Problema del agente viajero

El problema del agente viajero (TSP Travelling salesman problem), consiste en un agente de ventas que tiene que visitar *n* ciudades comenzando y terminando en una misma ciudad, visitando solamente una vez cada ciudad y haciendo el recorrido con el costo mínimo.

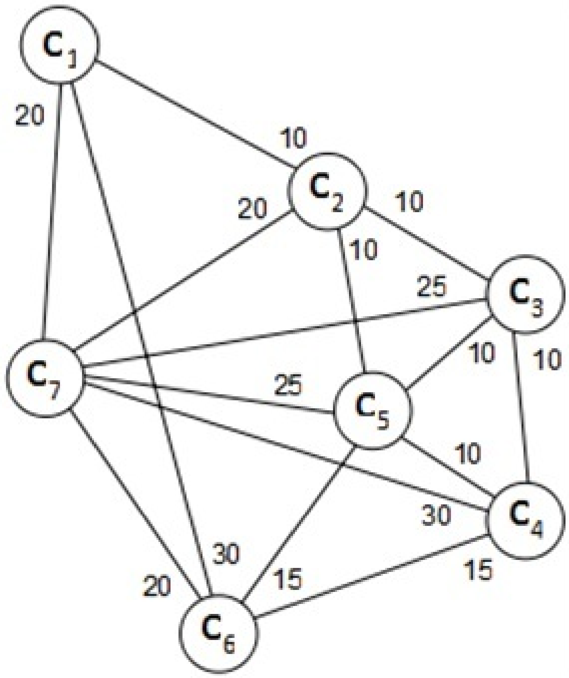
El costo del recorrido puede ser que esté representando distancia, tiempo u otro tipo de gasto asociado al recorrido.

En la práctica, para un problema del viajero con 5 ciudades hay 12 rutas diferentes y no es necesario usar una computadora para encontrar la mejor ruta, pero apenas aumentamos el número de ciudades las posibilidades crece exponencialmente:

- Para 10 ciudades hay 181440 rutas diferentes
- Para 30 ciudades hay más de $4{\cdot}10^{31}$ rutas posibles. Es decir una computadora que calcule un millón de rutas por segundo necesitaría 10^{18} años para resolverlo. Dicho de otra forma, si se hubiera comenzado a calcular al comienzo de la creación del universo (hace unos 13400 millones de años) todavía no se habría terminado.

El modelado del problema puede hacerse con un grafo ponderado dirigido donde los nodos están totalmente conectados, es decir, para cada par de nodos *i* y *j* siempre existe un lado que conecta al nodo *i* con el *j* y otro de *j* a *i*.

En general se considera que el costo de ir del nodo *i* al nodo *j* es diferente al costo de ir del nodo *j* al nodo *i*.



Para representar el costo asociado de ir de una ciudad a otra utilizaremos una matriz de costos donde el valor en la posición (*i*, *j*) representa el costo de ir de la ciudad *i* a la cuidad *j*.

Por ejemplo:

Matriz de costos

Ciudad	**0**	**1**	**2**	**3**	**…**	***n***
0	0	5	7	3	…	9
1	7	0	3	10	…	4
2	8	4	0	9	…	21
3	4	8	12	0	…	11
…	…	…	…	…	…	…
n	7	10	3	9	…	0

Los valores de la matriz serán generados aleatoriamente.

La representación de una posible solución al problema, es decir, un individuo de la población será representado por un vector con todas las ciudades en un orden dado, por ejemplo para 10 ciudades se tiene:

h1 = (0, 2, 4, 5, 1, 6, 7, 3, 9, 8, 0) (a)

Para generar la población inicial se tendrá que cada individuo en la población es un ordenamiento de las ciudades que se tienen que visitar.

La función de fitness será la suma de los costos de visitar las ciudades en un orden dado

Por ejemplo, para el individuo anterior (a):

Fitness(h1) = costo(0,2) + costo(2,4) + costo(4,5) + ….+ costo(8,0)

Operaciones

Mutación

La mutación en este caso será una permutación entre dos ciudades elegidas de forma aleatoria, por ejemplo:

h1 = (0,2,4,5,1,6,7,3,9,8) si para la mutación se eligen la ciudad 4 y 3

después de la mutación tendríamos al individuo h1' = (0,2,3,5,1,6,7,4,9,8)

Cruza

La cruza en este caso es un proceso más complicado que en la cruza que se realiza en un algoritmo genético porque hay que garantizar que todas las ciudades aparezcan en el vector y que no se repitan.

Hay algunas ideas de cómo realizar una cruza cuando se trabaja con ordenaciones, estas son: cruza parcialmente mapeada(PMX), cruza ordenada(OX) y cruza cíclica(CX).

Cruza parcialmente mapeada (PMX)

En esta cruza se construye un descendiente eligiendo una subsecuencia de un tour de alguno de los padres y preservando el orden y posición de tantas ciudades como sea posible del otro padre.

Una subsecuencia de un tour es seleccionada eligiendo dos puntos de cruza aleatorios los cuales servirán como fronteras de intercambio, por ejemplo:

h1 = (1 2 3| 4 5 6 7| 8 9)

y

h2 = (4 5 2| 1 8 7 6| 9 3)

Generamos descendientes de la siguiente manera. Primero, los segmentos entre los puntos de cruce son intercambiados (el símbolo x representa un valor que aún no se conoce)

d1 = (x x x| 1 8 7 6| x x)

y

d2 = (x x x| 4 5 6 7| x x)

Este intercambio define el siguiente mapeo:

1↔4, 8↔5, 7↔6 y 6↔7
Ahora hay que llenar con las ciudades faltantes con las que no haya conflicto, tomadas del padre original.

d1 = (x 2 3| 1 8 7 6| x 9)

y

d2 = (x x 2| 4 5 6 7| 9 3)

Finalmente, la primera x en h1, la cual debería ser 1, causa conflicto por lo que es reemplazada por 4 ya que así lo determina el mapeo 1↔4.

Similarmente la segunda x en h1 es reemplazada por 5, por el mapeo 8↔5.

De esta manera las x en h2 son reemplazadas por 1 y 8.

Así , con esta cruza los hijos serían

d1 = (4 2 3 1 8 7 6 5 9)

y

d2 = (1 8 2 4 5 6 7 9 3)

Cruza ordenada (OX)

En la cruza ordenada se construyen descendientes eligiendo una subsecuencia de un tour de un padre preservando el orden relativo de las ciudades del otro padre. Por ejemplo, dos padres con dos puntos de cruza:

h1 = (1 2 3| 4 5 6 7| 8 9)

y

h2 = (4 5 2| 1 8 7 6| 9 3)

Producirían dos descendientes de la siguiente manera.

Primero, los segmentos entre los puntos de cruza son copiados en los descendientes

d1 = (x x x| 4 5 6 7| x x)

y

d2 = (x x x| 1 8 7 6| x x)

Después, empezando desde el segundo punto de cruza de un padre, las ciudades del otro padre son copiadas en el mismo orden, omitiendo ciudades ya presentes. Al alcanzar el final de la cadena, continuamos desde la primera ciudad de la cadena. La secuencia de las ciudades en el segundo padre, desde el segundo punto de cruza, es:

9 – 3 – 4 – 5 – 2 – 1 – 8 – 7 - 6

Después de quitar las ciudades 4, 5,6 y 7las cuales ya se encuentran en el primer descendiente obtenemos:

9 – 3 – 2 – 1 - 8

Esta secuencia es puesta en el primer descendiente, empezando en el segundo punto de cruza.

d1 = (2 1 8| 4 5 6 7| 9 3)

Similarmente obtenemos el segundo descendiente

d2 = (3 4 5| 1 8 7 6| 9 2)

Cruza cíclica (CX)

En esta cruza se construyen dos descendientes en los cuales cada ciudad y su posición se toman de uno de los padres. Por ejemplo:

h1 = (1 2 3 4 5 6 7 8 9)

y

h2 = (4 1 2 8 7 6 9 3 5)

Podemos producir el primer descendiente tomando la primera cuidad del primer padre h1:

d1 = (1 x x x x x x x x)

Dado que la ciudad 4 en h2 se encuentra justo en la misma posición de la seleccionada ciudad 1, seleccionamos la ciudad 4. En h1 la ciudad 4 está en la posición 4 por lo que se tiene

d1 = (1 x x 4 x x x x x)

Siguiendo esta regla, ahora la ciudad 8 en h2 se encuentra en la misma posición que la ciudad 4 en h1 por lo que se elige 8

d1 = (1 x x 4 x x x 8 x)

Ahora la ciudad 3 en h2 se encuentra en la misma posición que la ciudad 8 en h1 por lo que se elige 3

d1 = (1 x 3 4 x x x 8 x)

La ciudad 2 en h2 se encuentra en la misma posición que la ciudad 3 en h1 por lo que se elige la ciudad 2

d1 = (1 2 3 4 x x x 8 x)

Ahora la ciudad que se encuentra en la misma posición en h2 que la ciudad 2 en h1 es la ciudad 1, que ya se encuentra en el descendiente, por lo que se cerró un ciclo, ahora las ciudades faltantes se toman del otro padre

d1 = (1 2 3 4 7 6 9 8 5)

Similarmente

d2 = (4 1 2 8 5 6 7 3 9)