Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: http://www.tandfonline.com/loi/goms20

An approximation algorithm for the longest path problem in solid grid graphs

Asghar Asgharian Sardroud & Alireza Bagheri

To cite this article: Asghar Asgharian Sardroud & Alireza Bagheri (2016): An approximation algorithm for the longest path problem in solid grid graphs, Optimization Methods and Software, DOI: 10.1080/10556788.2015.1130130

To link to this article: http://dx.doi.org/10.1080/10556788.2015.1130130

Optimization Methods & Software, 2016
http://dx.doi.org/10.1080/10556788.2015.1130130

Taylor & Francis
Taylor & Francis Group

An approximation algorithm for the longest path problem in solid grid graphs

Asghar Asgharian Sardroud and Alireza Bagheri*

Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran

(*Received 1 May 2015; accepted 6 December 2015*)

The longest path problem is a well-known NP-hard problem which can be used to model and solve some optimization problems. There are only a few classes of graphs for which this problem can be solved polynomially. In addition, the results show that the problem is hard to approximate within any reasonable factor for general graphs. We introduce a polynomial-time $\frac{2}{3}$-approximation algorithm for the longest path problem in solid grid graphs. Our algorithm can also approximate the longest path between any two given boundary vertices of a solid grid graph within the same approximation factor.

Keywords: longest path; Hamiltonian path; approximation algorithm; solid grid graph

AMS Subject Classification: 68R10; 05C85

1. Introduction

The well-known shortest path and longest path problems can be used as models to solve many types of optimization problems. Although, the shortest path problem can be solved efficiently, even with extra restrictions (see as examples [7,9,14]), the longest path and longest cycle problems are well-known NP-hard problems. There is a variety of results on approximating these problems showing that they are hard to approximate for general graphs. For example, assuming that $P \neq NP$, it has been shown that there is no polynomial-time constant-factor approximation algorithm for the longest path problem and neither it is possible to find a path of length $n - n^{\epsilon}$ in polynomial time in Hamiltonian graphs [18]. The color coding technique, introduced by Alon *et al.* [1], is one of the first approximation algorithms for these problems which can find paths and cycles of length $\log n$. Later, Björklund *et al.* introduced another technique with better approximation ratio, i.e. $O(n \log\log n/\log^2 n)$, for finding long paths [4,12]. To our knowledge, the result of Gabow [11], which can find a cycle or path of length $\exp(\Omega(\sqrt{\log l/\log\log l}))$ in graphs with the longest cycle of length l, is the best polynomial-time approximation algorithm for finding long cycles. Additionally, the results show that the problems are hard to approximate even in bounded-degree and Hamiltonian graphs [8,10]. These problems are even harder to approximate in the case of directed graphs as shown in [5]. For more related results on approximation algorithms for general graphs see [3,13,23].

*Corresponding author. Email: ar_bagheri@aut.ac.ir

There are a few classes of graphs for which the longest path or the longest cycle problems are polynomial-time solvable [6,15,16,19,20,24]. Grid graphs are vertex-induced subgraphs of the infinite grid G^{∞} and have wide applications in science and engineering such as VLSI design, CAD/CAM, computer graphics, graph drawing and information visualization, and robotics (see as examples [21,22]). In the case of grid graphs, first, Itai *et al.* [17] showed that the Hamiltonian path and cycle problems are NP-complete.

Later, Umans *et al.* showed that the Hamiltonicity of solid grid graphs, i.e. the grid graphs in which each internal face has length four, is decidable in polynomial time [25]. To our knowledge, the only two results on the problems of longest cycle and longest path for grid graphs is by Zhang and Liu [26] which gives a $\frac{5}{6}$-approximation algorithm for finding the longest path in grid graphs that have square-free cycle covers, and the result of [2] which gives an algorithm that can find a cycle of length at least $2n/3 + 1$ containing a given boundary edge in a given 2-connected n-vertex solid grid graph G.

In this paper, we introduce a polynomial-time constant-factor approximation algorithm for the longest path problem in solid grid graphs. We note that, comparing to the algorithm of Zhang and Liu [26], our algorithm covers a broader class of graphs, i.e. all solid grid graphs. The majority of the paper deals with the problem of approximating the longest path between two given boundary vertices s and t of a 2-connected solid grid graph G, that is, ALP(G, s, t). The main idea of our algorithm is to merge the boundary path B between s and t by a long cycle in $G \setminus B$ to construct a path containing more than $\frac{2}{3}$ of the vertices in G. For finding long cycles, we use the $\frac{2}{3}$-factor approximation algorithm for the longest cycle problem presented in [2].

The organization of the remainder of the paper is as follows. In Section 2, we present the terminology and some preliminary concepts. Section 3, contains the definition of forbidden problems and their solutions. The problem ALP(G, s, t) is *forbidden* if the n-vertex graph G does not contain any path of length at least $\frac{2}{3}$ between s and t. In Section 4, we introduce three types of split operations for splitting the ALP(G, s, t), and the longest path approximation algorithm is given in Section 5. Finally, Section 6 contains our conclusion.

2. Preliminaries

This section contains some necessary concepts and definitions used throughout the paper. Let G be a solid grid graph, that is, a vertex-induced subgraph of the infinite integer grid whose vertices are the integer coordinated points of the plane and has an edge between any two vertices of distance one. In addition, considering their natural embedding on the integer grid, we assume that solid grid graphs are plane graphs. The vertices of G adjacent to the outer face are called *boundary vertices*, and the set of boundary vertices of G forms its *boundary*. We call a pair of edges of G *parallel edges* if they are not incident to a common vertex, but are both adjacent to the same inner face. When G' is a subgraph of G, we use the notation $G \setminus G'$ to refer to the grid graph obtained from G after removing all the vertices of G' together with their incident edges. It is easy to show that $G \setminus G'$ is also a solid grid graph if the boundaries of G and G' are not disjoint. A connected graph G is *2-connected* if it contains no *cut vertex*, i.e. a vertex whose removal increases the number of connected components of G. Furthermore, two vertices of a 2-connected graph G are called cutting pair if their removal disconnects G. We define ALP(G, s, t) to be the problem of finding a path between the boundary vertices s and t of the 2-connected solid grid graph G, of length at least $\frac{2}{3}$ of the length of the longest such path. In the next sections, first, we give an algorithm for ALP(G, s, t), and in Section 5, we show that such an algorithm can be extended to approximate the general longest path problem in solid grid graphs. Our algorithm for solving ALP(G, s, t) works by merging the boundary path B between s and t by a long cycle

in $G \setminus B$. A path in ALP(G, s, t) is a boundary path if it contains only the boundary vertices of G. When $G \setminus B$ is not 2-connected, it may contain no long-enough cycle. In these cases, our algorithm first uses three types of split operations to convert the problem to subproblems in which $G \setminus B$ is 2-connected. Moreover, in some problems, we can show that the n-vertex graph G does not contain any path of length $2n/3$ or more between s and t. These problems are called *forbidden* problems, and our algorithm should avoid creating forbidden subproblems to ensure that the constructed path contains at least $\frac{2}{3}$ of the vertices of G. In the definition of spilt operations and forbidden problems, we call the vertex v on the path B and s to be a *critical cutting pair* if v and s are a cutting pair and there is no vertex u between s and v on B such that s and u are a cutting pair. In other words, v is a critical cutting pair to s if it is the nearest vertex to t on B which is a cutting pair to s. Analogously, we define a vertex v to be a critical cutting pair to t by swapping the roles of the end-vertices s and t. Because, there exists only two boundary paths between s and t, each of them are a critical cutting pair to at most two vertices of G. Also, we use the notation B_{uv} to denote the subpath of B between its vertices u and v.

3. The forbidden problems

The forbidden problems are the problems in which there is no path of length at least $2n/3$ between s and t in G. More precisely, ALP(G, s, t) is forbidden if it satisfies one of the conditions (F1), (F2), (F3) or (F4) defined in the following. For defining (F1) and (F2), we should first define (C1) and (C2).

(C1): ALP(G, s, t) satisfies (C1) if s and t are a cutting pair, or G has a vertex v such that s and v are a cutting pair and ALP$(G' \cup \{v\}, v, t)$ satisfies (C1), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

Considering the recursive definition of (C1), if ALP(G, s, t) satisfies (C1), there should be an integer $m \geq 1$ and a sequence of vertices $v_0 = s, v_1, \ldots, v_m = t$ such that each two consecutive vertices of the sequence are a cutting pair. For the problem ALP(G, s, t) which satisfies (C1), we define graphs $G_0, G_1, \ldots, G_{m-1}$ and $C_0, C_1, \ldots, C_m$ as follows. Graph G_0 is the same as G, and for $0 \leq i \leq m-2$, C_i is the connected component of $G_i \setminus \{v_i, v_{i+1}\}$ which does not contain t, and G_{i+1} is $G_i \setminus (C_i \cup \{v_i\})$. Furthermore, C_{m-1} and C_m are the connected components of $G_{m-1} \setminus \{v_{m-1}, v_m\}$. Note that, $C_0, C_1, \ldots, C_m$ are non-empty disjoint subgraphs of G by their definitions.

LEMMA 3.1 *If* ALP(G, s, t) *satisfies* (C1), *for each path between s and t in G, there is an $0 \leq i \leq m$ such that the path does not contain any vertex of C_i.*

Proof We prove the lemma by contradiction. Let P be a path between $s = v_0$ and t in G containing at least one vertex of each C_i, $0 \leq i \leq m$. Because v_0 and v_1 are a cutting pair and t is not in C_0, if P contains a vertex of C_0, it should come before v_1 along the path P, and v_1 should precede all the vertices of $G_1 \setminus v_1$ in P. Similarly, if P contains some vertices of C_{m-2}, they should precede v_{m-1} in P, and all the vertices of C_{m-1} and C_m should come after v_{m-1} along P. However, because v_{m-1} and v_m are a cutting pair, P cannot contain vertices from both of C_{m-1} and C_m, which is a contradiction. ■

(F1): ALP(G, s, t) satisfies (F1) if it satisfies (C1) such that all graphs C_i, $0 \leq i \leq m$, contain more than one vertex.

(C2): ALP(G, s, t) satisfies (C2) if G contains two vertices that are a critical cutting pair to s, or G has a vertex v cutting pair to s and ALP$(G' \cup \{v\}, v, t)$ satisfies (C2), in which G' is the connected component of $G \setminus \{s, v\}$ which contains t.

Similarly, considering the recursive definition of (C2), if ALP(G, s, t) satisfies (C2), G should contain a sequence of vertices $v_0 = s, v_1, \ldots, v_m$ such that each two consecutive vertices are a

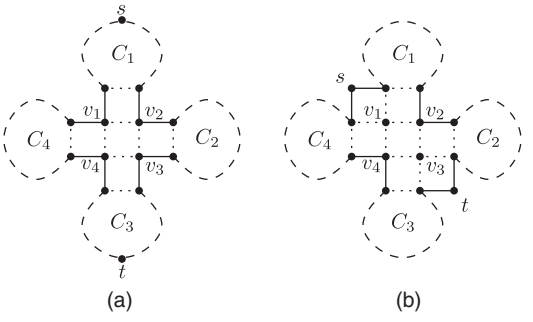

Figure 1. The cases (a) and (b) in which ALP(G, s, t) satisfies, respectively, (F3) and (F4). The dashed lines schematically represent the nonempty subgraphs.

cutting pair and two vertices v_{m+1} and v_{m+2} which are both cutting pairs to v_m, for an integer $m \geq 0$. When ALP(G, s, t) satisfies (C2), we define graphs $G_0, G_1, \ldots, G_m$ and $C_0, C_1, \ldots, C_{m+1}$ as follows. Graph G_0 is the same as G and, for $0 \leq i \leq m-1$, C_i is the connected component of $G_i \setminus \{v_i, v_{i+1}\}$ which does not contain t, and G_{i+1} is $G_i \setminus (C_i \cup \{v_i\})$. In addition, C_m and C_{m+1} are, respectively, the connected components of $G_m \setminus \{v_m, v_{m+1}\}$ and $G_m \setminus \{v_m, v_{m+2}\}$ which do not contain t. Note that, because ALP(G_m, v_m, t) satisfies (C2), both v_{m+1} and v_{m+2} are critical cutting pairs to v_m; therefore, C_m and C_{m+1} should be two non-empty disjoint subgraphs of G_m.

LEMMA 3.2 *If* ALP(G, s, t) *satisfies* (C2), *for each path between s and t in G, there is an $0 \leq i \leq m+1$ such that the path does not contain any vertex of C_i.*

Proof We prove the lemma by contradiction. Let P be a path between $s = v_0$ and t in G. Similar to the proof of Lemma 3.1, we can show that if P contains at least one vertex of each C_i, $0 \leq i \leq m$, v_m should precede all the vertices of $G_m \setminus v_m$ along P. But, because v_m is a critical cutting pair to both v_{m+1} and v_{m+2}, P cannot pass through both of C_m and C_{m+1}, which is a contradiction. ■

(F2): ALP(G, s, t) satisfies (F2) if it satisfies (C2) such that all graphs C_i, $0 \leq i \leq m+1$, contain more than one vertex.

Note that, both (F1) and (F2) may hold in a problem ALP(G, s, t) simultaneously. Two other types of forbidden problems, in which any path between s and t should miss some parts of the graph G, are illustrated in Figure 1.

(F3): ALP(G, s, t) satisfies (F3) if G contains four vertices v_1, v_2, v_3 and v_4, where each pair of them is a cutting pair, s lies on the boundary of G between v_1 and v_2, and t lies on the boundary of G between v_3 and v_4 (illustrated in Figure 1(a)).

(F4): ALP(G, s, t) satisfies (F4) if G contains four vertices v_1, v_2, v_3 and v_4 such that the structure of G around these vertices, s and t is as shown in Figure 1(b).

When ALP(G, s, t) satisfies (F3), let C_1, C_2, C_3 and C_4 be the connected components of $G \setminus \{v_1, v_2, v_3, v_4\}$ such that C_1 contains s and C_3 contains t. Similarly, if ALP(G, s, t) satisfies (F4), let C_1, C_2, C_3 and C_4 be the connected components of $G \setminus \{v_1, v_2, v_3, v_4, s, t\}$. Referring to Figure 1, it is clear that any path between s and t cannot contain vertices from both C_2 and C_4 in the case of (F3) and cannot contain vertices from all of C_1, C_2, C_3 and C_4 in the case of (F4).

LEMMA 3.3 *Any path between s and t in G can pass through only one of C_2 and C_4 if* ALP(G, s, t) *satisfies* (F3), *and can pass through at most three of C_1, C_2, C_3 and C_4 if* ALP(G, s, t) *satisfies* (F4).

A consequence of Lemmas 3.1, 3.2 and 3.3 is that there is no Hamiltonian path between s and t in G if G, s and t satisfy any of conditions (C1), (C2), (F3) or (F4).

3.1 *Solving the forbidden problems*

Our divide and conquer algorithm for solving ALP(G, s, t), presented in the next sections, requires that the given problem ALP(G, s, t) to not be a forbidden problem, and it never generates forbidden subproblems. Therefore, to complete our algorithm, we give a solution for the forbidden problems in this section. We show that there are limited number of cases in which ALP(G, s, t) is forbidden and provide a solution for each case.

First, let problem ALP(G, s, t) satisfy (F4). In this case, the structure of G around s and t should be as depicted in Figure 1(b), so the problem cannot satisfy any of the conditions (F1), (F2) or (F3) simultaneously, and the vertex set $\{v_1, v_2, v_3, v_4\}$ should be unique. Thus, we can convert it to a non-forbidden problem by removing the smallest subgraph among C_1, C_2, C_3 and C_4 from G. Note that, Lemma 3.3 ensures that the solution of the resulting non-forbidden problem has the length at least two-thirds of the length of the longest path between s and t in G.

For the case of (F3), (see Figure 1(a)), there may be more than one set of vertices $\{v_1, v_2, v_3, v_4\}$ which cause ALP(G, s, t) to satisfy (F3); however, since such sets may not overlap, we can make the problem non-forbidden by removing the smallest subgraph among C_2 and C_4 from G for each such set of vertices. In addition, it is possible that ALP(G, s, t) satisfies one or both of conditions (F1) and (F2) while satisfying (F3). In this case, as described above, we first convert the problem to the problem which only satisfies (F1) and/or (F2), and then convert it to a non-forbidden problem as follows. Again, Lemma 3.3 shows the correctness of our solution. Therefore, in the rest of this section, we can assume that ALP(G, s, t) does not satisfy (F3) or (F4).

If ALP(G, s, t) satisfies only (F1) or (F2), similar to the previous cases, we can convert it to a non-forbidden problem by removing the smallest subgraph among the subgraphs C_1, $C_2, \ldots, C_{m+1}$ from G, and the Lemmas 3.1 and 3.2 ensure the correctness of our solution. However, solving ALP(G, s, t) is a bit more tricky when both (F1) and (F2) hold simultaneously in ALP(G, s, t). We require the following lemma in that case.

LEMMA 3.4 *If* ALP(G, s, t) *satisfies* (F1) *or* (F2) *such that* $m > 4$, *then the structure of* G *around the vertices* $v_2, v_3, \ldots, v_{m-2}$ *must be as illustrated in Figure 2.*

Proof Let (F1) or (F2) hold in ALP(G, s, t). If G contains two critical cutting pairs to s, then $m = 0$. Therefore, G must have only one critical cutting pair to s. Let v be the vertex of G cutting pair to s, distinct from t as otherwise $m = 1$. We use Figures 3–5 to analyse all possible cases

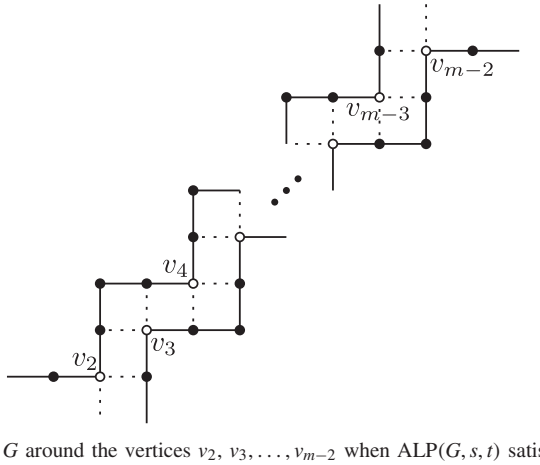

Figure 2. The structure of G around the vertices $v_2, v_3, \ldots, v_{m-2}$ when ALP(G, s, t) satisfies (F1) or (F2) such that $m > 4$.

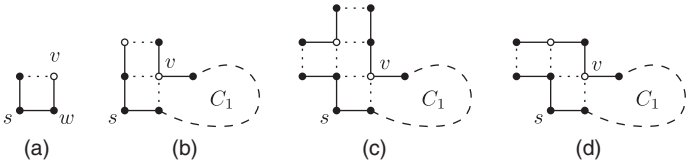

Figure 3. The possible structures for G around the degree-two boundary vertex s when it is critical cutting pair to only one vertex.

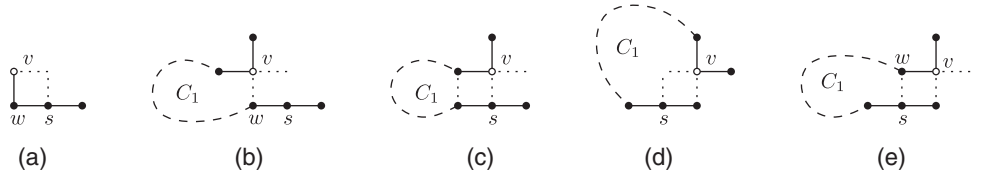

Figure 4. The possible structures for G around the degree-three boundary vertex s when it is critical cutting pair to only one vertex.

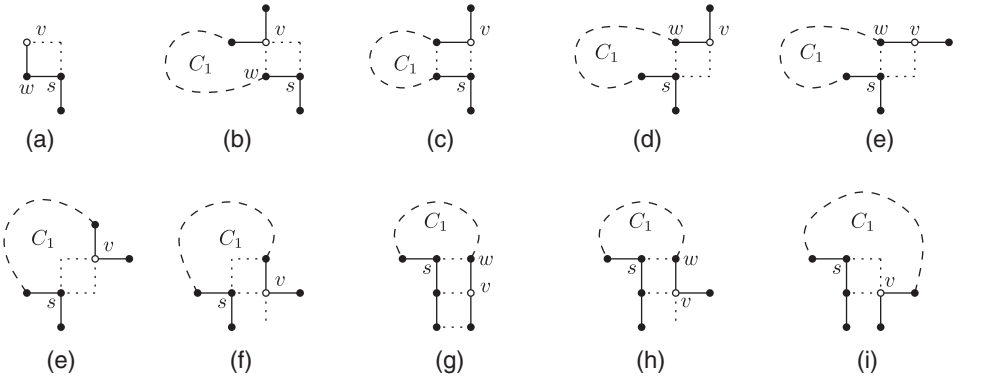

Figure 5. The possible structures for G around the degree-four boundary vertex s when it is critical cutting pair to only one vertex.

for G around s and v. Note that, the isomorphic cases are omitted in these figures. When s is a degree-two boundary vertex, the possible configurations are depicted in Figure 3. In Figure 3(a), one of the connected components of $G \setminus \{s, v\}$ has only one vertex, so (F1) or (F2) cannot hold. In the cases illustrated in Figure 3(b)–(d) consider the subgraph G_1 as defined before. For the cases of Figure 3(b) and 3(d), the structure of G_1 around v must be similar to the structure of G around s in Figure 3(a). Similarly, for the case of Figure 3(c), the structure of G_1 around v must be similar to the structure of G around s in Figure 3(b). Hence, when degree of s is two, m is at most 2.

Likewise, when s is a degree-three boundary vertex, the possible cases for G around s and v are shown in Figure 4. In the case of Figure 4(a), (F1) or (F2) cannot hold because $G \setminus \{s, v\}$ has a connected component containing only one vertex. In Figure 4(b)–(d), because v is a degree-two boundary vertex of G_1, m is at most three. Therefore, $m > 3$ is only possible in the case of Figure 4(e), which in this case the structure of G_1 around v must be similar to Figure 4(e) too.

Finally, if s is a degree-four boundary vertex, the possible cases are shown in Figure 5. In the case of Figure 5(a), (F1) or (F2) cannot hold because $G \setminus \{s, v\}$ has a connected component containing only one vertex. In the other cases of this figure, vertex v must be a degree-two or a degree-three boundary vertex of G_1. Hence, either $m \leq 4$ or the structure of G_1 around v must be similar to Figure 5(e).

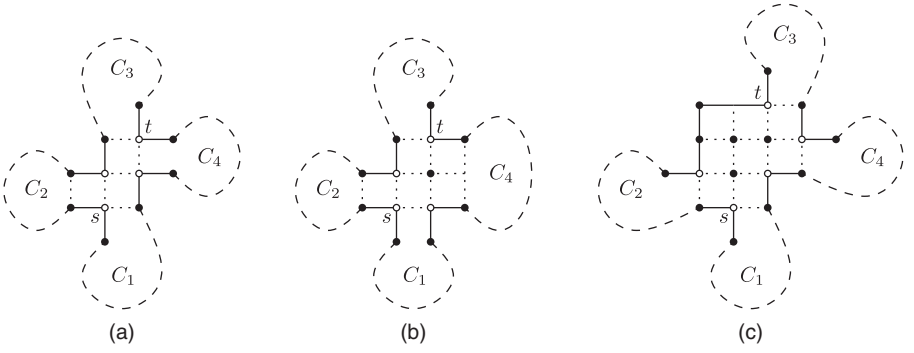

Figure 6. Some forbidden problems in which (F1) and (F2) hold simultaneously.

The above case analysis shows that, $m > 4$ is possible only when the structure depicted in Figure 4(e) is repeated in the subgraphs $G_1, G_2, \ldots, G_{m-2}$, respectively, around the vertices v_1, $v_2, \ldots, v_{m-2}$. Therefore, when $m \geq 4$ the structure of G around the vertices $v_1, v_2, \ldots, v_{m-2}$ must be as depicted in Figure 2. ■

Lemma 3.4 shows that there are finite number of possible structures for G around vertices s and t that make (F1) or (F2) hold in ALP(G, s, t) because, when (F1) or (F2) holds, either $m \leq 4$ or the structure of G around the vertices $v_1, v_2, \ldots, v_{m-2}$ is fixed. Thus, one can enumerate all possible structures of G around these vertices in which ALP(G, s, t) satisfies both (F1) and (F2), and provide a different solution for each case. More precisely, based on Lemmas 3.1 and 3.2, the longest path between s and t cannot go through all of some subgraphs of G, therefore in any case we can convert the problem to a non-forbidden problem by removing the smallest such subgraph. As examples, we enumerated three possible configurations for ALP(G, s, t) in which both (F1) and (F2) hold simultaneously, in Figure 6. In Figure 6(a), removing one of C_1 or C_2, and one of C_3 and C_4 from G makes the problem non-forbidden. Therefore, considering the Lemmas 3.1 and 3.2 and removing the smallest subgraph among C_1 and C_2, and the smallest subgraph among C_3 and C_4, one can solve the problem by converting it to a non-forbidden problem. For this purpose, in Figure 6(b), C_1 and C_3, C_1 and C_4, or C_2, and in Figure 6(c), C_2 and C_3, C_2 and C_4, or C_1 should be removed similarly.

4. The split operations

In this section, we introduce the split operations for dividing ALP(G, s, t) problem to smaller subproblems. Using these operations, we repeatedly split the problem until $G \setminus B$ becomes 2-connected, where B is a boundary path of G between s and t. In the split operations, we assume that ALP(G, s, t) is not a forbidden problem, and we never split it into forbidden subproblems. Therefore, we should show that subproblems created in the split operations are ALP problems and not forbidden. Moreover, in each split operation, we show that it is possible to construct a path between s and t containing at least $\frac{2}{3}$ of the nodes of G using the solutions of subproblems.

4.1 *Type I split operation*

We do type I split operation when G has a vertex cutting pair to s or t (without loss of generality say s). When, s and t are a cutting pair, since (F1) does not hold, a connected component of

$G \setminus \{s,t\}$ has only one vertex w, and the problem can be solved by finding an approximated longest cycle containing the edge (s,w) using the algorithm of [2].

Otherwise, first, let G contain only one vertex critical cutting pair to s, s and v be a cutting pair of G, and C_1 and C_2 be two connected components of $G \setminus \{s,v\}$ such that C_2 contains t. We illustrated all the possible cases for G around s and v in Figures 3–5, respectively, when s is a degree-two, degree-three and degree-four boundary vertex. In the following, type I split operation is described in each case.

- All The Cases Except The Cases of Figures 3(a), 4(a) and 5(a): In all of these cases, $C_2 \cup \{v\}$ is 2-connected. If $C_1 \cup \{s,v\}$ is not 2-connected, let w be the common adjacent vertex of s and v in C_1. Analysing all the cases in which $C_1 \cup \{s,v\}$ is not 2-connected, it is clear that such a vertex w always exists, and either $C_1 \cup \{s,w\}$ or $C_1 \cup \{v,w\}$ is 2-connected. Hence, we can split $\mathrm{ALP}(G,s,t)$ into $\mathrm{ALP}(C_2 \cup \{v\},v,t)$ and one of the three subproblems $\mathrm{ALP}(C_1 \cup \{s,v\},s,v)$, $\mathrm{ALP}(C_1 \cup \{s\},s,w)$ or $\mathrm{ALP}(C_1 \cup \{v\},v,w)$. Concatenating the solutions of subproblems and one of the edges (w,v) or (s,v), if needed, will result in a path of desired length between s and t in G.
- The Cases of Figures 4(a) and 5(a): In these cases, C_1 contains only one vertex w. If (C1) or (C2) do not hold, concatenating the edges (s,w) and (w,v) and the solution of $\mathrm{ALP}(C_2 \cup \{v\},v,t)$ will result in a solution for $\mathrm{ALP}(G,s,t)$. Otherwise, we solve $\mathrm{ALP}(G \setminus C_1,s,t)$ instead. The Lemmas 3.1 and 3.2 ensure that removing C_1 from G will not affect the ratio of our approximation.
- The Case of Figure 3(a): Again, in this case, C_1 contains only one vertex w, and if (C1) or (C2) do not hold, concatenating the edges (s,w) and (w,v) and the solution of $\mathrm{ALP}(C_2 \cup \{v\},v,t)$ will result in a solution for $\mathrm{ALP}(G,s,t)$. If (C1) or (C2) hold, we assume the structure of G around t to be similar to its structure around s, as otherwise we can split the problem using the vertex t. Therefore, $\mathrm{ALP}(C_2 \cup \{v\},v,t)$ cannot satisfy (F1). Furthermore, because (F3) does not hold in $\mathrm{ALP}(G,s,t)$, $\mathrm{ALP}(C_2 \cup \{v\},v,t)$ can satisfy (F2) only if the structure of G around s is as illustrated in Figure 7(a) or 7(b). But, in these two cases, $\mathrm{ALP}(C_2 \cup \{v\},w,t)$ does not satisfy (F1) and (F2). Therefore, we can solve the problem by solving $\mathrm{ALP}(C_2 \cup \{v\},v,t)$ or $\mathrm{ALP}(C_2 \cup \{v\},w,t)$ and adding the edges (s,u) and (u,v), or (s,w), respectively.

Next, we assume that G contains two vertices u and v critical cutting pairs to s, and C_u and C_v are the connected components of, respectively, $G \setminus \{s,u\}$ and $G \setminus \{s,v\}$ which does not contains t. In this case, C_u or C_v (without loss of generality say C_u) should contains only one vertex w because (F2) does not hold. All the possible cases for G around s are shown in Figure 8. In all the cases, except the cases of Figure 8(a) and 8(b), by removing the only vertex of C_u from G we can convert the problem to the case that s has only one critical cutting pair. In other words, in these cases, we can solve $\mathrm{ALP}(G \setminus C_u,s,t)$ instead. In the case of Figure 8(a), $\mathrm{ALP}(G \setminus C_u,s,t)$ may satisfy (F3), so we can solve one of $\mathrm{ALP}(G \setminus C_u,s,t)$ or $\mathrm{ALP}(G \setminus C_v,s,t)$ which does not

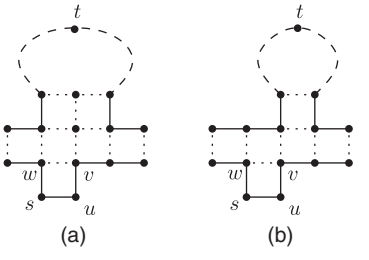

Figure 7. The cases in which (F2) can hold in the problem of Figure 3(a).

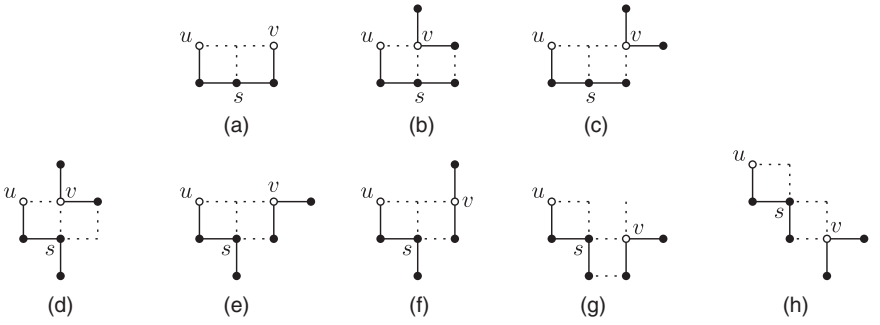

Figure 8. All the possible structures for G around s when it has two critical cutting pairs.

satisfy (F3) instead of ALP(G, s, t). Finally, in the case of Figure 8(b), we can split the problem into ALP$(C_v \cup \{s, v\}, s, v)$ and ALP$(G \setminus (C_v \cup C_u \cup \{s\}), v, t)$.

In type I split operation, presented in this subsection, we do not miss any of the vertices of G, except when (C1) or (C2) hold, in which we may miss one of the adjacent vertices of s. Therefore, considering Lemmas 3.1 and 3.2, and assuming that the solutions of the subproblems have the approximation ratio of $\frac{2}{3}$, the constructed path using this split operation should contain at least $\frac{2}{3}$ of the vertices of G. Additionally, in all the cases, it is easy to show that if none of (F1), (F2), (F3) or (F4) hold in ALP(G, s, t), they do not hold in any of the resulting subproblems either.

4.2 *Type II split operation*

Type II split is used to split ALP(G, s, t) when type I split is not possible and there is a cutting pair u and v in G whose removal disconnect s and t. Clearly, u and v should lie on different boundary paths B_1 and B_2 between s and t. Without loss of generality, let u be the nearest vertex to s along B_1 which is a cutting pair to a vertex of B_2, and v be the cutting pair to u nearest to s along B_2. In addition, let C_1 and C_2 be the connected components of $G \setminus \{u, v\}$ which, respectively, contains s and t. Based on the definition of u and v, there are only six possible structures for G around u and v, which are illustrated in Figure 9. In the following, we show how to split the problem in each case.

- The Case of Figure 9(a): We split the problem until $G \setminus B_1$ and $G \setminus B_2$ become 2-connected. However, in this case, the existence of the cutting pair u and v will not cause $G \setminus B_1$ or $G \setminus B_2$ to be not 2-connected, and we do not need to split the problem.
- The Cases of Figure 9(b) and 9(e): In these cases, we split the problem into ALP$(C_1 \cup \{u, v\}, s, v)$ and ALP$(C_2 \cup \{u, v\}, u, t)$. Because u and v in the subproblems are type I boundary vertices, (F1) and (F2) cannot hold in the subproblems. Let P_1 and P_2 be the solutions of the two subproblems. In both cases, if u is in P_1, it must be adjacent to v in P_1. Similarly, if v is in P_2, it must be adjacent to u in P_2. Therefore, we can merge P_1 and P_2 into a path of length at least $|P_1| + |P_2| - 1$, which is not less than $2n/3$.
- The Case of Figure 9(c): The split operation is like the case of Figure 9(b) but using vertices v and w.
- The Case of Figure 9(d): The split operation is like the case of Figure 9(b) but using vertices x and w.
- The Case of Figure 9(f):
  In this figure, w cannot be a boundary vertex because of the definition of u and v. In addition, without loss of generality, we can assume that r is not a boundary vertex, as otherwise we can

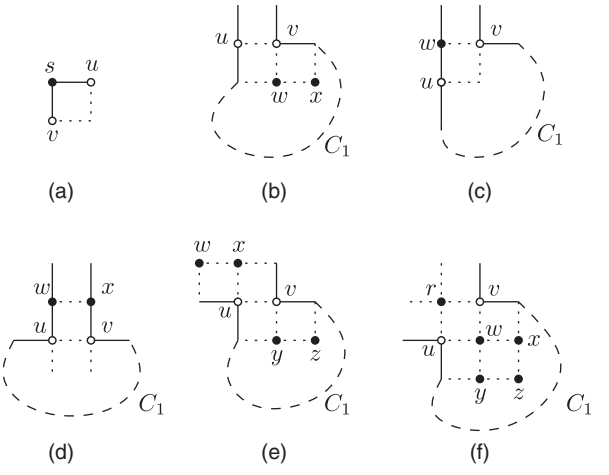

Figure 9. The possible structures for G around the vertices u and v.

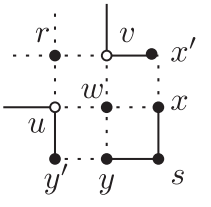

Figure 10. The graph of Figure 9 (f) when both x and y are boundary vertices.

solve ALP(G, t, s) instead of ALP(G, s, t). If at least one of x or y, say x, be a non-boundary vertex, both of $C_1 \cup \{u\}$ and $C_2 \cup \{u, v\}$ must be 2-connected, and we can split the problem into ALP$(C_1 \cup \{u\}, s, u)$ and ALP$(C_2 \cup \{u, v\}, u, t)$. Otherwise, i.e. when both of x and y are boundary vertices, the only possible case is illustrated in Figure 10. In this case, if both edges (x, x') and (y, y') are boundary edges, since (F4) does not hold, we can solve ALP(G, t, s) instead. Therefore, we can assume that (x, x') is a non-boundary edge, and solve the problem by solving ALP$(C_2 \cup \{u, v\}, u, t)$ and finding a cycle in the 2-connected graph $C_1 \cup \{u\} \setminus \{s, x, x'\}$ containing the edge (u, w) and then removing (u, w) and adding the edges (s, x) and (x, w).

4.3 *Type III split operation*

When type I and type II split operations are not possible and $G \setminus B$ is not 2-connected, in which B is a boundary path between s and t, we use type III split operation to split the problem. In this situation, considering the preconditions of type I and type II split operations, either B contains a cutting pair of G or G has a cutting set of three vertices two of which are in B. Let v be the nearest vertex to s along the path B that is in such a cutting set of vertices. Clearly, v is different from s, since type I split is not possible.

First, let there be a vertex u such that u and v are a cutting pair of G. We show that, ALP(G', v, t) is not a forbidden problem, in which G' is the result of removing the vertices of B_{sv} except v from G; therefore, we can solve the problem by concatenating B_{sv} and the solution of ALP(G', v, t). Knowing that B_{vt} is a boundary path between v and t in G', let B' be the other boundary path of G' between v and t. No vertex of B_{vt} could be a cutting pair to a vertex of B', as otherwise either type

I or type II split operations are possible on ALP(G, s, t) or $B_{s,v}$ contains a vertex different from v which is in a cutting set of three vertices two of which are in B, both cases leading contradictions. Therefore, none of the conditions (C1), (C2), (F3) or (F4) hold on ALP(G', v, t).

Next, let v be a vertex of a cutting set of three vertices two of which are in B, and u be the vertex of B next to v. Considering the preconditions of this split operation and the previous case, either $G \setminus B_{sv}$ or $G \setminus B_{su}$ should be 2-connected. Therefore, we can solve the problem by concatenating the solution of ALP$(G \setminus B_{sv}, u, t)$ and B_{su} when $G \setminus B_{sv}$ is 2-connected, and the solution of ALP$(G \setminus B_{su}, u', t)$ and $B_{su'}$ when $G \setminus B_{su}$ is 2-connected, in which u' is the vertex of B next to u. The argumentation that the subproblem ALP$(G \setminus B_{sv}, u, t)$ or ALP$(G \setminus B_{su}, u', t)$ is not forbidden is similar to the previous case.

5. The algorithm

In this section, using the results of Sections 3 and 4, we give an algorithm for ALP(G, s, t) problem, and show that this algorithm can be extended to approximate the longest path problem in solid grid graphs. The correctness of Algorithm 5.1 is shown in the following lemma.

Algorithm 5.1 The algorithm for solving ALP(G, s, t)

(1) **input:** A 2-connected solid grid graph G and its boundary vertices s and t
(2) **output:** A path between s and t of length at least $\frac{2}{3}$ of the length of the longest path between s and t
(3) **if** ALP(G, s, t) is forbidden **then**
(4) solve the problem as explained in Section 3
(5) **else if** s and t are incident to a common boundary edge of G **then**
(6) solve the problem using the longest cycle approximation algorithm given in [2]
(7) **else if** one of type I, type II, or type III splits are possible **then**
(8) split the problem and solve it recursively (see Section 4)
(9) **else**
(10) **let** B be the boundary path of G between s and t
(11) using a pair of parallel edges, merge B by a long cycle of $G \setminus B$ found by the algorithm of [2]

LEMMA 5.1 *There is a polynomial-time $\frac{2}{3}$-approximation algorithm for the version of the longest path problem in solid grid graphs in which the end-vertices are specified and forced to be on the boundary.*

Proof Let G be a solid grid graph. If G is not 2-connected, it should contain a cut vertex v. A path between two given vertices s and t in G cannot pass through the vertices of the connected components of $G \setminus \{v\}$ not containing s and t. Moreover, any path between s and t must pass through v, if s and t are in different connected components of $G \setminus \{v\}$. Therefore, when G is not 2-connected, one can split the problem easily using the cut vertices of G. Thus, without loss of generality, we assume that G is 2-connected, and reduce the problem to ALP(G, s, t). To complete our proof, we show that Algorithm 5.1 solves the ALP(G, s, t) in polynomial-time. In Section 3, we showed how we can convert a forbidden problem to a non-forbidden one by removing some of its vertices. Lemmas 3.1–3.3 show that removing these vertices does not reduce the approximation ratio of our algorithm in the case of forbidden problems. Also, in

Section 4 we showed that we never create a forbidden subproblem while spliting. Hence, the initially forbidden problems will be handled in the line 4 of the algorithm correctly, and during the recursions we never encounter forbidden problems. In the case when G contains a boundary edge (s,t), we can find a cycle of G containing (s,t) using the algorithm in [2] and remove the edge (s,t) to create a path between s and t of length at least $2|G|/3$, in line 6 of the algorithm. Furthermore, if one of the split operations of Section 4 is possible on ALP(G,s,t), we split the problem and solve it recursively. In Section 4, we showed that split operations only create consistent and non-forbidden subproblems, and how we can construct a solution to ALP(G,s,t) using the solutions of subproblems. Finally, if none of the above cases hold on ALP(G,s,t), $G \setminus B$ must be 2-connected and there must be a pair of parallel edges $e_1 \in B$ and $e_2 \in G \setminus B$. In this case, in line 11 of the algorithm, we find a long cycle in $G \setminus B$ containing the edge e_2 using the algorithm in [2] and merge it with B. Merging the cycle and B can be done by removing the parallel edges e_1 and e_2 from them and adding the two other adjacent edges of the common adjacent face of e_1 and e_2. Therefore, the algorithm works correctly.

The longest cycle approximation algorithm in [2] runs in linear time. Also, one can implement all the steps of our algorithm, except the recursive calls, to run in linear time with respect to n, the size of graph G. Thus, the worst case time complexity of Algorithm 5.1 is $O(n^2)$, because the total number of vertices of the subproblems created in split operations is always less than the number of the vertices of the graph G. ■

In Figure 11, we have illustrated an example application of Algorithm 5.1. The problem, i.e. a solid grid graph and two vertices s and t, is illustrated in Figure 11(a). First, the algorithm splits the problem using vertex s and its critical cutting pair v_2 (type I split). One of the resulting subproblems is illustrated in Figure 11(b). The other one is split by the algorithm using the vertices v_3 and v_4 (type II split) into two subproblems illustrated, respectively, in Figure 11(c) and 11(d). Finally, using the cutting set $\{v_5, v_8, t\}$, the algorithm splits the subproblem of Figure 11(d) (type III split). Figure 11(e) illustrates the resulting long path between s and t, obtained by concatenating the solutions of subproblems.

We note that, the best case and worst case scenarios of our algorithm depend on the best case and worst case scenarios of the algorithm for finding the longest cycle in [2]. Namely, for problem ALP(R,s,t) in which R is an even-by-even rectangular grid graph and s and t are two adjacent boundary vertices of R, our algorithms finds a Hamiltonian path between s and t because the algorithm in [2] finds a Hamiltonian cycle for R. Likewise, when R is a $3 \times n$ rectangular grid graph, the algorithm in [2] reports a cycle of length approximately $2n$, and consequently, our algorithm finds a path of length approximately $2n$. The following lemma shows that the above algorithm can be extended to solve the general longest path problem in solid grid graphs.

LEMMA 5.2 *There is a polynomial-time $\frac{2}{3}$-approximation algorithm for the longest path problem in solid grid graphs.*

Proof Let G be an n-vertex solid grid graph and the path P between vertices s and t be a longest path in G. Our approximation algorithm for finding a long path P' is as follows: for any pair of boundary vertices u and v of G find a long path using the algorithm of Lemma 5.1, and let P' be the path of maximum length among these paths. Because there are at most n^2 such pairs, the path P' can be found in polynomial time, and it just remains to prove that $|P'|$ is at least $2|P|/3$.

Considering Lemma 5.1, if both s and t are boundary vertices, we have $|P'| \geq 2|P|/3$. Similarly, if both s and t are in the same 2-connected maximal subgraph G' of G, then G' should contain all the vertices of P and at least two adjacent boundary vertices u and v of G. Because ALP(G',u,v) cannot be forbidden, the long path between u and v found by the algorithm of Lemma 5.1 must be at least $2|G'|/3$. Therefore, we must have $|P'| \geq 2|G'|/3 \geq 2|P|/3$.

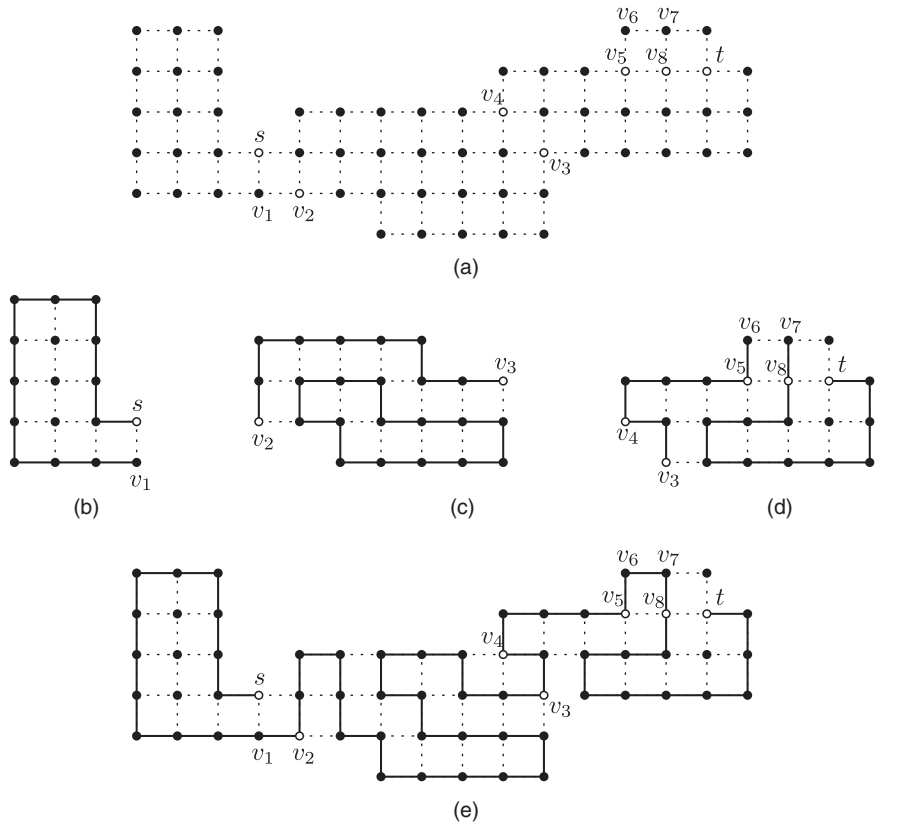

Figure 11. An example application of Algorithm 5.1.

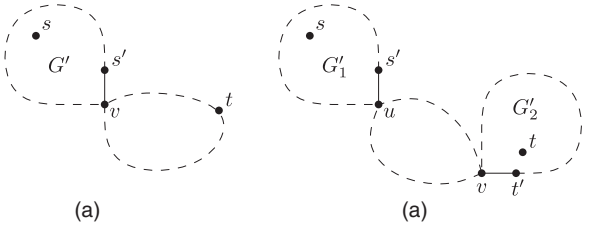

Figure 12. Illustration of graph G of Lemma 5.2 when only s is a non-boundary vertex (a) and both s and t are non-boundary vertices (b).

In addition, in the case when only one of s and t, say t, is a boundary vertex, let v be the nearest cut vertex of G to s along path P, G' be the connected component of $G \setminus v$ containing s, and s' be a boundary vertex of G adjacent to v in G' (see Figure 12(a) for illustration). Consider the algorithm of finding a long path between s' and t of Lemmas 5.1. Clearly, the algorithm splits the problem using vertex v. Because s', v and t are boundary vertices, and s' and v are adjacent, we can argue that the length of the long path found between s' and t by the algorithm is at least $2|P|/3$.

Similarly, in the case when both s and t are non-boundary vertices, let u and v be, respectively, the nearest cut vertex of G to s and t along the path P, and G'_1 and G'_2 be, respectively, the connected components of $G \setminus \{u, v\}$ containing s and t, and s' and t' be the boundary vertices of

G, respectively, adjacent to u and v in G'_1 and G'_2 (see Figure 12(b)). Again, because s', u, t' and v are boundary vertices, s' is adjacent to u and t' is adjacent to v, we can similarly argue that the length of the long path found between s' and t' by the algorithm is at least $2|P|/3$. ■

6. Conclusions

We introduced an algorithm that can find a path of length at least $\frac{2}{3}$ of the length of the longest path in solid grid graphs. The algorithm shows that the longest path problem can be approximated in polynomial time within factor of $\frac{2}{3}$, in this class of graphs. Our algorithm can also approximate the problem within the same factor when the end-vertices of the longest path are forced to be given boundary vertices.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

[1] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM (JACM) 42 (1995), pp. 844–856.
[2] A. Asgharian Sardroud and A. Bagheri, *An approximation algorithm for the longest cycle problem in solid grid graphs*, Discrete Appl. Math. (2014), to appear. doi:10.1016/j.dam.2015.10.022.
[3] N. Bansal, L.K. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko, *Further improvements in competitive guarantees for qos buffering*, Proceedings of the International Colloquium on Automata, Languages and Programming, Turku, Finland, Springer, 2004, pp. 196–207.
[4] A. Björklund and T. Husfeldt, *Finding a path of superlogarithmic length*, SIAM J. Comput. 32 (2003), pp. 1395–1402.
[5] A. Björklund, T. Husfeldt, and S. Khanna, *Approximating longest directed paths and cycles*, Proceedings of the International Colloquium on Automata, Languages and Programming, Turku, Finland, Springer, 2004, pp. 222–233.
[6] R.W. Bulterman, F.W. van der Sommen, G. Zwaan, T. Verhoeff, A.J.M. van Gasteren, and W.H.J. Feijen, *On computing a longest path in a tree*, Inform. Process. Lett. 81 (2002), pp. 93–96.
[7] R. Cerulli, R. De Leone, and G. Piacente, *A modified auction algorithm for the shortest path problem*, Optim. Methods Softw. 4 (1994), pp. 209–224.
[8] G. Chen, Z. Gao, X. Yu, and W. Zang, *Approximating longest cycles in graphs with bounded degrees*, SIAM J. Comput. 36 (2006), pp. 635–656.
[9] L. Di Puglia Pugliese and F. Guerriero, *Dynamic programming approaches to solve the shortest path problem with forbidden paths*, Optim. Methods Softw. 28 (2013), pp. 221–255.
[10] T. Feder and R. Motwani, *Finding large cycles in Hamiltonian graphs*, Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, Vancouver, BC, Canada, 2005, pp. 166–175.
[11] H.N. Gabow, *Finding paths and cycles of superpolylogarithmic length*, SIAM J. Comput. 36 (2007), pp. 1648–1671.
[12] H.N. Gabow and S. Nie, *Finding a long directed cycle*, ACM Trans. Algorithms (TALG) 4 (2008), pp. 7:1–7:21.
[13] H.N. Gabow and S. Nie, *Finding long paths, cycles and circuits*, Proceedings of the 19th Annual International Symposium on Algorithms and Computation, Surfers Paradise, Gold Coast, Australia, Springer, 2008, pp. 752–763.
[14] F. Guerriero and L. Di Puglia Pugliese, *Multi-dimensional labelling approaches to solve the linear fractional elementary shortest path problem with time windows*, Optim. Methods Softw. 26 (2011), pp. 295–340.
[15] G. Gutin, *Finding a longest path in a complete multipartite digraph*, SIAM J. Discrete Math. 6 (1993), pp. 270–273.
[16] K. Ioannidou, G.B. Mertzios, and S.D. Nikolopoulos, *The longest path problem is polynomial on interval graphs*, Proceedings of 34th International Symposium on Mathematical Foundations of Computer Science, Novy Smokovec, High Tatras, Slovakia, Springer, 2009, pp. 403–414.
[17] A. Itai, C.H. Papadimitriou, and J.L. Szwarcfiter, *Hamilton paths in grid graphs*, SIAM J. Comput. 11 (1982), pp. 676–686.
[18] D. Karger, R. Motwani, and G.D.S. Ramkumar, *On approximating the longest path in a graph*, Algorithmica 18 (1997), pp. 82–98.
[19] F. Kehsavarz Kohjerdi, A. Bagheri, and A. Asgharian Sardroud, *A linear-time algorithm for the longest path problem in rectangular grid graphs*, Discrete Appl. Math. 160 (2012), pp. 210–217.
[20] G.B. Mertzios and D.G. Corneil, *A simple polynomial algorithm for the longest path problem on cocomparability graphs*, SIAM J. Discrete Math. 26 (2012), pp. 940–963.
[21] T. Nishizeki and M.S. Rahman, *Planar Graph Drawing*, Vol. 12, World Scientific Pub Co Inc, Singapore, 2004.
[22] J.R. Sack and J. Urrutia, *Handbook of Computational Geometry*, North-Holland Publishing Co., Amsterdam, 2000.

[23] R. Uehara and Y. Uno, *Efficient algorithms for the longest path problem*, Proceedings of the 15th Annual International Symposium on Algorithms and Computation, Hong Kong, China, Springer, 2004, pp. 871–883.
[24] R. Uehara and Y. Uno, *On computing longest paths in small graph classes*, Internat. J. Found. Comput. Sci. 18 (2007), pp. 911–930.
[25] C. Umans and W. Lenhart, *Hamiltonian cycles in solid grid graphs*, Proceedings of 38th Annual Symposium on Foundations of Computer Science, Miami Beach, Florida, USA, 1997, pp. 496–505.
[26] W.-Q. Zhang and Y.-J. Liu, *Approximating the longest paths in grid graphs*, Theoret. Comput. Sci. 412 (2011), pp. 5340–5350.