Análisis de Algoritmos: A*, Floyd-Warshall y Prim - Matemática Discreta 2

Jonnathan Juárez 15377, Rodrigo Barrios 15009, Juan Andrés García 15046 - Universidad del Valle de Guatemala - 17.11.2016

Descripción del proyecto

El presente proyecto se centró en la solución de dos problemas distintos: encontrar la ruta más corta y el árbol de expansión mínima. Dado un grafo ponderado $G = (V, E, \omega)$, $\omega : E \mapsto \mathbb{R}$ se dice que la ruta más corta entre A y B $(A, B \in V)$ es el recorrido de aristas de E que conectan a A y B con el menor peso posible. Para la solución de este probelma se diseñaron 2 algoritmos: A* y Floyd-Warshall con el objetivo de compararlos en distintas situaciones sobre una rejilla de 25x25 y determinar cuál es más eficiente.
Tanto los algoritmos de A* como de Floyd-Warshall se emplearon para encontrar la ruta más corta en los siguientes laberintos:

- Se permitieron únicamente movimientos verticales y horizontales en la rejilla.
- Se trabajaron dos funciones de peso distintas para los movimientos horizontales $(H(m, n))$ y verticales $(V(m, n))$:
  - $H(m, n) = 1$
  - $V(m, n) = m + n$
  - $H(m, n) = m + n \mod 17$
  - $V(m, n) = m + n \mod 13$
- Se permitieron movimientos diagonales, con un peso de $\sqrt{2}$ y los movimientos verticales y horizontales con un peso de 1.
- Se colocaron obstáculos internos, sitios en la rejilla que el algoritmo no tiene permitido tocar.

El árbol de expansión mínima es aquel árbol compuesto por todos los vértices y aristas de G cuya la suma de sus aristas es la de menor peso. El algoritmo diseñado para este fue el de Prim, que luego de generado el árbol, fue resuelto con el algoritmo A* con la diferencia que se permitió el ingreso de coordenadas de origen y de destino.

Algoritmos utilizados

Algoritmo de Floyd-Warshall: recibe como entrada una matriz de distancias D_0, si hay una arista entre los vértices i y j, entonces la matriz D_0 contiene su longitud en las entradas correspondientes, caso contrario, se dice que la distancia entre ellos es infinito. En cada iteración la matriz es recalculada para que contenga las distancias entre cada par de vértices utilizando un conjunto de nodos intermedios, esta transformación puede ser descrita por:

$$D_{ij}^{n} = min(D_{ij}^{n-1}, D_{ik}^{n-1}, D_{kj}^{n-1})$$

Que resulta en las distancias más cortas entre todos los vértices de G y el camino que se debe tomar para llegar a cada uno de estos (Jungnickel, 2000; Kolár, 2004).

Algoritmo A:* A diferencia del algoritmo de Floyd-Warshall u otros, A* recibe como entrada un grafo $G = (V, E, \omega)$, $\omega : E \mapsto \mathbb{R}$ y devuelve también ese mismo grafo con la ruta más corta indicada. Sus ventajas yacen en estar optimizado en calcular rutas centradas en un solo destino pagando el costo en el diseño de una rejilla (*grid*) que sea soportada por el algoritmo. La esencia de A* es la definición de una función de heurística, que además de determinar la ruta óptima reordena los nodos de la rejilla para llegar a esta más rápido. Una función de heurística usualmente se define como:

$$f(v) = g(v) + h(v)$$

Donde $g(v)$ es la función de un algoritmo "convencional" (e.g. Dijkstra, Floyd) y $h(v)$ es la heurística, definida en casos donde se pueden tomar 4 direcciones como distancia Manhattan y en casos donde se pueden tomar 6 direcciones como distancia euclidiana.
La distancia Manhattan se define por:

$$d(\mathbf{p},\mathbf{q}) = ||\mathbf{p}\text{-}\mathbf{q}||$$

Donde $\mathbf{p}$ y $\mathbf{q}$ son vectores.
Y la distancia euclidiana está definida con:

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

(Patil, 2016).

Algoritmo de Prim: El algoritmo se centra en la simplificación de un árbol T al de expansión mínima, pero puede ser adaptado fácilmente para generar laberintos. La versión en pseudocódigo del algoritmo es como sigue:

Paso 1. Escoja un vértice arbitrario de G y agréguelo a un conjunto (inicialmente vacío) T.

Paso 2. Escoja un vértice al azar en G siempre y cuando se conecte con la frontera del laberinto (conjunto de vértices que se van desde el laberinto hacia afuera).

Paso 3. Agregar el vértice al árbol de expansión mínima (T).

Paso 4. Repetir pasos 2 y 3 hasta tener todos los vértices de G.

(Buck, 2011).

Discusión de resultados
Conclusiones
Referencias

- Jungickel, Dieter. 2000. *Graphs, Networks and Algorithms*. 2da. Ed. Springer. Augsburgo, Alemania.
- Kolár, Josef. 2004. *Floyd-Warshall Algorithm*. Web en línea. Disponible en: `http://www.programming-algorithms.net/article/45708/Floyd-Warshall-algorithm`. [Último acceso, 16 de noviembre de 2016].
- Patel, Amit. 2016. *Introduction to A* and implementation guide*. Web en línea. Disponible en: `http://www.redblobgames.com/pathfinding/a-star/introduction.html`. [Úlitmo acceso, 16 de noviembre de 2016].
- Buck, Jamis. 2011. *Maze generation: Prim's Algorithm*. Web en línea. Disponible en: `http://weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm`. [Último acceso, 16 de noviembre de 2016].