Optimización por Colonia de Hormigas para el Enrutamiento de Vehículos

Aldo Culquicondor

Computación Bioinspirada

Resumen

El problema de Enrutamiento de Vehículos es un problema clásico de Optimización Combinatoria donde se debe encontrar rutas óptimas para que varios *vehículos* transporten *paquetes* en una ciudad. Puede ser visto como una generalización del problema del Vendedor Viajero y es por tanto un problema $\mathcal{NP}$. Variantes del problema introducen paquetes de distinto valor y peso, vehículos de diferente capacidad, restricción de distancia total recorrida, entre otros. En este trabajo mostramos cómo resolver este problema usando la meta-heurística de Optimización por Colonia de Hormigas.

Consideremos los problemas de minimización para definir un modelo general de un problema de Optimización Combinatoria. Un modelo $P = (S, \Omega, f)$ de un problema de optimización combinatoria consiste de:

- un espacio de búsqueda S que es definido por un conjunto finito de variables de decisión, cada una con un dominio finito y un conjunto Ω de restricciones entre las variables;
- una función objetivo $f : S \to \mathbb{R}_0^+$ que debe ser minimizada.

1. Optimización por Colonia de Hormigas

La Optimización por Colonia de Hormigas (ACO - *Ant Colony Optimization*) es una meta-heurística de Inteligencia Colectiva. Las colonias de hormigas y sociedades de insectos, en general, son sistemas que, a pesar de la sencillez de sus individuos, presentan una organización social altamente estructurada y distribuida. Al observar la complejidad y perfección de algunas colonias y emjambres nos preguntamos si existe un gobierno en ellas. Aparentemente cada agente es autónomo, pero el resultado es algo organizado y complejo.

Un algoritmo de Optimización por Colonia de Hormigas puede verse como la interacción de tres procedimientos: construir soluciones, actualizar feromonas, búsqueda local. Estos algoritmos se organizan como se muestra en el Algoritmo 1.

Construir Soluciones Gestiona una colonia de hormigas que, al mismo tiempo y de forma asíncrona, cada una construye una solución del problema. Para

Algorithm 1 Algoritmo de la Optimización por Colonia de Hormigas

procedure METAHEURÍSTICAACO
  Initialización
  while condición de finalización no alcanzada **do**
    Construir Soluciones
    Búsqueda Local (opcional)
    Actualizar Feromonas
  end while
end procedure

ello, cada hormiga inicia con una solución vacía $s_p = \emptyset$. En cada paso, selecciona un componente de la solución factible $c_i^j \in \mathcal{N}(s_p) \subseteq C$ y es añadido a la solución parcial. $\mathcal{N}(s_p)$ es el conjunto de componentes que pueden ser añadidos manteniendo la factibilidad, estando en el estado s_p.

La selección de un componente es hecha de forma probabilística. La regla más usada es la del *Ant System* [1]:

$$p(c_i^j|s_p) = \frac{\tau_{ij}^{\alpha} \cdot [\eta(c_i^j)]^{\beta}}{\sum_{c_i^l \in \mathcal{N}(s_p)} \tau_{il}^{\alpha} \cdot [\eta(c_i^l)]^{\beta}} \quad \forall c_i^j \in \mathcal{N}(s_p), \tag{1}$$

donde $\eta(\cdot)$ es una función que asigna a cada componente $c_i^j \in \mathcal{N}(s_p)$ un valor heurístico. Los parámetros α y β determinan la relativa influencia de los rastros de feromona y la información heurística.

Actualizar Feromonas Es el proceso por el cual los rastros de feromona se modifican. El valor puede aumentar por el depósito de feromonas de hormigas en los componentes que utilizan, o disminuir, debido a la evaporación de feromona. La actualización de feromonas se implementa como sigue:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s), \tag{2}$$

donde S_{upd} es el conjunto de soluciones usadas para depositar feromona, $\rho \in (0, 1]$ es el parámetro llamado razón de evaporación, $g(\cdot) : S \to \mathbb{R}^+$ es una función tal que $f(s) < f(s') \Rightarrow g(s) \geq g(s')$. Determina la calidad de una solución y se llama función de evaluación. Al iniciar el algoritmo, $\tau ij = \tau_0$.

Búsqueda Local En inglés son llamadas *Daemon Actions*. Se utiliza para implementar acciones centralizadas que no se pueden realizar por las hormigas individualmente. Ejemplos de acciones *daemon* son la activación de un procedimiento de optimización local o el recojo de información global que se puede usar para decidir si es útil o no y depositar feromona adicional.

2. Problema de Enrutamiento de Vehículos

En el problema de Enrutamiento de Vehículos (VRP - *Vehicle Routing Problem*) el objetivo es encontrar rutas óptimas para un conjunto de vehículos a los que se asignan tareas en clientes distribuidos geográficamente. Una respuesta a

este problema es la mejor ruta que sirve a todos los clientes usando la flota de vehículos, respetando todas las restricciones operacionales como la capacidad del vehículo, la máxima distancia recorrida, entre otros, minimanzando el el costo total de transporte.

El problema se modela como un grafo donde los nodos son el centro de distribución y cada uno de los puntos de entrega. Para un mejor manejo de las feromonas, el nodo central se multiplicó, dándole un nodo propio a cada vehículo. El peso de las aristas es igual al costo de transporte de un nodo a otro. Se debe encontrar un conjunto de caminos en este grafo con inicio en el centro de distribución y que en conjunto recorran todos los nodos del grafo. La suma de los costos de los caminos debe ser el mínimo.

En este trabajo se han considerado las siguientes restricciones:

- un único centro de distribución, donde todos los caminos empiezan y terminan en él,
- cada cliente tiene un peso respectivo,
- cada vehículo tiene una capacidad distinta,
- un vehículo puede realizar cero o más salidas (caminos) y
- cada vehículo puede recorrer una distancia máxima total.

3. Modelamiento del VRP con ACO

Cada hormiga construye una solución completa del problema, es decir, una hormiga representa al primer vehículo y decide si realiza o no un recorrido. Tras terminar el recorrido, decide si vuelve a usar el mismo vehículo o utilizar el siguiente. El proceso se repite hasta que no queden vehículos. Es posible que una hormiga agote todos los vehículos sin haber visitado todos los nodos. Esta situación se penalizó en la función de evaluación g.

$$g(s) = \frac{\sqrt{peso\_total(s)} \cdot \left(1 + \frac{1}{longitud(s)}\right)}{N_h}, \tag{3}$$

donde N_h es el número de hormigas.

Además, se usaron los siguientes parámetros del algoritmo:

- una población de 20 hormigas,
- 2500 iteraciones,
- como feromona inicial τ_0 se estableció la evaluación de un camino que recorre todos los nodos, asumiendo que la distancia entre cada par de nodos de la solución es igual a la mitad de la distancia máxima,
- como heurística se utilizó la inversa del logaritmo de la distancia entre los nodos.
- se usó $\alpha = 0{,}5$, $\beta = 3$ y $\rho = 0{,}1$ y
- no se implementó un algoritmo de búsqueda local.

La implementación se encuentra disponible en GitHub (`https://github.com/alculquicondor/aco_vrp`)

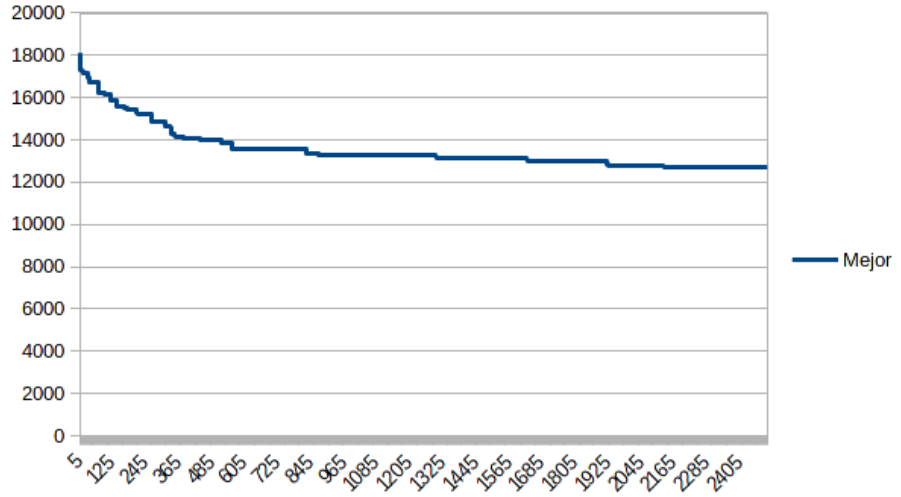

Figura 1: Curva de los Mejores

4. Resultados y Conclusiones

Se realizaron 10 ejecuciones y en cada iteración se registró la longitud de la mejor solución. La Figura 1 muestra el promedio de la mejor solución según el número de iteraciones. En promedio, tras las 2500 iteraciones, la distancia total es de 12727 ± 1069. La mejor solución obtenida tuvo una longitud de 10887 y se muestra en la Figura 2. Puede verse que la solución presenta cruces de caminos, lo cual podría mejorarse con un algoritmo de búsqueda local.

Se puede ver que la Optimización por Colonia de Hormigas va reduciendo el espacio de búsqueda enfocándose en soluciones similares a las que dieron mejores resultados en las iteraciones anteriores. Por su naturaleza, es muy sencillo modelar un problema de grafos con este algoritmo, mostrando resultados favorables.

Referencias

[1] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

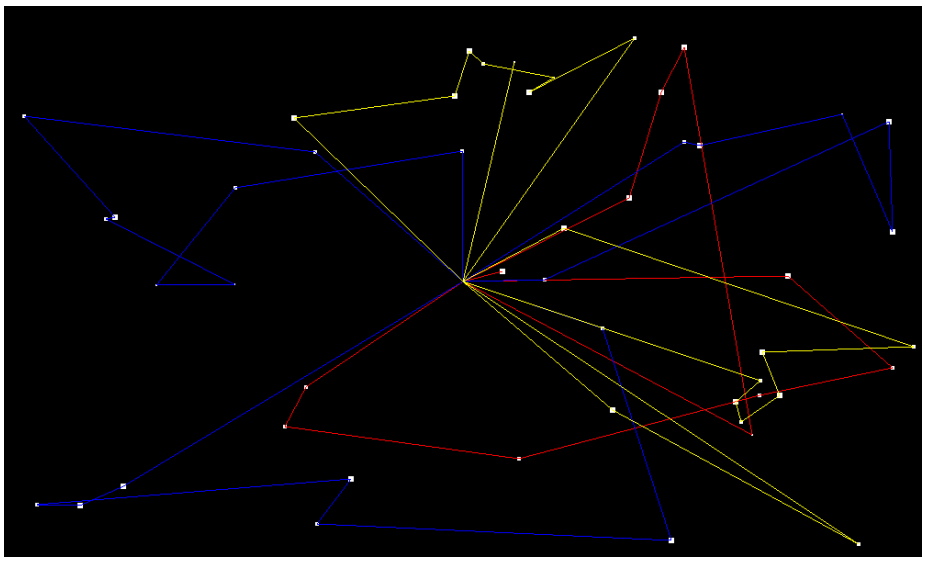

Figura 2: Mejor Solución. Cada color representa un tipo de algoritmo