Uso de técnicas de computación evolutiva en problemas de localización de instalaciones

Edgar Duarte-Forero

Bogotá, Colombia

Abstract

Los problemas de localización de instalaciones *(Facility Location Problems - FLP)* buscan la definición de la ubicación de varias instalaciones para brindar servicio a un conjunto de clientes, cumpliendo con restricciones de capacidad y buscando alcanzar medidas de desempeño como el costo total, la máxima distancia recorrida, entre otros. Este tipo de problemas es reconocido como NP complejos y por lo tanto requieren del uso de métodos heurísticos para encontrar soluciones aceptables. En este artículo se buscará explorar el uso de Algoritmos Evolutivos en problemas FLP a través de dos estrategias: algoritmos genéticos y estrategias evolutivas. El propósito del artículo consiste en evaluar la aplicabilidad de estas técnicas con miras a posteriores implementaciones en escenarios más específicos y complejos de investigación.

Keywords: Localización, Cobertura máxima, Algoritmos evolutivos, p-centro

1. Introducción

La ciencia de la localización *(Location Science)* estudia la toma de decisiones acerca de la configuración de redes de instalaciones que brindan servicios a clientes dentro de un área específica de trabajo. En este artículo se hará énfasis en técnicas para optimizar la localización de instalaciones que prestan servicios de carácter logístico: almacenes, puntos de venta, hospitales, entre otros. Algunas preguntas que surgen para esta toma de decisiones incluyen [1]:

- ¿Cuántas instalaciones deberían ser puestas a funcionar en una red de servicios?

- ¿Dónde deberían ubicarse esas instalaciones?
- ¿Qué tan grandes deberían ser en términos de su capacidad de respuesta?
- ¿Cómo se podrían asignar clientes a esas instalaciones?

La literatura contempla tres clasificaciones principales de problemas dentro del campo de la Ciencia de la localización, los cuales dependen del tipo de función objetivo utilizada: minimización de la suma de distancias, cobertura de servicios y minimización de la máxima distancia [2]. Las clasificaciones se realizan de acuerdo con criterios como el espacio de localización (continuo, en red o discreto) o el campo económico de aplicación (público, privado o mixto). Una descripción detallada de los criterios para clasificar problemas de localización puede ser encontrada en Daskin [1].

Las aplicaciones de la ciencia de localización abarcan problemas del mundo real en áreas como logística, telecomunicaciones, ruteo de vehículos, salud o transporte[3]. En algunas de estas áreas el principal criterio para la toma de decisiones es la mejora en el servicio a los clientes. Se considera que el servicio es adecuado si los clientes están ubicados dentro del área de cobertura de al menos una instalación prestadora de servicios [1]. Esta "cobertura" es definida como la posibilidad de brindar servicios a un cliente dependiendo de la distancia en la que se encuentre.

El problema de localización con cobertura máxima *(Maximum Covering Location Problem -MCLP)* busca la definición de la ubicación de instalaciones para maximizar la cantidad de demanda cubierta (lo cual es distinto del número de nodos de demanda) [4]. Este enfoque considera que existen algunas restricciones para el número de instalaciones a ubicar y por lo tanto no es posible cubrir toda la demanda de clientes. MCLP es un problema NP duro y por lo tanto muchos enfoques heurísticos han sido propuestos para su solución, incluyendo métodos Lagrangianos y algoritmos voraces [3].

Otro de los problemas frecuentemente examinados en la literatura de localización de instalaciones es el denominado p-centro. En lugar de maximizar la demanda cubierta por los nodos de oferta, el enfoque del problema p-centro consiste en minimizar la distancia de cobertura tal que cada nodo de demanda esté separado por la mínima distancia posible a alguna de las

instalaciones que ofertan servicios. Este problema también es conocido como *minimax*, pues busca precisamente minimizar la máxima distancia entre un nodo de demanda y su instalación más cercana [1].

Los algoritmos evolutivos han sido propuestos para resolver problemas FLP dado que brindan soluciones cerca al óptimo. Estos algoritmos simulan el proceso de evolución natural de una especie para obtener mejores características en una población. En cada generación, algunos operadores son aplicados a la población de individuos (soluciones), modificando así su estructura genética para alcanzar un óptimo. Los operadores más utilizados son los de selección, mutación y cruce [5]. Como lo mencionan Beyer y Schwefel [6], existen tres fuentes contemporáneas de algoritmos evolutivos (EA) que han persistido en la literatura científica durante los últimos años: Programación evolutiva [7], Algoritmos genéticos (GA)[8] y Estrategias evolutivas (ES)[9].

El uso de Algoritmos evolutivos para resolver problemas FLP ha sido reportado por Jaramillo et.al. [10]. Su trabajo evaluó el desempeño de Algoritmos Genéticos como un procedimiento alternativo para resolver tres diferentes versiones de problemas de localización: problemas de costo fijo con restricciones de capacidad, problemas de cobertura máxima y problemas de localización competitiva. Sus conclusiones establecen que los algoritmos genéticos utilizan mucho más tiempo que las heurísticas especializadas pero que al mismo tiempo arrojan mejores soluciones. Más recientemente, Atta [11] se enfocó en el mejoramiento de Algoritmos Genéticos por la incorporación de estrategias de mejora locales en las soluciones halladas. Esta idea es aplicada utilizando Estrategias de refinamiento local *(Local Refinement Strategies)* a través del remplazo de la localización de cada instalación por aquel punto que tenga la menor suma ponderada de distancias a otros puntos dentro del *cluster* obtenido luego de cada iteración.

El principal objetivo de este artículo consiste en la exploración de la aplicación de diferentes paradigmas de Algoritmos evolutivos para resolver problemas de localización.[1] El resto del artículo está organizado como sigue:

[1]Este artículo ha sido desarrollado en el marco de la asignatura de Computación Evolutiva de la Universidad Nacional de Colombia, bajo la dirección del PhD. Jonatan Gómez P. Previo al desarrollo de estas implementaciones se desarrollaron varios avances cuyos contenidos pueden ser evidenciados en los anexos Apéndice˜B.1, Apéndice˜B.2, Apéndice˜B.3

La sección 2 presenta el uso de Algoritmos Genéticos par resolver el problema de máxima cobertura MCLP. La sección 3 presenta la aplicación de Estrategias Evolutivas para resolver un problema p-centro en donde intervienen variables reales[2]. Finalmente, la sección 6 concluye el artículo y presenta recomendaciones para futuros trabajos.

2. Aplicación de Algoritmos Genéticos para el problema MCLP

Para presentar la aplicación de Algoritmos Genéticos al problema de localización con cobertura máxima se hará inicialmente una descripción matemática del problema, posteriormente se abordará la forma como se resolvió utilizando Algoritmos Genéticos y haciendo énfasis en la representación genética, la función de utilidad y restricciones, los operadores genéticos utilizados, el flujograma del algoritmo genético implementado, la instancia del problema utilizada y finalmente los resultados obtenidos[3].

2.1. Problema MCLP

Como se mencionó previamente, este problema busca maximizar la demanda cubierta por las instalaciones puestas en servicio dentro de una distancia de cobertura definida. Daskin [1] presenta a continuación una formulación algebraica del problema:

P: Número de instalaciones a ubicar
J: Conjunto de todos los nodos de oferta posibles para ubicar las P instalaciones
I: Conjunto de todos los nodos de demanda a abastecer
h_i: Demanda en el nodo i

$$z_i = \begin{cases} 1 & \text{si el nodo de demanda } i \in I \text{ es cubierto por uno de oferta } j \in J \\ 0 & \text{si no.} \end{cases}$$

y , Apéndice˜B.4

[2]Estas dos secciones todavía están pendientes de ser documentadas

[3]El código de la implementación del algoritmo genético en Python puede ser encontrado en el repositorio `https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/AG%20Objetos%20MAXCP.py`

$$x_j = \begin{cases} 1 & \text{si una instalación se localiza en el nodo } j \in J \\ 0 & \text{si no.} \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \text{si un nodo de demanda } i \in I \text{ puede ser atendido por un nodo } j \in J \\ 0 & \text{si no.} \end{cases}$$

Los valores de a_{ij} dependen de la distancia de cobertura establecida para el problema. Un nodo i puede ser atendido por un nodo j si la distancia entre ambos es menor a la distancia de cobertura. El modelo matemático para asegurar la máxima cobertura posible queda planteado a continuación:

$$\text{maximizar} \quad \sum_{i \in I} h_i z_i \tag{1a}$$

$$\text{sujeto a} \quad \sum_{j \in J} a_{ij} x_j \geq z_i \ \forall i \in I, \tag{1b}$$

$$\sum_{j \in J} x_j \leq P \ , \tag{1c}$$

$$x_j \in \{0, 1\} \ \forall j \in J, \tag{1d}$$

$$z_i \in \{0, 1\} \ \forall i \in I \tag{1e}$$

La ecuación 1a establece la necesidad de maximizar la suma de las demandas atendidas h_i, mas no el número de nodos de demanda atendidos (Cada nodo puede tener demandas distintas). El conjunto de restricciones 1b obliga a los nodos de demanda z_i sean atendidos por nodos de oferta x_j siempre y cuando se encuentren dentro de su cobertura a_{ij}. Finalmente la ecuación 1c establece un máximo para el número de nodos de oferta que se habilitan.

2.2. Representación

Para la construcción del algoritmo genético se contempla que los conjuntos de nodos de oferta y demanda deben ser codificados. Los elementos de esos conjuntos se representan por las variables z_i y x_j respectivamente. Cada individuo de la población consiste en un vector de números binarios que representa a una posible solución. La solución está integrada por dos partes,

la primera sección consiste en la representación binaria de los nodos de demanda que son atendidos ($Z = \{z_1, z_2, z_3, z_4...z_m\}$) y la segunda corresponde a los nodos de oferta que son habilitados ($X = \{x_1, x_2, x_3, x_4...x_n\}$). En la siguiente figura se presenta un ejemplo de esta representación.

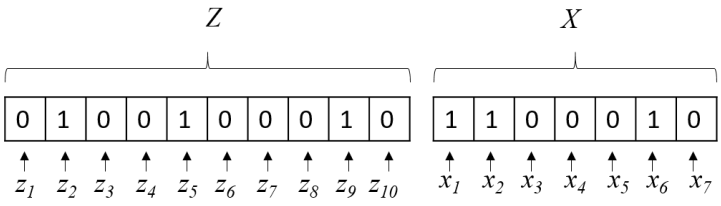


Figura 1: Representación de cromosomas para el problema MCLP

En este ejemplo, el número total de nodos de demanda existentes es de 10, y solamente se puede brindar cobertura a tres de ellos: $z_1, z_5, z_9 = 1$. Existen siete posibles nodos de oferta, pero en esta solución solamente se han habilitado tres: $x_1, x_2, x_6 = 1$. Dependiendo de la cantidad de nodos del problema, este vector aumenta o disminuye su tamaño. Es importante anotar que esta representación no requiere conversión a números reales o viceversa, puesto que se trata de un problema combinatorio.

2.3. Función de utilidad y restricciones

La función de utilidad calcula la suma de las demandas de cada nodo que es atendido. Para implementar las restricciones inicialmente se optó por penalizar los valores de la función de utilidad que no cumplan con ellas, pero los resultados obtenidos no fueron satisfactorios. En consecuencia se adoptó la estrategia de reparar los cromosomas defectuosos en cada iteración del algoritmo.

Para la restricción 1b la reparación implicó que cuando no hubiese cumplimiento, el valor de z_i pasase a ser de cero (0). En el caso de la restricción 1c, los cromosomas fueron reparados seleccionando aleatoriamente cromosomas con x_j igual a uno (1) y asignándoles valor cero (0) hasta que su suma total fuese igual a P.

2.4. Operadores genéticos

Para esta implementación se utilizaron los operadores de selección, cruce y mutación. El operador de selección utilizado consistió en un torneo de tamaño cuatro. La operación consiste en seleccionar aleatoriamente cuatro individuos de la población original y escoger aquel que tenga el mayor valor de función de utilidad. Ese individuo pasa entonces a hacer parte de la población de padres. El operador se repite hasta constituir una nueva población con el mismo tamaño de la original.

Con respecto al operador de cruce, éste se ejecuta seleccionando aleatoriamente dos individuos de la población de padres construida a partir del operador de selección. Se generaron dos posiciones aleatorias de cruce para cada uno de los dos componentes del cromosoma: nodos de demanda (Z) y nodos de oferta (X) (Ver figura 2). La probabilidad de cruce para un par de individuos es definida por el usuario con el parámetro $prob_{cross}$.

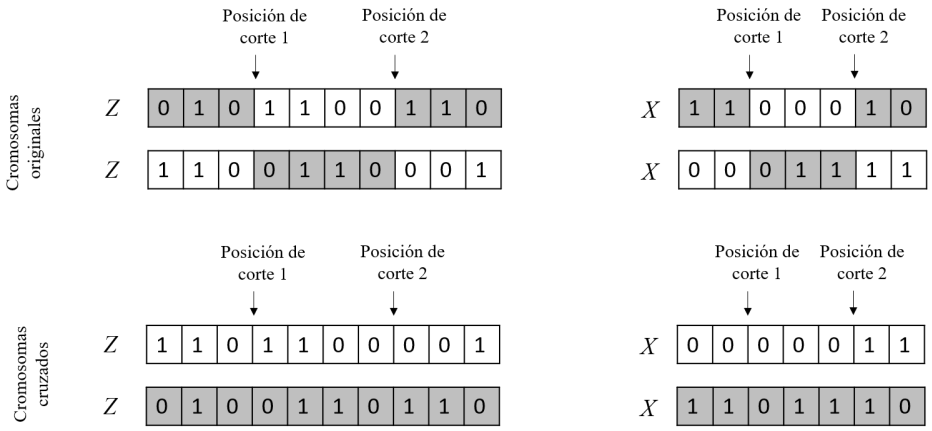


Figura 2: Representación de la operación de cruce de cromosomas para el problema MCLP

Producto del cruce, se genera una nueva población de hijos sobre la cual se ejecuta el operador de mutación. Este operador actúa de manera similar al de cruce, ejecutando mutaciones en los genes de manera aleatoria para cada componente del cromosoma (nodos de demanda y nodos de oferta). La probabilidad de mutación es un parámetro de gran importancia y la literatura sugiere que éste vaya decreciendo en la medida en que avanza el algoritmo genético. Para aplicar esta propuesta se utilizó como porcentaje de mutación

la siguiente función formulada por Bäck and Schütz [12] y explicada por Coello-Coello [13].

$$p_m(t) = \frac{L}{2 + \frac{L-2}{T} * t} \tag{2}$$

Donde, $0 <= t <= T$, L es la longitud de los cromosomas, t es la generación actual y T es el número máximo de generaciones.

La probabilidad de mutación para cada individuo se actualiza constantemente en la medida en que se van calculando nuevas generaciones.

2.5. Flujo del algoritmo

Para ejemplificar la interacción entre el algoritmo genético, la representación cromosómica, la función de utilidad y los operadores genéticos, se construyó un flujograma que representa la forma como se realizó la implementación (Ver figura 3).

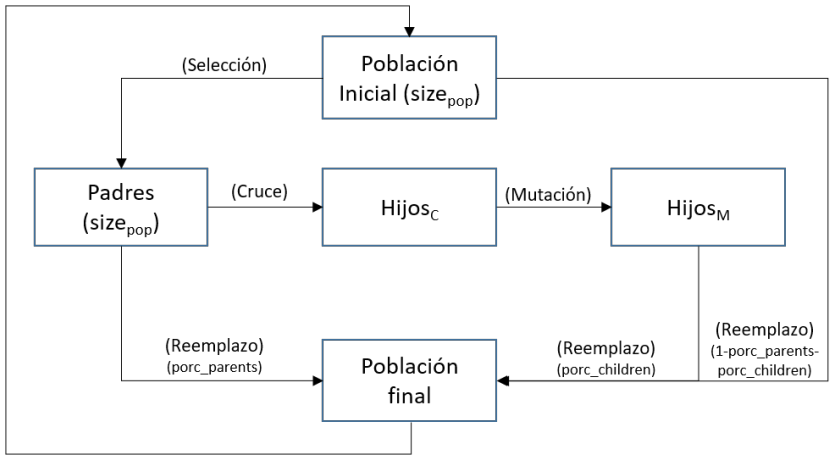


Figura 3: Flujograma del Algoritmo genético para el problema MCLP

El algoritmo parte de una población inicial compuesta por cromosomas cada uno con tamaño igual al número de nodos de demanda y nodos de oferta. El tamaño de la población viene dado por el parámetro $size_{pop}$ seleccionado por el usuario. Sobre esta población inicial actúa el operador de

selección construyendo una población de padres de igual tamaño $size_{pop}$.

Sobre la población de padres se aplican consecutivamente los operadores de cruce y mutación, llegando a constituir la población de hijos. El tamaño de esta población de hijos es igual al tamaño de población de padres $size_{pop}$. Finalmente para constituir una nueva población se seleccionan fracciones de los conjuntos de padres, hijos y población inicial. Los porcentajes de poblaciones seleccionadas son también parámetros definidos por el usuario ($porc_{parents}$ y $porc_{children}$ respectivamente). Sobre esta nueva población se calcula la función de utilidad y se repite el procedimiento nuevamente hasta que se alcance un número de generaciones definido también por el usuario con el parámetro *tmax*. La implementación desarrollada logra registrar los resultados de la función de utilidad de cada individuo de las poblaciones dentro de cada generación.

2.6. Instancia de aplicación

Para probar los resultados de la implementación desarrollada se utilizó la instancia de datos utilizada por Daskin [1] en la solución de problemas de localización. La instancia contiene las coordenadas de 88 ciudades en los Estados Unidos. Previo a la ejecución del algoritmo genético se ajustó la instancia calculando las distancias entre ciudades y los valores de las variables de cobertura a_{ij} para una distancia máxima de 410. Los resultados previamente obtenidos por Jaramillo et al. [10] son presentados a continuación[4].

Es importante reconocer que para este caso, la función de utilidad fue más allá de la minimización del costo, estableciendo como medida de eficiencia del algoritmo el porcentaje de nodos de demanda que son cubiertos por los nodos de oferta.

2.7. Resultados obtenidos

La implementación desarrollada fue puesta a prueba teniendo en cuenta como parámetros el tamaño de la población ($size_{pop}$), el tamaño de los

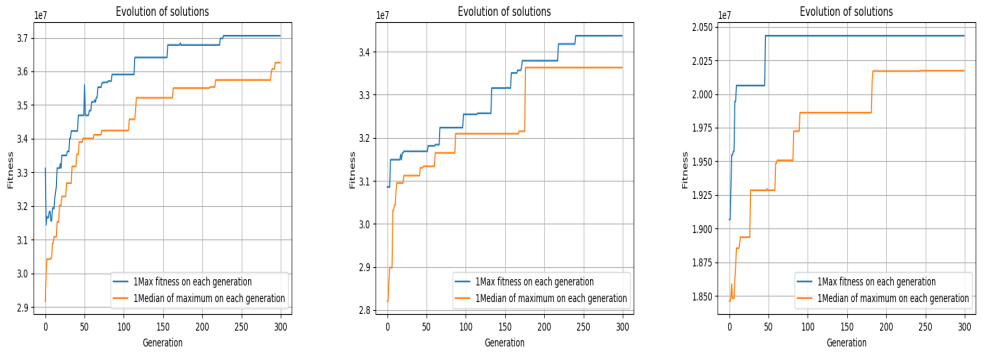
[4]Los datos de la instancia utilizada para validar la aplicación del algoritmo genético pueden ser consultados en el repositorio `https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/DataCities.xlsx`

P x_j	**Mejor solución (%)**
2	61.1
3	78.2
4	87.5
5	92.5
6	96.7
7	99.8
8	100

Cuadro 1: Resultados óptimos para el problema MCLP con la instancia de datos utilizada [10]

individuos(z_i y x_j), la probabilidad de cruce ($prob_{cross}$) y de mutación ($prob_{mut}$), los porcentajes de padres ($porc_{parents}$) e hijos ($porc_{children}$) en la población final, el número de generaciones ($tmax$) y el número de ejecuciones ($runs$).

La aplicación de algoritmos genéticos al problema MCLP permite realizar análisis basados en distintos valores de sus parámetros. Los porcentajes de cobertura para dos, cuatro y seis nodos de oferta fueron similares, aunque éste se combina con el efecto generado por las modificaciones en los porcentajes de padres e hijos. En la figura 4 se presenta la evolución generación por generación de las 30 ejecuciones (runs) desarrolladas en cada experimento.



(a) Experimento 1. P=6 (b) Experimento 2. P=4 (c) Experimento 3. P=2

Figura 4: Evolución del algoritmo genético en los tres escenarios estudiados

Parámetros	**Exp. 3**	**Exp. 2**	**Exp. 1**
size_pop	50	50	50
zi	88	88	88
xj	2	4	6
prob_cross	0.7	0.7	0.7
prob_mut	0.01	0.01	0.01
porc_parents	0.6	0.6	0.6
porc_children	0.3	0.3	0.3
tmax	300	300	300
runs	30	30	30
Percentage (%)	29,54	56.81	67.04
Max. Utility function	**20,432,739**	**34,368,212**	**37,057,637**

Cuadro 2: Resultados de la experimentación realizada con el Algoritmo Genético para el MCLP

3. Aplicación de Estrategias evolutivas para el problema p-centro

En esta sección se presentará la aplicación de estrategias evolutivas a problemas de localización, específicamente al problema p-centro. La organización de la sección es similar a la presentada en la sección . Inicialmente se describe el problema a ser resuelto, posteriormente se define la representación de los individuos, los detalles del cálculo de la función de utilidad y las restricciones del modelo, sus operadores genéticos, el flujo del algoritmo, la instancia sobre la cual se ejecuta la implementación y los resultados obtenidos.

La implementación del algoritmo de estrategias evolutivas se realizó utilizando el lenguaje de programación Python[5].

3.1. Problema p-centro

El problema p-centro busca identificar las instalaciones que pueden brindar servicio a una serie de clientes minimizando la distancia de cobertura

[5]El código de la implementación de estrategias evolutivas en Python puede ser encontrado en el repositorio `https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/ES%20Objetos%20MAXCP.py`

que pueda existir entre los nodos de oferta y de demanda. A cambio de utilizar una distancia de cobertura máxima constante y minimizar el número de instalaciones necesarias, este enfoque pretende minimizar la máxima distancia existente entre cualquiera de los nodos de demanda y su instalación más cercana.

El modelo matemático para la construcción del problema se presenta a continuación:
P: Número de instalaciones a ubicar
I: Conjunto de todos los nodos posibles para ubicar las P instalaciones
J: Conjunto de todos los clientes a abastecer
y_{ij}: Fracción de demanda del nodo $i \in I$ atendida por una instalación en el nodo $j \in J$
d_{ij}: Distancia entre el nodo $i \in I$ y el nodo $j \in J$
W: Máxima distancia existente entre los nodos de demanda y el nodo de oferta más cercano.

$$x_j = \begin{cases} 1 & \text{si una instalación se localiza en el nodo } j \in J \\ 0 & \text{si no.} \end{cases}$$

$$\text{minimizar} \quad W \tag{3a}$$

$$\text{sujeto a} \quad \sum_{j \in J} y_{ij} = 1 \ \forall i \in I, \tag{3b}$$

$$\sum_{j \in J} x_j = P, \tag{3c}$$

$$y_{ij} \leq x_j \ \ \forall i \in I; \forall j \in J, \tag{3d}$$

$$W \geq \sum_{j \in J} d_{ij} y_{ij} \ \ \forall i \in I, \tag{3e}$$

$$x_j \in \{0, 1\} \ \ \forall j \in J, \tag{3f}$$

$$y_{ij} \geq 0 \ \forall i \in I; \forall j \in J. \tag{3g}$$

La función objetivo 3a representa la máxima distancia existente en la red establecida entre cualquiera de los nodos de demanda y el nodo de oferta más cercano. La restricción 3b implica que si se atiende a un nodo de demanda en particular, ésta debe ser cubierta por completo entre uno o más nodos de oferta. La restricción 3d establece la asignación obligatoria de abrir un nodo de oferta j si se ha asignado una demanda con la variable y_{ij}. Finalmente, la restricción 3e es la que involucra la definición de la máxima distancia de cobertura en la red diseñada.

3.2. Representación

La estructura cromosómica de los individuos de este problema implica que se incluyan variables reales y binarias. Las variables reales consisten en los porcentajes y_{ij}, mientras que las binarias corresponden a la decisión de apertura de instalaciones en las ubicaciones j (z_j). De acuerdo con la lógica de las estrategias evolutivas, los individuos también tienen un componente que representa la desviación estándar de cada gen para poder realizar el operador de mutación (σ_{ij}).

Siendo así, la representación genética de los individuos queda como se presenta en la figura 5.

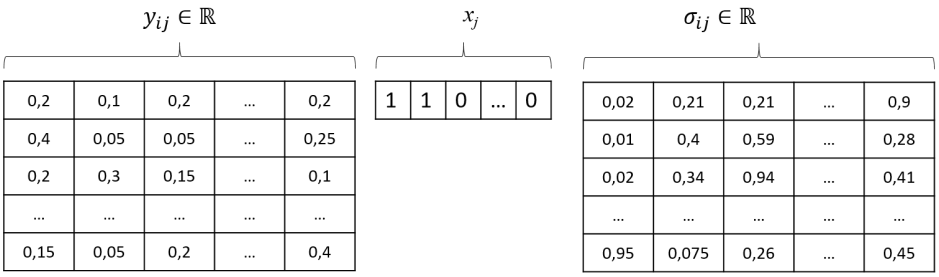


Figura 5: Representación de cromosomas para el problema MCLP

3.3. Función de utilidad y restricciones

Para este problema, la función de utilidad arroja el valor de la variable W, la cual se obtiene a su vez de la restricción 3e. Con el fin de cumplir

con las restricciones, se utilizó la estrategia de penalizar la función objetivo, utilizando para ello las siguientes funciones:

$$w_t = w_{t-1} + (\sum_i y_{ij}) * (1000) \tag{4}$$

$$w_t = w_{t-1} + (\sum_j x_j) * (1000) \tag{5}$$

El valor w_t corresponde a la actualización de la función de utilidad obtenida originalmente w_{t-1} tras aplicar la penalización. La ecuación 4 penaliza las soluciones que no cumplan con la restricción 3b. De otro lado, la ecuación 5 penaliza aquellas soluciones que no cumplan con las restricciones 3c o 3d.

3.4. Operadores genéticos

En Estrategias evolutivas, los operadores más utilizados son los de mutación y cruce. El operador de mutación se aplicó por separado para los componentes reales y binarios de cada individuo. Para el componente real (y_{ij}) de los individuos, se aplicó el siguiente operador de mutación:

$$y_{ij}^t = y_{ij}^{t-1} + N(0, \sigma_{ij}) \tag{6}$$

Los valores de σ_{ij} permanecieron constantes durante todo el proceso evolutivo. En cada mutación se ajustaron los valores resultantes de y_{ij}^t para que siempre estuvieran dentro del intervalo [0,1] dado que corresponden a los porcentajes de demanda de cada nodo i abastecido por el nodo de oferta j.

El operador de mutación para el componente binario x_j actuó de manera semejante al caso de algoritmos genéticos, seleccionando aleatoriamente una posición dentro del cromosoma y modificando su naturaleza.

El operador de selección utilizado para esta implementación es del tipo (μ, λ)-ES, en el cual los individuos μ producen una descendencia λ ($\lambda > \mu$) y el proceso de selección escoge los mejores μ individuos del conjunto λ como nueva generación [5].

3.5. Flujo del algoritmo

Esta implementación de estrategias evolutivas fue desarrollada siguiendo los pasos que se presentan en la figura 6.

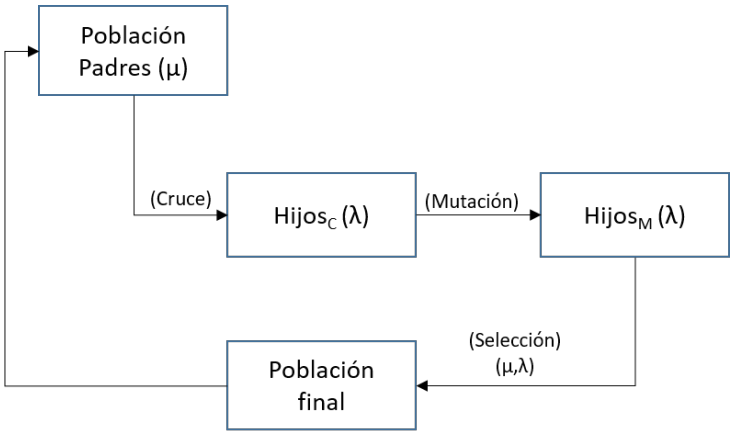


Figura 6: Flujograma de Estrategias evolutivas para el problema MCLP

La población inicial de padres es generada aleatoriamente con base en la estructura cromosómica explicada anteriormente. Posteriormente los operadores de cruce y mutación son aplicados secuencialmente a la población de padres generando una población de hijos. Finalmente, el operador de selección ordena la población de hijos y selecciona los mejores individuos para generar una nueva generación con el tamaño poblacional previamente establecido μ.

3.6. Instancia de aplicación

En la aplicación de estrategias evolutivas se utilizó la misma instancia de trabajo que en el caso de algoritmos genéticos [1]. Se aclara que en este caso se utilizaron las distancias entre pares de nodos (d_{ij}) y no se tuvo en cuenta a las variables de cobertura (a_{ij}), dado que no aplican para el modelo presentado.

Parámetros	**Exp. 1**	**Exp. 2**	**Exp. 3**
size_pop	50	50	50
yij	528	352	176
xj	6	4	2
prob_cross	0.7	0.7	0.7
prob_mut	0.2	0.2	0.2
tmax	100	100	100
runs	30	30	30
Utility function	**268,67**	**215,22**	**23,15**

Cuadro 3: Resultados de la experimentación realizada con Estrategias evolutivas para el problema p-centro

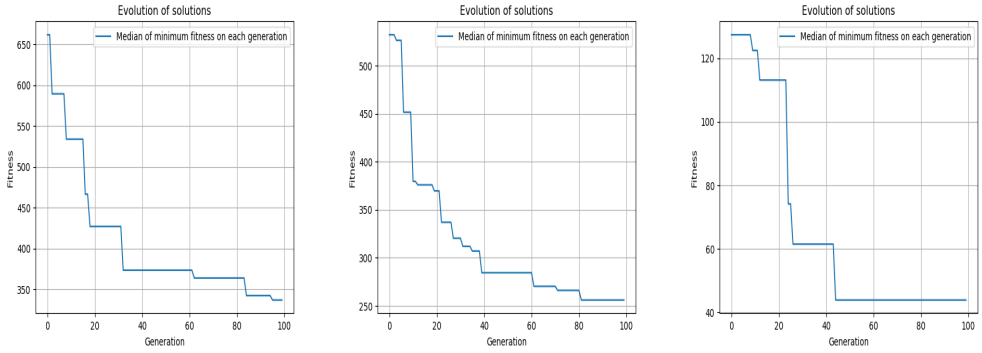
3.7. Resultados

Para comprobar la implementación de las estrategias evolutivas en el problema p-centro, se desarrollaron tres experimentos, cada uno de 20 ejecuciones, en donde se tuvieron en cuenta los siguientes parámetros: tamaño de la población ($size_{pop}$), el tamaño de los individuos(y_{ij} y x_j), la probabilidad de cruce ($prob_{cross}$) y de mutación ($prob_{mut}$), el número de generaciones ($tmax$) y el número de ejecuciones ($runs$). Los resultados para tres experimentaciones distintas pueden ser consultados en el cuadro 3.

Los resultados evidencian una aparente diferencia significativa entre las utilidades obtenidas por los tres experimentos, los cuales dependen a su vez del número de instalaciones a habilitar para cada red.

4. Análisis y conclusiones

En este trabajo se desarrolló la aplicación de dos paradigmas de computación evolutiva para problemas de localización. Inicialmente se abordó el uso de algoritmos genéticos, y para ello se utilizó un problema de localización con cobertura máxima (MCLP). La construcción del algoritmo permitió establecer una representación cromosómica combinatoria binaria para cada individuo. Durante la construcción de la implementación se evidenció que el uso de penalizaciones para el cumplimiento de restricciones no generaba resultados satisfactorios pues no se lograba cumplir con ellas. En consecuencia se adoptó la estrategia de reparación de cromosomas, lo cual introdujo una



(a) Experimento 1 (b) Experimento 2 (c) Experimento 3

Figura 7: Evolución de la estrategia evolutiva en los tres escenarios estudiados

mayor variabilidad en las respuestas, pero al mismo tiempo un mejor ajuste hacia el cumplimiento de restricciones.

En la ejecución del algoritmo genético se aplicaron los tres principales operadores genéticos de la computación evolutiva: cruce, mutación y selección. Se evidenciaron las ventajas de utilizar estrategias de torneo en la selección, así como generación de posiciones aleatorias en el cruce.

La implementación del algoritmo genético permitió un primer acercamiento a la exploración de parámetros de entrada en el desarrollo del algoritmo genético. Inicialmente solamente se exploraron los referentes a los porcentajes de padres e hijos en la nueva generación. Los resultados comparados con la literatura permiten evidenciar un acercamiento a los valores reportados por otros autores, pero todavía se evidencia un mal comportamiento con respecto a otras técnicas como relajación lagrangiana.

De otro lado, se realizó una implementación de estrategias evolutivas en el problema de localización p-centro. Esta implementación permitió evidenciar las posibilidades de codificación de individuos utilizando números reales y binarios en un mismo cromosoma. También fue posible realizar una primera aproximación a la experimentación con estrategias evolutivas utilizando como factor a los tamaños de cromosomas, como representación de modificaciones en el número de instalaciones disponibles. Se evidenció que a un mayor número de instalaciones disponibles, el valor de la distancia de cober-

tura aumentó considerablemente.

Como recomendación para continuar abordando este trabajo se plantean las siguientes posibilidades:

- Explorar un mayor número de escenarios para cada paradigma de manera que se puedan extraer conclusiones basadas en un diseño experimental formal.
- Utilizar otros paradigmas de la computación evolutiva como la programación multimodal o la optimización multicriterio para explorar sus aplicaciones en problemas de localización.
- Explorar la aplicación de técnicas de computación evolutiva a problemas más específicos en el marco de la ciencia de localización, utilizando componentes estocásticos, variables de flujo y otras.

Apéndice A. Ejemplo de resultados del algoritmo genético para el problema MCLP

****
DEFINITION OF THE EXPERIMENT
****

Population size: 50 Number of generations: 300
Parents percentage: 0.6 Children percentage: 0.3
Crossing probability: 0.7 Mutation probability: 0.01
Number of runs: 30
Model description:
$MaxZ = sum_i(hi * zi)$
S.T.: $zi <= sum_j(aij * xj)$ for every i
sum xj¡=P P=Max number of possible locations
xj and zi =[1,0] i=Order of the set of demand nodes
j=Order of the set of possible locations
xj:1 if location at j is opened, 0 elsewhere
zi:1 if demand node at i is covered in the solution, 0 elsewhere
Number of demand nodes(i): 88 Number of possible locations(j): 6
****

Highest fitness and chromosome in experiment:

1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

Apéndice B. Ejercicios de Computación evolutiva desarrollados durante el curso

Apéndice B.1. Max one Problem

Esta es una primera y sencilla implementación del problema de alcanzar el mayor número de números uno dentro de un cromosoma binario. El código respectivo puede ser encontrado en:

`https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/MaxOneProblem.py`

Apéndice B.2. Algoritmo de ascenso a la colina (Parte 1)

Este algoritmo genera un agente que busca el minimizador para la función fx ingresada en la línea 16. Se debe definir la cantidad de iteraciones en la línea 12. Como la cantidad de iteraciones es fija, el algoritmo continúa procesando a pesar de haber obtenido mínimos locales.

`https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/HillClimbingBasico.py`

Apéndice B.3. Algoritmo de ascenso a la colina y recocido simulado (Parte 2)

En esta versión del algoritmo no hay un número fijo de iteraciones. El algoritmo se detiene cuando ya no logra mejoras adicionales. Se declaran dos funciones: ascensocolina y recocido, las cuales operan sobre la función). Para este caso solo se consideran tres dimensiones.

`https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/HillClimbigRecocido.py`

Apéndice B.4. Algoritmo de ascenso a la colina función de Rastrigin (Parte 3)

Este algoritmo aplica el procedimiento de ascenso a la colina (Hill climbing) para la función que se pase como argumento en la función final del código.

`https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/HillClimbingRastrigin.py`

Referencias

[1] M. S. Daskin (Ed.), Network and Discrete Location: Models, Algorithms, and Applications, Second Edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 2013.

[2] H. A. Eiselt, V. Marianov (Eds.), Foundations of location analysis, number 155 in International series in operations research & management science, Springer, New York, NY, 2011. OCLC: 731761987.

[3] G. Laporte, S. Nickel, F. Saldanha da Gama (Eds.), Location Science, Springer International Publishing, Cham, 2015.

[4] R. Church, C. ReVelle, The maximal covering location problem, in: Papers of the Regional Science Association, volume 32, Springer, 1974, pp. 101–118.

[5] Z. Michalewicz, Genetic algorithms + data structures = evolution programs, Springer-Verlag, Berlin ; New York, 3rd rev. and extended ed edition, 1996.

[6] H.-G. Beyer, H.-P. Schwefel, Evolution strategies – A comprehensive introduction, Natural Computing 1 (2002) 3–52.

[7] L. J. Fogel, A. J. Owens, M. J. Walsh, Artificial intelligence through simulated evolution., Artificial intelligence through simulated evolution., John Wiley & Sons, Oxford, England, 1966.

[8] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, Complex adaptive systems, MIT Press, Cambridge, Mass, 1st mit press ed edition, 1992.

[9] I. Rechenberg, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Problemata, 15, Frommann-Holzboog, Stuttgart-Bad Cannstatt, 1973.

[10] J. H. Jaramillo, J. Bhadury, R. Batta, On the use of genetic algorithms to solve location problems, Computers & Operations Research 29 (2002) 761–779.

[11] S. Atta, P. R. Sinha Mahapatra, A. Mukhopadhyay, Solving maximal covering location problem using genetic algorithm with local refinement, Soft Computing 22 (2018) 3891–3906.

[12] T. Bäck, M. Schütz, Intelligent mutation rate control in canonical genetic algorithms, in: Z. W. Raś, M. Michalewicz (Eds.), Foundations of Intelligent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, pp. 158–167.

[13] C. Coello-Coello, Introducción a la Computación Evolutiva, 2004.