Traveling Salesman Problem
Edgar Osvaldo López Zúñiga

Introducción

El problema del agente viajero o TSP (Traveling Salesman Problem) es quizá el problema de optimización combinatoria más conocido. El problema consiste en N puntos o ciudades a visitar, se tiene un costo para el viaje entre cada par de puntos. Se asume que un vendedor, comenzando de cualquiera de las ciudades, tiene que visitar todos los puntos exactamente una vez y regresar al punto inicial. El objetivo es encontrar un recorrido óptimo para el que la distancia total o costo total del viaje se minimiza.

En este trabajo se trata con un caso especial de este problema y se intenta optimizar los recorridos mediante el uso de métodos de búsqueda local y el uso de un algoritmo genético con distintas consideraciones en cuanto a representación y operador de cruza. También se combinan los algoritmos utilizados para intentar brindar mejores soluciones al problema que las que se obtienen mediante el uso de estas técnicas por separado.

Desarrollo

Sea $G = (V, E)$ un grafo y $\mathbb{F}$ la familia de todos los ciclos Hamiltonianos en G. Para cada arista $e \in E$ se puede asignar un costo c_e. Después, se puede definir el TSP como el problema de encontrar un recorrido o ciclo Hamiltoniano en G tal que la suma de los costos de todas las aristas en el recorrido se minimice.

Si el conjunto de los nodos del grafo es $V = \{1, 2, ..., n\}$, se puede definir una matriz de costos o distancias $C = (c_{ij})_{n\times n}$ donde c_{ij} es el costo de la arista que une al nodo i con el j dentro de G. Sin perder la generalidad, se puede asumir que G es un grafo completo, y en caso de que no lo sea, se pueden reemplazar las aristas que no forman parte del grafo con aristas de un costo muy alto.

Dependiendo de la naturaleza de G y de su matriz de costos, se puede dividir al TSP en dos clases principales. Si G es un grafo no dirigido (C es simétrica), el TSP es llamado el problema del agente viajero simétrico (Symmetric Traveling Salesman Problem) o STSP. Si C es no necesariamente simétrica, entonces es llamado el problema del agente viajero asimétrico (o ATSP). Como cada grafo no dirigido puede verse con un grafo dirigido con aristas del mismo costo entre dos nodos, el STSP puede verse como un caso especial de ATSP. También es posible verificar que si se duplica el número de nodos, se puede representar al ATSP como un STSP.

Existen distintas formas de representar el TSP. Una de estas formas es tratar al TSP como un problema de permutaciones, en general existen dos variantes de la representación como permutación del TSP. La primera de ellas es la representación cuadrática del problema: Sea $\mathbb{P}_n$ la colección de todas las permutaciones del conjunto $\{1, 2, ..., n\}$. El TSP consiste en encontrar una permutación $\pi = \{\pi(1), \pi(2), ..., \pi(n)\}$ de $\mathbb{P}_n$ tal que

$$c_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

se minimiza. Para esta representación $(\pi(1), \pi(2), ...\pi(n))$ representa el orden en el que las ciudades son visitadas. Cada desplazamiento cíclico de π corresponde al mismo recorrido, por lo que existen n diferentes permutaciones representándolo.

Otra representación del TSP como permutación es nombrada representación lineal de permutaciones. En esta representación, solo se consideran factibles las permutaciones cíclicas. Sea $\mathbb{C}_n$ la colección de todas las permutaciones cíclicas de $\{1, 2, ..., n\}$. El TSP consiste en encontrar $\sigma = (\sigma(1), \sigma(2), ..., \sigma(n)) \in \mathbb{C}_n$ tal que

$$\sum_{i=1}^{n} c_{i\sigma(i)}$$

sea minimizado. En esta representación $\sigma(i)$ nos dice el nodo que sigue al nodo i en la permutación.

Variantes del TSP

Se mencionan algunas variantes del TSP que pueden ser reformuladas como un TSP realizando distintas transformaciones.

- **MAX TSP**: El objetivo consiste en encontrar un recorrido en G donde el costo total de las aristas del ciclo se maximizan. Este problema se puede transformar en el TSP al reemplazar los costos de cada arista por su inverso aditivo.
- **Bottleneck TSP**: El objetivo es encontrar un recorrido en G tal que el mayor costo de las aristas en el recorrido sea minimizado. Este problema se puede formular como un TSP donde los costos crecen exponencialmente.
- **TSP con múltiples visitas**: Se quiere encontrar un recorrido que comienza en un nodo i de G, visita cada nodo al menos una vez y regresa al nodo i de tal forma que la distancia recorrida sea minimizada. Este problema se puede transformar al TSP reemplazando los costos de las aristas con las distancias más cortas entre cada par de nodos en G.
- **Clustered TSP**: El conjunto de nodos de G se particiona en distintos grupos. El problema consiste en encontrar un recorrido de menor costo en G en donde las ciudades dentro del mismo grupo deben ser visitadas consecutivamente. El problema puede ser transformado en un TSP añadiendo un costo muy alto entre nodos de distintos grupos.
- **TSP Generalizado**: Consiste en particionar el conjunto de nodos de G en distintos grupos. Se quiere encontrar el ciclo más corto de G tal que se visite exactamente un nodo de cada grupo.
- **TSP con m agentes**: Se tiene m agentes localizados en un nodo inicial de G. Comenzando por ese nodo, cada uno de los agentes visita un subconjunto V_i de los nodos de G una sola vez y regresa al nodo inicial. El problema consiste en encontrar una partición del conjunto de nodos $V - \{1\}$ y un recorrido para cada uno de los m agentes tal que $|V_i| \geq 1$ para cada i, la unión de todos los recorridos sea el conjunto de los nodos menos el nodo inicial, y su intersección sea nula.

Problema tratado

Para este trabajo, se trataron distintas instancias del TSP simétrico. Las instancias utilizadas se obtuvieron de la TSPLIB, una librería con problemas específicos para distintas variantes del TSP y problemas relacionados.

Para cada instancia se cuenta con un archivo con una sección de especificación del problema y una sección de datos. En la sección de especificaciones se cuenta con información como el nombre de la instancia, el tipo de problema, la dimensión del problema, el tipo de aristas y su formato, el tipo de coordenadas, etc. En cuanto a la sección de los datos, el formato puede variar dependiendo de las especificaciones anteriores.

Para este trabajo, se trató con problemas específicos para el STSP donde los pesos de las aristas están dados por la distancia euclidiana entre pares de nodos. Los datos en las instancias euclidianas están dados por las coordenadas dentro del plano de cada uno de los nodos. También se trató con instancias en donde los costos son dados de forma explícita, ya sea como una matriz triangular superior, o como una matriz triangular inferior. En todos los casos considerados, se trata al problema de la misma manera, la única diferencia consiste en el preprocesamiento de los datos de entrada para cada instancia.

Instancia	Dimensión	Tipo	Óptimo
fri26	26	Matriz TI	937
kroA100	100	Euclidiana	21282
rd100	100	Euclidiana	7910
kroA200	200	Euclidiana	29368
rd400	400	Euclidiana	15281
pcb442	442	Euclidiana	50778
si535	535	Matriz TS	48450
rat575	575	Euclidiana	6773
pr1002	1002	Euclidiana	259045
si1032	1032	Matriz TS	92650

Tabla 1: Instancias utilizadas. TI es triangular inferior, TS es triangular superior.

Métodos de mejora

Los métodos de mejora son procedimientos que inician en una solución dada e intentan mejorar la solución mediante cambios iterativos manipulando componentes básicos de esta. En el caso de problemas relacionados con grafos, estos componentes pueden ser nodos, aristas, subcaminos y otras construcciones. Se pueden distinguir tres clases principales de métodos de mejora de acuerdo al tipo de estructuras de vecindario utilizadas:

- **Métodos constructivos**: Van agregando nuevas componentes para crear una nueva solución mientras se mantienen algunas componentes fijas. Entre estos métodos se encuentran algoritmos que utilizan componentes de distintas soluciones y algoritmos que eligen parámetros de construcción diferentes haciendo uso de iteraciones previas.
- **Métodos de trayectoria**: Usualmente llamados métodos de búsqueda local. Se mueven iterativamente de una solución a otra dependiendo de la definición del vecindario.
- **Métodos poblacionales**: Generalizan los métodos anteriores y consideran vecindarios de más de una solución como fundamento para crear nuevas.

Búsqueda local y k-intercambio

En un método de búsqueda local, se introduce una estructura de vecindario para generar movimientos de una solución a otra. Se dice que se ha llegado a un óptimo local cuando una solución no puede ser mejorada mendiante movimientos dentro del vecindario. Las estructuras de vecindario fundamentales para el TSP se basan en intarcambios de aristas e inserciones de nodos. Uno de los procedimientos clásicos de este tipo es el k-intercmabio. De acuerdo con la forma en la que se define un óptimo local, se dice que una solución es k-óptima (k-Opt) si ya no es posible mejorarla mediante movimientos definidos por el método de k-intercambios.

El método 2-intercambio o 2-opt es el más sencillo en la categoría de los k-intercambio. Este procedimiento es un método de mejora de búsqueda local y requiere de una solución inicial factible para iniciar. El algoritmo consiste en reemplazar dos aristas no adyacentes (v_i, v_i+) y (v_j, v_{j+}) por otras dos (v_i, v_j) y (v_{i+}, v_{j+}) que son las únicas aristas que pueden reemplazarlas una vez que se eliminan. Para mantener una orientación consistente, es necesario que el subcamino $(v_{i+}, ..., v_j)$ sea invertido. Cuando se invierte el camino $(v_{i+}, ..., v_j)$, el camino $(v_i, v_{i+}, ..., v_j, v_{j+})$ se reemplaza por $(v_i, v_j, ..., v_{i+}, v_{j+})$. El cambio en el costo producido por un movimiento 2-intercambio se puede expresar como $\Delta_{ij} = c_{ij} + c_{i+,j+} - c_{ii+} - c_{jj+}$. Se llega a un solución 2-óptima o 2-opt cuando se aplican movimientos de 2-intercambio iterativamente hasta que no haya un movimiento posible que resulte en un valor de Δ negativo.

La idea del procedimiento 2-opt puede ser generalizada a movimientos k-opt que eliminan k aristas y agregan k nuevas. Existen $\binom{n}{k}$ formas posibles de eliminar las k aristas en un recorrido y $(k-1)!2^{k-1}$ formas de reconectar los caminos desconectados. Es por esto que el uso de movimientos k-opt con $k > 3$ resulta impráctico, a menos que se reduzca el tamaño del vecindario utilizando técnicas especiales.

Algoritmos Evolutivos

Existen distintas variantes de algoritmos evolutivos. La idea común detrás de ellos es la misma: dada una población de individuos y un ambiente con recursos limitados, la competencia por estos recursos causará la selección natural. Esto a su vez, causa un aumento en la aptitud de la población.

Dada una función de calidad a maximizar, se puede crear aleatoriamente un conjunto de soluciones candidatas. Después se puede aplicar esta función a ellas para elegir a los mejores individuos que formarán parte de la siguiente generación. Esta transferencia de individuos a la siguiente generación se hace mediante operadores de recombinación o de mutación. La recombinación consiste en seleccionar dos o más candidatos (llamados padres) para producir una o más nuevas soluciones (llamada hijas). La mutación se aplica a un solo candidato y resulta en uno nuevo. Así utilizando estos operadores se generan nuevos candidatos (descendencia). La aptitud de estos candidatos es evaluada y después compiten con los anteriores para tomar lugar en la siguiente generación. Este proceso se repite hasta encontrar una solución lo suficientemente buena o hasta que se alcanzan las condiciones de terminación impuestas al algoritmo.

Varias componentes de un algoritmo evolutivo son estocásticas. Durante la selección, los mejores individuos no serán elegidos de forma determinista, y los individuos peor adaptados tienen una oportunidad de formar parte del conjunto de padres. Durante la recombinación, las partes que serán heredadas de cada padre son elegidas aleatoriamente. De la misma manera con la mutación, las piezas de una solución que serán modificadas, y las piezas que las sustituirán, son elegidas aleatoriamente.

Una iteración de un algoritmo evolutivo consiste en lo siguiente:

- Seleccionar a los padres
- Recombinar parejas de padres
- Mutar a la descendencia resultante
- Seleccionar individuos para la siguiente generación

En general, todos los algoritmos evolutivos siguen un esquema similar al descrito, difiriendo principalmente en detalles técnicos. Particularmente, algunas corrientes se caracterizan por su representación. Los algoritmos genéticos se caracterizan por representar las soluciones como cadenas o arreglos evaluados en un alfabeto finito.

Las partes que conforman un algoritmo evolutivo son:

- Representación
- Función de aptitud
- Población
- Mecanismo de selección de padres
- Operadores de variedad, recombinación y mutación.
- Selección de supervivientes.

Discusión de implementación y metodología

Para trabajar con el STSP se utiliizó una representación cuadrática del problema como permutación. Para la matriz de costos, se utiliizó un arreglo de C++, con los costos precalculados para cada par de nodos. Se implementó un método para cargar instancias euclidianas y con matrices explícitas de la TSPLIB.

En cuanto a los métodos de solución probados, se implementó una búsqueda local con movimientos 2-opt y un algoritmo genético en el que la representación utilizada es la misma que ya se mencionó. Para la búsqueda local, la implementación se hizo considerando los cambios en el costo que habría al intercambiar cada par de aristas y cuando este cambio es negativo, se procede a realizar un movimiento de 2-intercambio. Este procedimiento continúa hasta que ya no es posible mejorar la solución mediante movimientos de este tipo.

Para el algoritmo genético se hicieron más consideraciones. La primera de ellas a tomar en cuenta es la representación. Como se mencionó, la representación principal utilizada es la del recorrido como permutación. En esta representación, se almacena el recorrido en un arreglo que indica el orden en que las ciudades son visitadas. La segunda consideración a tomar en cuenta, es la selección de los padres, esta selección se hizo mediante torneo binario, es decir, se seleccionan dos soluciones candidatas y se elige de ellas la que tenga un costo total menor. Se recombina el material de dos soluciones padre para generar 4 descendientes.

Para la recombinación en el algoritmo genético se tuvieron que hacer distintas consideraciones. La primera de ellas es que en una representación cuadrática del recorrido como permutación no es posible aplicar los operadores clásicos de cruza del algoritmo genético debido a que se generarían soluciones con repetición de nodos. Por ejemplo, utiliizando un operador de cruza a dos puntos para las soluciones padre $(1, 2, 3, 4, 5, 6, 7)$ y $(7, 2, 4, 6, 1, 3, 5)$, si se aplica este operador entre los nodos en la posición 3 y 4 y los nodos en las posiciones 5 y 6, se generarían los siguientes descendientes: $(1, 2, 3, 4, 6, 6, 7)$ y $(7, 2, 4, 4, 5, 3, 5)$, que no solo son recorridos que no pasan por todos los nodos, sino que también visitan distintos nodos más de una vez, por lo que no son soluciones factibles al TSP. Una solución que se puede brindar para este problema es utilizar un operador de cruza especializado para el TSP. En el caso de este trabajo, se utilizó un operador de cruza ordenada, en donde se selecciona un porcentaje de la solución que es copiada de uno de los padres, y los nodos restantes, son copiados con el orden del otro padre. Por ejemplo, para los padres considerados anteriormente, supongamos que la solución $(1, 2, x, x, 5, 6, x)$ ha compiado las posiciones que no están marcadas con una x del primer padre. El resto de las posiciones correspondientes a los nodos 3, 4 y 7 se completan en el orden del otro padre, es decir, la solución final sería: $(1, 2, 7, 4, 5, 6, 3)$.

Otra forma de brindar una solución al problema de la recombinación, es utilizando los operadores de cruza clásicos del algoritmo genético, pero cambiando la representación. En este caso, es conveniente utilizar la representación en secuencias de inversión de una permutación. Para una permutación $(\pi(1), \pi(2), ..., \pi(n))$ se denota por a_j al número de enteros en la permutación que preceden a j pero que son mayores que j. La secuencia $a_1, a_2, ..., a_n$ es llamada la secuencia de inversión de la permutación $(\pi(1), \pi(2), ..., \pi(n))$. Por ejemplo, la secuencia de inversión de la permutación $(6, 2, 3, 4, 1, 7, 5)$ es $4, 1, 1, 1, 2, 0, 0$. Donde el 2 en la quinta posición indica que hay dos números en la permutación que son mayores a 5, pero se encuentran antes que el 5. La secuencia de inversión de una permutación satisface las condiciones

$$0 \leq a_i \leq n - i \text{ para } i = 1, 2, ..., n$$

y no hay restricciones en cuanto a repetición de elementos, lo que es conveniente para los operadores de cruza y mutación del algoritmo genético, ya que no hay que preocuparse por adaptarlos al problema específico, porque es posible aplicarlos sobre esta representación.

Tanto la cruza ordenada como la representación como secuencias de inversión de una pemutación fueron implementadas para el algortitmo genético. También se implementó una opción para aplicar una búsqueda local 2-opt a los hijos después de la recombinación, para de esta manera poder utilizar un algoritmo híbrido entre el genético y la búsqueda local.

Para una población de tamaño n se generan $2n$ hijos y de estos se seleccionan los n mejores para formar parte de la población final, siempre se incluye como parte de la población al elite histórico.

Tanto para la búsqueda local como para el algoritmo genético, las soluciones iniciales son generadas aleatoriamente asegurando que sean recorridos factibles para el TSP.

Configuración utilizada

Para la implementación se utilizó el lenguaje de programación C++. En cuanto a las ejecuciones, se realizaron 30 ejecuciones en paralelo para cada uno de los métodos utilizados para las instancias mencionadas en la Tabla 1. Las ejecuciones se realizaron en el cluster de supercómputo el Insurgente. Se utilizaron nodos Intel Xeon E-2620 v2 con 12 núcleos por nodo.

Resultados

Conclusiones

Referencias