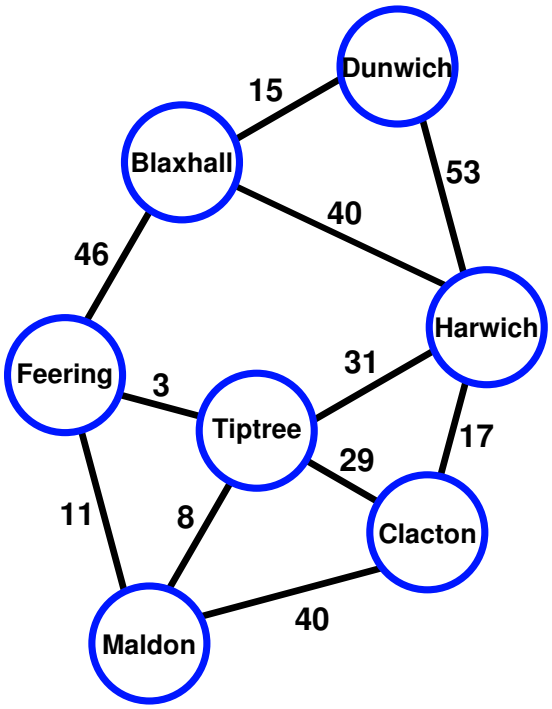
Widest path problem



In this graph, the widest path from Maldon to Feering has bandwidth 29, and passes through Clacton, Tiptree, Harwich, and Blaxhall.

In graph algorithms, the **widest path problem** is the problem of finding a path between two designated vertices in a weighted graph, maximizing the weight of the minimum-weight edge in the path. The widest path problem is also known as the **bottleneck shortest path problem** or the **maximum capacity path problem**. It is possible to adapt most shortest path algorithms to compute widest paths, by modifying them to use the bottleneck distance instead of path length.[1] However, in many cases even faster algorithms are possible.

For instance, in a graph that represents connections between routers in the Internet, where the weight of an edge represents the bandwidth of a connection between two routers, the widest path problem is the problem of finding an end-to-end path between two Internet nodes that has the maximum possible bandwidth.[2] The smallest edge weight on this path is known as the capacity or bandwidth of the path. As well as its applications in network routing, the widest path problem is also an important component of the Schulze method for deciding the winner of a multiway election,[3] and has been applied to digital compositing,[4] metabolic pathway analysis,[5] and the computation of maximum flows.[6]

A closely related problem, the **minimax path problem**, asks for the path that minimizes the maximum weight of any of its edges. It has applications that include transportation planning.[7] Any algorithm for the widest path problem can be transformed into an algorithm for the minimax path problem, or vice versa, by reversing the sense of all the weight comparisons performed by the algorithm, or equivalently by replacing every edge weight by its negation.

1 Undirected graphs

In an undirected graph, a widest path may be found as the path between the two vertices in the maximum spanning tree of the graph, and a minimax path may be found as the path between the two vertices in the minimum spanning tree.[8][9][10]

In any graph, directed or undirected, there is a straightforward algorithm for finding a widest path once the weight of its minimum-weight edge is known: simply delete all smaller edges and search for any path among the remaining edges using breadth first search or depth first search. Based on this test, there also exists a linear time algorithm for finding a widest *s*-*t* path in an undirected graph, that does not use the maximum spanning tree. The main idea of the algorithm is to apply the linear-time path-finding algorithm to the median edge weight in the graph, and then either to delete all smaller edges or contract all larger edges according to whether a path does or does not exist, and recurse in the resulting smaller graph.[9][11][12]

Fernandez, Garfinkel & Arbiol (1998) use undirected bottleneck shortest paths in order to form composite aerial photographs that combine multiple images of overlapping areas. In the subproblem to which the widest path problem applies, two images have already been transformed into a common coordinate system; the remaining task is to select a *seam*, a curve that passes through the region of overlap and divides one of the two images from the other. Pixels on one side of the seam will be copied from one of the images, and pixels on the other side of the seam will be copied from the other image. Unlike other compositing methods that average pixels from both images, this produces a valid photographic image of every part of the region being photographed. They weight the edges of a grid graph by a numeric estimate of how visually apparent a seam across that edge would be, and find a bottleneck shortest path for these weights. Using this path as the seam, rather than a more conventional

shortest path, causes their system to find a seam that is difficult to discern at all of its points, rather than allowing it to trade off greater visibility in one part of the image for lesser visibility elsewhere.[4]

A solution to the minimax path problem between the two opposite corners of a grid graph can be used to find the weak Fréchet distance between two polygonal chains. Here, each grid graph vertex represents a pair of line segments, one from each chain, and the weight of an edge represents the Fréchet distance needed to pass from one pair of segments to another.[13]

If all edge weights of an undirected graph are positive, then the minimax distances between pairs of points (the maximum edge weights of minimax paths) form an ultrametric; conversely every finite ultrametric space comes from minimax distances in this way.[14] A data structure constructed from the minimum spanning tree allows the minimax distance between any pair of vertices to be queried in constant time per query, using lowest common ancestor queries in a Cartesian tree. The root of the Cartesian tree represents the heaviest minimum spanning tree edge, and the children of the root are Cartesian trees recursively constructed from the subtrees of the minimum spanning tree formed by removing the heaviest edge. The leaves of the Cartesian tree represent the vertices of the input graph, and the minimax distance between two vertices equals the weight of the Cartesian tree node that is their lowest common ancestor. Once the minimum spanning tree edges have been sorted, this Cartesian tree can be constructed in linear time.[15]

2 Directed graphs

In directed graphs, the maximum spanning tree solution cannot be used. Instead, several different algorithms are known; the choice of which algorithm to use depends on whether a start or destination vertex for the path is fixed, or whether paths for many start or destination vertices must be found simultaneously.

2.1 All pairs

The all-pairs widest path problem has applications in the Schulze method for choosing a winner in multi-way elections in which voters rank the candidates in preference order. The Schulze method constructs a complete directed graph in which the vertices represent the candidates and every two vertices are connected by an edge. Each edge is directed from the winner to the loser of a pairwise contest between the two candidates it connects, and is labeled with the margin of victory of that contest. Then the method computes widest paths between all pairs of vertices, and the winner is the candidate whose vertex has wider paths to each opponent than vice versa.[3] The results of an election using this method are consistent with the Condorcet method – a candidate who wins all pairwise contests automatically wins the whole election – but it generally allows a winner to be selected, even in situations where the Concorcet method itself fails.[16] The Schulze method has been used by several organizations including the Wikimedia Foundation.[17]

To compute the widest path widths for all pairs of nodes in a dense directed graph, such as the ones that arise in the voting application, the asymptotically fastest known approach takes time $O(n^{(3+\omega)/2})$ where ω is the exponent for fast matrix multiplication. Using the best known algorithms for matrix multiplication, this time bound becomes $O(n^{2.688})$.[18] Instead, the reference implementation for the Schulze method uses a modified version of the simpler Floyd–Warshall algorithm, which takes $O(n^3)$ time.[3] For sparse graphs, it may be more efficient to repeatedly apply a single-source widest path algorithm.

2.2 Single source

If the edges are sorted by their weights, then a modified version of Dijkstra's algorithm can compute the bottlenecks between a designated start vertex and every other vertex in the graph, in linear time. The key idea behind the speedup over a conventional version of Dijkstra's algorithm is that the sequence of bottleneck distances to each vertex, in the order that the vertices are considered by this algorithm, is a monotonic subsequence of the sorted sequence of edge weights; therefore, the priority queue of Dijkstra's algorithm can be implemented as a bucket queue: an array indexed by the numbers from 1 to m (the number of edges in the graph), where array cell i contains the vertices whose bottleneck distance is the weight of the edge with position i in the sorted order. This method allows the widest path problem to be solved as quickly as sorting; for instance, if the edge weights are represented as integers, then the time bounds for integer sorting a list of m integers would apply also to this problem.[12]

2.3 Single source and single destination

Berman & Handler (1987) suggest that service vehicles and emergency vehicles should use minimax paths when returning from a service call to their base. In this application, the time to return is less important than the response time if another service call occurs while the vehicle is in the process of returning. By using a minimax path, where the weight of an edge is the maximum travel time from a point on the edge to the farthest possible service call, one can plan a route that minimizes the maximum possible delay between receipt of a service call and arrival of a responding vehicle.[7] Ullah, Lee & Hassoun (2009) use maximin paths to model the dominant reaction chains in metabolic networks; in their model, the weight of an edge is the free energy of the metabolic reaction represented

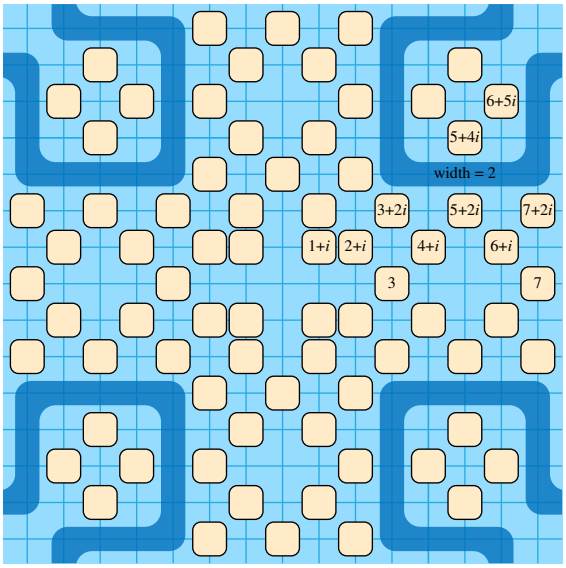
by the edge.[5]

Another application of widest paths arises in the Ford–Fulkerson algorithm for the maximum flow problem. Repeatedly augmenting a flow along a maximum capacity path in the residual network of the flow leads to a small bound, $O(m \log U)$, on the number of augmentations needed to find a maximum flow; here, the edge capacities are assumed to be integers that are at most U. However, this analysis does not depend on finding a path that has the exact maximum of capacity; any path whose capacity is within a constant factor of the maximum suffices. Combining this approximation idea with the shortest path augmentation method of the Edmonds–Karp algorithm leads to a maximum flow algorithm with running time $O(mn \log U)$.[6]

It is possible to find maximum-capacity paths and minimax paths with a single source and single destination very efficiently even in models of computation that allow only comparisons of the input graph's edge weights and not arithmetic on them.[12][19] The algorithm maintains a set S of edges that are known to contain the bottleneck edge of the optimal path; initially, S is just the set of all m edges of the graph. At each iteration of the algorithm, it splits S into an ordered sequence of subsets S_1, S_2, ... of approximately equal size; the number of subsets in this partition is chosen in such a way that all of the split points between subsets can be found by repeated median-finding in time $O(m)$. The algorithm then reweights each edge of the graph by the index of the subset containing the edge, and uses the modified Dijkstra algorithm on the reweighted graph; based on the results of this computation, it can determine in linear time which of the subsets contains the bottleneck edge weight. It then replaces S by the subset Si that it has determined to contain the bottleneck weight, and starts the next iteration with this new set S. The number of subsets into which S can be split increases exponentially with each step, so the number of iterations is proportional to the iterated logarithm function, $O(\log^* n)$, and the total time is $O(m \log^* n)$.[19] In a model of computation where each edge weight is a machine integer, the use of repeated bisection in this algorithm can be replaced by a list-splitting technique of Han & Thorup (2002), allowing S to be split into $O(\sqrt{m})$ smaller sets Si in a single step and leading to a linear overall time bound.[20]

3 Euclidean point sets

A variant of the minimax path problem has also been considered for sets of points in the Euclidean plane. As in the undirected graph problem, this Euclidean minimax path problem can be solved efficiently by finding a Euclidean minimum spanning tree: every path in the tree is a minimax path. However, the problem becomes more complicated when a path is desired that not only minimizes the hop length but also, among paths with the same hop length, minimizes or approximately minimizes the total length of the path. The solution can be approximated using geometric spanners.[21]



The dark blue band separates pairs of Gaussian prime numbers whose minimax path length is 2 or more.

In number theory, the unsolved Gaussian moat problem asks whether or not minimax paths in the Gaussian prime numbers have bounded or unbounded minimax length. That is, does there exist a constant B such that, for every pair of points p and q in the infinite Euclidean point set defined by the Gaussian primes, the minimax path in the Gaussian primes between p and q has minimax edge length at most B?[22]

4 References

[1] Pollack, Maurice (1960), "The maximum capacity through a network", *Operations Research*, **8** (5): 733–736, doi:10.1287/opre.8.5.733, JSTOR 167387

[2] Shacham, N. (1992), "Multicast routing of hierarchical data", *IEEE International Conference on Communications (ICC '92)*, **3**, pp. 1217–1221, doi:10.1109/ICC.1992.268047; Wang, Zheng; Crowcroft, J. (1995), "Bandwidth-delay based routing algorithms", *IEEE Global Telecommunications Conference (GLOBECOM '95)*, **3**, pp. 2129–2133, doi:10.1109/GLOCOM.1995.502780

[3] Schulze, Markus (2011), "A new monotonic, clone-independent, reversal symmetric, and Condorcet-consistent single-winner election method", *Social Choice and Welfare*, **36** (2): 267–303, doi:10.1007/s00355-010-0475-4

[4] Fernandez, Elena; Garfinkel, Robert; Arbiol, Roman (1998), "Mosaicking of aerial photographic maps via seams defined by bottleneck shortest paths", *Operations Research*, **46** (3): 293–304, doi:10.1287/opre.46.3.293, JSTOR 222823

[5] Ullah, E.; Lee, Kyongbum; Hassoun, S. (2009), "An algorithm for identifying dominant-edge metabolic pathways", *IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2009)*, pp. 144–150

[6] Ahuja, Ravindra K.; Magnanti, Thomas L.; Orlin, James B. (1993), "7.3 Capacity Scaling Algorithm", *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, pp. 210–212, ISBN 0-13-617549-X

[7] Berman, Oded; Handler, Gabriel Y. (1987), "Optimal Minimax Path of a Single Service Unit on a Network to Nonservice Destinations", *Transportation Science*, **21** (2): 115–122, doi:10.1287/trsc.21.2.115

[8] Hu, T. C. (1961), "The maximum capacity route problem", *Operations Research*, **9** (6): 898–900, doi:10.1287/opre.9.6.898, JSTOR 167055

[9] Punnen, Abraham P. (1991), "A linear time algorithm for the maximum capacity path problem", *European Journal of Operational Research*, **53** (3): 402–404, doi:10.1016/0377-2217(91)90073-5

[10] Malpani, Navneet; Chen, Jianer (2002), "A note on practical construction of maximum bandwidth paths", *Information Processing Letters*, **83** (3): 175–180, doi:10.1016/S0020-0190(01)00323-4, MR 1904226

[11] Camerini, P. M. (1978), "The min-max spanning tree problem and some extensions", *Information Processing Letters*, **7** (1): 10–14, doi:10.1016/0020-0190(78)90030-3

[12] Kaibel, Volker; Peinhardt, Matthias A. F. (2006), *On the bottleneck shortest path problem* (PDF), ZIB-Report 06-22, Konrad-Zuse-Zentrum für Informationstechnik Berlin

[13] Alt, Helmut; Godau, Michael (1995), "Computing the Fréchet distance between two polygonal curves" (PDF), *International Journal of Computational Geometry and Applications*, **5** (1–2): 75–91, doi:10.1142/S0218195995000064.

[14] Leclerc, Bruno (1981), "Description combinatoire des ultramétriques", *Centre de Mathématique Sociale. École Pratique des Hautes Études. Mathématiques et Sciences Humaines* (in French) (73): 5–37, 127, MR 623034

[15] Demaine, Erik D.; Landau, Gad M.; Weimann, Oren (2009), "On Cartesian trees and range minimum queries", *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009*, Lecture Notes in Computer Science, **5555**, pp. 341–353, doi:10.1007/978-3-642-02927-1_29

[16] More specifically, the only kind of tie that the Schulze method fails to break is between two candidates who have equally wide paths to each other.

[17] See Jesse Plamondon-Willard, Board election to use preference voting, May 2008; Mark Ryan, 2008 Wikimedia Board Election results, June 2008; 2008 Board Elections, June 2008; and 2009 Board Elections, August 2009.

[18] Duan, Ran; Pettie, Seth (2009), "Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths", *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '09)*, pp. 384–391. For an earlier algorithm that also used fast matrix multiplication to speed up all pairs widest paths, see Vassilevska, Virginia; Williams, Ryan; Yuster, Raphael (2007), "All-pairs bottleneck paths for general graphs in truly sub-cubic time", *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC '07)*, New York: ACM, pp. 585–589, doi:10.1145/1250790.1250876, MR 2402484 and Chapter 5 of Vassilevska, Virginia (2008), *Efficient Algorithms for Path Problems in Weighted Graphs* (PDF), Ph.D. thesis, Report CMU-CS-08-147, Carnegie Mellon University School of Computer Science

[19] Gabow, Harold N.; Tarjan, Robert E. (1988), "Algorithms for two bottleneck optimization problems", *Journal of Algorithms*, **9** (3): 411–417, doi:10.1016/0196-6774(88)90031-4, MR 955149

[20] Han, Yijie; Thorup, M. (2002), "Integer sorting in $O(n\sqrt{\log \log n})$ expected time and linear space", *Proc. 43rd Annual Symposium on Foundations of Computer Science (FOCS 2002)*, pp. 135–144, doi:10.1109/SFCS.2002.1181890.

[21] Bose, Prosenjit; Maheshwari, Anil; Narasimhan, Giri; Smid, Michiel; Zeh, Norbert (2004), "Approximating geometric bottleneck shortest paths", *Computational Geometry. Theory and Applications*, **29** (3): 233–249, doi:10.1016/j.comgeo.2004.04.003, MR 2095376

[22] Gethner, Ellen; Wagon, Stan; Wick, Brian (1998), "A stroll through the Gaussian primes", *American Mathematical Monthly*, **105** (4): 327–337, doi:10.2307/2589708, MR 1614871.

5 Text and image sources, contributors, and licenses

5.1 Text

- **Widest path problem** *Source:* https://en.wikipedia.org/wiki/Widest_path_problem?oldid=765885217 *Contributors:* Edward, Michael Hardy, Giftlite, Woohookitty, Rjwilmsi, KConWiki, David Eppstein, R'n'B, Breawycker, Volkan YAZICI, Baiyubin, Daveagp, Ost316, Addbot, Legobot, Luckas-bot, John of Reading, Primefac, Mark viking, Sizeofint, Fmadd and Anonymous: 1

5.2 Images

- **File:CPT-Graphs-undirected-weighted.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/5f/CPT-Graphs-undirected-weighted.svg *License:* CC0 *Contributors:* Own work *Original artist:* Pluke
- **File:Gaussian_moat_15x15.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/0f/Gaussian_moat_15x15.svg *License:* CC0 *Contributors:* Own work *Original artist:* David Eppstein

5.3 Content license

- Creative Commons Attribution-Share Alike 3.0