Minimización de la Distancia a una Estación de Emergencia mediante Algoritmos Genéticos

Marisol Calderón González

Universidad Tecnológica de Huejotzingo, Camino Real a San Mateo s/n Santa Ana Xalmimilulco, Huejotzingo, Puebla, (01-227) 2759300

marisol_c@hotmail.com

María Luisa Morales Hernández

Universidad Tecnológica de Huejotzingo, Camino Real a San Mateo s/n Santa Ana Xalmimilulco, Huejotzingo, Puebla, (01-227) 2759300

marialmor@hotmail.com

José David Alanis Urquieta

Universidad Tecnológica de Puebla, Antiguo Camino a la Resurrección No. 1002-A, Zona Industrial Oriente, Puebla, Puebla (01-222) 309.88.61

david.alanis@utpuebla.edu.mx

Griselda Saldaña González

Universidad Tecnológica de Puebla, Antiguo Camino a la Resurrección No. 1002-A, Zona Industrial Oriente, Puebla, Puebla (01-222) 309.88.61

griselda.saldana@utpuebla.edu.mx

Resumen

En este artículo se presenta la solución al problema de la minimización de la distancia a una estación de emergencia mediante algoritmos genéticos. El objetivo del algoritmo es señalar un lugar para la colocación de una estación en una ciudad y reducir el tiempo de respuesta a las emergencias generadas en la misma. Lo anterior se representa colocando una matriz que representa a la ciudad y sus secciones, utilizando coordenadas cartesianas para su representación. La instrumentación de la solución se llevó a cabo en MATLAB con el fin de obtener resultados que tengan una buena aproximación con un

grado alto de complejidad. Se ha realizado la implementación de los algoritmos siguiendo tres enfoques básicos, un enfoque de cromosomas continuos, cromosomas binarios y tradicionales.

Las generaciones han demostrado tener buenos resultados y se ha respetado la factibilidad del problema.

Palabras Claves: Algoritmo Genético, Cromosoma, Estación, Matlab

1. Introducción

El problema de encontrar la ruta más corta, mejor conocido como el algoritmo de Dijkstra, ya sea planteado como un problema de grafos ó de optimización combinatoria [6], consiste en partir de un origen determinado y llegar a uno fijo, además de requerir que la ruta tenga el mínimo peso. Otro problema de optimización combinatoria tradicional es el problema del agente viajero, que consiste en visitar desde un origen un conjunto de ciudades [6]. La solución de ambos problemas se plantea en la bibliografía de muchas formas y por diferentes métodos [1, 8].

El presente trabajo tiene como objetivo obtener la posición de una estación de emergencia en una población, de tal forma que se obtenga la distancia mínima cuando se genere una incidencia que requiera dicha estación. Para lograr esto, se plantea el problema como un problema de optimización combinatoria y se resuelve mediante algoritmos genéticos. Estos algoritmos se implementan de dos formas en representación binaria y continua.

Se pretende que las estaciones de emergencia sean Unidades de Respuesta a Emergencias tales como Unidades Médicas, Unidades de Bomberos, Unidades de Policía, etc., que mejoren el servicio a una Ciudad.

En función al problema que se plantea en el presente trabajo se asemeja a los mencionados anteriormente. Una combinación de ambos enfoques podría verse como el tener distintos orígenes con un destino único y fijo, pero con la ruta mínima.

El problema a resolver en este trabajo, fue inspirado por los hermanos Haupt [13] quiénes plantean algo muy similar pero sin obstáculos y considerando todas las rutas como posibles. Además de todo lo anterior se sabe que la solución por métodos combinatorios resulta cara computacionalmente hablando y en muchos casos puede complicarse demasiado.

Después de una investigación de campo, pero no exhaustiva de trabajos similares o relacionados con la minimización o reducción de la distancia a una estación de emergencia dado un conjunto de orígenes fijos, no ha sido reportada. Puede haber problemas específicos y parecidos en algunos aspectos como en [8], donde se discuten formas de solucionar con técnicas heurísticas de diversa índole las soluciones a problemáticas clásicas. Sin embargo como tal y específicamente hablando de la cuestión que ocupa este trabajo, no parece haber trabajo realizado al respecto con las técnicas que aquí se plantean.

En los últimos años, la comunidad científica internacional ha demostrado un creciente interés por resolver problemas de búsqueda y optimización a través de una técnica basada en la teoría de la evolución y que se conoce con el nombre de Algoritmo Genético.

Un Algoritmo Genético está basado en el proceso genético de los organismos vivos. A lo largo de generaciones, las poblaciones evolucionan en la naturaleza de acuerdo con los principios de selección natural y la supervivencia. Las especies que sobreviven no son las más fuertes, ni las más inteligentes, sino las que se logran adaptarse al medio o a transformarlo, esto fué postulado por Darwin [3]. Por imitación de este proceso, este método de programación es capaz de ir creando soluciones hacia valores óptimos en problemas del mundo real [1].

El algoritmo Genético trabaja con una población de individuos, para cada generación, un individuo representa una solución potencial para el problema dado. Cada solución es

evaluada en la función de aptitud o función objetivo para asignarle un grado de aptitud, de estas soluciones se eligen a los individuos más aptos para formar una nueva población, algunos de los miembros de la nueva población sufrirán transformaciones por medio de operadores genéticos para formar nuevas soluciones, estas transformaciones crean nuevos individuos por un pequeño cambio en un solo individuo (mutación), o combinando su material genético con otros individuos (cruzamiento), se espera que después de algún número de generaciones la población convergerá hacia una solución con un alto grado de aproximación a la solución óptima [1].

En este trabajo en la primera sección se describen los algoritmos genéticos y su representación tanto combinatoria como continua. Seguidamente se plantea el problema de optimización lineal combinatoria. En la segunda sección se muestra la implementación computacional de los algoritmos antes mencionados, así como la representación de la ciudad y sus posibilidades para la posición de una estación de emergencia, mediante un ejemplo. En la tercera sección, se muestran los resultados y experimentos obtenidos de las implementaciones y el planteamiento antes mencionado.

Por último se realizan las conclusiones pertinentes al trabajo, haciendo una discusión de los resultados y métodos utilizados. Terminado con las conclusiones obtenidas, los trabajos futuros y las aportaciones del trabajo de manera multidisciplinaria.

2. Desarrollo

La implementación computacional se desarrolló en base a la teoría de los algoritmos genéticos que a continuación se expone.

Algoritmos Genéticos

Componentes de los algoritmos genéticos

Función de Aptitud

La función de aptitud es la función objetivo de un problema de optimización. *i.e*. $\min f(x) = \max\big(g(x)\big) = \max\{(-f(x)).\}$ [14].

Selección de Variables

En la selección de variables, se inicia definiendo un cromosoma escrito como un arreglo de variables. Donde el cromosoma tiene *k* variables dadas por $x_1, x_2, x_3,\ldots,x_K$, *cromosoma* $=[x_1, x_2, x_3,\ldots,x_K]$. Cada cromosoma es evaluado en la función objetivo, $aptitud = f(cromosoma) = f(x_1, x_2, x_3, \ldots\ldots x_k)$ [13].

Representación de Variables

Las variables denominadas cromosomas pueden ser representado de dos formas: binaria y real.

Representación Binaria

Supongamos que queremos maximizar una función max *f*(*x*),

donde $x = (x_1, x_2, x_3,..,x_k)$. Cada variable real x_i, $i=1,2,...,k$, puede tomar valores en un intervalo cerrado $D_i = [a_i, b_i] \subseteq R$ y $f(x_1, x_2, x_3, ..., x_k) > 0$, $\forall x_i \varepsilon D_i$.

Se definen los cromosomas $x= (x_1, x_2, x_3,..,x_k)$ donde cada variable x_i es real, los cuales se representan por vectores binarios de longitud $m = \sum_{i=1}^{k} m_i$ Esta longitud *m* del vector depende de la longitud m_i de cada componente $\hat{X}_i$

Cada cadena binaria $\hat{X}_i$ de longitud m_i representa un valor codificado de la variable real x_i en [a_i , b_i], i=1,2,...,k. Para decodificar el cromosoma v se usa la siguiente ecuación:

$x_i = a_i + decimal\left(\hat{X}_i\right) * (b_i - a_i)/\left(2^{mi} - 1\right)$ [14].

El valor de aptitud se denotará por *eval* (v_i).

i.e. eval (v_i)= *f* (x)= *f* (x_1, x_2,...,x_k).

Representación real

En este caso, cada una de las variables o cromosomas del problema está representada por números en punto flotante [13].

Población Inicial

Se inicia construyendo una población inicial de cromosomas, y a partir de esta se generan nuevas poblaciones [13], está puede ser representada en forma binaria y real.

Representación binaria

Esta población inicial tiene *pop_size* cromosomas construido de ceros y unos con una longitud *m* [14].

Otra técnica de representación de los parámetros en la implementación computacional de los algoritmos genéticos es Código Gray que redefine el número binario de manera que números consecutivos tengan una distancia Hamming de uno, la cual está definida por el número de posiciones de bits diferentes [13].

Representación Real

La población inicial se puede generar a través de la siguiente matriz:

$IPOP = (b_i - a_i) * \text{random}\{ pop\_size, k \} + a_i;\ i = 1, 2,...,k.$

Donde $x_i \varepsilon [a_i, b_i]$, $i = 1, 2,...,k$

random{*pop_size*,*k*} es una función que genera una matriz de *pop_size* * *k* números uniformes aleatorios entre cero y uno [13].

Selección de Nueva Población

Para la selección de la nueva población se utiliza el método por ruleta que consiste en girar la rueda de la ruleta *pop_size* veces para determinar que cromosomas serán seleccionado, para esto se procede de la siguiente manera:

- Calcular el valor de aptitud *eval* (v_i) para cada cromosoma v_i, (i=1,2,...,*pop_size*)
- Encontrar la aptitud total de la población $F = \sum_{i=1}^{pop\_size} eval(v_i)$
- Calcular la probabilidad de selección p_i dada por: $p_i = eval(v_i)/F$, para cada cromosoma v_i , (i=1,2,...,*pop_size*)
- Calcular la probabilidad acumulativa q_i, $q_i = \sum_{j=1}^{i} p_j$ para cada cromosoma v_i, (i=1,2,...,*pop_size*

Cada vez que se selecciona un cromosoma para una nueva población se realiza lo siguiente:

- Generar un número aleatorio (float) *r* del rango [0,1],
- Si $r < q_1$ entonces seleccionamos al primer cromosoma (v_1); si $r \geq q_1$, seleccionamos el i-ésimo cromosoma v_i tal que $q_{i-1} < r \leq q_i$, $2 \leq i \leq pop\_size$ [14].

Cruzamiento o Reproducción Sexual

Para el cruzamiento, al inicio del algoritmo se da el valor $p_c \in (0,1)$ que representa el porcentaje de la población que sufrirá la operación de cruzamiento, y se procede de la siguiente manera:

Para cada cromosoma en la población, se genera un número aleatorio *r* (float) del rango [0,1]. Si $r < p_c$ seleccionamos al cromosoma dado para cruzamiento.

El método de cruzamiento que se utiliza es el punto único de cruza, en el cual para cada pareja de cromosomas se genera un número entero aleatorio positivo ("pos", punto de cruza) en el rango $[1, \ldots, \bar{m} - 1]$, (m es la longitud total de números de bits en un cromosoma) [14] (ver Fig. 1).

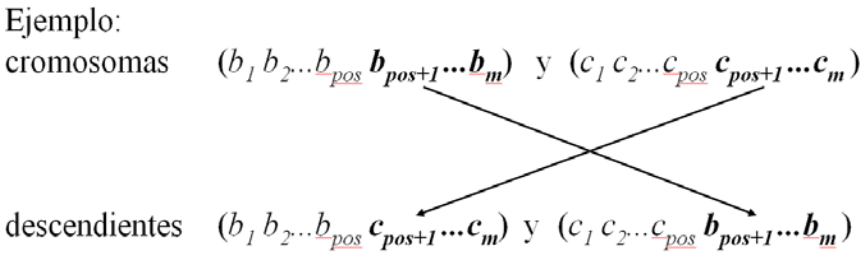

Fig. 1. Ejemplo de cruzamiento en representación binaria

Mutación

Para esto, se proporciona la probabilidad de mutación en la población, $p_m \in [0,1]$ Se genera un número aleatorio *r* (float) del rango [0,1]. Si $r < p_m$, se muta el bit. Si $r \geq p_m$, queda igual [14].

Otro método de mutación que se utilizó para el cruzamiento en este trabajo es elitismo que copia el mejor individuo a la siguiente generación [2].

Convergencia

La condición de terminación puede definirse utilizando dos criterios principales de terminación. Correr el algoritmo genético durante un número máximo de generaciones [6] o detenerlo cuando la población se haya estabilizado (es decir, cuando todos o la mayoría de los individuos tengan la misma aptitud) [13].

Planteamiento del problema

En este trabajo se localizará la ubicación de una estación de emergencia (estación de policía, estación médica, estación de bomberos, etc.) para mejorar los servicios de una ciudad.

El objetivo es establecer el punto de una ciudad donde se colocará la estación de tal manera que, se minimice el tiempo de respuesta a una llamada de emergencia que puede ocurrir en cualquier punto de la ciudad. Para esto, se hace un estudio de las últimas emergencias y se construye un mapa mostrando la frecuencia de las mismas en cada sección dada una ciudad. Este problema consiste en encontrar la localización de una estación de bomberos en una comunidad establecida en una cuadricula cuadrada de 6*6 [13].

En este estudio se incrementará el tamaño de la cuadrícula y se agregarán restricciones.

Se dividirá la ciudad en una cuadricula de 10 *10 km^2 con 100 secciones, y en cada sección se mostrará la frecuencia de llamadas.

Las restricciones que se incluyen en este problema serán un río horizontal con solamente dos puentes que lo cruzan, el río está localizado en $y = 6$ y el cruce de puentes en $x = 1.5$ y $x = 6.5$.

Se denotará por A al puente que se encuentra localizado en $x = 1.5$ y $y = 6$, y por *B* al puente que se encuentra localizado en $x = 6.5$ y $y = 6$. (ver Fig. 2)

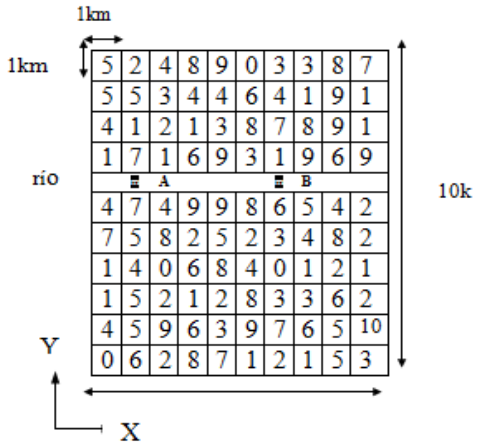

Fig. 2. Representación de la ciudad con río

Se deberá encontrar un par de números (x_{props} , y_{props}) que haga más corta la suma de las distancias entre las coordenadas propuestas de la estación y las coordenadas del centro de cada cuadrado *n.*

Si (x_{props} , y_{props}) son las coordenadas propuestas de la estación de emergencia, se deberá analizar si éstas se encuentran arriba o abajo del río.

Primer caso: si (x_{props} , y_{props}) se encuentra arriba del río, es decir 6 <= y_{props} <= 10, entonces para *n*=61,...,100, una función apropiada es la suma de las distancias pesadas por la frecuencia de emergencias.

$$aptitud(x_{props}, y_{props}) = \sum_{n=1}^{100} w_n \sqrt{(x_n - x_{props})^2 + (y_n - y_{props})^2}$$

Para n = 1,..., 60, se tomará la menor de las dos distancias pesadas a los cuadrados *n* pasando por los dos puentes.

$$\text{i)- DP1} = w_n \left[d\left((x_{props}, y_{props}), A \right) + d(A, (x_n, y_n)) \right]$$

$$= w_n \left[\sqrt{(1.5 - x_{props})^2 + (6 - y_{props})^2} + \sqrt{(x_n - 1.5)^2 + (y_n - 6)^2} \right]$$

ii) DP2=$w_n \left[d\left((x_{props}, y_{props}), B \right) + d(B, (x_n, y_n)) \right]$

$$= w_n \left[\sqrt{(6.5 - x_{props})^2 + (6 - y_{props})^2} + \sqrt{(x_n - 6.5)^2 + (y_n - 6)^2} \right]$$

Así, se denotará a la menor de las distancias i) y ii) por:

$d\left((x_{props}, y_{props}), (x_n, y_n) \right)$ n=1,…,60. Entonces la función de aptitud queda

$$\textit{aptitud } (x_{props}, y_{props}) = \sum_{n=61}^{100} w_n \sqrt{(x_n - x_{props})^2 + (y_n - y_{\text{props}})^2} + \sum_{n=1}^{60} d\left((x_{props}, y_{props}), (x_n, y_n) \right)$$

Segundo caso: Si (x_{props} , y_{props}) se encuentran abajo del río, es decir 0 <= y_{props} <= 6 entonces

$$\textit{aptitud } (x_{props}, y_{props}) = \sum_{n=61}^{100} d\left((x_{props}, y_{props}), (x_n, y_n) \right) + \sum_{n=1}^{60} w_n \sqrt{(x_n - x_{props})^2 + (y_n - y_{props})^2}$$

donde

(x_{props} , y_{props}) = coordenadas del centro del cuadrado n

(x_{props} , y_{props})= coordenadas propuestas de la estación

W_n= frecuencia de llamadas en el cuadrado n

Implementación computacional

El objetivo del desarrollo de este software es implementar un algoritmo genético para encontrar una solución o un óptimo.

El Software de Algoritmos de Búsqueda (SAO) desarrollado en Matlab (Versión. 8.2) R2012a [4, 5, 9, 10, 12], es un programa que utiliza dos técnicas de búsqueda, algoritmos genéticos y una técnica tradicional, para encontrar una solución mínima del problema de optimización planteado anteriormente. Principalmente se programó a los algoritmos

genéticos binarios y continuos, eligiendo desde una ventana principal alguna de las dos técnicas de búsqueda. Asimismo se incluye al programa una técnica tradicional de búsqueda, a partir de la siguiente secuencia de pasos.

- Seleccionar el algoritmo que se utilizará
- Binarios, Continuos o Técnica tradicional
- Definir los parámetros de entrada
- Elegir el tipo de técnica: Elitismo, No elitismo, Código Gray
- Salida de resultados

El software solicitará que algoritmo de búsqueda utilizará, así como los datos necesarios para obtener una solución óptima a través de los datos de entrada que el usuario proporcione.

A continuación se muestran la operación del programa SAO.

En la pantalla de inicio se muestran las tres técnicas de búsqueda del programa que son: algoritmo genético binario, algoritmo genético de parámetros continuos y técnica tradicional del software de MATLAB (ver Fig. 3).

Al seleccionar una de las opciones de la pantalla principal, el usuario podrá capturar, número de cromosomas (nc), número de generaciones (generaciones), precisión decimal (pd), probabilidad de cruzamiento (pm) y probabilidad de mutación (pm).

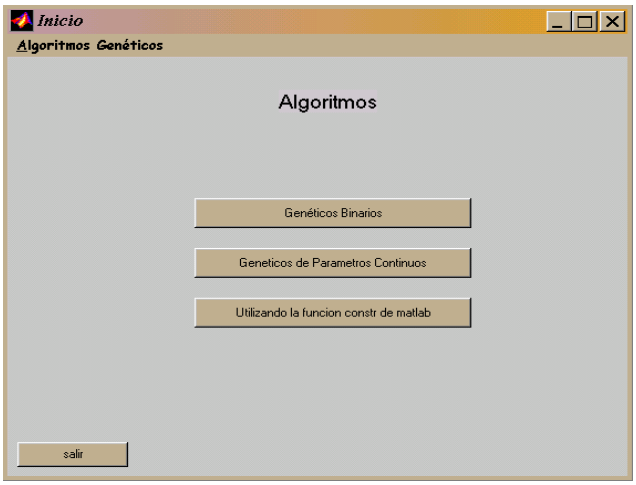

Fig. 3. Pantalla principal de SAO

Así como también se elegirá la técnica que se desea utilizar para dicho algoritmo: elitismo, no elitismo, ó código gray (ver Fig.4), después aparecerá una pantalla que mostrará los resultados de salida. (ver Fig. 5).

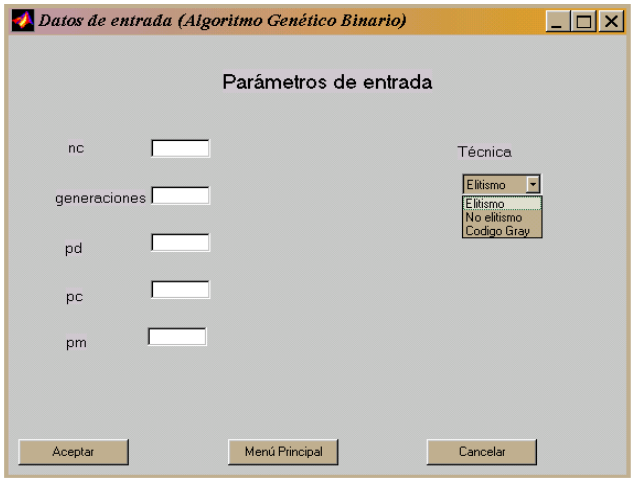

Fig. 4. Pantalla de entrada de parámetros

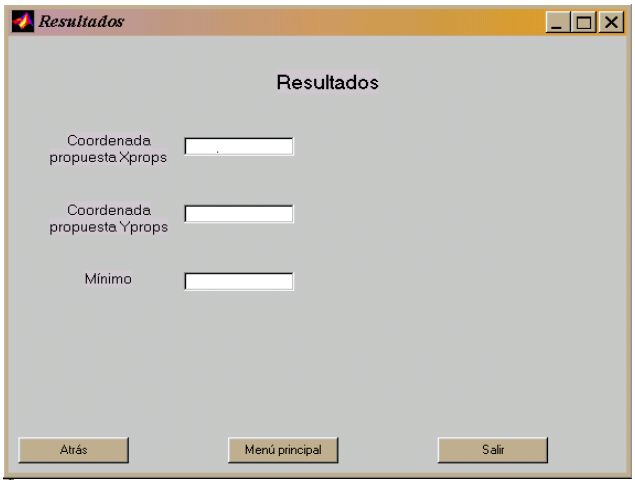

Fig. 5. Pantalla visualización de resultados

3. Resultados

Para el análisis de resultados se denota por:

AGB	**Algoritmo genético binario**
AGBE	**Algoritmo genético binario con elitismo**
AGBCG	**Algoritmo genético binario usando código gray**
AGC	**Algoritmo genético de parámetros continuos**
AGCE	**Algoritmo genético de parámetros continuos con elitismo**
AMCR	**Programa de optimización con restricciones de la aplicación de Matlab**.

A continuación se muestran los resultados obtenidos de las implementaciones de los programas de algoritmos genéticos, así como el análisis comparativo de los mismos con el método Constrained Minimization (Función fminsearchbnd) del toolbox de optimización de MATLAB [11].

Cada algoritmo se ejecuta hasta en 1000 generaciones con un tamaño de población mayor o igual a 20 cromosomas, un porcentaje de mutación menor o igual del 4% y un porcentaje de cruzamiento igual al 50%.

Comparando los resultados obtenidos de los algoritmos genéticos utilizando un número de 20, 30 y 40 cromosomas, un porcentaje de mutación del 4% y 3%, y un número de generaciones igual a 25, 50, 100, 500 y 1000 con el resultado obtenido del método Constrained Minimization, resulta que en el 98% de ejecuciones, los algoritmos genéticos lo superan en exactitud, sea cual fuere el método que se elija (ver tabla 1, 2). Consideramos que éste es uno de los resultados más importantes que se obtuvieron en este trabajo.

ALGORITMOS	***nc***	P_m	**Generaciones**	**Solución**		
				x_{props}	y_{props}	**mínimo**
AGB	500	0.04	500	6.4160	5.8374	1778.4921
AGBE	300	0.03	300	6.4075	5.7943	1778.3969
AGBCG	1000	0.04	1000	6.4010	5.7977	1778.4193
AGC	500	0.03	500	6.4045	5.7911	1778.3979
AGCE	1000	0.03	1000	6.4075	5.7932	1778.3971

Tabla 1. Comparativa de los mejores resultados de algoritmos genéticos con el método Constrained Minimization

ALGORITMOS	Solución		
	x_{props}	y_{props}	Mínimo
AGBE	**6.4075**	**5.7943**	**1778.3969**
AGCE	6.4075	5.7932	1778.3971
AMCR	6.2849	5.5847	1778.7040

Tabla 2. Mejor resultado en cada algoritmo

4. Discusión

De acuerdo a los resultados obtenidos se observa que en los algoritmos genéticos utilizando la técnica de elitismo se obtiene el mejor resultado con un tamaño de población igual a 20 cromosomas y un número de generaciones igual a 300. Es importante mencionar que si el número de cromosomas es mayor se obtienen los mismos resultados con un número de generaciones muy grande. Con respecto a la probabilidad de mutación los algoritmos genéticos binarios trabajan mejor con 0.03 y los algoritmos genéticos continuos con 0.04 (ver tabla 3,4,5,6,7,8).

ALGORITMOS	nc	P_m	Generaciones	Solución		
				x_{props}	y_{props}	mínimo
AGB	**20**	**0.04**	300	6.3752	5.6158	1779.4557
AGBE	**20**	**0.04**	300	6.4112	5.7782	1778.4437
AGBCG	**20**	**0.04**	500	6.3835	5.4950	1779.2552
AGC	**20**	**0.04**	500	6.3928	5.7701	1778.4198
AGCE	**20**	**0.04**	300	6.4115	5.8013	1778.3992

Tabla 3

ALGORITMOS	nc	P_m	Generaciones	Solución		
				x_{props}	y_{props}	mínimo
AGB	30	0.03	300	6.3146	5.6259	1778.6852
AGBE	30	0.03	300	6.4061	5.7928	1778.3970
AGBCG	30	0.03	1000	6.4010	5.7977	1778.4193
AGC	30	0.03	500	6.3737	5.7252	1778.5479
AGCE	30	0.03	1000	6.4075	5.7932	1778.3971

Tabla 4

ALGORITMOS	nc	P_m	Generaciones	Solución		
				x_{props}	y_{props}	MÍNIMO
AGB	30	0.04	500	6.4160	5.8374	1778.4921
AGBE	30	0.04	1000	6.4048	5.7997	1778.3975
AGBCG	30	0.04	1000	6.4010	5.7977	1778.4193
AGC	30	0.04	1000	6.3948	5.7639	1778.4233
AGCE	30	0.04	100	6.3901	5.7760	1778.4372

Tabla 5

ALGORITMOS	nc	P_m	Generaciones	Solución		
				x_{props}	y_{props}	mínimo
AGB	40	0.04	1000	6.3484	5.7031	1778.7736
AGBE	40	0.04	1000	6.4087	5.7966	1778.3972
AGBCG	40	0.04	50	6.4169	5.8258	1778.4319
AGC	40	0.04	1000	6.3822	5.7462	1778.4732
AGCE	40	0.04	1000	6.4064	5.7843	1778.4034

Tabla 6

ALGORITMOS	nc	P_m	Generaciones	Solución		
				x_{props}	y_{props}	mínimo
AGB	20	0.03	1000	6.4027	5.8058	1778.4348
AGBE	20	0.03	300	6.4075	5.7943	1778.3969
AGBCG	20	0.03	1000	6.3620	5.7094	1778.6478
AGC	20	0.03	1000	6.4027	5.8058	1778.4348
AGCE	20	0.03	1000	6.4085	5.8270	1778.4011

Tabla 7

ALGORITMOS	nc	P_m	Generaciones	Solución		
				x_{props}	y_{props}	mínimo
AGB	40	0.03	100	6.4148	5.7943	1778.4181
AGBE	40	0.03	1000	6.4061	5.7927	1778.3970
AGBCG	40	0.03	500	6.3855	5.7120	1778.6373
AGC	40	0.03	500	6.4045	5.7911	1778.3979
AGCE	40	0.03	1000	6.4090	5.7769	1778.4333

Tabla 8

Mejores resultados obtenidos en todas las ejecuciones

5. Conclusiones

Los AG trabajaron mejor con una probabilidad de mutación de .04 y con un número de cromosomas de 20. Los AG trabajaron mejor sin elitismo. Contrario a lo que se pensaba el AGB dio mejores resultados que el AGBCG. Cuando el número de generaciones varía entre 50 y 100 los AG superan los resultados de AMCR. Cuando el número de generaciones oscila entre 300 y 500 se obtienen los mejores resultados. Cuando el número es mayor de 1000 se mejora la exactitud sólo en decimales.

El algoritmo AMCR se ejecutó con una complejidad temporal menor que cualquiera de los algoritmos genéticos usado. En cuanto a la precisión decimal, se obtuvieron mejores resultados en los AGB, sin embargo, cualquier AG superó en precisión al algoritmo AMCR.

Se ha presentado la forma de implementación de un algoritmo genético programado en Matlab, con la utilización de los Toolbox pertinentes en la construcción de un software que resuelve el problema de colocar una estación de emergencia en una ciudad con el mínimo costo y la menor distancia. Se observa que el resultado x_{props} *=6.4075,*

y_{props}*=5.7943 con un mínimo de 1778.3069* ha resultado ser el más óptimo para este problema en particular y que la implementación ha resultado adecuada. El uso de la representación binaria tiene mejores resultados que la representación real, además de apoyarse en el código gray y elitismo. Con respecto al número de cromosomas se observa una mejora al aumentar la cantidad, hasta el punto donde con 1000 se estabiliza y la convergencia no obtiene una mejora sustantiva.

En trabajos futuros se puede realizar el mismo algoritmo con Java aprovechando el software libre, las bondades de la programación orientada a objetos, entre otras [7]. La viabilidad del proyecto también se puede aplicar a una ciudad del mundo real y demostrar la utilidad práctica del proyecto.

6. Referencias

[1] A. Diaz Fernández, Optimización Heurística y Redes Neuronales. 1° Edición. 2000. Paraninfo. Madrid, España. Páginas 159.

[2] A. E. Eiben, J.E. Smith, Introduction to Evolutionary Computing. 3ra. Edition. 2003. Springer. New York. Página 81.

[3] C. A. Villee, Biología. Octava edición. 1999. McGraw-Hill. México. Página 632.

[4] C. Pérez, Matlab y sus aplicaciones en las Ciencias y la Ingeniería. Primera Edición. 2002. Prentice Hall. Madrid. Páginas 45- 52, 54-58, 63, 68, 79-84.

[5] D. Báez López, O. Cervantes VillaGómez, MATLAB Con Aplicaciones a la Ingeniería, Física y Finanzas. 2a. Edición. 2012. Alfaomega. México D.F. Páginas 51-59, 94-121.

[6] F. S. Hillier, G. J. Lieberman. Introducción a la INVESTIGACIÓN DE OPERACIONES. Novena Edición. 2010. McGraw-Hill. México D.F. Página 591.

[7] J. D. Alanís Urquieta, M. Calderón González, M. L. Morales Hernández, G. Saldaña González, Reducción de la Distancia a Estaciones de a Estaciones de Emergencia mediante Algoritmos Genéticos, XXIV ENOAN, Escuela Nacional de Análisis Numérico PONENCIA SIN PUBLICACIÓN, Guanajuato, Gto., México 2014

[8] L. D. Chambers, Practical Handbook of Genetic Algorithms Complex Coding System. 2000. CRC Press.

[9] MathWorks. http://www.mathworks.com/products/new_products/release2013a.html. Abril 2013.

[10] MathWorks. http://www.mathworks.com/matlabcentral/fileexchange/27758-gui-layout-toolbox. Mayo 2013.

[11] MathWorks. https://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd--fminsearchcon. Mayo 2013.

[12] MathWorks, https://www.mathworks.com/matlabcentral/fileexchange/12122-handbook-of-graphical-user-interface--spanish--. Julio 2013

[13] R. L Haupt and S. E. Haupt, Practical Genetic Algorithms. 2da. Edition. 1998 John Willey & Sons. New York. Páginas 29-43,51-52,88-90.

[14] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Third Edition. 1999. Springer-Verlag. Berlin, Heidelberg, New York. Páginas 33-44.

7. Autores

Marisol Calderón González, tiene el título de Licenciado en Ciencias de la Computación por la Benemérita Universidad Autónoma de Puebla y la Maestría en Sistemas Computacionales por la Universidad Popular Autónoma del Estado de Puebla.

José David Alanís Urquieta, es Licenciado en Ciencias de la Computación por la Benemérita Universidad Autónoma de Puebla, Maestro en Ciencias de la computación con especialidad en cómputo científico. La candidatura a Doctor en Tecnologías de la Información y Comunicación por la Universidad Popular Autónoma del Estado de Puebla.

María Luisa Morales Hernández, tiene el título de Licenciado en Ciencias de la Computación por la Benemérita Universidad Autónoma de Puebla y la Maestría en Ingeniería Administrativa.

Griselda Saldaña González, tiene la Licenciatura en Electrónica por la Benemérita Universidad Autónoma de Puebla, la Maestría en Electrónica por la Universidad de las Américas Puebla y el Doctorado en Ciencias Computacionales por el Instituto Nacional de Astrofísica Óptica y Electrónica.