Apunte

Nelson Efrain A. Cruz

June 7, 2010

1 Distancias y Vecindarios

1.1 Distancias

La idea es definir los vecindarios tomando en cuenta, como medida para separar las soluciones, la distancia entre ellas. Teniendo en cuenta que en TSP todas las soluciones posibles no son mas que permutaciones de una cantidad n de nodos. Definimos distancia de las soluciones x_1 a x_2 como:

$$\rho(x_1, x_2) = \text{número de nodos en diferentes posiciones}$$

Es decir dadas dos soluciones factibles distintas x_1 y x_2 que están descriptas por la siguiente sucesión de $n = 6$ nodos:

$$x_1 = [a, b, c, d, e, f]$$
$$x_2 = [c, a, b, d, e, f]$$
$$\rho(x_1, x_2) = 3$$

Ya que tienen 3 elementos en posiciones distintas, es de notar que la distancia mínima que puede haber entre dos soluciones es 2, ya que si un elemento de la solución 1 esta en una posición distinta que en la solución 2 hay forzosamente otro elemento que esta en una posición distinta en solución 1. La distancia máxima es igual a n, que seria el caso en que todos los nodos se hallen en posiciones distintas en ambas soluciones.

$$2 < \rho(x, y) < n \text{ con n igual a la cantidad de nodos del problema}$$

1.2 Vecindarios

Una ves definida la distancia entre soluciones podemos definir nuestra estructura de vecindarios como sigue:

$$x' \in N_k(x) \Leftrightarrow \rho(x', x) = k$$

Restaría probar que los conjuntos definidos por N_k son disjuntos y además que describen el total de las permutaciones posibles de los n nodos, osea probar que:

$$1 + \sum_{k=2}^{n} \mid N_k \mid = n!$$

Después falta por esclarecer de que modo vamos a recorrer los vecindarios y cuantos vecindarios vamos a usar, ya que podríamos usar los $n-1$ vecindarios posibles pero esta acarrearía una mayor cantidad de iteraciones, sino podríamos utilizar como vecindarios rangos de los vecindarios, algo así como $x' \in N_2(x) \Leftrightarrow \rho(x, x') = 2, 3, 4$ por poner un ejemplo. Aparte como se recorren los vecindarios también es una cuestión importante, es decir si empezamos con $k = 2$ o $k = n$.

2 Una mejora para VNS básico

2.1 Elección estadística del vecino

Una vez probado VNS básico note que si bien el algoritmo producia, relativamente, buenas soluciones este era un poco tonto, pues en pos de la diversificación se generaban de forma aleatoria los elementos de las vecindades correspondientes y dado ello se conseguían demasiadas comparaciones, un poco acercandose a la fuerza bruta, que eran muy probablemente desechadas pues los óptimos locales conseguidos no eran mejores que el óptimo global conocido. Es decir precisábamos una mejor manera de elegir los vecinos, tratando de reducir así las comparaciones realizadas o desde otro punto de vista que los vecinos elegidos sean "buenos vecinos" a los que al aplicarse una búsqueda local generasen buenas soluciones.
Una aproximación a ello fue que a medida que se obtenían nuevos óptimos locales se iba tomando cuenta de las aristas usadas (si se quiere de los pares de nodos) en una memoria de largo plazo, es decir una solución se puede ver como una sucesión de n nodos o como una sucesión de $(n-1)+1$ aristas (al ser el camino de un ciclo) y entonces cada vez que se usa una arista esto es registrado en la memoria aumentando en uno el contador correspondiente a esa arista. Así las aristas que aparezcan mas frecuentemente en los óptimos locales tendrán valores de contador altos. La idea tras esto es que aquellas aristas que aparezcan de forma seguida en las (buenas) soluciones muy probablemente formen parte de la mejor solución o bien que estas aristas sean buenas candidatas para formar parte de una buena solución.

2.2 Dirigir la búsqueda

Elegir del modo descripto los vecinos pretende encausar la diversificación o sea que la búsqueda no se disperse tanto.
Un problema puede ser que nunca se alcance el óptimo global

2.3 Implementación

Dada una arista podemos referirnos a ella como un par de nodos, que serian los nodos que conecta esa arista, así la arista (a, b) conecta los nodos a con b. Entonces si la arista (a, b) tiene un contador con un valor alto quiere decir que en las soluciones (buenas u óptimos locales) que fuimos encontrado la sub-sucesíon $[a, b]$ o $[b, a]$ apareció muy frecuentemente.
Entonces si dado que una arista (y, x) tiene un valor de contador alto hubiéramos optado por que esta arista se siga usando si o si en el generado vecino, podríamos quedar atrapados en un óptimo local.(REVISAR!!!!!!)
Por ello se opto que la arista con contador mas elevado tendrá la probabilidad

mas alta de seguir siendo usada en el vecino y las demás aristas tendran una probabilidad proporcional dependiendo del valor de su contador.

Armar un vecino de x correspondiente al vecindario k se logra cambiando de posición k nodos en el vector solución x, asi para aplicar la memoria de largo plazo al algoritmo que crea los vecinos de x basta con que cada vez que estemos por elegir un nodo a ser cambiado de posición revisemos cual es la probabilidad de que este se mantenga en su posición y en base a ello decidamos si mover o no. Hay que tener en cuenta que mover un nodo de posición implica dejar de usar dos aristas, por ejemplo dada una solucion

$$[a, b, c, d, e, f]$$

mover de lugar c implica que las aristas

$$(b, c) \text{ y } (c, d)$$

se dejen de usar.