EL PROBLEMA DEL AGENTE VIAJERO USANDO LA HEURÍSTICA DE ACEPTACIÓN POR UMBRALES *(Threshold Accepting)*

Sandra del Mar Soto Corderi
No. cuenta: 315707267

13 de noviembre de 2020

RESUMEN

El Problema del Agente Viajero es un problema *NP-Hard* que responde la siguiente pregunta; "dada una lista de ciudades y las distancias entre cada par de ellas, ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?"
Que en su versión de optimización dice "Dada una gráfica completa con pesos $G = (V, E)$, encontrar un ciclo Hamiltonianto en G con un peso mínimo (si acaso existe)".
El método de Aceptación por Umbrales *(Threshold Accepting)* es mucho más simple que Recocido Simulado *(Simmulated Annealing)* y es el motivo por el cual se usará para TSP.

1. Introducción

Para la resolución de TSP, primero se contruyó una gráfica $G = (V, E)$ con las ciudades y sus respectivas distancias entre las ciudades con una base de datos que se nos fue dada. Esta gráfica ponderada tiene una función de de peso para las aristas $w : E \to \mathbb{R}^+$. Sea $S \subset V$ una instancia de TSP que se quiera resolver.

Para hacer más sencillo la versión de optimización de este problema se hara uso de una gráfica completa $G_s = (V_s, E_s)$, donde $V_s = S$ y $E_s = \{(u, v)|u, v \in S \wedge u \neq v\}$, con la función de peso amentada $w_s : E_s \to \mathbb{R}^+$ definida como:

$$w_s(u, v) = \begin{cases} w(u, v) & \text{si } (u, v) \in E \\ d(u, v) \times \text{máx}_d(S) & \text{e.o.c.} \end{cases}$$

donde $d(u, v)$ es la distancia natural entre dos vértices u y v; y $max_d(S)$ será la distancia máxima de S.

La **distancia natural** se utiliza para calcular la distancia natural entre dos elementos de S para de esta manera tener nuestra gráfica completa, se utilizarán las coordenadas de las ciudades correspondientes. Su fórmula está definida por

$$d(u, v) = R \times C$$

donde R es el radio de la Tierra y C está definida como $C = 2 \times \arctan(\sqrt{A}, \sqrt{1 - A})$.

La **distancia máxima** de S se define como:

$$\underset{d}{\text{máx}}(S) = \text{máx}\{w(u, v)|u, s \in S \wedge (u, v) \in E\}$$

Una solución de TSP dada una instancia S, es cualquier permutación de de los elementos de S y se puede asegurar que estas soluciones son válidas para G_s. Dándonos así, dos tipos de permutaciones: las factibles y las no factibles. Decimos que es factible si y sólo si las aristas entre dos elementos consecutivos de la permutación existen en E. Si al menos una arista no existe, decimos que no es factible.

La optimización de TSP necesitará una **función de costo** que evaluará "qué tan buena (o no) es una solución" para así poder comparar burdamente las evaluación de ésta y así poder decidir cuál es mejor utilizando el criterio de entre más *pequeña* sea la evaluación es *mejor*. Para esto se va a normallizar la función de costo, pero para esto se necesitará un normalizador.

El normalizador, como su nombre lo dice, normalizar la función de costo y nos dirá si una solución S es factible si esta evaluada entre 0 y 1, y que una solución no factible se evalue con un valor mayor que 1.

Función de costo, sea $S \subset V$ una instancia de TSP: la función de costo f de una permutación $P = v_{\rho(1)}, \ldots, v_{\rho(k)}$ de los elementos de S está definida como:

$$f(P) = \frac{\sum_{i=2}^{k} w_s(v_{\rho(i-1)}, v_{\rho(i)})}{\mathcal{N}(S)}$$

2. Aceptación por Umbrales *(Threshold Accepting)*

Dado un problema $\mathscr{P}$ de optimización y clasificado como NP-duro, sea S el conjunto de posibles soluciones a una instancia de $\mathscr{P}$. Se supondrá que se tiene una función $f : S \to \mathbb{R}^+$, (función objetivo), tal que $0 \leq f(s) \leq \infty$ para cualquier $s|inS$. Dadas s, s' si $f(s) < f(s')$, entonces se considerará a la solición s mejor que s'.

La idea central de **la aceptación por umbrales**, es dada una temperatura inicial $T \in \mathbb{R}^+$ y una solución inicial (obtenida de alguna manera). de forma aleatoria buscar una solución vecina s' tal que $f(s') \leq f(s) + T$, y entonces actualizar s para que sea s'; en este caso diremos que la solución s' es *aceptada*. Se continúa de esta manera mientras la temperatura T es disminuida paulatinamente siguiendo una serie de condiciones: el proceso termina cuando $T < \epsilon$; cuando se han generado un determinado número de soluciones aceptadas; o cuando otra serie de condiciones es satisfecha.

3. Elaboración del programa

Este proyecto se inició en el lenguaje de programación Rust

4. Configuración del sistema

5. Resultados

5.1. Instancia con 40 ciudades

Con los identificadores de las ciudades:

```
1,2,3,4,5,6,7,75,163,164,165,168,172,327,329,331,332,333,489,490,491,492,493,496,652,653,654,
656,657,792,815,816,817,820,978,979,980,981,982,984
```

Usando la semilla **635** nos da los siguientes resultados:

```
Ruta: [980, 327, 871, 331, 164, 984, 491, 492, 489, 4, 817, 978, 5, 6, 165, 3, 333, 981, 820,
332, 982, 816, 823, 7, 654, 490, 653, 656, 2, 661, 657, 168, 1, 815, 496, 172, 163, 329, 493,
979]
Costo: 0.21751778855705842
¿Es factible?: true
```

5.2. Instancia con 150 ciudades

Con los identificadores de las ciudades:

```
1,2,3,4,5,6,7,8,9,11,12,14,16,17,19,20,22,23,25,26,27,74,75,151,163,164,165,166,167,168,169,171,
172,173,174,176,179,181,182,183,184,185,186,187,297,326,327,328,329,330,331,332,333,334,336,339,
340,343,344,345,346,347,349,350,351,352,353,444,483,489,490,491,492,493,494,495,496,499,500,501,
502,504,505,507,508,509,510,511,512,520,652,653,654,655,656,657,658,660,661,662,663,665,666,667,
668,670,671,673,674,675,676,678,815,816,817,818,819,820,821,822,823,825,826,828,829,832,837,839,
840,871,978,979,980,981,982,984,985,986,988,990,991,995,999,1001,1003,1004,1037,1038,1073,1075
```

Usando la semilla **0** nos da los siguientes resultados:

```
Ruta: [350, 840, 151, 828, 12, 1038, 340, 502, 339, 675, 186, 520, 16, 671, 179, 183, 346, 17,
```

```
164, 11, 501, 499, 347, 817, 23, 176, 668, 352, 978, 6, 5, 1004, 22, 185, 991, 333, 27, 353, 990,
171, 652, 1075, 483, 512, 821, 75, 444, 826, 511, 504, 825, 500, 985, 674, 999, 8, 662, 331, 837,
979, 493, 509, 329, 839, 2, 656, 667, 184, 507, 7, 823, 678, 816, 187, 982, 14, 181, 332, 345,
820, 26, 654, 653, 344, 490, 676, 665, 673, 173, 815, 1, 829, 832, 661, 663, 657, 508, 986, 9,
168, 505, 19, 496, 182, 172, 163, 351, 981, 3, 165, 988, 174, 4, 489, 25, 492, 491, 984, 995, 349,
1003, 334, 871, 343, 510, 660, 20, 327, 670, 336, 980, 297, 1001, 74, 822, 166, 658, 666, 818, 655,
819, 1073, 1037, 494, 495, 167, 328, 326, 169, 330]
Costo: 0.1668609785897421
¿Es Factible?: true
```

6. Comentarios