay elitismo
P <- poblacion_inicial(NP)
evaluar(P)
for i<-1 hasta GMAX do
    P' <- []
    for hasta que size(P') = NP do
        h1, h2 = seleccion por torneo de P
        if rand(0,1) < crossover_p then
            Se aplica cruza entre h1 y h2
        end if
        if rand(0,1) < mutation_p then
            Se aplica mutacion a h1 y h2
        end if
        h1 y h2 se agregan a P'
    end for
    evaluar(P')
    if elitista = True then
        Se selecciona el mejor de P y agrega a P'
        Se elimina el peor de P'
    end if
    P <- P'
    evaluar(P)
end for
```

Para que se aplique la cruza entre los dos individuos hijos se genera un numero aleatorio entre 0 y 1 y si es menor a la probabilidad *crossover_p* entonces se aplica la cruza, ya sea la de 1 punto, 2 puntos o la cruza uniforme. El mismo caso aplica para la mutación, de acuerdo a su probabilidad *mutation_p*. Ya que fueron, o no, aplicados los operadores genéticos para esa selección de individuos hijos, ahora estos deben ser agregados a *P'*, estos pasos (*selección, cruce y mutación*) se repiten hasta que la longitud de *P'* llegue a su limite (*NP*). Por ultimo, cuando *P'* esta completo, se dice que se creo una nueva población en esa generación, entonces ésta se evalúa con **evaluar(P')** para que los individuos se ordenen. En caso de que el algoritmo sea elitista, entonces se modifica población *P'* intercambiando el mejor individuo de *P* por el peor de *P'*. Finalmente se reasigna la población *P* por *P'* generando una nueva población para la siguiente generación, y esto se termina hasta llegar a *GMAX* iteraciones. El mejor global se obtiene seleccionando el primer individuo de la ultima generación de *P* ya que al evaluarlo se ordenan de acuerdo a la aptitud.

7 Experimentacion

7.1 Instancias

For the experiments they will be used in a set of Benchmark instances for the CMST available at: http://people.brunel.ac.uk/mastjjb/jeb/orlib/capmstinfo.html. The instances are divided into two classes according to demand, which are those of unitary demand (UD) and not unitary (not UD). For the UD instances the number of vertices is $n \in \{80, 120, 160\}$ with a capacity of $Q \in \{5, 10, 20\}$. For non-UD instances it contains non-Eucledian distances with $n \in \{49, 99, 199\}$ and with capacities of $Q \in \{200, 400, 800\}$. Las instancias con las que se experimentaron en los algoritmos descritos en este documento son 20 las cuales se describen a continuación:

	Instancia	Num.Nodos	Nodo Central	Q (Capacidad)
1	cm50r1	50	0	200
2	:	:	:	400
3	:	:	:	800
4	cm50r1	50	15	200
5	:	:	:	400
6	:	:	:	800
7	cm50r2	50	0	200
8	:	:	:	400
9	:	:	:	800
10	cm50r2	50	25	200

7.2 Sintonización de parámetros para el Algoritmo Genético

La experimentación del algoritmo se llevo a cabo de acuerdo a una sintonización de parámetros con ayuda del siguiente Covering Array:

0	1	2	1	0	0	0
2	2	2	0	1	1	0
2	1	2	0	1	1	0
3	1	1	0	0	0	1
0	0	1	1	1	0	0
0	1	0	1	1	0	0
1	0	1	0	1	0	1
1	1	2	0	0	0	1
0	2	2	1	1	1	1
1	2	0	0	1	0	1
2	1	1	1	1	0	1
3	0	0	0	1	1	1
0	2	1	0	0	0	1
2	2	1	0	0	0	0
1	0	2	0	1	1	0
0	2	0	0	0	0	0
0	0	0	1	1	1	1
0	0	2	0	0	1	0
1	0	0	1	0	1	0
3	0	2	0	0	1	0
2	0	2	1	0	0	1
3	1	2	1	1	1	1
1	2	2	1	1	0	0
3	1	0	0	1	0	0
1	2	1	1	0	1	1
2	0	0	0	1	0	0
2	0	1	1	0	1	0
3	0	1	1	0	0	0
2	1	0	0	0	1	1
2	2	0	1	1	1	1
0	1	1	0	1	1	1
3	2	1	1	1	1	0
1	1	0	0	0	1	0
3	2	0	1	0	0	1
1	1	1	1	1	1	0
3	2	2	0	0	0	1

Cada una de las filas del Covering Array refiere al conjunto de parámetros de ejecución que requiere como entrada el algoritmo, cada valor de la fila es el índice de cada juego de parámetros según su posición. Los parámetros con los que se hicieron las pruebas son los siguientes:

P.Cruza	P.Mutacion	Población	Iteraciones	Torneo	Elitismo
0.9	1.0	2N	10000	3	Si
0.8	0.95	N	1000	2	No
0.5	0.9				

Cada valor de una fila del covering array es el índice correspondiente a la columna de la tabla de parámetros. Una vez teniendo estos datos, se definió como decidir que operador de mutación se debe usar por cada ejecución. Para ello se usaron las ecuaciones diofánticas las cuales constan de r numero de variables, en este caso tres, cada una por cada operador que se usa, y una granularidad G. Las ecuaciones para este problema se definen como:

$$x_1 + x_2 + x_3 = G$$

donde x_i es un operador de mutacion. Para calcular todo el conjunto de ecuaciones, que son las que nos definen la probabilidad de cada operador, se obtienen los GTPS de las combinaciones de:

$$total\ gtps = \binom{G + r - 1}{r - 1}$$

Una vez obtenidos todo el conjunto de GTPS, cada uno se le aplica la conversion:

$$x_1 = Z_0$$

$$x_i = Z_i - Z_{i-1} - 1$$

donde Z_i es un valor del *gtp*.

La siguiente tabla muestra el conjunto de ecuaciones diofánticas generadas con este procedimiento las cuales representan un conjunto de probabilidades que seran usadas para los operadores genéticos de cruza y mutacion.

0	0	5
0	1	4
0	2	3
0	3	2
0	4	1
0	5	0
1	0	4
1	1	3
1	2	2
1	3	1
1	4	0
2	0	3
2	1	2
2	2	1
2	3	0
3	0	2
3	1	1
3	2	0
4	0	1
4	1	0
5	0	0

Ya teniendo todas las ecuaciones diofánticas, todo el conjunto se agrega a cada una de las filas del covering array. En este experimento el total de ecuaciones diofanticas generadas fueron 21 ecuaciones para cruce y 21 para mutación, con 3 tipos para cada uno, y una granularidad $G = 5$. El total de lineas de ejecucion para las pruebas del algoritmo son: (21 ecuaciones diofanticas de cruce) * (21 ecuaciones diofanticas de mutación) * (36 filas del CA) * (31 ejecuciones) = 492156 ejecuciones totales.

Una vez realizadas todas las ejecuciones con el Algoritmo Genético se obtienen los parámetros que convergieron para el algoritmo con los cuales se obtuvieron los mejores resultados:

P.Cruza	P.Mutacion	Población	Iteraciones	Torneo	Elitismo	C1	C2	C3	M1	M2	M3
0.8	0.9	N	1000	3	Si	60%	20%	20%	0%	0%	100%

Los primeras seis columnas de la tabla muestra los parámetros del algoritmo GA que convergieron, los últimos seis parámetros representan las probabilidades de los operadores de cruce (C) y mutación (M) que se usan para obtener mejores resultados, que para el caso de cruce los tres operadores juegan su posibilidad pero la cruce 1 tiene mayor posibilidad, en el caso de la mutación solo juega la posibilidad la mutación 3.

8 Resultados

En la siguiente sección se muestra un conjunto de gráficas las cuales muestran un análisis de los resultados del algoritmo genético.

10 instancias para el problema CMST para el Algoritmo Genético

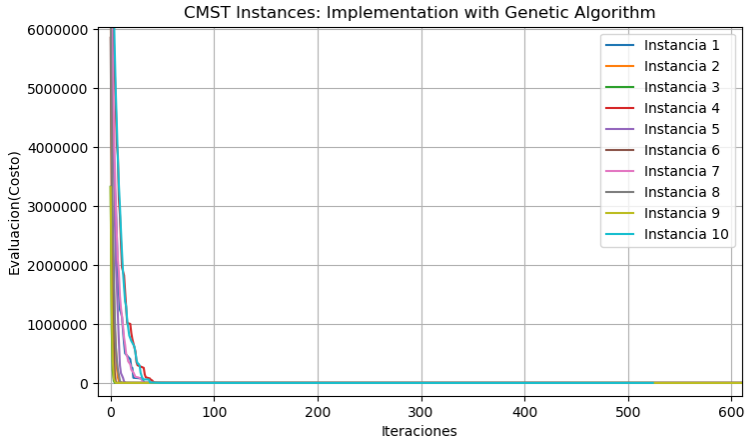


Fig. 4. La siguiente gráfica muestra el desempeño del algoritmo para la 10 instancias de prueba. Se puede ver claramente que el AG realiza pocas iteraciones debido a que la mayoría del tiempo encuentra mejores soluciones y encuentra al final el valor mas bajo a diferencia de los otros. El Algoritmo Genético es el que muestra menor calidad de resultados ya que obtiene una convergencia prematura.

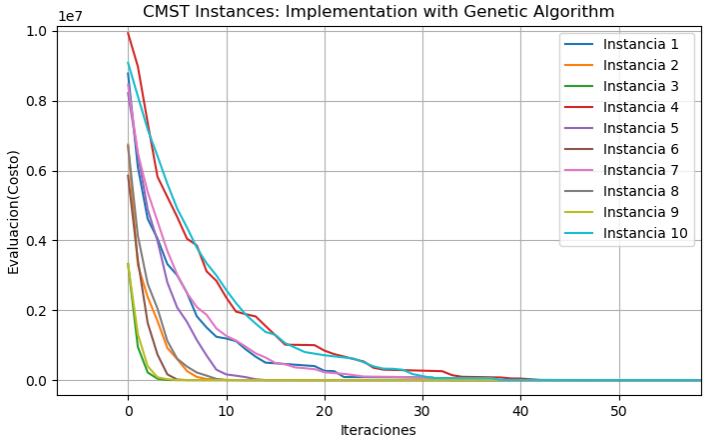


Fig. 5. La siguiente gráfica muestra una perspectiva mas amplia de la gráfica anterior donde se puede ver mas claramente los valores aproximados de cada una de las instancias.

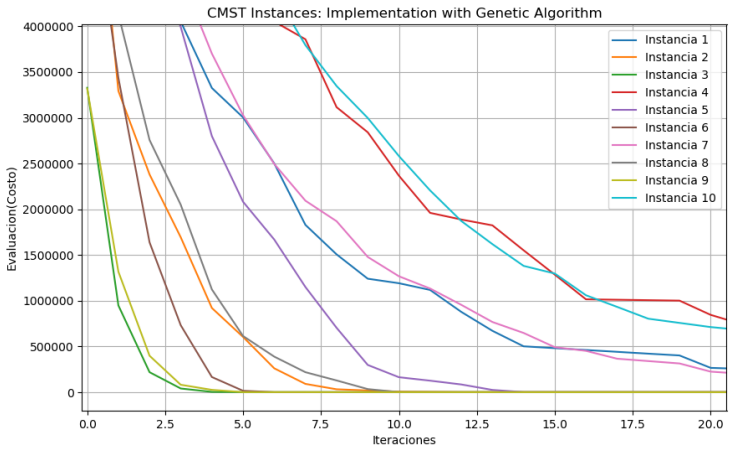


Fig. 6. Aqui se ve mas el comportamiento de cada instancia.

8.1 Análisis de Resultados

En la siguiente tabla se muestran los mejores resultados conocidos del algoritmo AG en comparación con otras técnicas implementadas para este problema usando las 10 instancias:

	Instancia	Num.Nodos	Nodo Central	Q (Capacidad)	Blind Optim.	Sim.Annealing	Genetic Algorithm
1	cm50r1	50	0	200	695	558	1008
2	:	:	:	400	704	541	1036
3	:	:	:	800	704	541	909
4	cm50r1	50	15	20	709	541	11230
5	:	:	:	400	709	541	1099
6	:	:	:	800	695	558	1025
7	cm50r2	50	0	200	714	570	1593
8	:	:	:	400	729	559	1330
9	:	:	:	800	717	581	1120
10	cm50r2	50	25	200	744	559	1527

9 Conclusiones

En este documento se trabaja con el problema del árbol de costo mínimo capacitado (CMST). Dado un árbol con un nodo concentrador el cual tiene un conjunto de nodos terminales cada uno con un valor de una demanda, a excepción del nodo central, todas las terminales están conectados al nodo concentrador ya que toda la información debe llegar hacia él. El objetivo del problema es encontrar una red entre los nodos de tal forma que el costo de sus flujos sea el más mínimo posible. Cada enlace entre las terminales consta de un costo, por lo que la restricción de este problema requiere que la suma de los costos de los subredes conectadas al nodo central no supere una capacidad de Q. Para el CMST se planteo hacer una representación con secuencias de arboles usando el método de codificación de Neville para la conversión de secuencia a árbol y de árbol a secuencia ya que era indispensable usar las dos formas para generar correctamente las soluciones. Una vez implementado cada algoritmo se realizo un método para sintonizar un conjunto de parámetros de entrada para realizar pruebas y así saber que combinación obtiene mejores resultados. Después de la experimentación, y ya teniendo los parámetros adecuados, se obtuvieron resultados razonables.

References

[1] M Kritikos and George Ioannou. "A greedy heuristic for the capacitated minimum spanning tree problem". In: *Journal of the Operational Research Society* 68.10 (2017), pp. 1223–1235.

[2] Manolis N Kritikos and George Ioannou. "Two heuristics for the capacitated minimum spanning tree problem with time windows". In: *Journal of the Operational Research Society* 70.4 (2019), pp. 555–567.

[3] Efrain Ruiz et al. "A biased random-key genetic algorithm for the capacitated minimum spanning tree problem". In: *Computers & Operations Research* 57 (2015), pp. 95–108.

[4] Cesar Rego, Frank Mathew, and Fred Glover. "Ramp for the capacitated minimum spanning tree problem". In: *Annals of Operations Research* 181.1 (2010), pp. 661–681.

[5] Christos A Pappas et al. "Heuristics for the multi-level capacitated minimum spanning tree problem". In: *Optimization Letters* 8.2 (2014), pp. 435–446.

[6] Ángel Kuri, José Galaviz, et al. *Algoritmos genéticos*. Sirsi) i9789681663834. IPN, 2002.

[7] Julián; Pazos Alejandro) Marcos; Rivero Gestal (Daniel; Rabuñal Juan Ramón; Dorado and Marcos Gestal. *Introducción a los algoritmos genéticos y la programación genética*. Universidade da Coruña, 2010.

[8] DE Goldberg. "Genetic algorithms in search, optimization, and machine learning, addison-wesley, reading, ma, 1989". In: *NN Schraudolph and J* 3.1 (1989).

[9] Marcos Gestal Pose. “Introducción a los algoritmos genéticos”. In: *Departamento de Tecnologías de la Información y las Comunicaciones Universidad de Coruña* (2000).

[10] Jorge Arranz de la Peña and Antonio Parra Truyol. “Algoritmos genéticos”. In: *Universidad Carlos III* (2007).

[11] Pablo Estévez Valencia. “Optimización mediante algoritmos genéticos”. In: *Anales del Instituto de Ingenieros de Chile*. Vol. 109. 2. 1997, pp. 83–92.

[12] Esteban Manuel Gil Sagás. “Programación de la generación de corto plazo en sistemas hidrotérmicos usando algoritmos genéticos”. In: *MsC Tesis, Universidad Técnica Federico Santa María* (2001).