Algoritmo de Colonia de Hormigas para el problema de Cobertura de Vértices de Peso Mínima

Juan C. Castrejón

Mayo 2019

Resumen

Las recientes meta-heurísticas inspiradas por fenómenos de la naturaleza, han mostrado un buen desempeño, a menudo superando los métodos clásicos.

En esta ocasión se usará el algoritmo de colonia de hormigas, donde las hormigas deambulan aleatoriamente mientras dejan caminos de feromonas a su paso, lo cual atrae a otras hormigas, reforzando así, las mejores rutas a su objetivo.

Este documento explica una adaptación del algoritmo de colonia de hormigas aplicado al problema de cobertura de vértices de peso mínima.

1 Introducción al problema

Dada una gráfica no dirigida $G = (V, E)$, la cobertura de vértices V' (fig. 1) es un subconjunto de V, tal que para toda arista:

$$(u, v) \in E \Rightarrow u \in V' \vee v \in V' \tag{1}$$

En otras palabras, que toda arista de la gráfica tiene al menos un vértice en la cobertura V'.

Una cobertura de vértices de peso mínima (fig. 2) de una gráfica no dirigida, con peso en los vértices $G = (V, E)$ es una cobertura de vértices V' tal que la suma de pesos de los vértices de V' es mínima.

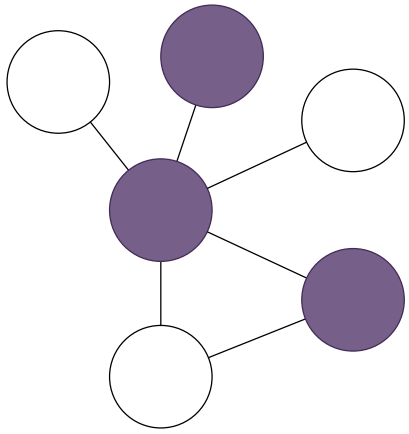

Figura 1: Cobertura de vértices

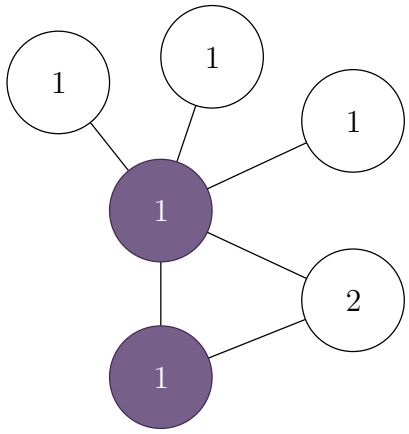

Figura 2: Cobertura de vértices de peso mínima

La cobertura de vértices de peso mínima (el acrónimo MWVC, por sus siglas en inglés, será usado a partir de ahora), puede ser usada para modelar muchas situaciones del mundo real en las areas de telecomunicaciones, diseño de circuitos, etc. MWVC es un problema NP-duro de optimización, por lo que no se conoce una solución en tiempo polinomial.

En este documento se usará una meta-heurística como una alternativa viable para encontrar soluciones *suficientemente buenas* en tiempo razonable, dada una instancia de MWVC.

2 Optimización por Colonia de Hormigas

En la naturaleza, algunas especies de hormigas, deambulan aleatoriamente y cuando encuentran comida, regresan a su colonia, dejando caminos de féromonas en su paso. Si otras hormigas encuentran estos caminos, es probable que dejen de moverse aleatoriamente y sigan el camino, reforzando las feromonas si encuentran comida eventualmente.

Sin embargo, los caminos de feromonas se evaporan con el tiempo, reduciendo así la atractividad para otras hormigas. Mientras más tiempo le tome a una hormiga recorrer un camino, más se evaporan las féromonas. Un camino corto, es atravesado más frecuentemente, lo que refuerza su atractividad, provocando que otras hormigas lo prefieran sobre otros caminos más largos.

La optimización por colonia de hormigas, a la cual se referirá ahora como ACO (Ant Colony Optimization) por sus siglas en inglés, es un algoritmo miembro de los métodos de inteligencia de enjambre, en el cual de manera probabilística, se desea encontrar caminos óptimos en una gráfica, usando el comportamiento natural basado en féromonas usado por las hormigas cuando buscan un camino desde una fuente de comida, hasta su colonia.

Cuando se usa ACO, el problema se transforma en una gráfica con pesos, después se distribuye un conjunto de hormigas en la gráfica para encontrar caminos que correspondan a soluciones potencialmente *buenas*.

La idea general del algoritmo se describe a continuación:

Procedimiento 1: ACO

1. Posicionar cada hormiga en un nodo inicial arbitrario.
2. Cada hormiga avanza a un nodo siguiente usando una regla de transición.
3. Aplicar la regla de actualización local de feromonas.
4. Repetir pasos 2-4 hasta que todas las hormigas completen una solución.
5. Aplicar la regla de actualización global de feromonas.
6. Repetir pasos 1-5 hasta que se cumpla la condición de terminación.

2.1 Representación de la gráfica

Sea $G = (V, E)$ con peso en los vértices, la gráfica que denota una instancia de MWVC; para garantizar que siempre exista un camino que cruce todos los vérticed de V, se construirá una gráfica completa $G_c = (V, E_c)$. La gráfica completa servirá para poder garantizar que una hormiga puede visitar cualquier vértice a partir de cualquier otro vértice. En la gráfica completa, las aristas tendrán peso 0 o 1, indicando si la arista ya ha sido cubierta por algún vértice; al crear la gráfica completa, el peso de las aristas que no pertenzcan a E, será 0; y 1 en el caso contrario.

2.2 Regla de transición

La regla de transición es una relación aleatoria que asigna la probabilidad de que un hormiga k que se encuentra actualmente en un vértice i, escoja el vértice j como su siguiente vértice para visitar. La siguiente ecuación muestra cómo se calcula ésta probablidad:

$$p_j^k = \begin{cases} 1, & \text{si } q < q_0 \text{ y } j = max(\tau_r \eta_{rk}) \text{ con } r \in A_k; \\ 0, & \text{si } q < q_0 \text{ y } j \neq max(\tau_r \eta_{rk}) \text{ con } r \in A_k; \\ \dfrac{\tau_j \eta_{rk}}{\sum_{r \in A_k} \tau_j \eta_{rk}} & \text{si } q \geq q_0; \end{cases} \tag{2}$$

donde A_k es el conjunto de vértices accesibles para la hormiga k, τ_j es el valor de feromona actual en el vértice j y η_{rk} representa el número de aristas no cubiertas por la hormiga k que están conectadas al vértice j, dividido entre el peso del vértice j; básicamente es una heurística local para favorecer al vértice que cubra la mayor cantidad de aristas y que tenga menor peso.

2.3 Actualización local de feromonas

Cada vez que una hormiga selecciona un vértice i como su siguiente vértice, se realiza la actualización local de feromonas, la cual ajusta la feromona actual de un vértice disminuyendo la deseabilidad frente a otras hormigas, con el fin de dirigir la exploración de las demás hormigas a otros vértices que aún no hayan sido visitados. La feromona en el vértice i se actualiza localmente usando la siguiente regla:

$$\tau_i = (1 - \varphi)\tau_i + \varphi\tau_0 \tag{3}$$

donde φ es un parametro que indica el *grado de evaporación* de las feromonas, esto es, que tan rápido decrecen las fermonas de un vértice con cada actualización.

2.4 Actualización global de feromonas

Después de que todas las hormigas hayan completado una solución (ya no quedan aristas por cubrir), se ejecuta la actualización global de feromonas, la cual aumenta la intensidad de las feromonas en los vértices que pertenecen a la mejor solución conocida actualmente y reduce el resto, con el fin de que en la siguiente iteración del algoritmo se favorezca a los vértices que pertenecen a la siguiente solución y por consecuente, se generen soluciones similares a ésta y se olviden las malas soluciones generadas en iteraciones anteriores. Sea $V'_c \subseteq V$ la mejor solución actualmente conocida; la actualización global de feromonas se realiza para cada vértice en la gráfica, usando la siguiente regla:

$$\tau_i = \begin{cases} (1 - \varphi)\tau_i + \varphi\Delta\tau_i, & \text{si } i \in V'_c; \\ (1 - \varphi)\tau_i, & \text{en otro caso;} \end{cases} \tag{4}$$

donde

$$\Delta\tau_i = \frac{1}{\sum_{j \in V'_c} w(j)} \tag{5}$$

y $w(v)$ es la función que regresa el peso del vértice v.

3 Función de costo

Dada un conjunto solución $V' \subseteq V$ la función de costo que permite medir la utilidad de V' se usa como sigue:

$$f(V') = \sum_{v \in V'} w(v) \tag{6}$$

como se puede observar, la función de costo sólo toma en cuenta la suma de los pesos de los vértices en la solución actual, ya que la heurística local

en la regla de transición ya favorece la obtención de soluciones con menos elementos.

Sean $V_1', V_2' \subseteq V$ dos cubiertas de la gráfica, decimos que V_1' es mejor solución que V_2' si $f(V_1') < f(V_2')$.

4 Resultados

A continuación se muestran los resultados obtenidos para una gráfica con $|V| = 15$ y $|E| = 24$, con pesos iguales en todos los vértices, construida de tal forma que el algoritmo *greedy*, descrito en el procedimiento 2, no obtiene la solución óptima para el problema.

En la figura 3 y 4 se puede observar que ACO fue capaz de mejorar efectivamente la solución obtenida por el algoritmo *greedy*. ACO muestra un desempeño similar en gráficas mucho más grandes, donde el espacio de búsqueda mayor, también favorece la obtención de más soluciones distintas a la del algoritmo *greedy*.

Los parámetros usados en ACO se establecieron como: $k = 10$, $q_0 = 0.9$, $\varphi = 0.9$, $INTENTOS = 150$, $\tau_0 = |V|(|V| - a)/C$, donde k es el número de hormigas, C es el costo total de una soluciñon incial aproximaday a es el número de vértices de la solución inicial.

Procedimiento 2: Algoritmo greedy

1. Para cada vértice, tomar el vértice v con mayor proporción de aristas sin cubrir/peso del vértice.
2. Para cada arista de v indicar que esa arista ya está cubierta en la solución.
3. Agregar v a la solución.
4. Repetir los pasos 1-4 hasta que no queden aristas por cubrir en la gráfica.

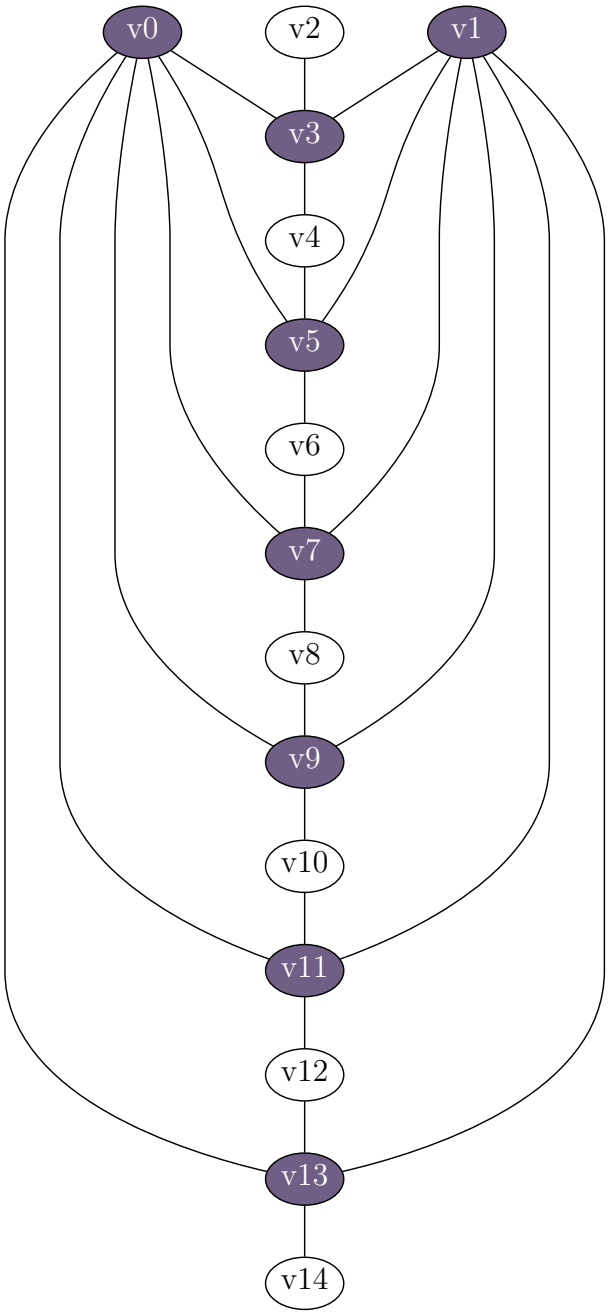

Figura 3: Solución obtenida por el algoritmo greedy.

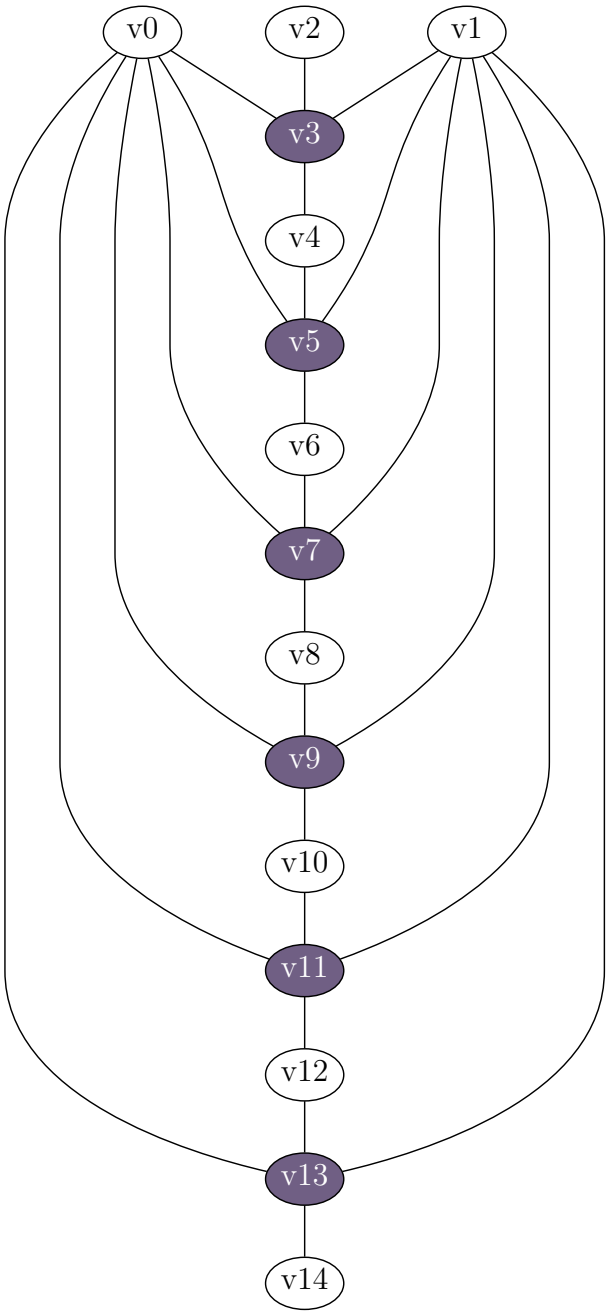

Figura 4: Solución obtenida por ACO.

Referencias

[1] M. Dorigo (1992). "Optimization, Learning and Natural Algorithm", *PhD thesis*, Politecnico di Milano, Italy.

[2] Shyu, S.J., P.Y. Yin, B.M.T. Lin (2004). "An Ant Colony Optimization Algorithm for the Minimum Weight Vertex Cover Problem", *Annals of Operations Research*, 131, 283–304.