Optimización Multiobjetivo de Posicionamiento de Antenas en Zonas Pobladas

Franco Olivera
Instituto de Computación
Factultad de Ingeniería, UdelaR
Montevideo, Uruguay
franco.olivera@fing.edu.uy

Nicolás Mayora
Instituto de Computación
Factultad de Ingeniería, UdelaR
Montevideo, Uruguay
nicolas.mayora@fing.edu.uy

***Abstract*—Se plantea un algoritmo evolutivo para el problema de posicionamiento de antenas, explora conceptos teóricos referentes a su naturaleza multiobjetivo, e implementa en Python utilizando algoritmos conocidos en la industria. Posteriormente se compara con un algoritmo alternativo greedy y extraen resultados y conclusiones a partir del análisis.**

***Index Terms*—telecomunicaciones, algoritmos genéticos, optimización, nsga-ii**

I. Introducción

En la industria de telecomuniaciones inalámbricas es de interés minimizar costos de despliegue al planificar una nueva red móvil. Un problema central en esta área consiste en proveer servicio a la mayor cantidad de gente mientra se tiene un costo aceptable. Obtener óptimos a este problema no es una tarea trival dado que algoritmos deterministas convencionales no suelen ser capaces de obtener buenas soluciones multiobjetivo en un tiempo razonable. Por esto se propone una solución evolutiva al problema.

II. Descripción del Problema

El problema a resolver consta en decidir posiciones óptimas para antenas en zonas pobladas teniendo un enfoque multiobjetivo; dado a que el costo de las mismas no es nulo se desea obtener una diversa gama de soluciones pertenecientes al frente de Pareto, de forma que si es de interés se consideran aquellas soluciones con mayor cubrimiento—con la desventaja de tener mayor costo, o aquellas con un costo menor—al igual que cubrimiento. Como se puede observar, el problema consta en minimizar el costo y maximizar el cubrimiento de la instancia proveída.

A. Parámetros de entrada

Los parámetros de entrada para el problema son:

- La instancia, en la representación detallada mas adelante, junto a sus dimensiones.
- El costo de cada antena.
- El radio de cubrimiento de cada antena.
- La población en cada generación del algoritmo
- La cantidad de generaciones.

III. Representación abstracta

Para la representación del problema se particiona la región en cuestión en una grilla de $M \times N$ celdas, y en base a si se está considerendo una solución o una instancia de entrada, se tienen distintos dominios y semántica para los valores encontrados.

A. Representación de Instancias

Para representar las instancias, las celdas de la matriz toma valores enteros no negativos representantes de la población presente en esa zona.

B. Representación de Soluciones

Para representar las soluciones, las mismas toman valores binarios representantes de la presencia de una antena en determinada zona, la cual cubre un area circular a su alrededor. Éste es un parámetro de entrada pues consideramos de interés ver distintas soluciones variando este valor.

IV. Estrategia de Resolución

En esta sección se detalla el trasfondo teórico el cual con el fin de fundamantar conceptualmente la solución implementada.

A. Algoritmo utilizado

Para solucionar el problema presentado se utiliza el algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm II*). Las principales características del mismo es el hecho que es elitista, que utiliza un algoritmo de ordenamiento rápido, y los operadores en juego—los cuales serán expuestos en breve. Más información disponible en [1] y [2]

B. Funciones de Fitness

Al ser un problema de optimización multiobjetivo, se tienen dos métricas a optimizar; se desea (i) minimizar el costo brindado al tener una mayor cantidad de antenas y (ii) maximizar el cubrimiento de población. Las funciones de fitness para cada una de las métricas entonces son: i

1) Para minimizar el costo simplemente multiplica la cantidad de antenas por el costo de cada una, es decir se tiene:

$$F_{costo}(S) = k \sum_{x:=s_{ij}} x,$$

para alguna solución $S \in \mathcal{M}_{M\times N}[0,1]$, y donde k es el costo de cada antena.

2) Por otro lado, para el cubrimiento de la población se toma como fitness la población total cubierta por todas las antenas. Para ello se itera sobre cada antena creando una matriz de cubrimiento, la cual se inizializa con ceros y se le asigna un uno a cada celda cubierta por almenos una antena. Luego de finalizar la matriz de cubrimiento se multiplica cada coordenada con el respectivo cuadrante en la matriz de poblacion, para ser sumada a un total de cubrimiento.

C. Operadores evolutivos

A continuación se detallan los operadores evolutivos. Notar que para poder ser aplicados se interpreta la solución como un vector de bits; no una matriz de los mismos.

1) Selección: Como especificado por NSGA-II, se utiliza binary tournament selection, el cual selecciona pares de elementos aleatorios, y sobrevive aquel más fit. Sin embargo, el concepto de "más fit" presenta ambigüedad en el contexto multiobjetivo, por lo que [3] presenta una forma de extender el operador de selección a problemas de esta naturaleza:

Primero, si un individuo domina a otro, sobrevive aquel dominante. Luego, si no hay dominancia entre los individuos, para cada uno se calcula la diferencia entre la cantidad de individuos que domina y que es dominado por, sea $D_i, i \in 1, 2$. El individuo con el mayor D_i sobrevive.

2) Cruzamiento: Para cruzamiento entre soluciones se utiliza cruzamiento binario uniforme con probabilidad 0.5, lo cual significa que para cada alelo en un par de individuaos hay una probabilidad del 50% que estos sean intercambiados.

3) Mutación: Para la mutación de las soluciones se utiliza bitflip mutation, la cual con una problabilidad $P(mutación) = \frac{1}{M\times N}$ para cada alelo cambia el valor del mismo, donde $M \times N$ es la cantidad de celdas en la matriz de entrada.

V. Implementación del Algoritmo

Para implementar en código el algoritmo se utiliza el lenguaje Python 3.9.9 y la librería Pymoo (Python multi-objective optimization) [4]. Esta herramienta cuenta fue seleccionada dado a su amplia documentación y cuenta con implementaciones de los previamente mencionados operadores al igual que NSGA-II, además de tener utilidades convenientes a la hora de generar plots de los resultados obtenidos.

En el presente zip se puede encontrar el código fuente de nuestra implementación. Para correrlo es necesario tener Python y pip instalados. Para instalar Pymoo basta con correr `$ pip install pymoo` en la consola de comandos. Luego, para correr el algoritmo en si, descomprimir el archivo y navegar hasta el directorio conteniendo `/src/` y correr `$ python src/main.py`. Finalmente seguir las indicaciones en pantalla. Tener en cuenta que al final de la ejecución se muestra un scatter plot con las soluciones no dominadas; donde el eje x contiene el costo total de antenas y el eje y el opuesto de la población cubierta.

A. Generación de instancias

Entre el código fuente se puede encontrar un generador de instancias, `generador.py` y una variedad de instancias pregeneradas con el script. El mismo tiene diversos parámetros que modifican la naturaleza de la solución en el cabezal del script. Las instancias son generadas en el formato esperador por `main.py`: Una línea conteniendo la altura de la matriz, seguida por otra conteniendo el ancho y finalmente la matriz en si, con saltos de línea luego de cada fila.

VI. Análisis de la solución propuesta

Se realizaron 100 corridas de la instancia de 20x20, con el radio de las antenas siendo 3, la poblacíon por generación 50 y 500 generaciones en total. A continuación se exponen histogramas para La cantidad de antenas y el cubrimiento. Cabe mencionar también que realizar estas corridas tomó 40 minutos en total.

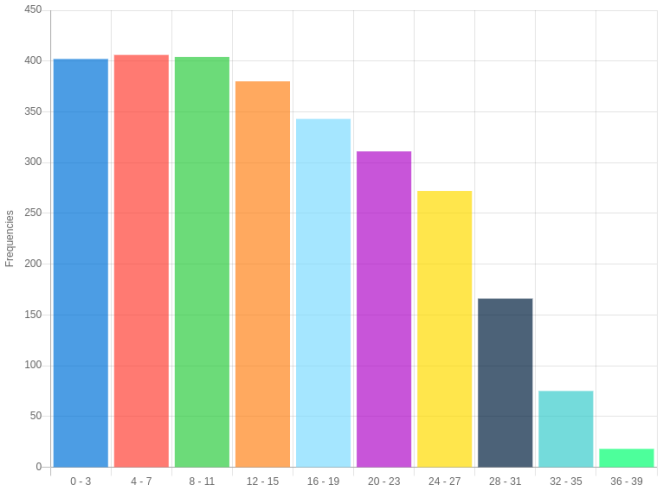

Fig. 1. Histograma del costo de las soluciones.

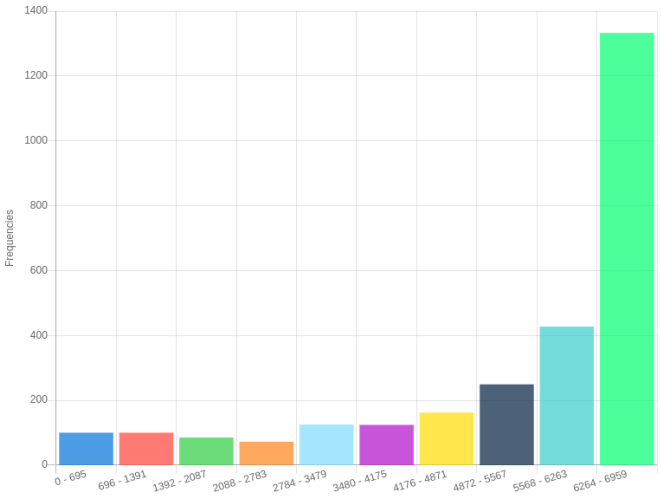

Fig. 2. Histograma del cubrimiento de las soluciones.

Como se observa en 1 se tiene una gran cantidad de soluciones con un bajo número de antenas, como es de esperar ya que es de interés minimizar la cantidad de éstas. Se nota tambien que a partir de 12 de ellas disminuye rápidamente la cantidad de soluciones con determinado número de ellas.

Por otro lado, se puede observar en 2 que se tiene un muy bajo número de individuos con valores de cubrimiento menores a 4900, y que hay una gran concentración de valores en el mayor rango.

Se concluye en base a los últimos dos párrafos, que nuestro algoritmo evolutivo efectivamente converge a el frente de Pareto de la instancia.

A. *Variedad de Soluciones*

Al graficar todas las soluciones no dominadas en una misma figura se obtiene una figura que trasmite que tan esparcidos estan los valores. Se computó analíticamente y la longitud promedio de las "barras" es 322, y el el eje x están equiespaciadas a un intervalo de 1. El cubrimiento promedio siendo 5378, se tiene que la desviación mencionada cuenta para 5.99% del mismo.

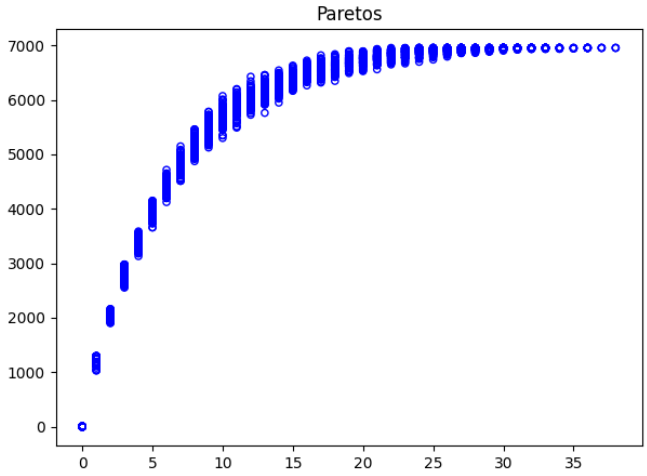

Fig. 3. Soluciones no dominadas luego de 100 corridas

B. *Comparación con algoritmo greedy*

En los archivos fuente proveídos se encuentra la implementación de un algoritmo greedy. Para él, en cada iteración se computa el cubrimiento aportado por cada celda en caso que contenga una antena, se ordenan todos los valores, y selecciona el mayor de todos. Una antena es colocada en ese lugar y se asignan todas las celdas en su rango a cero, antes de repetir el proceso y finalizando una vez el cubrimiento máximo es cero, es decir, una antena mas no aportaría nada.

El algoritmo *no* es óptimo, ya que hemos observado una oportunidad en la cual ha alcanzado el fitness máximo con una antena más que el algoritmo evolutivo, y removiendo cualquiera de ellas se obtiene uno no óptimo. Esto se debe a que el algoritmo greedy carece la habilidad de remover antenas una vez puestas para explorar mejores posiciones. Sin embargo, en la mayoría de los casos obtiene muy buenas soluciones que compiten con o sobrepasan el algoritmo evolutivo.

A continuación se tiene un scatter plot con el algoritmo greedy en rojo y el evolutivo en azul.

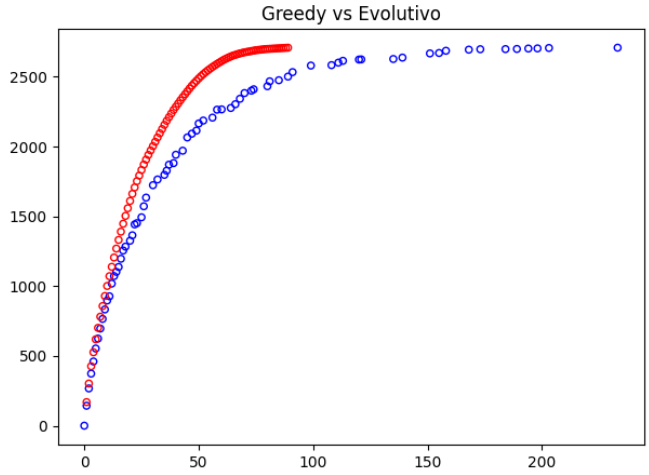

Fig. 4. Greedy vs Evolutivo. dim=50x50, r=3, gen=1500, pop=75

Se puede ver en la figura 4 que las soluciones producidas por el algoritmo greedy efectivamente domina a las soluciones obtenidas mediante una estrategia evolutiva. Sin embargo, creemos que para instancias del problema mas grandes no es viable utilizar el algoritmo greedy pues recomputar los costos para cada celda y obtener el mayor no es viable.

VII. Conclusión

Se pudo implementar un algoritmo evolutivo que resuelve el problema en cuestión con velocidades y niveles de convergencia aceptables. Por más que se pudo a su vez encontrar un algoritmo greedy que produce mejores soluciones, el mismo no es viable para producción dado a su complejidad temporal.

References

[1] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002, doi: 10.1109/4235.996017.

[2] Pymoo, "NSGA-II: Non-dominated Sorting Genetic Algorithm", https://pymoo.org/algorithms/moo/nsga2.html, retr. 4/12/2021.

[3] University of Kent, Department of Computing, "Multi-objective Algorithms for Attribute Selection in Data Mining" Pappa, Freitas, Kaestner, https://www.cs.kent.ac.uk/people/staff/aaf/pub_papers.dir/MOEA-bk-chapter-Gisele.pdf, retr. 4/12/2021.

[4] Pymoo multiobjective optimization library, https://pymoo.org/, retr. 6/12/2021.