Introducción a la computación evolutiva

Trabajo de cursada

Estudiante

Figeiredo Benjamín

Docente

Dr. Virginia Yannibelli

Introducción

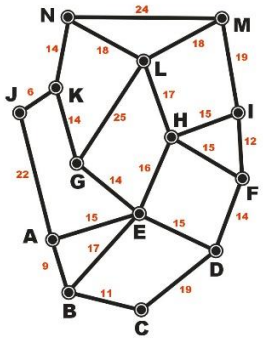
La computación evolutiva es un area perteneciente a las ciencias de la computación, que particularmente se centra en el estudio de algoritmos basados en la teoría de la evolución de Darwin, llamados algoritmos evolutivos. Estos algoritmos son meta-heurísticos (métodos aproximados diseñados para resolver problemas de optimización combinatoria, mejores que las heurísticas tradicionales) de búsqueda y optimización que respetan las bases de la evolución natural: la selección natural y las variaciones genotípicas (cruzamiento y mutación genética). Se enfocan en tres tipos de problemas: de optimización, de modelamiento y de simulación. En el presente trabajo se diseñará un algoritmo de este tipo para resolver el reconocido problema del viajante, un problema de optimización, el cual será presentado a continuación.

Problema presentado

Como se mencionó anteriormente, la problemática a abordar es el problema del viajante. Más concretamente, es el siguiente:

Dado un conjunto de N ciudades, y el costo del viaje entre cada par de esas ciudades, el problema del viajante consiste en encontrar el camino de mínimo costo que permita visitar todas las ciudades del conjunto (pasando sólo una vez por cada ciudad) y retornar a la ciudad de partida.

Para poder comprender el contexto, se presenta una representación visual del mismo



El objetivo entonces será diseñar un algoritmo evolutivo que sea capaz de encontrar un camino optimo dado un conjunto de puntos conectados entre sí, con pesos asociados a esas conexiones.

Algoritmo evolutivo diseñado

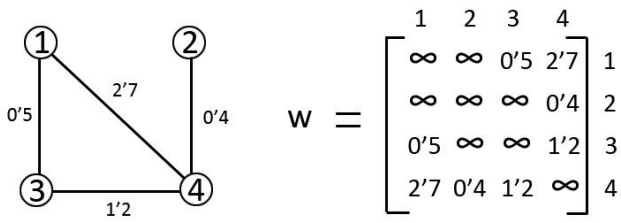
Datos concretos del problema

Como datos del problema se pueden encontrar:

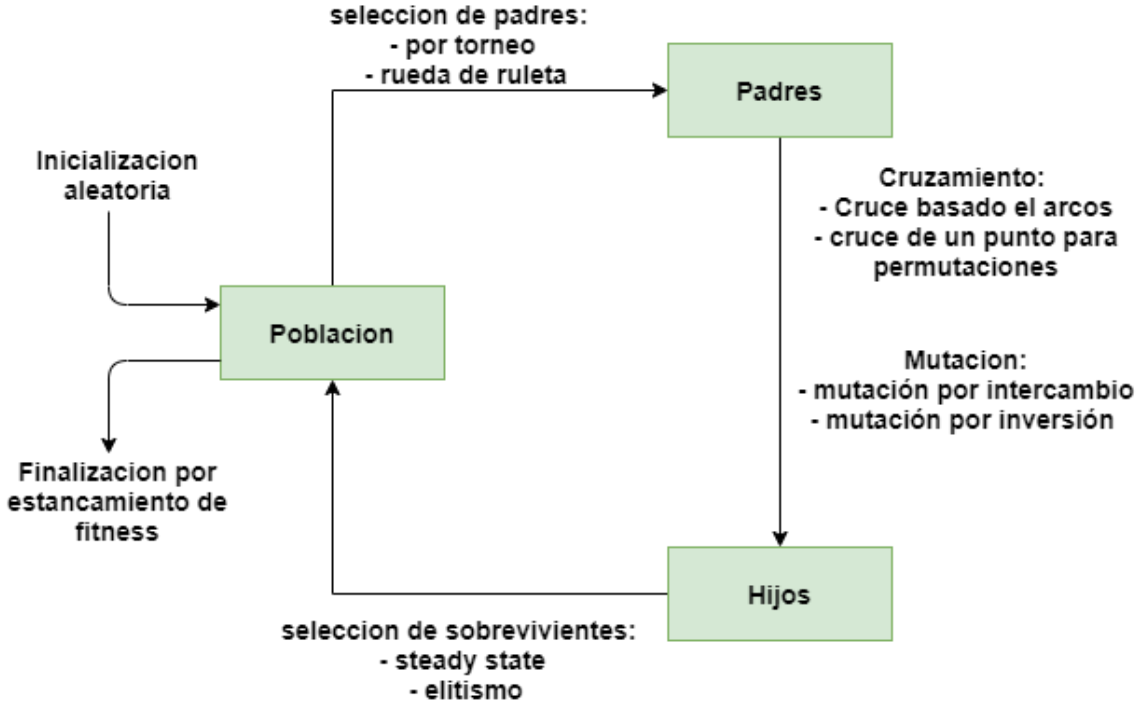
- ➔ N ciudades
- ➔ Costo (i, j): costo de ir desde la ciudad i a la ciudad j

Entrada

En este caso, la entrada será representada como una matriz de NXN donde las posiciones de la misma son ciudades, y los valores contenidos en las celdas representa el costo de viajar entre las mismas.

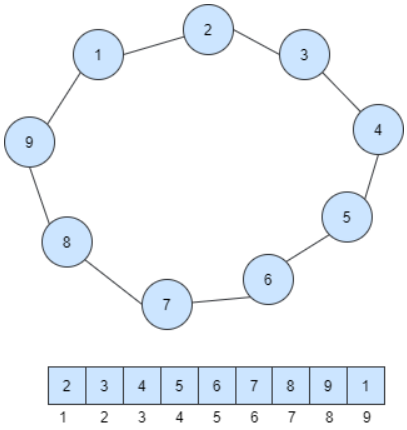


Esquema de ejecución general del algoritmo



Representación o codificación de las soluciones

El problema presentado corresponde a un problema de secuenciación (permutaciones). En estos algoritmos se debe decidir el orden en el que deberían ocurrir ciertos eventos; particularmente, en el problema del viajante es importante que ciudad es la siguiente inmediatamente después de la ciudad actual. Entonces, la representación de la solución se da mediante un arreglo de tamaño N (N arcos recorridos), en donde la posición del arreglo indica la ciudad actual y el valor en esa posición la ciudad a la que se ira a partir de la actual.



Proceso de decodificación

Para posteriormente poder calcular la función de evaluación, se necesita saber el costo de los arcos entre ciudades. Dado que las soluciones no contienen esta información, es necesario decodificarlas; para esto, cada posición del arreglo se toma en conjunto con su valor, para luego con estos datos acceder a la matriz de entrada, la cual contiene los datos necesarios para la función de fitness.

Función de evaluación

Según lo planteado en la problemática, se debe encontrar un camino mínimo que pase por las N ciudades y retorne a la de origen. La función de evaluación determina la calidad de las soluciones, y en este caso la calidad será medida mediante la longitud del camino que la solución representa. Entonces, la función de evaluación es

$$fitness(i) = \frac{1}{longitud\ del\ camino}$$

El hecho de utilizar la inversa de la longitud del camino hace que el problema de minimización pase a ser de maximización; dicho esto, lo que se busca es maximizar esta función de fitness.

Proceso de generación de la población inicial

El proceso de la construcción de la población inicial será a partir de la generación de soluciones aleatorias; estas deben ser compatibles con la codificación, por lo que se generaran arreglos que sigan la dupla <ciudad actual, ciudad siguiente>. Además, se debe asegurar en cada solución que todas las ciudades sean alcanzadas y no repetidas (ya que es una permutación), así como también que la última ciudad alcanzada sea adyacente a la ciudad de origen. Se podría aplicar alguna heurística que consiga soluciones buenas, pero se considera que no vale el esfuerzo ya que, en cuestiones de tiempo y resultado, se obtienen salidas similares.

Proceso de selección de padres

En primer lugar, se define un operador de selección de padres basado en ranking, ya que soluciona los problemas de la selección basada en fitness (convergencia prematura, selección aleatoria). Este operador define a cada solución una probabilidad en base al fitness relativo, planteando un ranking ascendente y asociando una probabilidad según la posición en el mismo. Para mapear el conjunto de ciudades a una probabilidad, se utiliza el mapeo exponencial de manera de tener una mayor presión selectiva, es decir, una mayor diferencia entre las probabilidades de las distintas soluciones.

Selección de padres: algoritmo de selección por torneo

Una de las posibles variantes de selección de padres ante una permutación es la selección por torneo. Para poder llevar acabo esta, se toman dos parejas (cuatro conjuntos de ciudades) al azar de la población, y de cada pareja se selecciona al mejor conjunto de ciudades, quedando así una única pareja de padres. Esta selección no requiere conocimiento global de la población, lo que la hace una buena opción ante poblaciones grandes. Como se mencionó, de cada pareja seleccionada al azar se toma el mejor conjunto; pero, ¿Cómo se comparan los mismos? Esto se hace estableciendo una política de comparación.

Esta política es determinada según cuatro factores, los cuales son: posición de del conjunto de ciudades en la población, tamaño k de la muestra (en este caso se eligió k = 2, pero hay que tener en cuenta que cuanto más grande sea k, más se incrementa la presión selectiva), probabilidad p de que el mejor conjunto de ciudades de la muestra sea seleccionada (en este caso se eligió p=1, de manera que sea determinístico) y si el conjunto de ciudades es elegido con o sin reemplazo (si al ser elegido el mismo, vuelven a la población o ya no son tenidos en cuenta).

Selección de padres: algoritmo de selección por rueda de ruleta

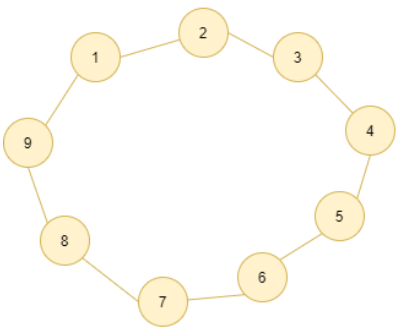
En el caso de la selección por rueda de ruleta, las ciudades son ubicadas en una rueda de ruleta en donde cada una ocupa un espacio proporcional a su probabilidad de selección; la rueda es girada N veces de manera de seleccionar a N conjuntos de ciudades que estarán en el mating pool.

Operadores de cruce genético

Operador de cruce: cruce basado en arcos

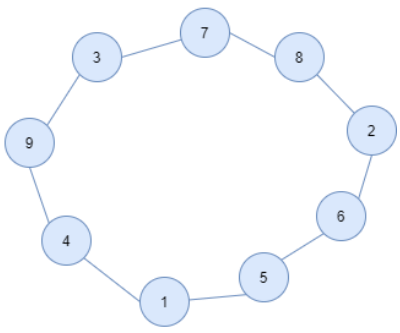
Este operador se basa en la idea de que un hijo debería ser creado solamente a partir de los arcos que están presente en uno o en los dos padres. Para poder lograr esto, se construye una tabla en donde cada ciudad hija tiene asociadas las adyacencias de sus padres, como se muestra a continuación

PADRE 1



1	2	3	4	5	6	7	8	9

PADRE 2



1	2	3	4	5	6	7	8	9

Ciudad	Arcos
1	2,5,4,9
2	1,3,6,8
3	2,4,7,9
4	1,3,5,9
5	1,4,6+
6	2,5+,7
7	3,6,8+
8	2,7+,9
9	1,3,4,8

HIJO CON ADYACENCIAS ASOCIADAS

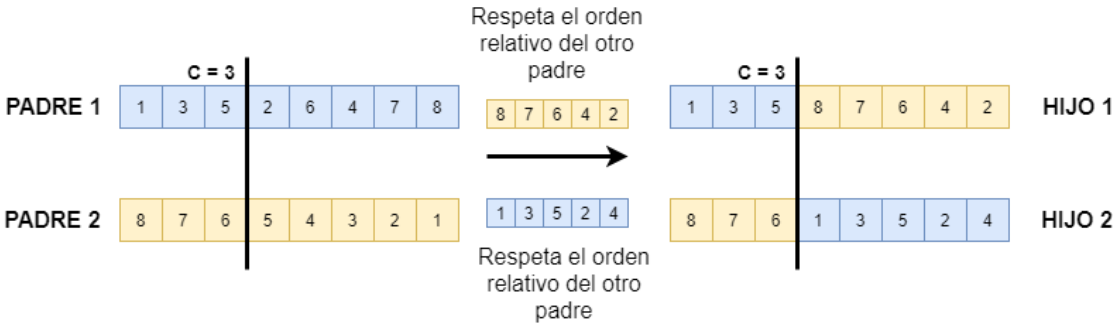
Con la tabla ya armada, se hace un proceso de selección de ciudades para el hijo según criterios (random, lista más corta de arcos, arcos en común) y se va construyendo el mismo. Se muestra un ejemplo

Choices	Element selected	Reason	Partial result
All	1	Random	[1]
2,5,4,9	5	Shortest list	[1 5]
4,6	6	Common edge	[1 5 6]
2,7	2	Random choice (both have two items in list)	[1 5 6 2]
3,8	8	Shortest list	[1 5 6 2 8]
7,9	7	Common edge	[1 5 6 2 8 7]
3	3	Only item in list	[1 5 6 2 8 7 3]
4,9	9	Random choice	[1 5 6 2 8 7 3 9]
4	4	Last element	[1 5 6 2 8 7 3 9 4]

Y de esta forma, se van generando los hijos a partir de la información de adyacencia de los padres.

Operador de cruce: cruce de un punto para permutaciones

Este operador combina la información de ambos padres según el cruce de un punto, pero respetando la condición de las permutaciones (cada valor ocurre solamente una vez en la solución). Para realizar esto, se toma un punto de referencia c ($1 \leq c < N$) y mediante la información de los padres, se generan los hijos de la siguiente manera

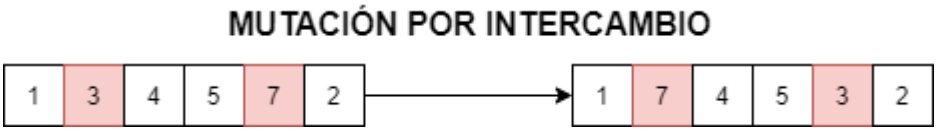


De esta manera, se generan los nuevos hijos a partir de información de ambos padres; al ser la solución representada como una permutación, es útil este cruce en el contexto del problema, ya que los hijos generados representan posibles secuencias de caminos válidos.

Operadores de mutación genética

Operador de mutación: mutación por intercambio

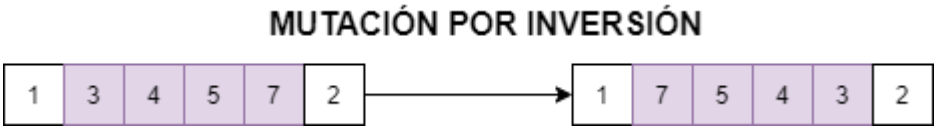
Este tipo de mutación es compatible con el problema ya que preserva información de adyacencias. El mismo consta de elegir dos posiciones al azar e intercambiar sus valores. Gráficamente se visualiza de la siguiente manera



Se puede observar que la información sobre adyacencias es preservada; en este caso, solamente se rompen cuatro links (1-3, 3-4, 5-7, 7-2).

Operador de mutación: mutación por inversión

Al igual que el operador anterior, este tipo de mutación también preserva la mayor parte de información sobre adyacencias. Para aplicar el mismo, se eligen dos posiciones al azar de la solución y se invierten los valores existentes entre dichas posiciones. Se muestra a continuación una representación gráfica como ejemplo



En este caso, la cantidad de links rotos es dos (1-3, 7-2), por lo que también es un buen operador.

Proceso de selección de sobrevivientes

Selección de sobrevivientes: Steady-state

Este mecanismo tiene como objetivo eliminar los peores individuos; para esto, se reemplazan los n peores conjuntos de ciudades por los n mejores conjuntos generados. Para evitar la convergencia prematura, se combina con la política de no permitir duplicados, aunque esto no sería indispensable en caso de que la población sea grande.

Selección de sobrevivientes: elitismo

El proceso de elitismo siempre preservará al mejor individuo hasta el momento, de esta manera asegurando que la solución obtenida hasta el momento no empeorará de una generación a la siguiente. En este caso, se podría combinar con la selección de padres por torneo (para preservar la diversidad).

Análisis respecto a parámetros

Probabilidad de cruce

La probabilidad de cruce debe ser elevada ya que el mismo combina y genera soluciones nuevas. Ya que el cruce sirve para la exploración del area de búsqueda, se propone un 100% de probabilidad de cruce.

Probabilidad de mutación

La probabilidad de mutación se encarga de incorporar características nuevas a la población. La misma explota áreas locales en busca de óptimos en las mismas, siendo este el motivo para que la probabilidad no sea muy elevada (convergencia prematura). En este caso, se propone un 5% de probabilidad de mutación.

Tamaño de la población inicial

El tamaño de la población inicial es importante para saber que procesos elegir a la hora de diseñar. Se determina un tamaño de población inicial proporcional a 100, ya que es suficiente para operar con los cruzamientos y mutaciones; de todas maneras, no se descuida que el tamaño de la población inicial tiene como factor influyente la cantidad de ciudades (N) y la cantidad de arcos entre ellas.

Condición de finalización para la ejecución del algoritmo

La condición de finalización del algoritmo se determina como la generación de 500 generaciones, y/o por estancamiento de fitness. El estancamiento de fitness se dará, en este caso, cuando de una generación a la otra la diferencia entre los fitness sea cercana a cero.

Conclusión

Los algoritmos evolutivos son de gran utilidad para la mejora de heurísticas ya existentes, ya que proveen la flexibilidad necesaria para la generalización de los problemas, así como también simpleza a la hora de diseñarlos, teniendo una gran variedad de implementaciones posibles las cuales se ajustan a las necesidades de los problemas.

Es importante tener un enfoque distinto sobre la resolución de problemas, para el día de mañana tener las herramientas necesarias para abordarlos. Con respecto a lo aprendido, los temas y problemas abordados fueron de gran interés, poniéndose por encima de la complejidad de los mismos, lo que es un punto a destacar.

A futuro, se espera seguir profundizando en este tema, con el objetivo de poder manejar con más dinamismo el area de computación evolutiva.