Titulo del paper

Carlos Cruz
Carné: 10-10168

Gabriel Gedler
Carné: 10-10272

Abstract

El Problema del Agente Viajero o TSP es uno de los problemas más estudiados en la ciencia de la computación. En este trabajo se estudian las mejoras de tiempo que puede ofrecer el uso de metaheurísticas para obtener soluciones óptimas (o subóptimas).

1. Introducción

Actualmente, el Problema del Agente Viajero o TSP (abreviado del inglés *Travelling Salesman Problem*), es uno de los problemas NP-duro más estudiados en la ciencia de la computación; y, debido al gran número de aplicaciones que tiene, se busca poder encontrar soluciones óptimas lo más rápido posible. En este trabajo se estudia el uso de metaheurísticas que permitan obtener soluciones óptimas (o subóptimas bastante cercanas a la óptima) en un tiempo considerablemente menor al que utilizaría un algoritmo voraz (*greedy*).

2. Descripción del problema

El Problema del Agente Viajero consiste en, dado un conjunto de ciudades, encontrar el camino más corto posible que pase exactamente una vez por cada ciudad y regrese a la ciudad de partida. Este problema es de complejidad NP-duro (lo que implica que es de complejidad NP). Esto implica que su orden de crecimiento no es polinomial si no, en este caso, el orden es de $O(n!)$.

Este es uno de los problemas de la ciencia de la computación más estudiados actualmente, por el gran número de aplicaciones que tiene, tanto dentro como fuera de la computación, como: planificación de viajes por ciudades, diseño de microchips, estudio de problemas genéticos como secuencias de ADN, entre muchos otros.

3. Trabajos previos

Johnson y McGeough[1] sugieren varios algoritmos de recorrido sencillo para generar soluciones iniciales en un problema TSP y, además, recopilan distintas optimizaciones de soluciones iniciales propuestas por diversos autores.

Antosiewicz, Koloch y Kamiński[2] comparan las metaheurísticas más comunes aplicadas al Problema del Agente Viajero.

4. Representación de la solución

Para modelar este problema se utiliza un grafo implícito en el que cada vértice representa una ciudad, cada arco representa la existencia de un camino entre cada par de ciudades y el peso de cada arco representa el costo de viajar entre cada par de ciudades

Para representarlo computacionalmente se utiliza:

- numNodos: almacena el número de nodos que tiene el grafo.
- caminos: es un arreglo de enteros que almacena el número de identificación de la ciudad a la que se viajará a continuación. Esto permite conocer el sucesor de cada nodo en orden de complejidad lineal, lo cual ayuda enormemente para optimizacion 2-opt. Tambien otorga la idea de un ciclo dirigido, que tiene mas sentido que con una matriz de adyacencias.
- costos: es una matriz de números reales que almacena el costo de viajar entre cualquier par de ciudades (nodos). Esto permite realizar busquedas de costos en orden constante.

5. Implementación de búsqueda local

Búsqueda local.

Algorithm 1: 2-opt

Input: Arreglo de caminos A
matriz de costos M
costo del recorrido C
Output: Nuevo arreglo de caminos A'
Nuevo costo del recorrido C'

```
1  Boolean mejora = true
2  while mejora do
3      mejora = false
4      C' = C
5      foreach i = 1...|A| do
6          foreach j = 1...|A| do
7              int i' = A[i]
8              int j' = A[j]
9              CostoViejo = M[i][i'] + M[j][j']
10             CostoNuevo = M[i][j'] + M[j][i']
11             if CostoViejo > CostoNuevo then
12                 mejora = true
13                 C' = C' - CostoViejo + CostoNuevo
14                 Actualizar los caminos
15 return C'
```

Conclusiones

Aquí irán las conclusiones.

Referencias

[1] Johnson, D. and McGeoch, L. (1995). *The Traveling Salesman Problem: A Case Study in Local Optimization*. pp. 8-27.

[2] Antosiewicz, M., Koloch, G. and Kamiński, B. (2013). *Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed.*

Apéndice

Aquí irá el apéndice.