Finding a Path of Superlogarithmic Length

Andreas Björklund and Thore Husfeldt

Department of Computer Science, Lund University

Abstract. We consider the problem of finding a long, simple path in an undirected graph. We present a polynomial-time algorithm that finds a path of length $\Omega\big((\log L/\log\log L)^2\big)$, where L denotes the length of the longest simple path in the graph. This establishes the performance ratio $O\big(|V|(\log\log|V|/\log|V|)^2\big)$ for the Longest Path problem, where V denotes the graph's vertices.

1 Introduction

Given an unweighted, undirected graph $G=(V,E)$ the *longest path* problem is to find the longest sequence of distinct vertices $v_1\cdots v_k$ such that $v_iv_{i+1}\in E$. This is a classical NP-hard problem (number ND29 in Garey and Johnson [5]) with a considerable body of research devoted to it, yet its approximability remains elusive:

> "For most canonical NP-hard problems, either dramatically improved approximation algorithms have been devised, or strong negative results have been established, leading to a substantially improved understanding of the approximability of these problems. However, there is one problem which has resisted all attempts at devising either positive or negative results — longest paths and cycles in undirected graphs. Essentially, there is no known algorithm which guarantees approximation ratio better than $|V|/\text{polylog}|V|$ and there are no hardness of approximation results that explain this situation." [4]

Indeed, the quoted ratio has been obtained only for special classes of graphs (for example, Hamiltonian graphs), while in the general case the best known ratio prior to the present paper was of order $|V|/\log|V|$.

We present a polynomial-time algorithm for the general case that finds a path of length $\Omega\big((\log L/\log\log L)^2\big)$ in a graph with longest path length L; the best previous bound was $\Omega(\log L)$. This corresponds to a performance ratio of order

$$O\bigg(\frac{|V|\big(\log\log|V|\big)^2}{\log^2|V|}\bigg). \qquad (1)$$

For bounded degree graphs we improve the ratio to $O\big(|V|\log\log|V|/\log^2|V|\big)$. For three-connected graphs we establish the perormance ratio (1) for the *longest cycle* problem.

Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden. Email: thore@cs.lth.se.

Previous work

The first approximation algorithms for longest path are due to Monien [7] and Bodlaender [2], both finding a path of length $\Omega(\log L/\log\log L)$. Neither of these algorithms can be used to find a $\log|V|$ path if it exists, but Papadimitriou and Yannakakis conjectured that such a polynomial-time algorithm exists [8]. This was confirmed by Alon, Yuster, and Zwick [1], introducing the important method of *colour-coding.* Especially, this algorithm finds an $\Omega(\log L)$-path and corresponds to a performance ratio of

$$O\left(\frac{|V|}{\log|V|}\right),$$

which is the best ratio known prior to the present paper.

The problem has received additional study for restricted classes of graphs, where the "$\log|V|$-barrier" has been broken by Vishwanathan [9]. His algorithm achieves the same performance ratio (1) as ours, but works only for Hamiltonian graphs. In *sparse* Hamiltonian graphs, Feder, Motwani, and Subi [4] find even longer paths.

The hardness results for this problem are mainly due to Karger, Motwani, and Ramkumar [6]: The longest path problem does not belong to APX and cannot be approximated within $2^{\log^{1-\epsilon}|V|}$ unless NP $\subseteq$ DTIME$\big(2^{O(\log^{1/\epsilon} n)}\big)$ for any $\epsilon > 0$.

2 Preliminaries

In the remainder, we consider a connected graph $G = (V, E)$ with $n = |V|$ vertices and $e = |E|$ edges. We write $G[W]$ for the graph induced by the vertex set W.

Paths and cycles

The *length* of a path and a cycle is its number of edges. The length of a cycle C is denoted $l(C)$. A k-cycle is a cycle of length k, a k^+-cycle is a cycle of length k or larger. A k-path and k^+-path is defined similarly. For vertices x and y, an xy-path is a (simple) path from x to y, and if P is a path containing u and v we write $P[u,v]$ for the subpath from u to v. We let $L_G(v)$ denote the length of the longest path from a vertex v in the graph G, and sometimes abbreviate $L_W(v) = L_{G[W]}(v)$. The *path length* of G is $\max_{v\in V} L_G(v)$.

We need the following result, Theorem 5.3(i) of [2]:

Theorem 1 (Bodlaender) *Given a graph, two of its vertices s, t, and an integer k, one can find a k^+-path from s to t (if it exists) in time $O\big((2k)!2^{2k}n + e\big)$.*

Corollary 1 *A k^+-cycle through a given vertex can be found in time $t(k) = O\big(((2k)!2^{2k}n + e)n\big)$, if it exists.*

Proof. Let s be the given vertex. For all neighbours t of s apply the Theorem on the graph with the edge st removed. □

We also need the following easy lemma.

Lemma 1 *If a connected graph contains a path of length r then every vertex is an endpoint of a path of length at least $\frac{1}{2}r$.*

Proof. Given vertices $u, v \in V$ let $d(u, v)$ denote the length of the shortest path between u and v.

Let $P = p_0 \cdots p_r$ be a path and let v be a vertex. Find i minimising $d(p_i, v)$. By minimality there is a path Q from v to p_i that contains no other vertices from P. Now either $QP[p_i, p_r]$ or $QP[p_i, p_0]$ has length at least $\frac{1}{2}r$. □

The next lemma is central to our construction: Assume that a vertex v originates a long path P and v lies on a cycle C; then the removal of C decomposes G into connected components, one of which must contain a large part of P.

Lemma 2 *Assume that a connected graph G contains a simple path P of length $L_G(v) > 1$ originating in vertex v. There exists a connected component $G[W]$ of $G[V - v]$ such that the following holds.*

1. *If $G[W + v]$ contains no k^+-cycle through v then every neighbour $u \in W$ of v is the endpoint of a path of length*

$$L_W(u) \geq L_G(v) - k.$$

2. *If C is a cycle in $G[W+v]$ through v of length $l(C) < L_{G[W+v]}(v)$ then there exists a connected component H of $G[W - C]$ that contains a neighbour u of $C - v$ in $G[W + v]$. Moreover, every such neighbour u is the endpoint of a path in H of length*

$$L_H(u) \geq \frac{L_G(v)}{2l(C)} - 1.$$

Proof. Let $r = L_G(v)$ and $P = p_0 \cdots p_r$, where $p_0 = v$. Note that $P[p_1, p_r]$ lies entirely in one of the components $G[W]$ of $G[V - v]$.

First consider statement 1. Let $u \in W$ be a neighbour of v. Since $G[W]$ is connected, there exists a path Q from u to some vertex of P. Consider such a path. The first vertex p_i of P encountered on Q must have $i < k$ since otherwise the three paths vu, $Q[u, p_i]$ and $P[p_0, p_i]$ form a k^+-cycle. Thus the path $Q[u, p_i]P[p_i, p_r]$ has length at least $r - k + 1 > r - k$.

We proceed to statement 2. Consider any cycle C in $G[W + v]$ through v.

Case 1. First assume that $P \cap C = v$, so that one component H of $G[W - C]$ contains all of P except v. Let N be the set of neighbours of $C - v$ in H. First note that N is nonempty, since $G[W]$ is connected. Furthermore, the path length of H is at least $r - 1$, so Lemma 1 gives $L_H(u) \geq (r - 1)/2$ for every $u \in N$.

Case 2. Assume instead that $|P \cap C| = s > 1$. Enumerate the vertices on P from 0 to r and let $i_1, \ldots, i_s$ denote the indices of vertices in $P \cap C$, in particular

$i_1 = 0$. Let $i_{s+1} = r$. An averaging argument shows that there exists j such that $i_{j+1} - i_j \geq r/s$. Consequently there exists a connected component H of $G(W - C)$ containing a simple path of length $r/s - 2$. At least one of the i_jth or i_{j+1}th vertices of P must belong to $C - v$, so the set of neighbours N of $C - v$ in H must be nonempty. As before, Lemma 1 ensures $L_H(u) \geq r/2s - 1$ for every $u \in N$, which establishes the bound after noting that $s \leq l(C)$. □

3 Result and Algorithm

The construction in this section and its analysis establishes the following theorem, accounting for the performance ratio (1) claimed in the introduction in the worst case.

Theorem 2 *If a graph contains a simple path of length L then we can find a simple path of length*

$$\Omega\biggl(\Bigl(\frac{\log L}{\log\log L}\Bigr)^2\biggr)$$

in polynomial time.

3.1 Construction of the Cycle Decomposition Tree

Given a vertex v in G, our algorithm constructs a rooted node-weighted tree $T_k = T_k(G, v)$, the cycle decomposition tree. Every node of T_k is either a *singleton* or a *cycle* node: A singleton node corresponds to a single vertex $u \in G$ and is denoted $\langle u \rangle$, a cycle node corresponds to a cycle C with a specified vertex $u \in C$ and is denoted $\langle C, u \rangle$. Every singleton node has unit weight and every cycle node $\langle C, u \rangle$ has weight $\frac{1}{2}l(C)$.

The tree is constructed as follows. Initially T_k contains a singleton node $\langle v \rangle$, and a call is made to the following procedure with arguments G and v.

1. For every maximal connected component $G[W]$ of $G[V - v]$, execute step 2.
2. Search for a k^+-cycle through v in $G[W + v]$ using Theorem 1. If such a cycle C is found then execute step 3. Otherwise pick an arbitrary neighbour $u \in G[W + v]$ of v, insert the node $\langle u \rangle$ and the tree edge $\langle v \rangle\langle u \rangle$, and recursively compute $T_k\bigl(G[W], u\bigr)$.
3. Insert the cycle node $\langle C, v \rangle$ and the tree edge $\langle v \rangle\langle C, v \rangle$. For every connected component H of $G[W - C]$ choose an arbitrary neighbour $u \in H$ of $C - v$, and insert the singleton node $\langle u \rangle$ and the tree edge $\langle C, v \rangle\langle u \rangle$. Then, recursively compute $T_k(H, u)$.

Note that each recursive step constructs a tree that is connected to other trees by a single edge, so T_k is indeed a tree. Also note that the ancestor of every cycle node must be a singleton node. The root of T_k is $\langle v \rangle$.

3.2 Paths in the Cycle Decomposition Tree

The algorithm finds a path of greatest weight in T_k. This can be done in linear time by depth first search. The path found in T_k represents a path in G, if we interpret paths through cycle vertices as follows. Consider a path in T_k through a cycle vertex $\langle C, u\rangle$. Both neighbours are singleton nodes, so we consider the subpath $\langle u\rangle\langle C, u\rangle\langle v\rangle$. By construction, v is connected to some vertex $w \in C$ with $w \neq u$. One of the two paths from u to w in C must have length at least half the length of C, call it P. We will interpret the path $\langle u\rangle\langle C, u\rangle\langle v\rangle$ in T_k as a path uPv in G. If a path ends in a cycle node $\langle C, u\rangle$, we may associate it with a path of length $l(C) - 1$, by moving along C from u in any of its two directions. Thus a path of weight m in T_k from the root to a leaf identifies a path of length at least m in G.

We need to show that T_k for some small k has a path of sufficient length:[1]

Lemma 3 *If G contains a path of length $r \geq 2^8$ starting in v then $T_k = T_k(G, v)$ for*

$$k = \left\lceil \frac{2\log r}{\log\log r} \right\rceil$$

contains a weighted path of length at least $\frac{1}{8}k^2 - \frac{1}{4}k - 1$.

Proof. We follow the construction of T_k in §3.1.

We need some additional notation. For a node $x = \langle w\rangle$ or $x = \langle C, w\rangle$ in T_k we let $L(x)$ denote the length of the longest path from w in the component $G[X]$ corresponding to the subtree rooted at x. More precisely, for every successor y of x (including $y = x$), the set X contains the corresponding vertices w' (if $y = \langle w'\rangle$ is a singleton node) or C' (if $y = \langle w', C'\rangle$ is a cycle node).

Furthermore, let $\mathbf{S}(n)$ denote the singleton node children of a node n and let $\mathbf{C}(n)$ denote its cycle node children. Consider any singleton node $\langle v\rangle$.

Lemma 2 asserts that

$$L(v) \leq \max\left\{ \max_{w\in\mathbf{S}\langle v\rangle} L(w) + k, \max_{\substack{\langle C,v\rangle\in\mathbf{C}\langle v\rangle \\ w\in\mathbf{S}\langle C,v\rangle}} \big(2L(w) + 2\big)l(C) \right\}. \tag{2}$$

Define $n(v) = w$ if $\langle w\rangle$ maximises the right hand side of the inequality (2) and consider a path $Q = \langle x_0\rangle \cdots \langle x_t\rangle$ from $\langle v\rangle = \langle x_0\rangle$ described by these heavy nodes. To be precise we have either $n(x_i) = x_{i+1}$ or $n(x_i) = x_{i+2}$, in the latter case the predecessor of $\langle x_{i+2}\rangle$ is a cycle node.

We will argue that the gaps in the sequence

$$L(x_0) \geq L(x_1) \geq \cdots \geq L(x_t).$$

[1] All logarithms are to the base 2 and the constants involved have been chosen aiming for simplicity of the proof, rather than optimality.

cannot be too large due to the inequality above and the fact that $L(x_t)$ must be small (otherwise we are done), and therefore Q contains a lot of cycle nodes or even more singleton nodes.

Let s denote the number of cycle nodes on Q. Since every cycle node has weight at least $\frac{1}{2}k$ the total weight of Q is at least $\frac{1}{2}sk + (t-s) = s(\frac{1}{2}k - 1) + t$.

Consider a singleton node that is followed by a cycle node. There are s such nodes, we will call them *cycle parents*. Assume $\langle x_j \rangle$ is the first cycle parent node. Thus according to the first part of Lemma 2 its predecessors $\langle x_0 \rangle, \ldots, \langle x_j \rangle$ satisfy the relation $L(x_{i+1}) \geq L(x_i) - k$, so

$$L(x_j) \geq r - jk \geq r - \tfrac{1}{8}k^3 \geq \tfrac{7}{8}r,$$

since $j \leq t \leq \frac{1}{8}k^2$ (otherwise we are finished) and $r \geq k^3$.

From the second part of Lemma 2 we have

$$L(x_{j+2}) \geq \frac{7r}{16l(C)} - 1 \geq \frac{r}{k^2}.$$

where we have used $l(C) \leq \frac{1}{4}k^2$ (otherwise we are finished) and $r \geq \frac{4}{3}k^2$.

This analysis may be repeated for the subsequent cycle parents as long as their remaining length after each cycle node passage is at least k^3. Note that Q must pass through as many as $s' \geq \lceil \frac{1}{4}k - 1 \rceil$ cycle nodes before

$$\frac{r}{k^{2s'}} < k^3,$$

at which point the remaining path may be shorter than k^3. Thus we either have visited $s \geq s'$ cycle nodes, amounting to a weighted path Q of length at least

$$s(\tfrac{1}{2}k + 1) \geq \tfrac{1}{8}k^2 - \tfrac{1}{4}k - 1$$

(remembering that any two consecutive cycle nodes must have a singleton node in-between), or there are at most $s < s'$ cycle nodes on Q. In that case there is a tail of singleton nodes starting with some $L(x) \geq k^3$. Since $L(x_j) \leq L(x_{j+1}) + k$ for the nodes on the tail, the length of the tail (and thus the weight of Q) is at least k^2. □

3.3 Summary

Our algorithm divides the input graph into its connected components and performs the following steps for each. It picks a vertex v in the component and constructs cycle decomposition trees T_k for all $k = 6, \ldots, \lceil 2\log n/\log\log n \rceil$. Corollary 1 tells us that this is indeed a polynomial time task. Moreover, Lemma 1 ensures that v originates a path of at least half the length of the longest path in the component. The algorithm then finds paths in G identified by the longest weighted paths in T_k in linear time. Finally, Lemma 3 establishes the desired approximation ratio.

4 Extensions

4.1 Bounded Degree Graphs

As in [9], the class of graphs with their maximum degree bounded by a constant admits a relative $\log\log n$-improvement over the performance ratio shown in this paper. All paths of length $\log n$ can be enumerated in polynomial time for these graphs. Consequently, we can replace the algorithm from Theorem 1 by an algorithm that efficiently finds cycles of logarithmic length or larger through any given vertex if they exist.

Proposition 1 *If a constant degree graph contains a simple path of length L then we can find a simple path of length*

$$\Omega\left(\frac{\log^2 L}{\log\log L}\right)$$

in polynomial time.

This gives the performance ratio $O\big(|V|\log\log|V|/\log^2|V|\big)$ for the longest path problem in constant degree graphs.

4.2 Three-Connected Graphs

Bondy and Locke [3] have shown that every 3-connected graph with path length l must contain a cycle of length at least $2l/5$. Moreover, their construction is easily seen to be algorithmic and efficient. This implies the following result on the longest cycle problem:

Proposition 2 *If a 3-connected graph contains a simple cycle of length L then we can find a simple cycle of length*

$$\Omega\left(\left(\frac{\log L}{\log\log L}\right)^2\right)$$

in polynomial time.

This gives the performance ratio $O\big(|V|(\log\log|V|/\log|V|)^2\big)$ for the longest cycle problem in 3-connected graphs. Note that for 3-connected cubic graphs, [4] show a consiberably better bound.

Acknowledgement

We thank Andrzej Lingas for bringing [9] to our attention, and Gerth Stølting Brodal for commenting on a previous version of this paper.

References

1. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(2):844–856, 1995.
2. H. L. Bodlaender. On linear time minor tests with depth-first search. *Journal of Algorithms*, 14(1):1–23, 1993.
3. J. A. Bondy and S. C. Locke. Relative length of paths and cycles in 3-connected graphs. *Discrete Mathematics*, 33:111–122, 1981.
4. T. Feder, R. Motwani, and C. S. Subi. Finding long paths and cycles in sparse Hamiltonian graphs. In *Proc. 32th STOC*, pages 524–529. ACM, 2000.
5. M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.
6. D. Karger, R. Motwani, and G.D.S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997.
7. B. Monien. How to find long paths efficiently. *Annals of Discrete Mathematics*, 25:239–254, 1985.
8. C. H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the V–C dimension. *Journal of Computer and Systems Sciences*, 53(2):161–170, 1996.
9. S. Vishwanathan. An approximation algorithm for finding a long path in Hamiltonian graphs. In *Proc. 11th SODA*, pages 680–685, 2000.