

Aplicación de técnicas de computación evolutiva en problemas de localización de instalaciones para prestación de servicios logísticos

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 cuatro estrategias: problema de localización con máxima cobertura, problema de localización p-centro, y aplicaciones multimodales y multicriterio. El propósito del artículo consiste en evaluar la aplicabilidad de estas técnicas con miras a posteriores implementaciones en escenarios más específicos 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. Muchas preguntas son consideradas en este campo de trabajo. En este artículo se hará particular énfasis en instalaciones que prestan servicios de carácter logístico: almacenes, puntos de venta, hospitales, entre otros. Algunas incluyen [1]:

- ¿Cuántas instalaciones deberían ser puestas a funcionar en una red de servicios?
- ¿Dónde deberían ser ubicadas 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]. De hecho, las clasificaciones se realizan de acuerdo con distintos criterios, tales como: espacio de localización (continuo, en red o discreto), campo económico de aplicación (público, privado o mixto), entre otros. 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 muchas áreas: 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 de un radio específico de distancia desde al menos una instalación prestadora de servicios [1]. Por lo tanto, cada instalación brinda una “cobertura” para sus clientes la cual es definida como la posibilidad de brindar servicios a un cliente dependiendo de la distancia desde la cual 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 de servicios, el enfoque del problema p-centro consiste en minimizar la distancia de cobertura tal que cada nodo de demanda sea cubierto dentro de una determinada distancia por alguna de las instalaciones. 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].

De manera más reciente, el enfoque de los Algoritmos evolutivos ha sido propuesto para resolver problemas MCLP dado que brindan soluciones adecuadas cerca al óptimo. Los algoritmos evolutivos están inspirados por mecanismos de evolución natural para obtener mejores características en una población a través de la simulación de su evolución. En cada generación, algunos operadores son aplicados a la población de individuos (soluciones), modificando así su estructura genético 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 quince años: Programación evolutiva [7], Algoritmos genéticos (GA)[8] y Estrategias evolutivas (ES)[9].

El uso de Algoritmos evolutivos para resolver problemas MCLP ha sido reportado por Jaramillo et.al. [10]. Su trabajo evaluó el desempeño de Algoritmos Genéticos como un procedimiento alternativo para resolver cuatro 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 de la actualización de la localización de cada instalación por el 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. El resto del artículo está organizado como sigue: 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. Una aproximación a los problemas multimodales y a los problemas multicriterio es presentada en las secciones 4 y 5 respectivamente¹. Finalmente, la sección 6 concluye el artículo y presenta recomendaciones para futuros trabajos².

2. Aplicación de Algoritmos Genéticos para el problema MCLP

Para presentar la aplicación de este paradigma de la computación evolutiva al problema de localización con cobertura máxima, se utilizar la siguiente lógica. En primer lugar se hará una descripción matemática del problema, posteriormente se abordará la forma como se resolvió utilizando Algoritmos Genéticos. Para ello se detallará 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 obtenidos³.

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:

¹Estas dos secciones todavía están pendientes de ser documentadas

²Este 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 envidenciados en los anexos Apéndice~B.1, Apéndice~B.2, Apéndice~B.3 y , Apéndice~B.4

³El 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>

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

$$z_i = \begin{cases} 1 & \text{si el nodo } i \in I_{\text{escubierto}} \\ 0 & \text{si no.} \end{cases}$$

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

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

El modelo matemático para asegurar la máxima cobertura posible queda planteado a continuación:

$$\text{maximize} \quad \sum_{i \in I} h_i Z_i \quad (1a)$$

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

$$\sum_{j \in J} x_j \leq P, \quad (1c)$$

$$x_j \in \{0, 1\} \quad \forall j \in J, \quad (1d)$$

$$z_i \in \{0, 1\} \quad \forall i \in I \quad (1e)$$

La ecuación 1a establece la necesidad de maximizar el producto de las demandas atendidas h_i . La restricción 1b obliga a que solamente 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 las variables z_i y x_j son las que requieren ser codificadas. Así, cada individuo de la

población consiste en un vector de números binarios que representa a una posible solución. La solución consiste en un número de nodos de demanda que son atendidos y otro número de nodos de oferta que son habilitados. En la siguiente figura se presenta un ejemplo de esta representación.

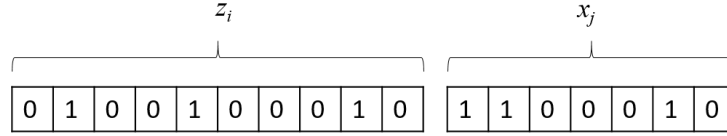


Figura 1: Representación De cromosomas para el problema MCLP

En este ejemplo, el número de nodos de demanda es de 10, mientras que el número de nodos de oferta es de siete. 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, en tanto que se trata de un problema combinatorio.

2.3. Función de utilidad y restricciones

La función de utilidad simplemente calcula la sumatoria del producto de las demandas de cada nodo de demanda que es atendido. Para implementar las restricciones se optó por penalizar los valores de la función de utilidad que no cumplan con ellas. Básicamente se adoptaron las siguientes expresiones:

$$f_i = f_{i-1} - (z_i - a_{ij} * x_j) * (h_i * 0,1) \quad (2)$$

$$f_i = f_{i-1} - \left(\sum_j x_j \right) * (h_i * 0,1) \quad (3)$$

Las dos ecuaciones anteriores denotan cómo se penalizan las funciones de utilidad obtenidas, reduciendo su valor de manera proporcional a su propio valor.

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 genera una posición aleatoria de cruce para cada uno de los dos componentes del cromosoma: nodos de demanda (z_i) y nodos de oferta (x_j). La probabilidad de cruce para un par de individuos es definida por el usuario con el parámetro $prob_{cross}$.

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 para cada individuo es definida por el usuario con el parámetro $prob_{mut}$.

2.5. Flujo del algoritmo

Para ejemplificar la interacción del 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 2).

El algoritmo parte de una población inicial compuesta por cromosomas cada uno con tamaño de acuerdo con el 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

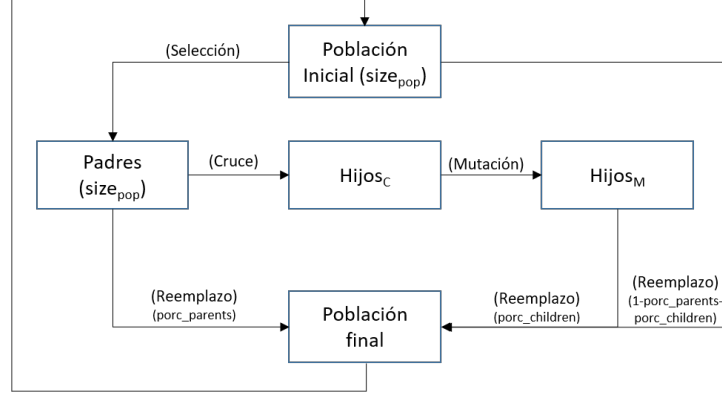


Figura 2: Flujograma del Algoritmo genético para el problema MCLP

de esta población de hijos es igual al tamaño de población de padres $size_{pop}$. Finalmente para constituir una nueva población se seleccionan porcentajes de las poblaciones de padres, hijos y de la 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 en Python 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. 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 continuación⁴.

⁴Los datos de la instancia utilizada para validar la aplicación del algo-

Tamaño 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]

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 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 3 se presenta la evolución generación por generación de las 20 ejecuciones ($runs$) desarrolladas en cada experimento.

Parámetros	Exp. 1	Exp. 2	Exp. 3
size_pop	20	20	20
zi	88	88	88
xj	2	4	6
prob_cross	0.7	0.7	0.7
prob_mut	0.2	0.2	0.2
porc_parents	0.1	0.3	0.5
porc_children	0.7	0.5	0.3
tmax	30	30	30
runs	20	20	20
Percentage (%)	77.27	76.13	74.99
Max. Utility function	39,271,360.8	38,807,852.5	38,905,354.7

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 del paradigma 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 Python⁵.

ritmo genético pueden ser consultados en el repositorio <https://github.com/Evolutionary-Computing-2019/Edgar-Duarte/blob/master/DataCities.xlsx>

⁵El 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>

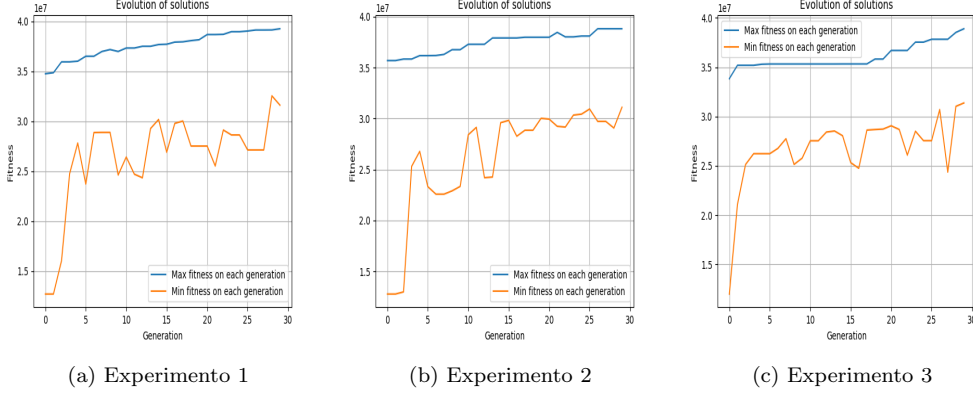


Figura 3: Evolución del algoritmo genético en los tres escenarios estudiados

3.1. Problema p -centro

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

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

$$\text{minimizar } W \quad (4a)$$

$$\text{sujeto a } \sum_{j \in J} y_{ij} = 1 \quad \forall i \in I, \quad (4b)$$

$$\sum_{j \in J} x_j = P, \quad (4c)$$

$$y_{ij} \leq x_j \quad \forall i \in I; \forall j \in J, \quad (4d)$$

$$W \geq \sum_{j \in J} d_{ij} y_{ij} \quad \forall i \in I, \quad (4e)$$

$$x_j \in \{0, 1\} \quad \forall j \in J, \quad (4f)$$

$$y_{ij} \geq 0 \quad \forall i \in I; \forall j \in J. \quad (4g)$$

La función objetivo 4a 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 4b 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 4d 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 4e 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} , las binarias 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.

Siendo así, la representación genética de los individuos queda como se presenta en la figura 4.

$y_{ij} \in \mathbb{R}$					x_j					$\sigma_{ij} \in \mathbb{R}$				
0,2	0,1	0,2	...	0,2	1	1	0	...	0	0,02	0,21	0,21	...	0,9
0,4	0,05	0,05	...	0,25						0,01	0,4	0,59	...	0,28
0,2	0,3	0,15	...	0,1						0,02	0,34	0,94	...	0,41
...
0,15	0,05	0,2	...	0,4						0,95	0,075	0,26	...	0,45

Figura 4: Representación de cromosomas para el problema MCLP

3.3. Función de utilidad y restricciones

Para este problema, la función de utilidad consiste en el valor mismo de la variable W , la cual se obtiene a su vez de la restricción 4e. Las penalizaciones para que se cumpliera con las restricciones del modelo se ejercieron de acuerdo con las siguientes funciones:

$$w_t = w_{t-1} + \left(\sum_i y_{ij} \right) * (1000) \quad (5)$$

$$w_t = w_{t-1} + \left(\sum_j x_j \right) * (1000) \quad (6)$$

La ecuación 5 penaliza las soluciones que no cumplan con la restricción 4b. De otro lado, la ecuación 6 penaliza aquellas soluciones que no cumplan con las restricciones 4c o 4d.

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 cada individuo, se aplicó el siguiente operador de mutación:

$$y_{ij}^t = y_{ij}^{t-1} + N(0, \sigma_{ij}) \quad (7)$$

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 $[0,1]$ en tanto que la variable se refiere a un porcentaje.

El operador de mutación para el componente binario x_j actuó de manera semejante al operador explicado para el caso de algoritmos genéticos.

El operador de selección utilizado para esta implementación es del tipo (μ, λ) -ES, en la cual los mejores individuos de la mezcla de hijos (λ) y padres (μ) son seleccionados para hacer parte de la próxima generación.

3.5. Flujo del algoritmo

Esta implementación de estrategias evolutivas fue llevada siguiendo los pasos que se presentan en la figura 5.

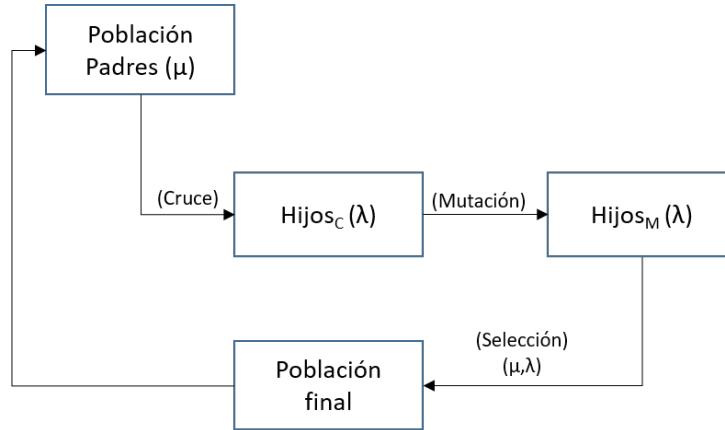


Figura 5: Flujograma de Estrategias evolutivas para el problema MCLP

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

Parámetros	Exp. 1	Exp. 2	Exp. 3
size_pop	20	20	20
y _{ij}	176	352	528
x _j	2	4	6
prob_cross	0.7	0.7	0.7
prob_mut	0.2	0.2	0.2
tmax	30	30	30
runs	20	20	20
Max. Utility function	59.22	193.12	319.16

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

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.

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 de 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 valores de y_{ij} y de z_j están relacionados dado que tienen el mismo número de columnas. Los resultados evidencian una aparente diferencia significativa entre las utilidades obtenidas por los tres experimentos, los cuales

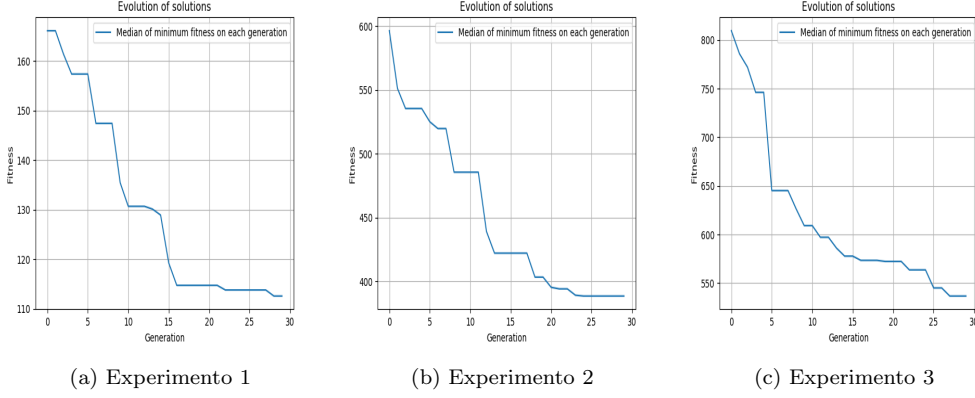


Figura 6: Evolución de la estrategia evolutiva en los tres escenarios estudiados

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. También se planteó el uso de penalizaciones para representar a las restricciones del problema. 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 en Python 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 cobertura 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: 20 Number of generations: 30

Parents percentage: 0.5 Children percentage: 0.3

Crossing probability: 0.7 Mutation probability: 0.2

Number of runs: 20

Model description:

$$MaxZ = \sum_i (h_i * z_i)$$

S.T.: $z_i \leq \sum_j (a_{ij} * x_j)$ for every i

$\sum_j x_j = P$ $P = \text{Max number of possible locations}$

x_j and $z_i = [1, 0]$ i =Order of the set of demand nodes
 j =Order of the set of possible locations
 x_j :1 if location at j is opened, 0 elsewhere
 z_i :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. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0.
 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1.
 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0.
 1. 0. 1. 1. 1.]
 Percentage of covered nodes:
 0.7045454545454541
 Time Consumed
 1112.3115017414093 seconds

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 f_x 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/HillClimbingRecocido.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 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.