Edmonds' Algorithm

# Edmonds' Algorithm

In graph theory (http://www.popflock.com/learn?s=Graph_theory), **Edmonds' algorithm** or **Chu-Liu/Edmonds' algorithm** is an algorithm (http://www.popflock.com/learn?s=Algorithm) for finding a spanning (http://www.popflock.com/learn?s=Spanning_subgraph) arborescence (http://www.popflock.com/learn?s=Arborescence_(graph_theory)) of minimum weight (sometimes called an *optimum branching*). It is the directed (http://www.popflock.com/learn?s=Directed_graph) analog of the minimum spanning tree (http://www.popflock.com/learn?s=Minimum_spanning_tree) problem. The algorithm was proposed independently first by Yoeng-Jin Chu and Tseng-Hong Liu (1965) and then by Jack Edmonds (http://www.popflock.com/learn?s=Jack_Edmonds) (1967).

## Algorithm

### Description

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 (http://www.popflock.com/learn?s=Arborescence_(graph_theory)) $A$ 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 $v \in C$ (an edge coming into the cycle), 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 $v \notin C$ (an edge going away from the cycle), 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 $v \notin C$ (an edge unrelated to the cycle), 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 $C$, 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

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

## References

- 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 (http://www.popflock.com/learn?s=Digital_object_identifier):10.6028/jres.071b.032

(//doi.org/10.6028%2Fjres.071b.032)

- Tarjan, R. E. (http://www.popflock.com/learn?s=Robert_Tarjan) (1977), "Finding Optimum Branchings", *Networks*, **7**: 25-35, doi (http://www.popflock.com/learn?s=Digital_object_identifier):10.1002/net.3230070103 (//doi.org/10.1002%2Fnet.3230070103)
- Camerini, P.M.; Fratta, L.; Maffioli, F. (1979), "A note on finding optimum branchings", *Networks*, **9**: 309-312, doi (http://www.popflock.com/learn?s=Digital_object_identifier):10.1002/net.3230090403 (//doi.org/10.1002%2Fnet.3230090403)
- Gibbons, Alan (1985), *Algorithmic Graph Theory*, Cambridge University press, ISBN (http://www.popflock.com/learn?s=International_Standard_Book_Number) 0-521-28881-9 (http://www.popflock.com/learn?s=Special:BookSources/0-521-28881-9)
- Gabow, H. N.; Galil, Z.; Spencer, T.; Tarjan, R. E. (http://www.popflock.com/learn?s=Robert_Tarjan) (1986), "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs", *Combinatorica*, **6**: 109-122, doi (http://www.popflock.com/learn?s=Digital_object_identifier):10.1007/bf02579168 (//doi.org/10.1007%2Fbf02579168)

## External links

- Edmonds's algorithm ( edmonds-alg ) (http://edmonds-alg.sourceforge.net/) - An open source (http://www.popflock.com/learn?s=Open_source) implementation of Edmonds's algorithm written in C++ (http://www.popflock.com/learn?s=C%2B%2B) and licensed under the MIT License (http://www.popflock.com/learn?s=MIT_License). This source is using Tarjan's implementation for the dense graph.