Approximating the Longest Cycle Problem in Sparse Graphs

TOMÁS FEDER[*] RAJEEV MOTWANI[†] CARLOS SUBI[‡]

January 30, 2002

Abstract

We consider the problem of finding long paths and cycles in Hamiltonian graphs. The focus of our work is on sparse graphs, e.g., cubic graphs, that satisfy some property known to hold for Hamiltonian graphs, e.g., k-cyclability.

We first consider the problem of finding long cycles in 3-connected cubic graphs whose edges have weights $w_i \geq 0$. We find cycles of weight at least $(\sum w_i^a)^{\frac{1}{a}}$ for $a = \log_2 3$. Based on this result, we develop an algorithm for finding a cycle of length at least $m^{(\log_3 2)/2}$ in 3-cyclable graphs with vertices of degree at most 3. As a corollary of this result, for arbitrary graphs with vertices of degree at most 3 that have a cycle of length l (or more generally a 3-cyclable minor with degrees at most 3 and with l edges), we find a cycle of length at least $l^{(\log_3 2)/2}$.

We consider the graph property of *1-toughness* that is common to Hamiltonian graphs and 3-connected cubic graphs, and try to determine if 1-toughness implies the existence of long cycles. We show that 2-connectivity and 1-toughness, for constant degree graphs, may give cycles that are only of logarithmic length. However, we exhibit a class of 3-connected 1-tough graphs with degrees up to 6 where we can find cycles of length at least $m^{\log_3 2}/2$.

1 Introduction

Over the last decade, there has been tremendous progress in the area of approximation algorithms. 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 $n/\text{polylog}(n)$, and there are no hardness of approximation results that explain this situation. This is true even for the problem of finding long paths and cycles in bounded-degree Hamiltonian graphs, and indeed it has been conjectured that this special case is already very hard to approximate.

We examine the following question: If a graph has a Hamiltonian cycle, what is the longest cycle that can be found in polynomial time? Our approach is to select properties of Hamiltonian graphs, and use them to construct long cycles. For instance, Hamiltonian graphs are k-cyclable for all k, i.e., any k vertices lie on a common cycle. For 3-cyclable graphs with vertices of degree at most 3, we find a cycle of length at least $m^{(\log_3 2)/2}$. Before describing these and other results in greater detail, we briefly review the previous work in this area.

[*]E-mail: tomas@theory.stanford.edu.

[†]Department of Computer Science, Stanford University, CA 94305. E-mail: rajeev@cs.stanford.edu. Supported by ARO MURI Grant DAAH04–96–1–0007 and NSF Grant IIS–9811904.

[‡]E-mail: carlossubi@genmagic.com.

Previous Work. The first approximation algorithm was due to Monien [9], who gave a polynomial-time algorithm for finding paths of length $\Omega(\log n/\log\log n)$ in Hamiltonian graphs. Furer and Raghavachari [6] and Karger, Motwani, and Ramkumar [8] improved this to finding paths of length $\Omega(\log n)$ in Hamiltonian graphs. Karger et al also pointed out that the Hamiltonian cycle problem appears difficult even in *cubic* graphs and devised an algorithm for finding paths of length $\Omega(\sqrt{n}\log n)$ in *random* cubic Hamiltonian graphs. Alon, Yuster, and Zwick [1] presented an algorithm for finding paths of length $c\log n$, for any constant c, as long as the graph had such a path (no assumption of Hamiltonicity). A recent result of Vishwanathan [12] gives an algorithm for finding paths of length $\Omega((\log n/\log\log n)^2)$ in Hamiltonian graphs.

The situation from the point of view of hardness results is equally dismal. Karger, Motwani, and Ramkumar [8] showed that unless P = NP, it is impossible to find paths of length $n - n^{\epsilon}$ in Hamiltonian graphs for any ϵ. They also established that there is no constant-factor approximation algorithm for the longest path problem (unless P = NP), and that if there is a polynomial-time algorithm with ratio $2^{O(\log^{1-\epsilon} n)}$ then NP $\subseteq$ DTIME$(2^{O(\log^{1/\epsilon} n)})$. They extend the latter result to bounded-degree graphs, and conjecture that the problem is as hard even in bounded-degree Hamiltonian graphs. More recent work of Bazgan, Santha, and Tuza [2] establishes that the longest path problem cannot be approximated within constant factors in *cubic* Hamiltonian graphs. They also showed that for any $\epsilon > 0$, this problem is not approximable within ratio $2^{O(\log^{1-\epsilon} n)}$ unless NP $\subseteq$ DTIME$(2^{O(\log^{1/\epsilon} n)})$.

Overview of Our Results. The focus of our work is on sparse graphs, e.g., cubic graphs, that satisfy some property known to hold for Hamiltonian graphs, e.g., k-cyclability. It was conjectured by Bondy and Simonovits [3] that every 3-connected cubic graph has a cycle of length at least n^c for some constant $c > 0$. This was verified by Jackson [7], who showed that 3-connected cubic graphs have cycles of length at least n^c for $c = \log_2(1+\sqrt{5}) - 1 \approx 0.69$. A key tool in our work is a constructive and generalized version of this result. In Section 2, we consider the problem of finding long cycles in 3-connected cubic graphs whose edges have weights $w_i \geq 0$. We find cycles of weight at least $(\sum w_i^a)^{\frac{1}{a}}$ for $a = \log_2 3$; the exponent $\frac{1}{a} \approx 0.63$ is smaller than that of Jackson, but the proof is much simpler and constructive, and also gives the first result for the weighted case. (We note that the best upper bound known [3] for 3-connected cubic graphs has exponent $\log 8/\log 9 \approx 0.96$.)

Based on our algorithm for weighted cycles in 3-connected cubic graphs, together with an algorithm given in Section 3 for finding a cycle through three given edges in 3-connected cubic graphs when such a cycle exists, we describe in Section 4 an algorithm for finding cycles of length at least $m^{(\log_3 2)/2}$ in 3-cyclable graphs with vertices of degree at most 3. As a corollary, we find a cycle of length at least $l^{(\log_3 2)/2}$ in any graph with vertices of degree at most 3 that has a 3-cyclable minor with degrees at most 3 and with l edges. (Note that a cycle of length l is such a minor.) A similar approximation for the longest path problem in graphs with vertices of degree at most 3 follows from this result.

In Section 5, we consider the graph property of *1-toughness* that is common to Hamiltonian graphs and 3-connected cubic graphs, and try to determine if 1-toughness implies the existence of long cycles. We show that 2-connectivity and 1-toughness, for constant degree graphs, may give cycles that are only of logarithmic length, and only slightly longer paths. However, we exhibit a class of 3-connected 1-tough graphs with degrees up to 6 where we can find cycles of length at least $m^{\log_3 2}/2$.

A preliminary version of this paper has appeared as an extended abstract in a conference proceedings [5].

2 Weighted 3-Connected Cubic Graphs

In this section we establish the following result.

Theorem 1 *Let $G(V,E)$ be an n-vertex, 3-connected, cubic graph. Suppose that each edge $e_i \in E$ has associated with it a weight $w_i \geq 0$. Then, G has a cycle of total weight at least*

$$L_a(w) = \left(\sum {w_i}^a\right)^{\frac{1}{a}}$$

for $a = \log_2 3 \approx 1.58$, and such a cycle can be found in polynomial time. In the unit weight case, this gives a cycle of length at least $\left(\frac{3}{2}n\right)^d$ for $d = \log_3 2 \approx 0.63$.

The rest of this section is devoted to the proof of this theorem. We first prove some relevant lemmas. The bound will follow from combining and dropping weights as follows:

Lemma 1 *Consider a collection of weights $w_i \geq 0$. Suppose we repeatedly perform one of the following operations: either select two weights and replace them with their sum; or, select three weights, drop the smallest one, and replace the other two with their sum. In the end, when only one weight remains, it will be at least $L_a(w) = (\sum {w_i}^a)^{\frac{1}{a}}$ for $a = \log_2 3$.*

PROOF: We show that the quantity $\sum {w_i}^a$ is not reduced by either operation. That is, if $0 \leq x \leq y \leq z$, then $x^a + y^a + z^a \leq (y+z)^a$. We can assume $x = y$. If $z = y$, then $3y^a = (2y)^a$. As z increases, $\frac{d}{dz} 2y^a + z^a = az^{a-1} \leq \frac{d}{dz}(y+z)^a = a(y+z)^{a-1}$.

■

We shall simplify the graph by removing and combining edges, so that the weights are treated as in Lemma 1. We define an operation on *cubic graphs* which selects an edge h, removes it from the graph, and then collapses the two resulting paths of length 2 into single edges. We will refer to this as *the H-transform applied at edge h*; the transformation is illustrated in Figure 1. Note that when we apply the H-transform to a cubic graph G with n vertices and $3n/2$ edges, we will obtain a cubic graph G' with $n-2$ vertices and $(3n/2) - 3$ edges.

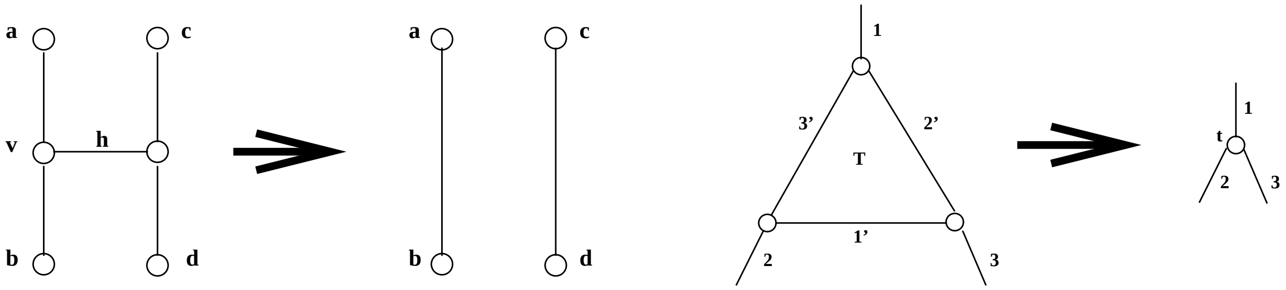

Figure 1: An illustration of the H-transform and the T-transform.

Lemma 2 *Let G be a 3-connected, cubic graph with no triangles. Then, there exists a vertex v such that applying the H-transform at any of the three edges incident on v will result in a new graph G' which is also a 3-connected, cubic graph.*

PROOF: We claim that applying the H-transform at an edge h of a 3-connected, cubic graph G with no triangles, results in a new graph G' which is also 3-connected, cubic if and only if the graph G does not have two other edges e, f such that e, f, h are not all three incident to the same vertex, and removing e, f, h disconnects G.

The only if part is immediate, since if G has two such edges e, f, then in G', the two edges e, f, or possibly edges obtained from them after collapsing paths of length 2 into single edges, will disconnect G', contrary to 3-connectivity.

For the if part, suppose that G' is not 3-connected. Then G' has two edges l, m that disconnect it. If l is a contracted path of length 2 from G, then the middle vertex of this path, on which h is incident, together with m, disconnect G, contrary to the 3-connectivity of G. So h, l, m are edges in G that separate it, and not all three incident to the same vertex, as required.

We can assume that G has three edges e, f, g whose removal disconnects it into two components K and K' having at least two vertices each. (Otherwise, any v will have the desired property, by the preceding claim.) Choose such a configuration with K minimal. Since K is not a triangle, it has some vertex v on which none of e, f, g is incident. Let h be an edge incident on v, drop h and combine each of the two resulting paths of length 2 into single edges, to obtain G'. Suppose G' is not 3-connected. Then by the claim G has two edges l, m such that such that h, l, m are edges in G that separate it, and not all three incident to the same vertex. If either K or K', say K', has exactly one of h, l, m in it, say l, then l would have to separate one of e, f, g from the other two, say e, but then e, l would separate G, contradicting 3-connectivity. So we can assume that none of h, l, m is in K'. But this contradicts the minimality of K.

■

While Lemma 2 assumed the absence of triangles, we must be able to handle triangles. To this end, we define an operation for removing a triangle $T = \{u, v, w\}$ which involves replacing the three vertices in T by a "super-vertex" t whose three incident edges are the three edges incident on u, v, and w that were *not* a part of the triangle T (see Figure 1). It is obvious that applying this transformation to a cubic graph G will result in a cubic graph G'. Moreover, since any two edges separating G' would also separate G, this transformation also preserves 3-connectivity.

Lemma 3 *Let $G \neq K_4$ be a 3-connected, cubic graph having at least one triangle T. If T is replaced by a super-vertex, then the resulting graph G' is also 3-connected and cubic.*

We now provide the proof of Theorem 1. Let G be a 3-connected cubic graph. If G has no triangle, then we select v as in Lemma 2, and if the three edges incident on v are 1, 2, and 3 of weights $w_1 \leq w_2 \leq w_3$, we remove edge 1 and combine edges 2 and 3. (The two edges at the other end of 1 also get combined.) Notice that the combination of weights is as in Lemma 1.

If $G \neq K_4$ has triangles, we can replace a triangle with a super-vertex as in Lemma 3. We must decide what to do with the weights inside the triangle. For a triangle T with incident edges 1, 2, and 3, we can label the edges of T with $1'$, $2'$, and $3'$, where if 1 was incident on a vertex of T, then label $1'$ goes to the edge joining the other two vertices of T, and similarly for 2 and 3 (see Figure 1). The weights of $1'$, $2'$, and $3'$ are then added to 1, 2, and 3 respectively. Notice that if 1 is later removed by an application of Lemma 2, then removing also $1'$ gives a path 2, $3'$, $2'$, and 3, thus justifying the way the weights are handled. This works recursively for super-vertices consisting of triangles whose vertices are themselves super-vertices consisting of triangles, and so on.

Finally, if $G = K_4$, then label disjoint pairs of edges $1, 1'$, $2, 2'$, and $3, 3'$. Then removing $1, 1'$ gives a cycle $2, 3', 2', 3$, so we can remove the pair of disjoint edges of least weight. This completes the proof.

3 Cycle through Three Edges

In this section we establish the following result.

Proposition 1 *Let G be a 3-connected, cubic graph, and let e, f, g be three edges in it. Then G has a cycle through e, f, g, if and only if removing e, f, g, does not disconnect G. Such a cycle can be found in polynomial time.*

If removing e, f, g, disconnects G into two components K and K', then the cycle would have to go between K and K' an even number of times, and so could not include all three edges. This proves the only if part.

The proof of the if part is based on reducing the graph to a K_4 by repeatedly applying the H-transform and the T-transform. If e, f, g, are three edges in K_4 that do not disconnect it, then they are not all three incident to the same vertex, so they either form a triangle or a path of length 3 that can be completed to a cycle of length 4. This produces the desired cycle.

We observe that we can find a cycle through two given edges l, m in G, by selecting an endpoint u of l and an endpoint v of m, finding three disjoint paths from u to v, and observing that some two of these three paths form the desired cycle.

We denote by G^* the graph obtained from G by removing e, f, g. By assumption G^* is connected.

Suppose first that G has no triangles. Then we can find a vertex v such that applying the H transform at any of the three edges h_1, h_2, h_3 incident on v will result in a new graph G' which is also a 3-connected, cubic graph. There are then three cases.

Case 1: None of e, f, g is incident on v.
If removing h_1 from G^* does not disconnect G^*, then applying the H-transform at h_1 will result in a graph G' such that the three edges e, f, g, or edges obtained from them by contracting paths of length 2, do not disconnect it. Otherwise removing h_1 from G^* disconnects it, creating in particular a component C_1 not containing v. Similarly, removing h_2 or h_3 from G^* creates components C_2, C_3 not containing v, respectively. Since G is 3-connected, the three edges e, f, g must go from C_1 to C_2, from C_1 to C_3, and from C_2 to C_3, respectively. But then h_1, e, f separate G, and we could not have applied the H-transform at h_1, unless C_1 consists of a single vertex. In this last case e, f get collapsed to a single edge l, so that we just need to find a cycle through l, g in G', and this can be done as observed above.

Case 2: Precisely one of e, f, g is incident on v.
Say $g = h_3$. As before, if removing h_1 from G^* does not disconnect G^*, then we can apply the H-transform at h_1. Otherwise removing h_1 or h_2 from G^* creates components C_1, C_2 not containing v, respectively. The edge $g = h_3$ goes from v to either C_1 or C_2, say C_2. The edges e, f must go from C_1 to C_2, since G is 3-connected. But then h_1, e, f separate G, and as before we would not have applied the H-transform at h_1, unless C_1 consists of a single vertex, in which case e, f get collapsed to a single edge l and we can find a cycle through l, g in G'.

Case 3: Precisely two of e, f, g are incident on v.
Say $f = h_2$ and $g = h_3$. Then the H-transform at h_1 collapses f, g to a single edge l, and we can find a cycle through l, e in G'.

It is not possible that all three of e, f, g are incident on v, since they would disconnect G. This completes the study of the H-transform. We consider next the T-transform, with four cases.

Case 1: None of e, f, g are inside the triangle T.
Then after replacing T with a super-vertex t to obtain G', the three edges e, f, g will still not disconnect G'.

Case 2: Precisely one of e, f, g is inside the triangle T.
Say e is in T. Then after replacing T with a super-vertex t, it is sufficient to find a cycle in G' through f, g and t; we can always ensure that e gets visited inside T. If either f or g is incident on t, then we just need to find a cycle through f, g, and this is done as observed before. Otherwise, to ensure that t is visited, we can let e' be one of the three edges $1, 2, 3$ incident on T, and find a cycle through e', f, g in G', provided that these three edges do not disconnect G'. If they do, then each of the edges $1, 2, 3$ disconnects G^* (even if e is included in G^*), creating components C_1, C_2, C_3 not containing T, respectively. Then adding two edges f, g to G^* to obtain G will not make it 3-connected, a contradiction.

Case 3: Precisely two of e, f, g are inside the triangle T.
Say e, f are the edges $1', 2'$ in T. Let e', f' be the two corresponding edges $1, 2$ incident to T. Clearly e', f', g disconnect G if and only if e, f, g disconnect G, so we can assume they do not. A cycle entering and leaving T at e', f' can be made to go through e, f inside T; we can thus carry out the T-transform.

Case 4: The three edges e, f, g constitute the triangle T.
Then T is the desired cycle.

This completes the proof.

4 3-Cyclable Graphs

A graph is said G to be *k-cyclable* if for any choice of k vertices, there is a cycle in G going through all k of them. Clearly, if a graph has a Hamiltonian cycle, it is k-cyclable for all k. If k is a constant, then given k vertices, one can find in polynomial time a cycle going through all k of them, if one exists: this requires finding k disjoint paths connecting the vertices, and a polynomial-time algorithm for the disjoint paths problem with a constant number of paths was given by Robertson and Seymour [10, 11] in their work on graph minors. For 3-cyclable graphs with vertices of degree at most 3, however, finding a cycle through three vertices will only require the result in Section 3. Note that in particular, 3-connected cubic graphs are 3-cyclable.

In the rest of this section, we will establish the following result (and its corollary):

Theorem 2 *Let G be a 3-cyclable graph with vertices of degree at most 3, and with m edges. Then we can find, in polynomial time, a cycle in G of length at least $m^{\frac{1}{2a}}$, where $a = \log_2 3$.*

Proof: We will need three properties of the 3-cyclable graph G.

1. G is 2-connected, since it is 2-cyclable.

2. If there is an edge $f = uv$ such that the subgraph $G - \{u, v\}$ on the vertices other than u, v has two or more connected components, then we can remove the edge f from G and still have a 3-cyclable graph. For if a cycle uses f, it could use instead a path from u to v in one of the connected components of $G - \{u, v\}$. (The justification for counting edges so removed will be given in the following corollary.)

3. For any two vertices u, v, the graph $G - \{u, v\}$ on the remaining vertices has at most two connected components. For if it has at least three components, then three vertices in the three components cannot be on an cycle, contrary to 3-cyclability.

Suppose there are two edges e_1, e_2 that separate G into 2 components K and K'. If K has no edge separating the two vertices incident on e_1, e_2, then we say that K is a *2-component*. The *graph* of a 2-component K is obtained from K by connecting the two vertices incident on e_1, e_2 by an edge e, and replacing all maximal 2-components contained in it by single super-vertices. The graph of a 2-component is a 3-connected graph with edges replaced by paths. This follows from the fact that if we replace all maximal paths whose vertices other than the starting and ending vertices have degree 2, by single edges, and add the edge e, then there will be no parallel edges, by properties (1), (2) and (3). The resulting graph must be 3-connected, since otherwise we could identify a new 2-component contained in it, contrary to maximality. At the top level of G, if G is 3-connected, apply Theorem 1. Otherwise, select a 2-component K, adjacent via two edges e_1, e_2 to its complement K', which has some $r \geq 0$ edges each separating the two vertices incident on e_1, e_2, thus decomposing K' into $(r+1)$ 2-components; so G is a cycle of $(r+2)$ 2-components.

So suppose G is not 3-connected, and perform the above decomposition into 2-components. The 2-components so selected have a tree structure: at the top level, we have a cycle of 2-components, which are viewed as the children of the root; for a 2-component in this tree, its children are the maximal 2-components contained in it.

We assign a weight w_i to each 2-component in the tree, so that if the 2-component has m_i edges, then $w_i \geq {m_i}^d$, where $d = \frac{1}{a} = \log_3 2$. We do this inductively, starting at the leaves. So suppose for a given 2-component, we have assigned weights to the maximal 2-components contained in it. The graph of the 2-component is a 3-connected graph with edges replaced by paths. To every such path, assign a weight equal to the sum of the weights of the super-vertices in it, plus the number of edges connecting these super-vertices. The special edge e is given weight 0. Apply then Theorem 1 to the 3-connected graph, obtaining a cycle of weight $w \geq (\sum w_i^a)^{\frac{1}{a}} \geq (\sum m_i)^{\frac{1}{a}} = m^d$. Assign weight w to the 2-component, completing the induction. At the top level, we have a cycle of 2-components, so we add their weights plus the number of edges in the cycle.

Unfortunately, the cycles that we selected for each 2-component to define the weights of the 2-component do not necessarily connect, since the cycle for a 2-component may not go through the special edge e that links it to its parent in the tree. We shall define cycles that do go through e so that if a 2-component has weight w, then the cycle through e visits at least $w^{\frac{1}{2}} \geq m^{\frac{1}{2a}}$ edges other than e, and this will complete the proof.

Consider a 2-component whose weight w was defined by finding a cycle C of weights w_i, and suppose for each i we have found a path inside the corresponding maximal 2-component of length at least ${w_i}^{\frac{1}{2}}$. We write $w_i = \epsilon_i w$, with $\sum \epsilon_i = 1$. Clearly, if C goes through e, we can just select C and have a cycle through e of lenght at least $\sum {w_i}^{\frac{1}{2}} \geq w^{\frac{1}{2}}$, since $\sum {\epsilon_i}^{\frac{1}{2}} \geq 1$ where the sums are over the weights in C.

Suppose C does not go through e. By 3-connectivity, we can join an endpoint of e to C with three disjoint paths, and in particular obtain two disjoint paths joining the two endpoints of e to C, respectively. The two disjoint paths enter C at two distinct vertices, thus decomposing the weights of C into two subsets S and T delimited by these two vertices. If $\sum {\epsilon_i}^{\frac{1}{2}} \geq 1$ when the sum is taken over the weights in either S or T, then we are done. If this is not the case, then we will show that the largest weight $w_1 = \epsilon_1 w$ in S and the largest weight $w_2 = \epsilon_2 w$ in T satisfy ${\epsilon_1}^{\frac{1}{2}} + {\epsilon_2}^{\frac{1}{2}} > 1$. If $w_2 = 1$, then the cycle through e and S goes through the edge of weight w_1 and at least one other edge (of weight at least $w_2 = 1$). So we can assume $w_1, w_2 > 1$. It will then be sufficient to make the cycle through e go through the edges corresponding to the two weights w_1, w_2. Each of these two edges is a path with a maximal 2-component in it, then we can select two representative vertices from the two maximal 2-components, plus a vertex from the complement of the 2-component to represent edge e; a cycle through these three vertices must exist by 3-cyclability; this implies that there is a cycle through the three selected edges in the graph of the 2-component, and since this graph is a 3-connected, cubic graph, such a cycle can be found in polynomial time by Proposition 1.

It remains to prove the claim about the sum of the $\epsilon_i^{\frac{1}{2}}$. Notice that if $w_i \geq w_{i'}$ and $0 \leq \delta \leq w_{i'}$, then $w_i^{\frac{1}{2}} + w_{i'}^{\frac{1}{2}} \geq (w_i + \delta)^{\frac{1}{2}} + (w_{i'} - \delta)^{\frac{1}{2}}$. For the argument, we modify the weights by choosing the appropriate values δ, as follows. We can ensure that S will have at most one non-zero weight smaller than w_1, and similarly that T will have at most one non-zero weight smaller than w_2. Thus S has $s \geq 1$ weights equal to w_1 and one weight $0 \leq w_1' < w_1$; similarly T has $t \geq 1$ weights equal to w_2 and one weight $0 \leq w_2' < w_2$.

We thus have $s\epsilon_1 + \epsilon_1' + t\epsilon_2 + \epsilon_2' = 1$, with $s\epsilon_1^{\frac{1}{2}} + {\epsilon_1'}^{\frac{1}{2}} < 1$ and $t\epsilon_2^{\frac{1}{2}} + {\epsilon_2'}^{\frac{1}{2}} < 1$. We write $\epsilon_1' = \lambda\epsilon_1$ with $0 \leq \lambda < 1$, and let $s' = s + \lambda$. Similarly, we write $\epsilon_2' = \mu\epsilon_2$ with $0 \leq \mu < 1$, and let $t' = t + \mu$. Then $s'\epsilon_1 + t'\epsilon_2 = 1$. Also, $s'\epsilon_1^{\frac{1}{2}} \leq (s + \lambda^{\frac{1}{2}})\epsilon_1^{\frac{1}{2}} < 1$, and similarly $t'\epsilon_2^{\frac{1}{2}} < 1$. But then $s'\epsilon_1 < \epsilon_1^{\frac{1}{2}}$ and $t'\epsilon_2 < \epsilon_2^{\frac{1}{2}}$, so $\epsilon_1^{\frac{1}{2}} + \epsilon_2^{\frac{1}{2}} > 1$. This completes the proof.

■

Corollary 1 *Let G be a graph with vertices of degree at most 3. Let l be the maximum number of edges in a 3-cyclable minor of G with degrees at most 3. (So l is at least the length of the longest cycle in G.) Then we can find, in polynomial time, a cycle of length at least $l^{(\log_3 2)/2}$.*

PROOF: Note that a 3-cyclable minor K with degrees at most 3 is obtained from a subgraph H of G by skipping some vertices of degree 2 in H, so that an edge in K corresponds to a path with internal vertices of degree 2 in H.

We can assume G is 2-connected. Otherwise some edge f separates G into two components, and the 3-cyclable minor must be contained in one of these two components. We may thus remove f; the problem decomposes into 2-connected subproblems.

The structure of the graph as a tree of 2-components is still the same as before, even if the graph is not 3-cyclable. The only difference is that we cannot avoid parallel edges in the graph of a 2-component. Parallel edges occur only in one new kind of 2-component: Let P_3 be the graph consisting of three parallel edges joining two vertices; the graph of a 2-component may now also be P_3, with one of the three parallel edges being the special edge e linking to the parent in the tree, and one or both of the other two parallel edges f_1, f_2 being a path of vertices corresponding to 2-components that are children of this 2-component in the tree.

We proceed to simplify the graph as follows, from the leaves of the tree, up to the root. For the special P_3 2-component, we remove one of the two parallel edges f_1, f_2, the one corresponding the a subgraph of smaller size. Thus we have no parallel edges anymore, as before.

As we go up the tree, we must also make sure we will be able to obtain, from a cycle in a 2-component, a new cycle going through the special edge e. This is not possible if the two edges f_1 and f_2 with weights w_1 and w_2, together with e, separate the 3-connected graph of the 2-component. All such possible pairs (f_1, f_2) which together with e separate the 3-connected graph must be disjoint. We replace one of f_1, f_2, the one corresponding to a subgraph of smaller size, with a single edge (of weight 1).

If the 3-cyclable minor of size l reaches the root of the tree of 2-components, then the simplified graph will have $m \geq l$ edges, by induction from the leaves up to the root of the tree of 2-components. The reason is that when f_1 or f_2 is removed or reduced to a single edge, the 3-cyclable minor of size l necessarily does have vertices in one of the two subgraphs corresponding to f_1 and f_2. We then get a cycle of length at least $m^{(\log_3 2)/2}$ with the previous algorithm.

It may seem that one should not count the edges removed to satisfy property 2 above. However, when a single edge f_1 is removed from P_3, at least one edge corresponding to e is later added, and one can count both edges since $x^{\frac{1}{2a}} + 1 \geq (x+2)^{\frac{1}{2a}}$ for $a, x \geq 1$.

If the 3-cyclable minor of size l does not reach the root of the tree of 2-components, consider the 2-component closest to the root that it reaches. The 3-cyclable minor does not have vertices corresponding to the special edge e connecting this 2-component to its parent. On this 2-component, we do not simplify the graph for pairs of edges f_1, f_2 which together with e disconnect the graph of the 2-component (neither in the case of P_3 nor in the case of a 3-connected graph); we assign to e weight 1. As before, we have $m \geq l$ edges by induction from the leaves of the tree up to this 2-component, and get a cycle of length at least $m^{(\log_3 2)/2}$ with the previous algorithm.

■

An algorithm that approximates the longest cycle problem can be used to approximate the longest path problem by guessing and adding the edge that would complete the long path into a cycle, then removing this edge after a long cycle is found and getting back a path. In our case, since adding an edge could create two vertices of degree four, we can instead add the edge between two new vertices inserted in the middle of two existing edges.

5 Long Cycles and Toughness

A graph G is said to be *1-tough* if it has the following property: *For every set S of vertices, the graph $G - S$ on the remaining vertices has at most $|S|$ components.* There has been interest in devising algorithms for long paths/cycles in graphs that are 1-tough [4, 8, 12], because Hamiltonian and weakly Hamiltonian graphs are 1-tough. There has been some work that shows that for graphs with large degree sums, 1-toughness implies the existence of long cycles. In the above results, 3-connected cubic graphs are 1-tough, since every component is delimited by at least three separating vertices, while each separating vertex delimits at most three components. The graphs in Theorem 2 are not necessarily 1-tough. However, all 2-connected graphs with vertices of degree at most 3 can be made 1-tough without significantly altering the structure of the graph, namely by replacing each vertex of degree 3 with a triangle; then each vertex of S delimits at most two components of $G - S$, and each component of $G - S$ is delimited by at least two vertices of S, giving 1-toughness.

We show that there are 1-tough, cubic graphs that are only 2-connected, and where the length of the longest cycle is only logarithmic in the number of vertices, thus giving an exponential gap with respect to the 3-connected case, for cubic graphs.

Theorem 3 *There are arbitrarily large 1-tough, 2-connected, k-regular graphs, for $k \geq 3$, whose longest cycle has length $O(k \log_k n)$, and whose longest path has length $O(k \log_k^2 n)$. The graphs for $k = 3$ are planar.*

Proof: We give an inductive construction, and for the induction to go through we will need a condition that is slightly stronger than 1-toughness. We say that a graph with some chosen special edges is extra 1-tough if, when counting the number of components of $G - S$, we also count as components any special edges both of whose endpoints are in S, the total is still at most $|S|$. Notice that K_{k+1}, with all edges incident to one chosen vertex v as special, is extra 1-tough.

Suppose we have constructed a 2-connected k-regular graph G with some special edges, starting with K_{k+1} as just described, and this graph is extra 1-tough. We select a special edge e and replace it with $H = K_{k+1}$ as follows. Break one of the special edges (v, w) in H, and replace e with H by connecting v and w to the vertices to which e is incident. The resulting graph is still 2-connected, k-regular, and extra 1-tough. To see this last property, notice that the t vertices in S not in H still give at most t components, while the $|S| - t$ vertices in H subdivide the component corresponding to e as follows. The vertices v and w may each create at most one extra component, while the remaining vertices in H create no component except for the special edge (v, u) for each u chosen, if v is also chosen. This proves extra 1-toughness.

We apply this construction inductively. Starting with K_{k+1}, we replace the k special edges with H, where each copy of H brings in $k-1$ special edges. We then replace the $k(k-1)$ special edges with a copy of H again, so we now have $k(k-1)^2$ special edges, and continue in this fashion. The resulting graph will have the structure of two balanced trees of degree k connected at the leaves, where one tree corresponds to the subdivision into special edges at v, while the other three brings the special edges back together into the remaining vertices of $H - v$. The trees have height $O(\log_k n)$, and the longest cycle will go from one root to the other and back, visiting the k vertices in $H - v$ for each vertex in the second tree. The cycle will thus have length at most $O(k \log_k n)$. The longest path has a middle section that goes from one root v_1 to the other root v_2, then back to a child of v_1, then back to a child of a child of v_2, and so on. This gives the $O(k \log_k^2 n)$ bound for paths.

■

It remains open whether 3-connected graphs always have cycles or paths of length n^c for some $c < 1$, by also assuming the graph to be planar or of bounded degree. It was shown by Karger, Motwani, and Ramkumar [8] that there are 3-connected graphs that are 1-tough, have a 2-factor, and are 3-cyclable, with longest path of length $O(\log n)$; the examples, however, have two vertices adjacent to all vertices.

Theorem 4 *Let G be a 3-connected graph with maximum degree at most 6, such that every vertex of degree 4 belongs to at least one triangle, every vertex of degree 5 belongs to at least two edge-disjoint triangles, and every vertex of degree 6 belongs to at least three edge-disjoint triangles. Then G is 1-tough. Let T be a minimal set of triangles satisfying the conditions on vertices of degree 4, 5, and 6. If no two vertices of degree 5 or 6 are adjacent, then no triangle in T shares two different edges with two other triangles of T. If this condition on the triangles of T holds, then we can find in polynomial time a cycle of length $r^{\log_3 2}$, where r is the number of edges of G minus 2/3 the number of edges belonging to triangles of T.*

Proof: The triangles that are assumed to exist for vertices of degree 4, 5, and 6, imply that the neighborhood of every vertex contains at most three independent vertices. As a consequence, if we select a set S of vertices, each vertex of S will be adjacent to at most three components of $G - S$. On the other hand, each component of $G - S$ is delimited by at least three vertices of S, by 3-connectivity. It follows that $G - S$ has at most $|S|$ components, that is, G is 1-tough.

Suppose now that the vertices of degree 5 or 6 are independent; thus their neighbors are of degree 3 or 4. Since degree 4 vertices require only one triangle, it follows that a vertex of degree 6 will not have any triangle in T other than the three edge-disjoint triangles required to be in T. If a triangle t_1 in T shares two distinct edges with two triangles t_2 and t_3 in T, then some vertex v of t_1 must be of degree 5, say it is a vertex v common to t_1 and t_2. If the vertex of t_2 not on t_1 and the vertex of t_3 not on t_1 are the same, then we can drop t_1 or t_2 from T. Otherwise, either v also belongs to t_3, in which case t_2 and t_3 can be used as the two edge-disjoint triangles for v and we can drop t_1, or v does not belong to t_3. In this last case, either the triangle t_4 which is edge-disjoint from t_1 is also edge-disjoint from t_2, in which case we can drop t_1, or it shares an edge with t_2, in which case we can drop t_2.

So we assume that every triangle of T shares at most one edge with other triangles of T. We first consider the case where all triangles of T are edge-disjoint. It is known that the operation of replacing a vertex u of degree at least 4 with two adjacent vertices v and w having each at least two neighbors, the neighborhoods of v and w being disjoint and giving the neighborhood of u when combined, preserves 3-connectivity. We give weight 1/2 to the edges of T and weight 1 to the remaining edges. We replace each vertex of degree 4 with two vertices as just explained, with one of the two vertices incident to a triangle in T, and give weight 0 to the edge joining the two new vertices. For degree 5 vertices, we replace them with a path of length 2 of weight 0 edges, and distribute the adjacencies to make the new vertices have degree 3, in such a way that the first and last vertex of the weight 0 path are incident to two edge-disjoint triangles in T. For degree 6 vertices, we replace them with a star of degree 3 consisting of three weight 0 edges, so that the three leaves are incident to three edge-disjoint triangles in T. By Theorem 1, we can find a cycle of weight at least $r^{\log_3 2}$ in the resulting cubic graph. This cycle, however, might visit some vertex of the original graph more than once. But then, it must visit two edges of the same selected triangle in T, and these two edges of weight 1/2 can be replaced with a single direct edge, so we avoid such conflicts and obtain a cycle of length at least $r^{\log_3 2}$ in the original graph.

Notice that a vertex of degree 6 belongs only to three edge-disjoint triangles in T, while a vertex of degree 5 can have at most two triangles sharing an edge in T. So only two or three triangles can share an edge e. When this happens, we cannot preserve all two or three triangles when introducing weight 0 edges, but we can preserve one of them, say t, and introduce single weight 0 edges at the endpoints of e separating t from the other one or two triangles that used to share e. Notice that if there are two separated triangles, they cannot both give a conflict at their vertices not on e, so we can restrict attention to the case of only one separated triangle. We thus have a gadget with five edges of weight 1/2, at most two edges of weight 0, and at most four incoming edges. This gadget is traversed by at most one or two paths. We can verify that these paths can be modified within the gadget so as to avoid conflicts, and in such a way that the resulting path lengths are at least the sum of the original path weights. This completes the proof.

■

A graph is said to be α-tough if for every set S of vertices that leaves a graph $G - S$ on the remaining vertices with $k \geq 2$ connected components, we have $|S| \geq \alpha k$. It has been conjectured that there is a constant α such that all α-tough graphs have a Hamiltonian cycle. It is known that it is not enough to take $\alpha = 1.5$: Consider a 3-connected cubic graph that does not have a Hamiltonian cycle, and replace every vertex with a triangle; since every vertex of the resulting graph has at most two independent neighbors, it follows that every vertex of S is adjacent to at most two components of $G - S$, while every component of $G - S$ is delimited by at least three vertices of S by 3-connectivity, giving 1.5-toughness. In this direction, we would like to know, if $(1 + \epsilon)$-tough graphs with $\epsilon > 0$, which are in particular 3-connected, always have cycles of length at least n^c for some $c = c(\epsilon) < 1$.

Acknowledgements We are grateful to Sudipto Guha for several helpful discussions.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing,* 1995, pp. 326–335.

[2] C. Bazgan, M. Santha, and Z. Tuza. On the Approximability of Finding A(nother) Hamiltonian Cycle in Cubic Hamiltonian Graphs. In: *Proceedings of the Symposium on Theoretical Aspects of Computer Science,* 1998, pp. 276–286.

[3] J.A. Bondy and M. Simonovits. Longest cycles in 3-connected cubic graphs. *Canadian Journal of Mathematics* 32 (1980): 987–992.

[4] V. Chvatal. *Hamiltonian cycles.* In: *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* E.L. Lawler et al (Eds.), 1985, pp. 402–430.

[5] T. Feder, R. Motwani, C. and Subi. Finding Long Paths and Cycles in Sparse Hamiltonian Graphs. In: *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing,* 2000, pp. 524-529.

[6] M. Fürer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In: *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms,* 1992, pp. 317–324.

[7] B. Jackson. Longest Cycles in 3-Connected Cubic Graphs. *Journal of Combinatorial Theory (Series B)* 41 (1986): 17–26.

[8] D. Karger, R. Motwani, and G.D.S. Ramkumar. On Approximating the Longest Path in a Graph. *Algorithmica* 18 (1997): 82–98.

[9] B. Monien. How to find long paths efficiently. *Annals of Discrete Mathematics* 25 (1985): 239–254.

[10] N. Robertson and P.D. Seymour. An outline of a disjoint paths algorithm. In: *Paths, Flows, and VLSI-Layout*, N. Korte et al (Eds.), Berlin: Springer-Verlag, 1990, pp. 267–292.

[11] N. Robertson and P.D. Seymour. Graph Minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory (Series B)* 63 (1995): 65–110.

[12] S. Vishwanathan. An Approximation Algorithm for Finding a Long Path in Hamiltonian Graphs. In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms,* 2000, pp. 680–685.