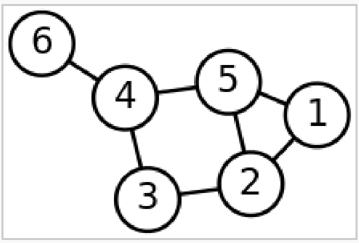
Shortest path problem

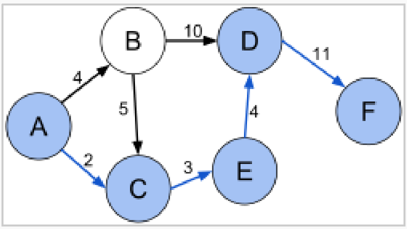
From Wikipedia, the free encyclopedia

In graph theory, the **shortest path problem** is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.

The problem of finding the shortest path between two intersections on a road map (the graph's vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of its road segment) may be modeled by a special case of the shortest path problem in graphs.



(6, 4, 5, 1) and (6, 4, 3, 2, 1) are both paths between vertices 6 and 1



Shortest path (A, C, E, D, F) between vertices A and F in the weighted directed graph

Contents

Definition [edit]

The shortest path problem can be defined for graphs whether undirected, directed, or mixed. It is defined here for undirected graphs; for directed graphs the definition of path requires that consecutive vertices be connected by an appropriate directed edge.

Two vertices are adjacent when they are both incident to a common edge. A path in an undirected graph is a sequence of vertices $P = (v_1, v_2, \ldots, v_n) \in V \times V \times \ldots \times V$ such that v_i is adjacent to v_{i+1} for $1 \leq i < n$. Such a path P is called a path of length n from v_1 to v_n. (The v_i are variables; their numbering here relates to their position in the sequence and needs not to relate to any canonical labeling of the vertices.)

Let $e_{i,j}$ be the edge incident to both v_i and v_j. Given a real-valued weight function $f : E \rightarrow \mathbb{R}$, and an undirected (simple) graph G, the shortest path from v to v' is the path $P = (v_1, v_2, \ldots, v_n)$ (where $v_1 = v$ and $v_n = v'$) that over all possible n minimizes the sum $\sum_{i=1}^{n-1} f(e_{i,i+1})$. When each edge in the graph has unit weight or $f : E \rightarrow \{1\}$, this is equivalent to finding the path with fewest edges.

The problem is also sometimes called the **single-pair shortest path problem**, to distinguish it from the following variations:

- The **single-source shortest path problem**, in which we have to find shortest paths from a source vertex *v* to all other vertices in the graph.
- The **single-destination shortest path problem**, in which we have to find shortest paths from all vertices in the directed graph to a single destination vertex *v*. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.
- The **all-pairs shortest path problem**, in which we have to find shortest paths between every pair of vertices *v*, *v'* in the graph.

These generalizations have significantly more efficient algorithms than the simplistic approach of running a single-pair shortest path algorithm on all relevant pairs of vertices.

Algorithms [edit]

The most important algorithms for solving this problem are:

- Dijkstra's algorithm solves the single-source shortest path problem.
- Bellman–Ford algorithm solves the single-source problem if edge weights may be negative.
- A* search algorithm solves for single pair shortest path using heuristics to try to speed up the search.
- Floyd–Warshall algorithm solves all pairs shortest paths.
- Johnson's algorithm solves all pairs shortest paths, and may be faster than Floyd–Warshall on sparse graphs.
- Viterbi algorithm solves the shortest stochastic path problem with an additional probabilistic weight on each node.

Additional algorithms and associated evaluations may be found in Cherkassky et al.[1]

Single-source shortest paths [edit]

Undirected graphs [edit]

Weights	Time complexity	Author
I$_+$	$O(E + V \log V)$	Dijkstra 1959
I	$O(E)$	Thorup[2] (requires constant-time multiplication).

Unweighted graphs [edit]

Algorithm	Time complexity	Author
Breadth-first search	$O(E + V)$	

Directed acyclic graphs [edit]

An algorithm using topological sorting can solve the single-source shortest path problem in linear time, $\Theta(E + V)$, in weighted DAGs.

Directed graphs with nonnegative weights [edit]

The following table is taken from Schrijver (2004).[3] A green background indicates an asymptotically best bound in the table.

Algorithm	Time complexity	Author
	$O(V^2EL)$	Ford 1956
Bellman–Ford algorithm	$O(VE)$	Bellman 1958, Moore 1959
	$O(V^2 \log V)$	Dantzig 1958, Dantzig 1960, Minty (cf. Pollack & Wiebenson 1960), Whiting & Hillier 1960
Dijkstra's algorithm with list	$O(V^2)$	Leyzorek et al. 1957, Dijkstra 1959
Dijkstra's algorithm with modified binary heap	$O((E + V) \log V)$	

. . .	. . .	. . .
Dijkstra's algorithm with Fibonacci heap	$O(E + V \log V)$	Fredman & Tarjan 1984, Fredman & Tarjan 1987
	$O(E \log \log L)$	Johnson 1982, Karlsson & Poblete 1983
Gabow's algorithm	$O(E \log_{E/V} L)$	Gabow 1983b, Gabow 1985b
	$O(E + V\sqrt{\log L})$	Ahuja et al. 1990

This list is incomplete; you can help by expanding it.

Planar directed graphs with nonnegative weights [edit]

Directed graphs with arbitrary weights [edit]

Algorithm	Time complexity	Author
Bellman–Ford algorithm	$O(VE)$	Bellman 1958, Moore 1959

This list is incomplete; you can help by expanding it.

Planar directed graphs with arbitrary weights [edit]

All-pairs shortest paths [edit]

The all-pairs shortest path problem finds the shortest paths between every pair of vertices v, v' in the graph. The all-pairs shortest paths problem for unweighted directed graphs was introduced by Shimbel (1953), who observed that it could be solved by a linear number of matrix multiplications that takes a total time of $O(V^4)$.

Undirected graph [edit]

Weights	Time complexity	Algorithm
$\mathbf{I}_+$	$O(V^3)$	Floyd-Warshall algorithm
$\mathbf{I}_+$	$O(EV \log \alpha(E,V))$	Pettie-Ramachandran[4]
I	$O(EV)$	Thorup[2] (requires constant-time multiplication).

Directed graph [edit]

Weights	Time complexity	Algorithm
I (no negative cycles)	$O(V^3)$	Floyd-Warshall algorithm
I (no negative cycles)	$O(EV + V^2 \log V)$	Johnson-Dijkstra
I (no negative cycles)	$O(EV + V^2 \log \log V)$	Johnson-Pettie[5]
I	$O(EV + V^2 \log \log V)$	Hagerup[6]

Applications [edit]

Shortest path algorithms are applied to automatically find directions between physical locations, such as driving directions on web mapping websites like MapQuest or Google Maps. For this application fast specialized algorithms are available.[7]

If one represents a nondeterministic abstract machine as a graph where vertices describe states and edges describe possible transitions, shortest path algorithms can be used to find an optimal sequence of choices to reach a certain goal state, or to establish lower bounds on the time needed to reach a given state. For example, if vertices represent the states of a puzzle like a Rubik's Cube and each directed edge corresponds to a single move or turn, shortest path algorithms can be used to find a solution that uses the minimum possible number of moves.

In a networking or telecommunications mindset, this shortest path problem is sometimes called the min-delay path problem and usually tied with a widest path problem. For example, the algorithm may seek the shortest (min-delay) widest path, or widest shortest (min-delay) path.

A more lighthearted application is the games of "six degrees of separation" that try to find the shortest path in graphs like movie stars appearing in the same film.

Other applications, often studied in operations research, include plant and facility layout, robotics, transportation, and VLSI design".[8]

Road networks [edit]

A road network can be considered as a graph with positive weights. The nodes represent road junctions and each edge of the graph is associated with a road segment between two junctions. The weight of an edge may correspond to the length of the associated road segment, the time needed to traverse the segment or the cost of traversing the segment. Using directed edges it is also possible to model one-way streets. Such graphs are special in the sense that some edges are more important than others for long distance travel (e.g. highways). This property has been formalized using the notion of highway dimension.[9] There are a great number of algorithms that exploit this property and are therefore able to compute the shortest path a lot quicker than would be possible on general graphs.

All of these algorithms work in two phases. In the first phase, the graph is preprocessed without knowing the source or target node. The second phase is the query phase. In this phase, source and target node are known.The idea is that the road network is static, so the preprocessing phase can be done once and used for a large number of queries on the same road network.

The algorithm with the fastest known query time is called hub labeling and is able to compute shortest path on the road networks of Europe or the USA in a fraction of a microsecond.[10] Other techniques that have been used are:

- ALT
- Arc Flags
- Contraction hierarchies
- Transit Node Routing
- Reach based Pruning
- Labeling

Related problems [edit]

For shortest path problems in computational geometry, see Euclidean shortest path.

The travelling salesman problem is the problem of finding the shortest path that goes through every vertex exactly once, and returns to the start. Unlike the shortest path problem, which can be solved in polynomial time in graphs without negative cycles, the travelling salesman problem is NP-complete and, as such, is believed not to be efficiently solvable for large sets of data (see P = NP problem). The problem of finding the longest path in a graph is also NP-complete.

The Canadian traveller problem and the stochastic shortest path problem are generalizations where either the graph isn't completely known to the mover, changes over time, or where actions (traversals) are probabilistic.

The shortest multiple disconnected path [11] is a representation of the primitive path network within the framework of Reptation theory.

The widest path problem seeks a path so that the minimum label of any edge is as large as possible.

Linear programming formulation [edit]

There is a natural linear programming formulation for the shortest path problem, given below. It is very simple compared to most other uses of linear programs in discrete optimization, however it illustrates connections to other concepts.

Given a directed graph (V, A) with source node s, target node t, and cost w_{ij} for each edge (i, j) in A, consider the program with variables x_{ij}

$$\text{minimize } \sum_{ij\in A} w_{ij}x_{ij} \text{ subject to } x\geq 0 \text{ and for all } i, \sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & \text{if } i=s; \\ -1, & \text{if } i=t; \\ 0, & \text{otherwise.} \end{cases}$$

The intuition behind this is that x_{ij} is an indicator variable for whether edge (i, j) is part of the shortest path: 1 when it is, and 0 if it is not. We wish to select the set of edges with minimal weight, subject to the constraint that this set forms a path from s to t (represented by the equality constraint: for all vertices except s and t the number of incoming and outcoming edges that are part of the path must be the same (i.e., that it should be a path from s to t).

This LP has the special property that it is integral; more specifically, every basic optimal solution (when one exists) has all variables equal to 0 or 1, and the set of edges whose variables equal 1 form an *s*-*t* dipath. See Ahuja et al.[12] for one proof, although the origin of this approach dates back to mid-20th century.

The dual for this linear program is

maximize $y_t - y_s$ subject to for all ij, $y_j - y_i \le w_{ij}$

and feasible duals correspond to the concept of a consistent heuristic for the A* algorithm for shortest paths. For any feasible dual *y* the reduced costs $w'_{ij} = w_{ij} - y_j + y_i$ are nonnegative and A* essentially runs Dijkstra's algorithm on these reduced costs.

General algebraic framework on semirings: the algebraic path problem [edit]

This section requires expansion.
(August 2014)

Many problems can be framed as a form of the shortest path for some suitably substituted notions of addition along a path and taking the minimum. The general approach to these is to consider the two operations to be those of a semiring. Semiring multiplication is done along the path, and the addition is between paths. This general framework is known as the algebraic path problem.[13][14][15]

Most of the classic shortest-path algorithms (and new ones) can be formulated as solving linear systems over such algebraic structures.[16]

More recently, an even more general framework for solving these (and much less obviously related problems) has been developed under the banner of valuation algebras.[17]

See also [edit]

- Pathfinding
- IEEE 802.1aq
- Flow network
- Shortest path tree
- Euclidean shortest path
- Min-plus matrix multiplication
- Bidirectional search, an algorithm that finds the shortest path between two vertices on a directed graph

References [edit]

1. ^ Cherkassky, Boris V.; Goldberg, Andrew V.; Radzik, Tomasz (1996). "Shortest paths algorithms: theory and experimental evaluation". *Mathematical Programming*. Ser. A **73** (2): 129–174. doi:10.1016/0025-5610(95)00021-6. MR 1392160.
2. ^ *a* *b* Thorup, Mikkel (1999). "Undirected single-source shortest paths with positive integer weights in linear time". *Journal of the ACM (JACM)* **46** (3): 362–394. Retrieved 28 November 2014.
3. ^ Schrijver, Alexander (2004). *Combinatorial Optimization — Polyhedra and Efficiency*. Algorithms and Combinatorics **24**. Springer. ISBN 3-540-20456-3. Here: vol.A, sect.7.5b, p.103
4. ^ *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*. 2002. pp.267–276
5. ^ *Theoretical Computer Science* **312**. 2004. pp.47–74
6. ^ *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*. 2000. pp.61–72
7. ^ Sanders, Peter (March 23, 2009). "Fast route planning". Google Tech Talk.
8. ^ Chen, Danny Z. (December 1996). "Developing algorithms and software for geometric path planning problems". *ACM Computing Surveys* **28** (4es): 18. doi:10.1145/242224.242246.
9. ^ Abraham, Ittai; Fiat, Amos; Goldberg, Andrew V.; Werneck, Renato F. "Highway Dimension, Shortest Paths, and Provably Efficient Algorithms". ACM-SIAM Symposium on Discrete Algorithms, pages 782-793, 2010.
10. ^ Abraham, Ittai; Delling, Daniel; Goldberg, Andrew V.; Werneck, Renato F. research.microsoft.com/pubs/142356/HL-TR.pdf "A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks". Symposium on Experimental Algorithms, pages 230-241, 2011.
11. ^ Kroger, Martin (2005). "Shortest multiple disconnected path for the analysis of entanglements in two- and three-dimensional polymeric systems". *Computer Physics Communications* **168** (168): 209–232. doi:10.1016/j.cpc.2005.01.020.
12. ^ Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall. ISBN 0-13-617549-X.
13. ^ John Baras; George Theodorakopoulos (4 April 2010). *Path Problems in Networks*. Morgan & Claypool Publishers. pp. 9–. ISBN 978-1-59829-924-3.
14. ^ Mehryar Mohri, "Semiring frameworks and algorithms for shortest-distance problems", Journal of Automata, Languages and Combinatorics, Volume 7 Issue 3, January 2002, Pages 321 - 350

[illegible]

15. ^ http://www.iam.unibe.ch/~run/talks/2008-06-05-Bern-Jonczy.pdf
16. ^ Michel Gondran; Michel Minoux (2008). *Graphs, Dioids and Semirings: New Models and Algorithms*. Springer Science & Business Media. chapter 4. ISBN 978-0-387-75450-5.
17. ^ Marc Pouly; Jürg Kohlas (2011). *Generic Inference: A Unifying Theory for Automated Reasoning*. John Wiley & Sons. Chapter 6. Valuation Algebras for Path Problems. ISBN 978-1-118-01086-0.

- Bellman, Richard (1958). "On a routing problem". *Quarterly of Applied Mathematics* **16**: 87–90. MR 0102435.
- Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2001) [1990]. "Single-Source Shortest Paths and All-Pairs Shortest Paths". *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill. pp. 580–642. ISBN 0-262-03293-7.
- Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs" (PDF). *Numerische Mathematik* **1**: 269–271. doi:10.1007/BF01386390.
- Fredman, Michael Lawrence; Tarjan, Robert E. (1984). *Fibonacci heaps and their uses in improved network optimization algorithms*. 25th Annual Symposium on Foundations of Computer Science. IEEE. pp. 338–346. doi:10.1109/SFCS.1984.715934. ISBN 0-8186-0591-X.
- Fredman, Michael Lawrence; Tarjan, Robert E. (1987). "Fibonacci heaps and their uses in improved network optimization algorithms". *Journal of the Association for Computing Machinery* **34** (3): 596–615. doi:10.1145/28869.28874.
- Leyzorek, M.; Gray, R. S.; Johnson, A. A.; Ladew, W. C.; Meaker, S. R., Jr.; Petry, R. M.; Seitz, R. N. (1957). *Investigation of Model Techniques — First Annual Report — 6 June 1956 — 1 July 1957 — A Study of Model Techniques for Communication Systems*. Cleveland, Ohio: Case Institute of Technology.
- Moore, E. F. (1959). "The shortest path through a maze". *Proceedings of an International Symposium on the Theory of Switching (Cambridge, Massachusetts, 2–5 April 1957)*. Cambridge: Harvard University Press.

pp. 285–292.
- Shimbel, Alfonso (1953). "Structural parameters of communication networks". *Bulletin of Mathematical Biophysics* **15** (4): 501–507. doi:10.1007/BF02476438.

Further reading [edit]

- Frigioni, D.; Marchetti-Spaccamela, A.; Nanni, U. (1998). "Fully dynamic output bounded single source shortest path problem". *Proc. 7th Annu. ACM-SIAM Symp. Discrete Algorithms*. Atlanta, GA. pp. 212–221.

Authority control GND: 4138403-9

Categories: Network theory | Polynomial-time problems | Graph algorithms | Computational problems in graph theory