Edmonds' algorithm

From Wikipedia, the free encyclopedia
(Redirected from Edmonds's algorithm)

This article is about the optimum branching algorithm. For the maximum matching algorithm, see Blossom algorithm.

In graph theory, a branch of mathematics, **Edmonds' algorithm** or **Chu–Liu/Edmonds' algorithm** is an algorithm for finding a spanning arborescence of minimum weight (sometimes called an *optimum branching*). It is the directed analog of the minimum spanning tree problem. The algorithm was proposed independently first by Yoeng-jin Chu and Tseng-hong Liu (1965) and then by Jack Edmonds (1967).

Graph and tree search algorithms
α–β · A* · B* · Backtracking · Beam · Bellman–Ford · Best-first · Bidirectional · Borůvka · Branch & bound · BFS · British Museum · D* · DFS · Depth-limited · Dijkstra · **Edmonds** · Floyd–Warshall · Fringe search · Hill climbing · IDA* · Iterative deepening · Johnson · Jump point · Kruskal · Lexicographic BFS · Prim · SMA*
Listings
Graph algorithms · *Search algorithms* · *List of graph algorithms*
Related topics
Dynamic programming · Graph traversal · Tree traversal · Search games
v · t · e

Algorithm [edit]

Description [edit]

The algorithm takes as input a directed graph $D = \langle V, E \rangle$ where V is the set of nodes and E is the set of directed edges, a distinguished vertex $r \in V$ called the *root*, and a real-valued weight $w(e)$ for each edge $e \in E$. It returns a spanning arborescence A rooted at r of minimum weight, where the weight of an arborescence is defined to be the sum of its edge weights, $w(A) = \sum_{e \in A} w(e)$.

The algorithm has a recursive description. Let $f(D, r, w)$ denote the function which returns a spanning arborescence rooted at r of minimum weight. We first remove any edge from E whose destination is r. We may also replace any set of parallel edges (edges between the same pair of vertices in the same direction) by a single edge with weight equal to the minimum of the weights of these parallel edges.

Now, for each node v other than the root, find the edge incoming to v of lowest weight (with ties broken arbitrarily). Denote the source of this edge by $\pi(v)$. If the set of edges $P = \{(\pi(v), v) \mid v \in V \setminus \{r\}\}$ does not contain any cycles, then $f(D, r, w) = P$.

Otherwise, P contains at least one cycle. Arbitrarily choose one of these cycles and call it C. We now define a new weighted directed graph $D' = \langle V', E' \rangle$ in which the cycle C is "contracted" into one node as follows:

The nodes of V' are the nodes of V not in C plus a *new* node denoted v_C.

If (u, v) is an edge in E with $u \notin C$ and $v \in C$, then include in E' a new edge $e = (u, v_C)$, and define $w'(e) = w(u, v) - w(\pi(v), v)$.

If (u, v) is an edge in E with $u \in C$ and $v \notin C$, then include in E' a new edge $e = (v_C, v)$, and define $w'(e) = w(u, v)$.

If (u, v) is an edge in E with $u \notin C$ and $v \notin C$, then include in E' a new edge $e = (u, v)$, and define $w'(e) = w(u, v)$.

For each edge in E', we remember which edge in E it corresponds to.

Now find a minimum spanning arborescence A' of D' using a call to $f(D', r, w')$. Since A' is a spanning arborescence, each vertex has exactly one incoming edge. Let (u, v_C) be the unique incoming edge to v_C in A'. This edge corresponds to an edge $(u, v) \in E$ with $v \in C$. Remove the edge $(\pi(v), v)$ from C.

Wikipedia
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
فارسی
Italiano
Српски / srpski
ไทย
Edit links

breaking the cycle. Mark each remaining edge in C. For each edge in A', mark its corresponding edge in E. Now we define $f(D,r,w)$ to be the set of marked edges, which form a minimum spanning arborescence.

Observe that $f(D,r,w)$ is defined in terms of $f(D',r,w')$, with D' having strictly fewer vertices than D. Finding $f(D,r,w)$ for a single-vertex graph is trivial (it is just D itself), so the recursive algorithm is guaranteed to terminate.

Running time [edit]

The running time of this algorithm is $O(EV)$. A faster implementation of the algorithm due to Robert Tarjan runs in time $O(E \log V)$ for sparse graphs and $O(V^2)$ for dense graphs. This is as fast as Prim's algorithm for an undirected minimum spanning tree. In 1986, Gabow, Galil, Spencer, and Tarjan produced a faster implementation, with running time $O(E + V \log V)$.

References [edit]

- Chu, Y. J.; Liu, T. H. (1965), "On the Shortest Arborescence of a Directed Graph", *Science Sinica* **14**: 1396–1400
- Edmonds, J. (1967), "Optimum Branchings", *J. Res. Nat. Bur. Standards* **71B**: 233–240, doi:10.6028/jres.071b.032
- Tarjan, R. E. (1977), "Finding Optimum Branchings", *Networks* **7**: 25–35, doi:10.1002/net.3230070103
- Camerini, P.M.; Fratta, L.; Maffioli, F. (1979), "A note on finding optimum branchings", *Networks* **9**: 309–312, doi:10.1002/net.3230090403
- Gibbons, Alan (1985), *Algorithmic Graph Theory*, Cambridge University press, ISBN 0-521-28881-9
- Gabow, H. N.; Galil, Z.; Spencer, T.; Tarjan, R. E. (1986), "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs", *Combinatorica* **6**: 109–122, doi:10.1007/bf02579168

External links [edit]

- Edmonds's algorithm (edmonds-alg) – An open source implementation of Edmonds's algorithm written in C++ and licensed under the MIT License. This source is using Tarjan's implementation for the dense graph.

Categories: Graph algorithms

This page was last modified on 3 January 2015, at 16:00.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Mobile view