Optimización Industrial con Computación Evolutiva

Proyecto Final: Usando computación evolutiva para encontrar localizaciones óptimas de una cadena de supermercados

Repositorio: https://github.com/jpereiran/OICE-GeneticAlgorithms

Alumnos: José Pereira - Alex Villanueva

PROBLEMÁTICA

Una cadena de supermercados desea ingresar al mercado limeño. Para esto, la cadena debe localizar sus supermercados a partir de una lista de posibles locaciones. Sin embargo, la cadena tiene presupuesto para instalar solo 10.

Debido a esta limitante, el objetivo de la empresa es escoger los supermercados de manera que se maximice la suma de la población que vive a 500 m alrededor de ellos y la suma de las distancias entre los supermercados escogidos para lograr una mayor captación de público. Para esto, se solicita proponer una solución a partir del uso de algoritmos genéticos que permitan encontrar una solución que maximice los objetivos de la empresa sin excederse en los recursos.

ALGORITMO MONO-OBJETIVO

También conocido algoritmo genético simple, se utiliza para resolver problemas de optimización y búsqueda basado en la valoración del fitness o función objetivo única para una población. Asimismo, para variar los resultados en cada generación y simular la evolución, se aplica un proceso de cruzamiento y mutación que dará paso a una nueva generación, la cual se espera que posea un mejor fitness.

- **Representación de individuos:** Para este caso se optó representar los individuos o cromosomas mediante una clase que agrupa sus características, la cual tiene un conjunto de genes representado por un lista de valores de los id's de distintas localizaciones, con un tamaño total de 10 genes por cromosoma y otras características como su valor de fitness agregado y disgregado.
- **Operadores de cruzamiento:** Este operador que genera una nueva generación, basado en el intercambio de cromosomas entre dos padres para generar nuevos hijos, en otras palabras utiliza la combinación de soluciones existentes, se basa en el proceso de

búsqueda guiada, donde se establece un número de n puntos o genes a utilizar para crear una nueva generación.

- **Operadores de mutación:** Es operador mediante el valor de la probabilidad modifica los genes de los individuos, de forma aleatoria activa o desactiva un bit del gen.
- **Función de Fitness:** Para este caso, se tomó como función de fitness a la suma de las distancias entre todos los posibles pares de localizaciones escogidas en el individuo más la suma de las poblaciones alrededor de cada uno de los individuos.

ALGORITMO MULTI-OBJETIVO

A diferencia del algoritmo mono-objetivo donde se establece una función para cubrir una necesidad, el multi-objetivo debe de cubrir varias funciones objetivo diferentes, el cual ya no solo busca una solución óptima, sino un conjunto de posibles soluciones equivalentes, las cuales busquen optimizar al menos una de las variables y ser la mejor en conjunto frente al resto de posibles soluciones.

Para este caso, se utilizó el algoritmo **NSGA-II** (Kalyanmoy Deb) y se utilizó una función de distancia de Krowding para obtener los resultados de la frontera de Pareto para el caso analizado.

- **Representación de individuos:** Para este caso los individuos o cromosomas, tienen un conjunto de genes representado por el valor del id de distintas localizaciones, con un tamaño total de 10 genes por cromosoma. Con el fin de simplificar la ejecución del algoritmo, este caso fue manejado con el uso de listas y ya no de una clase para los individuos, siendo representados de la siguiente forma:

$$Ind[n] = [0, 15, 2, 36, 4, 7, 59, 44, 31, 27]$$

- **Operadores de cruzamiento:** Este operador que genera una nueva generación, basado en el intercambio de cromosomas entre dos padres para generar nuevos hijos, en otras palabras utiliza la combinación de soluciones existentes, se basa en el proceso de búsqueda guiada, donde se establece un número de n puntos o genes a utilizar para crear una nueva generación.
- **Operadores de mutación:** Es operador mediante el valor de la probabilidad modifica los genes de los individuos, de forma aleatoria activa o desactiva un bit del gen y lo intercambia por otro disponible.
- **Funciones de Fitness:** Para este caso, se tomó como primera función de fitness a la suma de las distancias entre todos los posibles pares de localizaciones escogidas y como segunda función de fitness a la suma de las poblaciones alrededor de cada uno de los individuos.

RESULTADOS

- **ALGORITMO MONO-OBJETIVO**

Caso Torneo

Ejecución	Individuo	Fitness	Dist X	Dist Y	Población
1	[47, 7, 22, 42, 0, 51, 23, 58, 52, 4]	159703.793	1.87564	1.91767	159700
2	[58, 59, 48, 25, 1, 22, 46, 10, 34, 58]	163303.89	1.82614	2.06423	163300
3	[24, 53, 53, 50, 1, 41, 4, 2, 50, 22]	163503.674	1.74855	1.92574	163500
4	[28, 7, 22, 42, 0, 51, 23, 58, 52, 4]	146604.789	1.87278	2.91623	146600
5	[58, 37, 20, 32, 54, 45, 42, 15, 47, 5]	141255.169	2.10011	3.06884	141250
6	[21, 8, 13, 48, 16, 19, 2, 10, 3, 58]	170402.296	1.39622	0.90011	170400
7	[1, 48, 51, 59, 37, 53, 3, 18, 20, 24]	158253.683	1.76545	1.91784	158250
8	[19, 29, 53, 1, 22, 57, 43, 47, 3, 59]	143804.654	1.91794	2.73639	143800
9	[12, 59, 34, 51, 22, 9, 47, 56, 50, 10]	161904.068	1.97122	2.09682	161900
10	[20, 36, 15, 10, 29, 25, 18, 18, 51, 51]	135504.517	1.74516	2.772	135500

Caso Óptimo

Ejecución	Individuo	Iteración	Fitness	Dist X	Dist Y	Población
1	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	315	176502.8571	1.74214	1.114950	176500
2	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	416	176502.8571	1.74214	1.114950	176500
3	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	80	176502.8571	1.74214	1.114950	176500
4	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	151	176502.8571	1.74214	1.114950	176500
5	[0, 1, 2, 2, 7, 14, 46, 54, 57, 59]	188	175803.13105	1.8808	1.31206	175800
6	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	194	176502.8571	1.6481	1.19898	176100
7	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	308	176502.8571	1.74214	1.11495	176500
8	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	222	176502.8571	1.74214	1.11495	176500
9	[1, 6, 9, 13, 19, 50, 53, 54, 57, 58]	142	176502.8571	1.75831	1.10940	175850
10	[0, 1, 2, 2, 7, 14, 46, 54, 57, 59]	103	175803.13105	1.8808	1.3299	176400

Como se observa en la tabla de resultados, se muestra que se obtienen diferentes valores para la velocidad o generación en que se obtiene el resultado óptimo y el valor de este óptimo en sí, ya que en este caso, no en todas las ejecuciones se logra conseguir ese valor.

Asimismo, se puede apreciar que este es un problema sencillo, el cual no toma muchas generaciones para converger en el resultado óptimo, sin importar la forma en que se ha inicializado los valores, y que el uso de la mutación y el cruzamiento representa una estrategia eficaz para encontrar la solución.

- **ALGORITMO MULTI-OBJETIVO**

Ejecución	Individuo	Distancia	Población
1	[51, 37, 26, 48, 36, 10, 47, 41, 45, 30]	5.8445699	136600
2	[1, 19, 53, 51, 53, 17, 47, 1, 17, 20]	6.244560	96850
3	[56, 19, 34, 12, 34, 49, 44, 36, 37, 37]	6.018599	143950
4	[21, 18, 41, 2, 47, 38, 17, 7, 50, 47]	6.19338000	135000
5	[50, 31, 45, 10, 4, 17, 55, 16, 29, 26]	6.158690	121600
6	[59, 55, 2, 3, 41, 44, 35, 1, 25, 22]	6.036499	137600
7	[11, 54, 14, 17, 47, 58, 15, 47, 32, 5]	5.95919	120850
8	[10, 53, 13, 50, 54, 9, 10, 19, 27, 22]	6.34548	143100
9	[16, 5, 40, 14, 44, 34, 28, 36, 19, 18]	6.07271	173550
10	[17, 51, 44, 28, 27, 38, 29, 21, 21, 34]	5.8675	153550

Como se observa en la tabla de resultados, se muestra que la ahora todos los valores resultantes en las ejecuciones son distintos entre sí. Esto se debe a que debido a la presencia de más de un objetivo, existen muchos resultados óptimos para este problema (frontera de Pareto).

Asimismo, dado que este es un problema sencillo, utilizar este método genera que se necesiten realizar un mayor número de iteraciones (y más costosas a nivel computacional) para poder encontrar el grupo de resultados óptimos.

Gráfico 1 - Resultado de las 10 iteraciones:

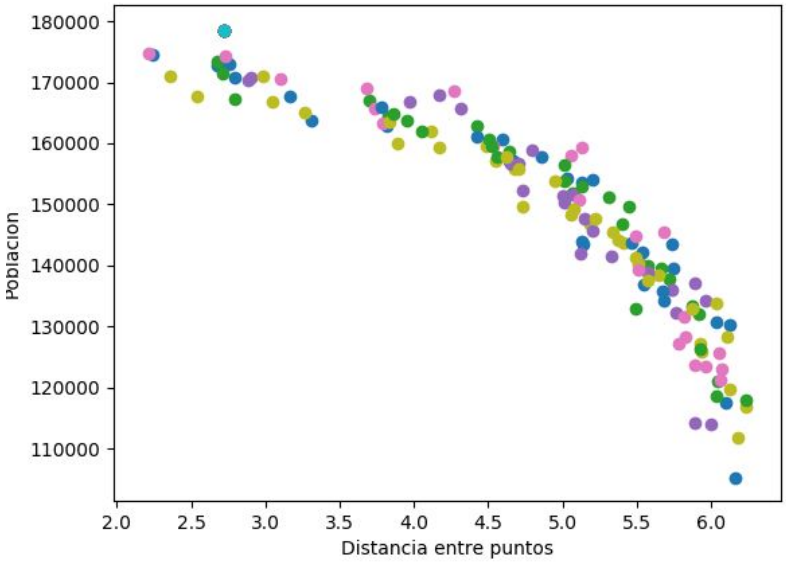
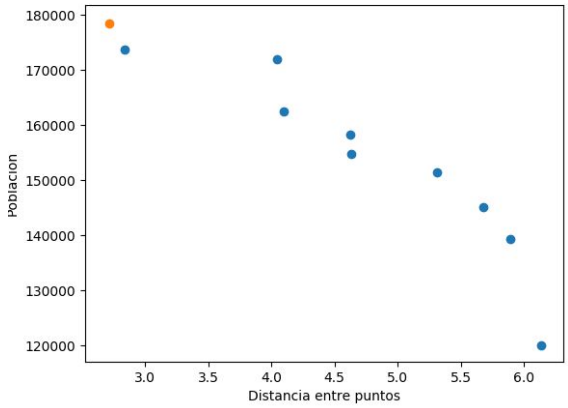
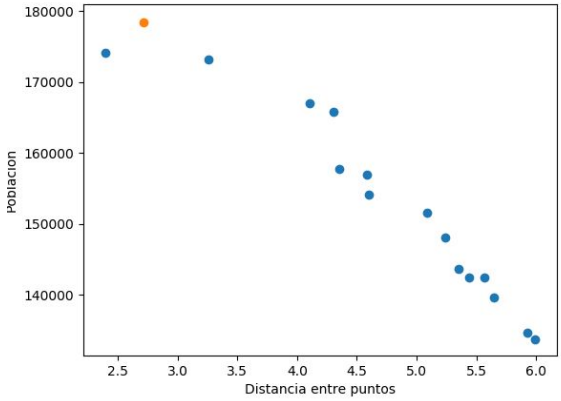


Gráfico 2 - Resultado de iteraciones independientes:





Del mismo modo, como se observa en los gráficos de la frontera de Pareto, se ve que no en todas las iteraciones se pueden obtener los mismo resultados.

COMPARACIÓN

Como se puede observar en los gráficos mostrados previamente, debido al uso de diferentes funciones objetivos para cada uno de los casos (mono-objetivo y multi-objetivo), las soluciones obtenidas por cada uno de los algoritmos son distintas. Esto ocasiona que los puntos mostrados en cada caso nunca sean los mismo.

De igual manera, en cada iteración de algoritmo multi-objetivo, se obtiene un universo distinto de respuestas óptimas ya que cada uno parte con una población inicial distinta. Asimismo, este universo se da debido a que al tener más de una función de evaluación y se tenga que comparar

más de una dimensión, se tenga más de una respuesta óptima frente al caso del mono-objetivo en el que solo se tiene una respuesta óptima ya que solo se evalúa una dimensión.

También, se puede apreciar en los resultados que debido a la aleatoriedad que poseen estos algoritmos, la obtención de la solución óptima puede darse de manera muy rápida (80ra iteración) o tomar un mayor tiempo en encontrarse (416va iteración).

CONCLUSIONES

- Como se muestra en las tablas de resultados, se puede determinar que la cantidad de generaciones necesarias para la obtención de la solución óptima en el caso del algoritmo mono-objetivo está altamente correlacionada a la población inicial. Esto se puede apreciar viendo la cantidad máxima y mínima de iteraciones que fueron necesarias, las cuales fluctuaron entre 80 y 416 para una misma solución y diferentes valores para otras soluciones no tan óptimas.
- Al momento de utilizar un operador de selección como torneo o ruleta, se puede observar que la variabilidad aumenta en el resultado final ya que no siempre se logra obtener la mejor generación en este caso y más bien se usa la mejor de un pequeño subgrupo que generalmente se determina al azar.
- Dependiendo de la cantidad de variaciones que se realicen en los individuos a través de cada generación se puede modificar el proceso de convergencia con la solución óptima. En los casos mostrados solo trabajamos con una variación de dos individuos por generación la cual nos permitió tener un ratio de crecimiento/decrecimiento estable, pero si se aumenta la cantidad de transformaciones y cruzamientos que se dan en cada generación, se puede obtener una respuesta óptima de manera más rápida.
- Del mismo modo, en caso se tenga una variación de individuos muy alta en cada generación y no se maneje de manera adecuada el elitismo en la supervivencia de los individuos, se pueden obtener resultados no tan adecuados ya que se altera de sobremanera la población y se pueden introducir nuevos individuos con soluciones menos óptimas.
- Para el caso del algoritmo multi-objetivo empleado, la utilización del elitismo es primordial para su ejecución, puesto que al crear una población mayor tras la creación de la población hija, es necesario poder descartar los peores individuos de ambas de acuerdo a la comparación de los múltiples objetivos. De esta forma, se puede determinar una frontera de Pareto de manera más rápida y con un costo computacional menor que en otros algoritmos.
- Luego de realizar las 10 iteraciones con el algoritmo multi-objetivo, se aprecia que cada uno cuenta con una frontera de Pareto distinta, esto se debe a que cada uno parte con una población inicial distinta y pasa por transformaciones distintas en cada generación, lo que ocasiona que las respuestas óptimas de cada pasada no sean las mismas.
- En caso se desee hallar el universo de respuestas óptimas (Pareto) para el caso del algoritmo multi-objetivo, se debe de asegurar que se recorran el universo de todos los posibles individuos del problema. Sin embargo, esto implicaría realizar un algoritmo de fuerza bruta con un costo computacional muy elevado. Debido a esto, es que con el uso de algoritmos genéticos se sacrifica la obtención de algunas de las soluciones óptimas frente a una mayor velocidad en encontrar los resultados.

- Asimismo, utilizar estos algoritmos para el problema dado a pesar de que este no es tan complejo, resulta bastante óptimo, ya que si por ejemplo tomamos un aproximamiento de fuerza bruta en el caso del mono-objetivo, tendríamos que evaluar n! / ((n - m)! * m!) = 75394027566 combinaciones frente a las 10 x 500 = 5000 utilizadas. Sin embargo, debido a que no hemos recorrido todo el universo, no es posible asegurar que para ambos casos la respuesta final se la misma.
- Finalmente, el uso de estos algoritmos si es beneficioso ya que reduce de gran manera el costo computacional para obtener una respuesta adecuada en casos que se trabaje con problemas con una complejidad muy alta.