WIKIPEDIA

Longest path problem

In graph theory and theoretical computer science, the **longest path problem** is the problem of finding a simple path of maximum length in a given graph. A path is called simple if it does not have any repeated vertices; the length of a path may either be measured by its number of edges, or (in weighted graphs) by the sum of the weights of its edges. In contrast to the shortest path problem, which can be solved in polynomial time in graphs without negative-weight cycles, the longest path problem is NP-hard, meaning that it cannot be solved in polynomial time for arbitrary graphs unless P = NP. Stronger hardness results are also known showing that it is difficult to approximate. However, it has a linear time solution for directed acyclic graphs, which has important applications in finding the critical path in scheduling problems.

Contents

NP-hardness

The NP-hardness of the unweighted longest path problem can be shown using a reduction from the Hamiltonian path problem: a graph *G* has a Hamiltonian path if and only if its longest path has length $n - 1$, where *n* is the number of vertices in *G*. Because the Hamiltonian path problem is NP-complete, this reduction shows that the decision version of the longest path problem is also NP-complete. In this decision problem, the input is a graph *G* and a number *k*; the desired output is "yes" if *G* contains a path of *k* or more edges, and *no* otherwise.[1]

If the longest path problem could be solved in polynomial time, it could be used to solve this decision problem, by finding a longest path and then comparing its length to the number *k*. Therefore, the longest path problem is NP-hard. The question "does there exist a simple path in a given graph with at least *k* edges" is NP-complete.[2]

In weighted complete graphs with non-negative edge weights, the weighted longest path problem is the same as the Travelling salesman path problem, because the longest path always includes all vertices.[3]

Acyclic graphs and critical paths

A longest path between two given vertices *s* and *t* in a weighted graph *G* is the same thing as a shortest path in a graph $-G$ derived from *G* by changing every weight to its negation. Therefore, if shortest paths can be found in $-G$, then longest paths can also be found in *G*.[4]

For most graphs, this transformation is not useful because it creates cycles of negative length in $-G$. But if *G* is a directed acyclic graph, then no negative cycles can be created, and a longest path in *G* can be found in linear time by applying a linear time algorithm for shortest paths in $-G$, which is also a directed acyclic graph.[4] For instance, for each vertex *v* in a given DAG, the length of the longest path ending at *v* may be obtained by the following steps:

1. Find a topological ordering of the given DAG.
2. For each vertex *v* of the DAG, in the topological ordering, compute the length of the longest path ending at *v* by looking at its incoming neighbors and adding one to the maximum length recorded for those neighbors. If *v* has no incoming neighbors, set the length of the longest path ending at *v* to zero. In either case, record this number so that later steps of the algorithm can access it.

Once this has been done, the longest path in the whole DAG may be obtained by starting at the vertex *v* with the largest recorded value, then repeatedly stepping backwards to its incoming neighbor with the largest recorded value, and reversing the sequence of vertices found in this way.

The critical path method for scheduling a set of activities involves the construction of a directed acyclic graph in which the vertices represent project milestones and the edges represent activities that must be performed after one milestone and before another; each edge is weighted by an estimate of the amount of time the corresponding activity will take to complete. In such a graph, the longest path from the first milestone to the last one is the critical path, which describes the total time for completing the project.[4]

Longest paths of directed acyclic graphs may also be applied in layered graph drawing: assigning each vertex *v* of a directed acyclic graph *G* to the layer whose number is the length of the longest path ending at *v* results in a layer assignment for *G* with the minimum possible number of layers.[5]

Approximation

Björklund, Husfeldt & Khanna (2004) write that the longest path problem in unweighted undirected graphs "is notorious for the difficulty of understanding its approximation hardness".[6] The best polynomial time approximation algorithm known for this case achieves only a very weak approximation ratio, $n/\exp(\Omega(\sqrt{\log n}))$.[7] For all $\epsilon > 0$, it is not possible to approximate the longest path to within a factor of $2^{(\log n)^{1-\epsilon}}$ unless NP is contained within quasi-polynomial deterministic time; however, there is a big gap between this inapproximability result and the known approximation algorithms for this problem.[8]

In the case of unweighted but directed graphs, strong inapproximability results are known. For every $\epsilon > 0$ the problem cannot be approximated to within a factor of $n^{1-\epsilon}$ unless P = NP, and with stronger complexity-theoretic assumptions it cannot be approximated to within a factor of $n/\log^{2+\epsilon} n$.[6] The color-coding technique can be used to find paths of logarithmic length, if they exist, but this gives an approximation ratio of only $O(n/\log n)$.[9]

Parameterized complexity

The longest path problem is fixed-parameter tractable when parameterized by the length of the path. For instance, it can be solved in time linear in the size of the input graph (but exponential in the length of the path), by an algorithm that performs the following steps:

1. Perform a depth-first search of the graph. Let d be the depth of the resulting depth-first search tree.
2. Use the sequence of root-to-leaf paths of the depth-first search tree, in the order in which they were traversed by the search, to construct a path decomposition of the graph, with pathwidth d.
3. Apply dynamic programming to this path decomposition to find a longest path in time $O(d!2^d n)$, where n is the number of vertices in the graph.

Since the output path has length at least as large as d, the running time is also bounded by $O(\ell!2^\ell n)$, where ℓ is the length of the longest path.[10] Using color-coding, the dependence on path length can be reduced to singly exponential.[9][11][12][13] A similar dynamic programming technique shows that the longest path problem is also fixed-parameter tractable when parameterized by the treewidth of the graph.

For graphs of bounded clique-width, the longest path can also be solved by a polynomial time dynamic programming algorithm. However, the exponent of the polynomial depends on the clique-width of the graph, so this algorithms is not fixed-parameter tractable. The longest path problem, parameterized by clique-width, is hard for the parameterized complexity class $W[1]$, showing that a fixed-parameter tractable algorithm is unlikely to exist.[14]

Special classes of graphs

A linear-time algorithm for finding a longest path in a tree was proposed by Dijkstra in 1960's, while a formal proof of this algorithm was published in 2002.[15] Furthermore, a longest path can be computed in polynomial time on weighted trees, on block graphs, on cacti,[16] on bipartite permutation graphs,[17] and on Ptolemaic graphs.[18]

For the class of interval graphs, an $O(n^4)$-time algorithm is known, which uses a dynamic programming approach.[19] This dynamic programming approach has been exploited to obtain polynomial-time algorithms on the greater classes of circular-arc graphs[20] and of co-comparability graphs (i.e. of the complements of comparability graphs, which also contain permutation graphs),[21] both having the same running time $O(n^4)$. The latter algorithm is based on special properties of the Lexicographic Depth First Search (LDFS) vertex ordering[22] of co-comparability graphs. For co-comparability graphs also an alternative polynomial-time algorithm with higher running time $O(n^7)$ is known, which is based on the Hasse diagram of the partially ordered set defined by the complement of the input co-comparability graph.[23]

Furthermore, the longest path problem is solvable in polynomial time on any class of graphs with bounded treewidth or bounded clique-width, such as the distance-hereditary graphs. Finally, it is clearly NP-hard on all graph classes on which the Hamiltonian path problem is NP-hard, such as on split graphs, circle graphs, and planar graphs.

See also

- Gallai–Hasse–Roy–Vitaver theorem, a duality relation between longest paths and graph coloring
- Longest uncrossed knight's path
- Snake-in-the-box, the longest induced path in a hypercube graph

References

1. Schrijver, Alexander (2003), *Combinatorial Optimization: Polyhedra and Efficiency, Volume 1* (https://books.google.com/books?id=mqGeSQ6dJycC&pg=PA114), Algorithms and Combinatorics, **24**, Springer, p. 114, ISBN 9783540443896.
2. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001), *Introduction To Algorithms* (https://books.google.com/books?id=NLngYyWFl_YC&pg=PA978) (2nd ed.), MIT Press, p. 978, ISBN 9780262032933.
3. Lawler, Eugene L. (2001), *Combinatorial Optimization: Networks and Matroids* (https://books.google.com/books?id=m4MvtFenVjEC&pg=PA64), Courier Dover Publications, p. 64, ISBN 9780486414539.
4. Sedgewick, Robert; Wayne, Kevin Daniel (2011), *Algorithms* (https://books.google.com/books?id=idUdqdDXqnAC&pg=PA661) (4th ed.), Addison-Wesley Professional, pp. 661–666, ISBN 9780321573513.
5. Di Battista, Giuseppe; Eades, Peter; Tamassia, Roberto; Tollis, Ioannis G. (1998), "Layered Drawings of Digraphs", *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, pp. 265–302, ISBN 978-0-13-301615-4.
6. Björklund, Andreas; Husfeldt, Thore; Khanna, Sanjeev (2004), "Approximating longest directed paths and cycles", *Proc. Int. Coll. Automata, Languages and Programming (ICALP 2004)*, Lecture Notes in Computer Science, **3142**, Berlin: Springer-Verlag, pp. 222–233, MR 2160935 (https://www.ams.org/mathscinet-getitem?mr=2160935).
7. Gabow, Harold N.; Nie, Shuxin (2008), "Finding long paths, cycles and circuits", *International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science, **5369**, Berlin: Springer, pp. 752–763, doi:10.1007/978-3-540-92182-0_66 (https://doi.org/10.1007%2F978-3-540-92182-0_66), MR 2539968 (https://www.ams.org/mathscinet-getitem?mr=2539968). For earlier work with even weaker approximation bounds, see Gabow, Harold N. (2007), "Finding paths and cycles of superpolylogarithmic length" (http://www.cs.colorado.edu/~hal/u.pdf) (PDF), *SIAM Journal on Computing*, **36** (6): 1648–1671, doi:10.1137/S0097539704445366 (https://doi.org/10.1137%2FS0097539704445366), MR 2299418 (https://www.ams.org/mathscinet-getitem?mr=2299418) and Björklund, Andreas; Husfeldt, Thore (2003), "Finding a path of superlogarithmic length", *SIAM Journal on Computing*, **32** (6): 1395–1402, doi:10.1137/S0097539702416761 (https://doi.org/10.1137%2FS0097539702416761), MR 2034242 (https://www.ams.org/mathscinet-getitem?mr=2034242).

8. Karger, David; Motwani, Rajeev; Ramkumar, G. D. S. (1997), "On approximating the longest path in a graph", *Algorithmica*, **18** (1): 82–98, doi:10.1007/BF02523689 (https://doi.org/10.1007%2FBF02523689), MR 1432030 (https://www.ams.org/mathscinet-getitem?mr=1432030).
9. Alon, Noga; Yuster, Raphael; Zwick, Uri (1995), "Color-coding", *Journal of the ACM*, **42** (4): 844–856, doi:10.1145/210332.210337 (https://doi.org/10.1145%2F210332.210337), MR 1411787 (https://www.ams.org/mathscinet-getitem?mr=1411787).
10. Bodlaender, Hans L. (1993), "On linear time minor tests with depth-first search", *Journal of Algorithms*, **14** (1): 1–23, doi:10.1006/jagm.1993.1001 (https://doi.org/10.1006%2Fjagm.1993.1001), MR 1199244 (https://www.ams.org/mathscinet-getitem?mr=1199244). For an earlier FPT algorithm with slightly better dependence on the path length, but worse dependence on the size of the graph, see Monien, B. (1985), "How to find long paths efficiently", *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, North-Holland Math. Stud., **109**, Amsterdam: North-Holland, pp. 239–254, doi:10.1016/S0304-0208(08)73110-4 (https://doi.org/10.1016%2FS0304-0208%2808%2973110-4), MR 0808004 (https://www.ams.org/mathscinet-getitem?mr=0808004).
11. Chen, Jianer; Lu, Songjian; Sze, Sing-Hoi; Zhang, Fenghui (2007), "Improved algorithms for path, matching, and packing problems", *Proc. 18th ACM-SIAM Symposium on Discrete algorithms (SODA '07)* (http://faculty.cse.tamu.edu/shsze/papers/kpath.pdf) (PDF), pp. 298–307.
12. Koutis, Ioannis (2008), "Faster algebraic algorithms for path and packing problems", *International Colloquium on Automata, Languages and Programming* (http://ccom.uprrp.edu/~ikoutis/papers/MultilinearDetection.pdf) (PDF), Lecture Notes in Computer Science, **5125**, Berlin: Springer, pp. 575–586, doi:10.1007/978-3-540-70575-8_47 (https://doi.org/10.1007%2F978-3-540-70575-8_47), MR 2500302 (https://www.ams.org/mathscinet-getitem?mr=2500302).
13. Williams, Ryan (2009), "Finding paths of length k in $O^*(2^k)$ time", *Information Processing Letters*, **109** (6): 315–318, arXiv:0807.3026 (https://arxiv.org/abs/0807.3026), doi:10.1016/j.ipl.2008.11.004 (https://doi.org/10.1016%2Fj.ipl.2008.11.004), MR 2493730 (https://www.ams.org/mathscinet-getitem?mr=2493730).
14. Fomin, Fedor V.; Golovach, Petr A.; Lokshtanov, Daniel; Saurabh, Saket (2009), "Clique-width: on the price of generality", *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (SODA '09)* (https://www.siam.org/proceedings/soda/2009/SODA09_090_fominf.pdf) (PDF), pp. 825–834.
15. Bulterman, R.W.; van der Sommen, F.W.; Zwaan, G.; Verhoeff, T.; van Gasteren, A.J.M. (2002), "On computing a longest path in a tree" (http://www.sciencedirect.com/science/article/pii/S0020019001001983?via%3Dihub), *Information Processing Letters*, **81** (2): 93–96, doi:10.1016/S0020-0190(01)00198-3 (https://doi.org/10.1016%2FS0020-0190%2801%2900198-3).
16. Uehara, Ryuhei; Uno, Yushi (2004), "Efficient algorithms for the longest path problem" (https://link.springer.com/chapter/10.1007%2F978-3-540-30551-4_74), *ISAAC 2004*: 871–883.
17. Uehara, Ryuhei; Valiente, Gabriel (2007), "Linear structure of bipartite permutation graphs and the longest path problem" (http://www.sciencedirect.com/science/article/pii/S0020019007000464?via%3Dihub), *Information Processing Letters*, **103** (2): 71–77, doi:10.1016/j.ipl.2007.02.010 (https://doi.org/10.1016%2Fj.ipl.2007.02.010).
18. Takahara, Yoshihiro; Teramoto, Sachio; Uehara, Ryuhei (2008), "Longest path problems on Ptolemaic graphs", *IEICE Transactions*, **91-D**: 170–177.
19. Ioannidou, Kyriaki; Mertzios, George B.; Nikolopoulos, Stavros D. (2011), "The longest path problem has a polynomial solution on interval graphs" (https://link.springer.com/article/10.1007%2Fs00453-010-9411-3), *Algorithmica*, **61** (2): 320–341, doi:10.1007/s00453-010-9411-3 (https://doi.org/10.1007%2Fs00453-010-9411-3).
20. Mertzios, George B.; Bezakova, Ivona (2014), "Computing and counting longest paths on circular-arc graphs in polynomial time" (http://www.sciencedirect.com/science/article/pii/S0166218X12003241?via%3Dihub), *Discrete Applied Mathematics*, **164** (2): 383–399, doi:10.1016/j.dam.2012.08.024 (https://doi.org/10.1016%2Fj.dam.2012.08.024).
21. Mertzios, George B.; Corneil, Derek G. (2012), "A simple polynomial algorithm for the longest path problem on cocomparability graphs" (http://epubs.siam.org/doi/10.1137/100793529), *SIAM Journal on Discrete Mathematics*, **26** (3): 940–963, arXiv:1004.4560 (https://arxiv.org/abs/1004.4560), doi:10.1137/100793529 (https://doi.org/10.1137%2F100793529).
22. Corneil, Derek G.; Krueger, Richard (2008), "A unified view of graph searching" (http://epubs.siam.org/doi/10.1137/050623498), *SIAM Journal on Discrete Mathematics*, **22** (4): 1259–1276, doi:10.1137/050623498 (https://doi.org/10.1137%2F050623498).

23. Ioannidou, Kyriaki; Nikolopoulos, Stavros D. (2011), "The longest path problem is polynomial on cocomparability graphs" (http://www.cs.uoi.gr/~stavros/J-Papers/J-2012-ALGO.pdf) (PDF), *Algorithmica*, doi:10.1007/s00453-011-9583-5 (https://doi.org/10.1007%2Fs00453-011-9583-5).

External links

- "Find the Longest Path (http://valis.cs.uiuc.edu/~sariel/misc/funny/longestpath.mp3)", song by Dan Barrett