https://doi.org/10.1016/j.ipl.2021.106176

End Vertices of Graph Searches on Bipartite Graphs

Meibiao Zou* Zhifeng Wang* Jianxin Wang* Yixin Cao*†

Abstract

For a graph search algorithm, the end vertex problem is concerned with which vertices of a graph can be the last visited by this algorithm. We show that for both lexicographic depth-first search and maximum cardinality search, the end vertex problem is NP-complete on bipartite graphs, even if the maximum degree of the graph is bounded.

1 Introduction

A graph search algorithm is a systematic way of exploring a graph and the core of such an algorithm lies on the strategy of choosing the next vertex to visit. Breadth-first search (BFS) and depth-first search (DFS) are the most fundamental graph algorithms and widely used, sometimes implicitly. Two other search algorithms, lexicographic breadth-first search (LBFS) [10] and maximum cardinality search (MCS) [11], were invented for the purpose of recognizing chordal graphs, and Lexicographic depth-first search (LDFS) was proposed by Corneil and Krueger [6]. Corneil et al. [5] noticed that the last vertex of a graph search is very important and they formulated the end vertex problem, which asks whether a vertex of a graph can be the last visited vertex by a specific graph search algorithm. It may sound a little surprising, but this ostensibly simple problem is NP-hard for all the well-known graph searches; we refer the reader to Cao et al. [3] for the state of the art.

This paper is focused on the end vertex problem on bipartite graphs. Charbit et al. [4] proved that the BFS end vertex problem is NP-complete on degree-bounded bipartite graphs. Gorzny and Huang [7, 8] proved that the end vertex problems for LBFS and DFS are also NP-complete on bipartite graphs. We show that this is also the case for two more graph searches.

Theorem 1.1. *The* LDFS *end vertex problem is NP-complete on bipartite graphs, even when the maximum degree of the input graph is seven.*

Theorem 1.2. *The* MCS *end vertex problem is NP-complete on bipartite graphs, even when the maximum degree of the input graph is fifteen.*

*School of Computer Science and Engineering, Central South University, Changsha, China. Supported by National Natural Science Foundation of China (61828205, 61672536), Hunan Provincial Key Lab on Bioinformatics, and Hunan Provincial Science and Technology Program (2018WK4001). mubyzou@csu.edu.cn, moluwang@csu.edu.cn, jxwang@mail.csu.edu.cn

†Department of Computing, Hong Kong Polytechnic University, Hong Kong, China. Supported in part by the Hong Kong Research Grants Council (RGC) under grants 15201317 and 15226116 and the National Natural Science Foundation of China (NSFC) under grant 61972330. yixin.cao@polyu.edu.hk.

Now for five of the six major graph searches that are well studied in this regard, the complexity of their end vertex problems are known to be NP-complete on bipartite graphs. The only remaining one is maximal neighborhood search (MNS), and we tend to believe it is also NP-complete. Let us also mention that Gorzny and Huang [8, 9] have also studied the end vertex problems on asteroidal triple–free bipartite graphs.

Notations. We denote the vertex set of a graph G by $V(G)$. A graph is *bipartite* if its vertex set can be partitioned into two disjoint independent sets. The two ends of an edge are *neighbors* of each other, and they are *adjacent*. A set of vertices are *false twins* if all of them have the same set of neighbors; note that by definition, there is no edge among false twins. An ordering σ of vertices in G is a bijection from $V(G) \to \{1, 2, \ldots, |V(G)|\}$. For two vertices u and v, we use $u <_\sigma v$ to denote $\sigma(u) < \sigma(v)$. The last vertex of σ, i.e., z with $\sigma(z) = |V(G)|$, is the *end vertex* of σ. Let S be a graph search algorithm. The S *end vertex* problem is defined as follows.

S end vertex

Input: A graph G and a vertex $z \in V(G)$.
Output: Is there an S-ordering of G of which z is the end vertex.

We prove both theorems by reductions from the 3-satisfiability problem (3-SAT): Given a Boolean formula in conjunctive normal form in which each clause comprises precisely three literals, we are asked whether there is an assignment to the variables to make the formula evaluated to be true. To have the stronger statements on degree-bounded graphs, we need the observation of Ahadi and Dehghan [1], which states that 3-SAT remains NP-complete even if every literal occurs exactly twice. The reductions are inspired by and adapted from Beisegel et al. [2].

2 Lexicographic depth-first search

Lexicographic depth-first search (LDFS) is a tie-breaking version of DFS [6]. Similar as DFS, LDFS visits a neighbor of the most recent vertex, or backtracks if all its neighbors have been visited. The difference between DFS and LDFS lies in the choice when the current vertex has more than one unvisited neighbor. Each unvisited vertex is assigned a label, which is the set of numbers its visited neighbors have received. When the most recent vertex has more than one unvisited neighbor, LDFS picks one with the lexicographically largest label. The algorithm is described in Figure 1.

1. **for each** $v \in V(G)$ **do** label$(v) \leftarrow \emptyset$;
2. **for** $i = 1, \ldots, n$ **do**
2.1. $v \leftarrow$ an unvisited vertex with the lexicographically largest label;
2.2. $\sigma(v) \leftarrow i$;
2.3. **for each** unvisited neighbor x of v **do** add i to label(v);
3. **return** σ.

Figure 1: The algorithm Lexicographic depth-first search.

Given an instance $\mathcal{I}$ of 3-SAT with p variables, $x_1, x_2, \ldots, x_p$, and q clauses, $c_1, c_2, \ldots, c_q$, we construct a graph G as follows; an illustration of this construction can be found in Figure 2. For

each variable $x_i, i = 1, \ldots, p$, we introduce four vertices, x_i^{in}, x_i^{out} for literal x_i, and $\bar{x}_i^{\text{in}}$, $\bar{x}_i^{\text{out}}$ for literal $\neg x_i$, with edges $x_i^{\text{in}} x_i^{\text{out}}$ and $\bar{x}_i^{\text{in}} \bar{x}_i^{\text{out}}$. We call the vertex with superscript "in" the *in-vertex*, and the other the *out-vertex* for a literal. As we will see, the two vertices for a literal are always visited consecutively, with the in-vertex followed by the out-vertex. For each clause c_j with $j = 1, \ldots, q$, we introduce a vertex c_j and add three edges to make the vertex c_j adjacent to the in-vertices of the *negations* of the three literals in c_j; e.g., if $c_j = x_1 \vee \neg x_2 \vee x_3$, then the three neighbors of vertex c_j are $\bar{x}_1^{\text{in}}$, x_2^{in}, and $\bar{x}_3^{\text{in}}$.

For each $i = 1, \ldots, p-1$, and for each of the four literal pairs $\{x_i, \neg x_i\} \times \{x_{i+1}, \neg x_{i+1}\}$, we introduce two adjacent vertices, and make one of them adjacent to the two in-vertices and the other to the two out-vertices. They make the *joint* for this pair of literals. A joint between a literal with index i and a literal with index $i+1$ is a *forward joint* for the literal with index i, and a *backward joint* for the other. We add adjacent vertices t_1 and t_2, called *turning points*, and make them adjacent to $x_p^{\text{out}}, \bar{x}_p^{\text{out}}$ and $x_p^{\text{in}}, \bar{x}_p^{\text{in}}$ respectively.

The gadget *starter* comprises $4p + 8$ vertices, $a_1, \ldots, a_5$ and $b_0, \ldots, b_{4p+2}$, which are connected as follows. First, all the edges between a_3, a_4, a_5 and a_2, b_0, b_1 are present, and so are $a_1 a_2$ and $b_0 b_2$. Note that a_3, a_4, and a_5 are pairwise false twins. For each $i = 1, \ldots, p-1$, we add seven edges $b_{4i-3}b_{4i-2}$, $b_{4i-2}b_{4i-1}$, $b_{4i-1}b_{4i}$, $b_{4i}b_{4i+1}$, $b_{4i-3}b_{4i}$, $b_{4i-2}b_{4i+1}$, and $b_{4i-1}b_{4i+2}$. For each of the four joints between $\{x_i, \bar{x}_i\}$ and $\{x_{i+1}, \bar{x}_{i+1}\}$, we make the neighbors of the out-vertices in them adjacent to b_{4i}. Moreover, we make b_{4p} adjacent to t_1, make b_{4p+1} adjacent to x_1^{out} and $\bar{x}_1^{\text{out}}$, and make b_{4p+2} adjacent to x_1^{in} and $\bar{x}_1^{\text{in}}$. Finally, we add the vertex z and make it adjacent to a_2 and t_1. This concludes the construction of the graph G. Let $n = |V(G)| = 16p + q + 3$.

It is quite obvious that the constructed graph is indeed bipartite. The vertices are depicted as either solid or empty, and they form the two separate parts.

Proposition 2.1. *The graph* G *constructed above is bipartite.*

Proof. The first part consists of a_2, b_0, b_1, b_3, …, b_{4p+1}, the in-vertices of all the literals, all the neighbors of the out-vertices in joints, and t_1, all denoted as solid nodes in Figure 2. All the other vertices, denoted as empty nodes in Figure 2, belong to the other part. Since there is no edge between two vertices in the same part, G is bipartite. □

One direction of the proof of the correctness of this reduction is straightforward. Suppose that the 3-SAT instance $\mathcal{I}$ is satisfiable. We start from a_1, visit all the vertices in the starter, and then go through vertices for all true literals, followed by the turning points. We then turn back to visit all vertices for the false literals, the remaining joint vertices, and all the clause vertices. Finally, we backtrack to t_1 and visit z.

Lemma 2.2. *If the 3-*SAT *instance* $\mathcal{I}$ *is satisfiable, then* z *is an* LDFS *end vertex of* G.

Proof. We may assume without loss of generality that setting every variable to be true satisfies the instance; otherwise we may negate all the occurrences of some variables, which does not change the graph. We can construct a corresponding LDFS ordering that ends at z as follows. Starting from a_1, we visit a_2, a_3, b_0, a_4, b_1, a_5, and b_2, b_3, …, b_{4p+1}, b_{4p+2} in order. After the starter, we enter the main box to visit x_1^{in} and x_1^{out},. Then for $i = 1, \ldots, p-1$, we visit the vertices in the x_i–x_{i+1} joint, and then x_{i+1}^{in} and x_{i+1}^{out} in order. To see that this is a valid LDFS ordering, note that the two neighbors of x_i^{out} in the forward joints have the same label, and if $i > 1$, its unvisited neighbor in the backward joints has a smaller label; moreover, after visiting the solid vertex in the joint, we must visit the other vertex in the same joint immediately.

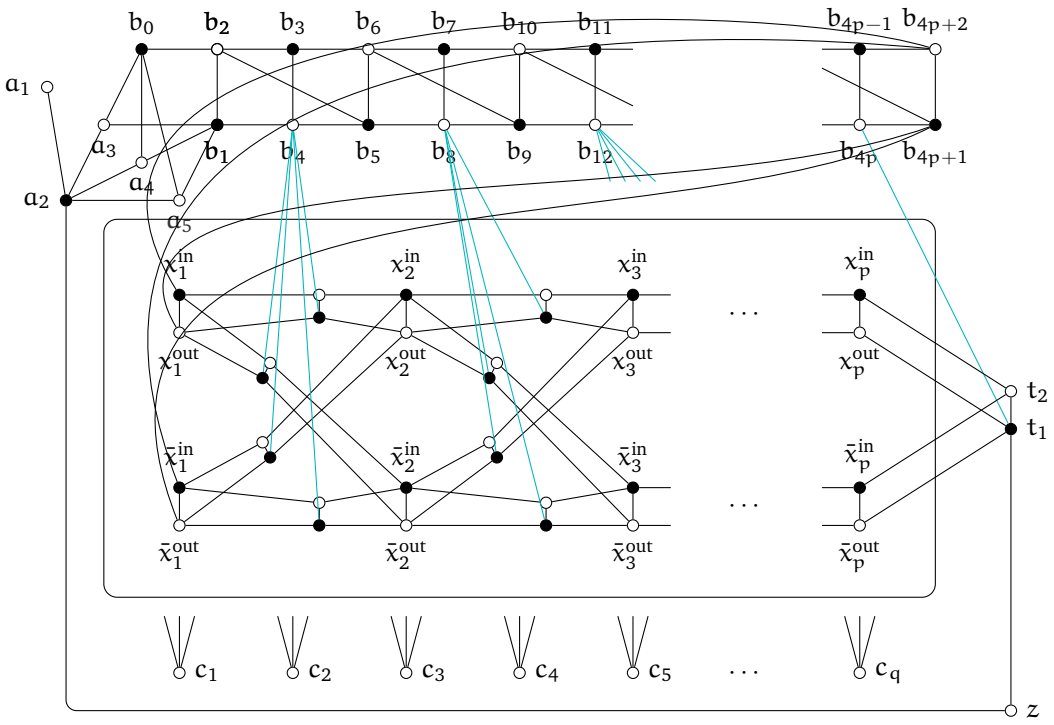

Figure 2: Illustration of the reduction for the LDFS end vertex problem. In the main box are the $4p$ literal vertices and their $8p-8$ joint vertices. At the top is the starter, the bottom the clause vertices, and the right the turning points and the vertex z.

After visiting x_p^{out}, we have to visit the turning points t_1 and t_2 in order because t_1 has a larger label than the other unvisited neighbor of x_p^{out}, which is in the $\neg x_{p-1}$–x_p joint. Then we turn back to visit other literal vertices. Note that $\bar{x}_p^{in}$ is the only unvisited neighbor of t_2, which is followed by $\bar{x}_p^{out}$ because it is adjacent to t_1. The only two unvisited neighbors of $\bar{x}_p^{out}$ are in the backward joints for $\neg x_p$, and we should visit the one in the x_{p-1}–$\neg x_p$ joint because it has another visited neighbor x_{p-1}^{out}. After both vertices in this joint, we backtrack to $\bar{x}_p^{out}$, and visit vertices in the $\neg x_{p-1}$–$\neg x_p$ joint, followed by $\bar{x}_{p-1}^{in}$ and $\bar{x}_{p-1}^{out}$ in order, and then the two vertices in the $\neg x_{p-1}$–x_p joint. We then backtrack to $\bar{x}_{p-1}^{out}$ and proceed backward similarly. In this manner we visit all the remaining literal vertices and joint vertices. Note that we never backtrack to an in-vertex during this stage, and thus we will not visit any clause vertex.

After finishing all the literal vertices and vertices in the joints, we backtrack. Since each clause c_j is satisfied, it contains at least one true literal, and hence the vertex c_j is adjacent to at least one $\bar{x}_i^{in}$ for some $i = 1, \ldots, p$. If vertex c_j has not been visited when backtracking to $\bar{x}_i^{in}$, it has to be visited now. Note that all the three neighbors of vertex c_j are literal vertices, and hence have already been visited, and thus we backtrack immediately after visiting vertex c_j. In summary, all the clause vertices are visited before we backtrack to t_1. Then z is the only unvisited vertex of the graph, and visited the last. □

For the other direction, we argue that the visiting order given above is essentially the only LDFS of G if we want to keep z to the end.

Lemma 2.3. *If z is an* LDFS *end vertex of the graph* G*, then* $\mathcal{I}$ *is satisfiable.*

Proof. Let σ be an LDFS ordering of G such that $\sigma(z) = n$.

Our first claim is that σ must start from one of $a_1, \ldots, a_5$. If $t_1 <_\sigma a_2$, then the label of z is always larger than a_1, and z cannot be the end vertex of σ. In the rest, we assume $a_2 <_\sigma t_1$, and consider when a_2 is visited. Since a_3, a_4, and a_5 are false twins, we may assume without loss of generality, $a_3 <_\sigma a_4 <_\sigma a_5$; note that we can always switch their positions in σ without changing the validity of this ordering. Let $A = \{a_1, \ldots, a_5, b_0\}$, whose neighbors include b_1, b_2, and z. First, suppose that the first vertex of σ is not in A, then the last visited vertex before entering A is either b_1 or b_2. In either case, the vertices in A are visited consecutively, after which we need to visit z immediately. Since $a_2 <_\sigma t_1$, the vertex t_1 has not been visited, contradicting that z is the end vertex. Now suppose that σ starts from b_0, then the next is either b_2 or a_3. If $\sigma(b_2) = 2$, then the vertex we visit before coming back to A has to be b_1, (possibly after a long sequence of other vertices,) followed by a_3, a_2, a_4, a_5, and then a_1, z or directly z. But t_1 has not been visited, a contradiction to $\sigma(z) = n$. If $\sigma(a_3) = 2$, then we must visit either b_1, a_4, a_2, a_5, or a_2, a_4, b_1, a_5, b_2 in sequence. In the first case, we need to visit a_1 and z immediately, while in the second, we will visit t_1 and z before backtracking to a_2; in either case, z cannot be the end vertex. Therefore, the first vertex of σ has to be one of $a_1, \ldots, a_5$, and it is easy to verify that the first two visited vertices in $V(G) \setminus A$ are b_1 and b_2 in order.

The second claim is that for $i = 1, \ldots, p$, the four vertices b_{4i-1}, b_{4i}, b_{4i+1}, and b_{4i+2} are visited consecutively, and b_{4i+2} is the last of them. After b_{4i-2} is visited, its unvisited neighbors are b_{4i-1} and b_{4i+1}. Depending the choice on them, the next four vertices we visit in order are either b_{4i-1}, b_{4i}, b_{4i+1}, and b_{4i+2}, or b_{4i+1}, b_{4i}, b_{4i-1}, and b_{4i+2}—note that b_{4i} is adjacent to b_{4i-3}, while both b_{4i-1} and b_{4i+1} are adjacent to b_{4i-2}. The claim follows by an inductive application of this argument. As a result, the last visited vertex in the starter has to be b_{4p+2}, and before finishing all vertices in the starter, we will not jump to a literal vertex from b_{4i}. After finishing all the vertices in the starter, we visit either x_1^{in} or $\bar{x}_1^{\mathrm{in}}$, followed by the out-vertex for the same literal, because it is an unvisited neighbor of the last vertex and it is adjacent to b_{4p+1}.

We then argue that after visiting x_i^{out} or $\bar{x}_i^{\mathrm{out}}$, $i = 1, \ldots, p-1$, we visit the vertices for literal x_{i+1} or $\neg x_{i+1}$, via a forward joint; in particular, the in-vertex and the out-vertex for the literal are visited in order. We may consider x_i^{out}, and the other is symmetric. All the unvisited neighbors of x_i^{out} are in joints. Those in forward joints are adjacent to b_{4i}, and hence have larger labels than the one in the backward joint. After visiting the solid vertex in a forward joint, we have to finish the other vertex of the same joint because it is adjacent to x_i^{in}. With an inductive application of this argument, we reach x_p^{out} or $\bar{x}_p^{\mathrm{out}}$. For the same reason, we continue to t_1, which is adjacent to b_{4p}, and then t_2.

We are now ready to construct the assignment for $\mathcal{I}$. Let $i = 1, \ldots, p$. If the vertices for literal x_i have been visited before t_1, (or $x_i^{\mathrm{in}} <_\sigma t_1$,) then we set x_i to be true; we set x_i to be false otherwise (if $\bar{x}_i^{\mathrm{in}} <_\sigma t_1$). Suppose for contradiction that clause $c_j, j = 1, \ldots, q$ is not satisfied by this assignment. For each literal ℓ in c_j, the in-vertex for the negation of ℓ has been visited before t_1. Thus, all the neighbors of vertex c_j have been visited before t_1, and to visit c_j we have to backtrack to one of these literal vertices, which can only happen after finishing all the neighbors of t_1. But then $z <_\sigma c_j$, a contradiction to $\sigma(z) = n$. This completes the proof. □

Proof of Theorem 1.1. Since the LDFS end vertex problem is clearly in NP, its NP-completeness follows from Lemmas 2.2 and 2.3. If we start from a 3-SAT instance in which every literal occurs exactly twice, then the maximum degree is seven, attained by b_4, b_8, …, b_{4p-4}, and x_2^{in}, $\bar{x}_2^{\mathrm{in}}$, …, x_{p-1}^{in},

$\bar{x}^{in}_{p-1}$. □

3 Maximum cardinality search

Given an instance $\mathcal{I}$ of 3-SAT with p variables, $x_1, x_2, \ldots, x_p$, and q clauses, $c_1, c_2, \ldots, c_q$, we construct a graph G as follows; an illustration of this construction can be found at Figure 3. For each literal, we introduce two vertices; in particular, vertices x_i and x'_i for literal x_i, and vertices $\bar{x}_i$ and $\bar{x}'_i$ for literal $\neg x_i$, where $i = 1, \ldots, p$. For each clause c_j with $j = 1, \ldots, q$, we introduce a set C_j of three vertices. Since the vertices in C_j are false twins, we are not going to distinguish them. The literal vertex x_i (resp., $\bar{x}_i$) is adjacent to all three vertices in C_j if c_j contains literal $\neg x_i$ (resp., x_i). For example, if $c_j = x_1 \vee \neg x_2 \vee x_3$, then the three neighbors of each vertex in C_j are $\bar{x}_1$, x_2, and $\bar{x}_3$.

For each $i = 1, \ldots, p-1$, and for each of the four literal pairs $\{x_i, \neg x_i\} \times \{x_{i+1}, \neg x_{i+1}\}$, we introduce two vertices, and make each of them adjacent to all the four vertices for the two corresponding literals. They are called the *joint* for this pair of literals, and a joint between a literal with index i and a literal with index $i+1$ is a *forward joint* for the literal with index i, and a *backward joint* for the other. For each $j = 1, \ldots, q-1$, we introduce three vertices, forming the *clause joint* for C_j and C_{j+1}, and make each of them adjacent to all the six vertices in $C_j \cup C_{j+1}$. For each literal vertex and each literal joint vertex, we introduce an image, and connect the vertex and its image. Let I_L denote the set of $4p$ images of literal vertices, and I_J the set of $8(p-1)$ images of literal joint vertices. Note that $I_L \cup I_J$ is an independent set.

The *starter* comprises four vertices, a_1, …, a_4, and the three edges among them are all incident to a_2. Both a_3 and a_4 are adjacent to all the literal vertices for literals x_1 and $\neg x_1$ as well as all the $3q-3$ clause joint vertices. There are three set of vertices: a pair F of *flash points*; a triple T_L of *transmitters* to the images of literal vertices, and a triple T_J of *transmitters* to the images of literal joint vertices. Every vertex in F is adjacent to all the literal vertices for literals x_p and $\neg x_p$. Every vertex in T_L is adjacent to all the vertices in I_L and all the vertices in F . Every vertex in T_J is adjacent to all the vertices in I_J and all the vertices in T_L. Note that for each of the sets F, T_L, and T_J, vertices in it are false twins, and so are a_3 and a_4. To finish the construction, we add a vertex z, and make it adjacent to all the three vertices in C_q. Let $n = |V(G)| = 24p + 6q - 6$.

Proposition 3.1. *The graph* G *constructed above is a bipartite graph.*

Proof. We can partition $V(G)$ into two parts. The first part comprises all the literal vertices, all the clause joint vertices, all the images of literal joints, all the vertices in T_L, a_2, and z, while the other comprises all the literal joint vertices, all the clause vertices, all the images of literal vertices, all the vertices in F and T_J, a_1, a_3, and a_4. Since there is no edge between two vertices in the same part, G is bipartite. □

At each step, the maximum cardinality search (MCS) algorithm picks an unvisited vertex that has the largest number of visited neighbors. The basic idea of the construction is as follows. The clause vertices have higher connectivity among each other, and z is connected to all the three vertices in C_q. To make sure that z is visited last, we need to go through the main box without having to enter the clause vertices. This can only be guaranteed by a satisfying assignment: We can reach the flash point through only the vertices for true literals, and then reach I_L and I_J through transmitters in T_L and T_J, which increases the visited neighbors for each literal vertex and each literal joint vertex by one.

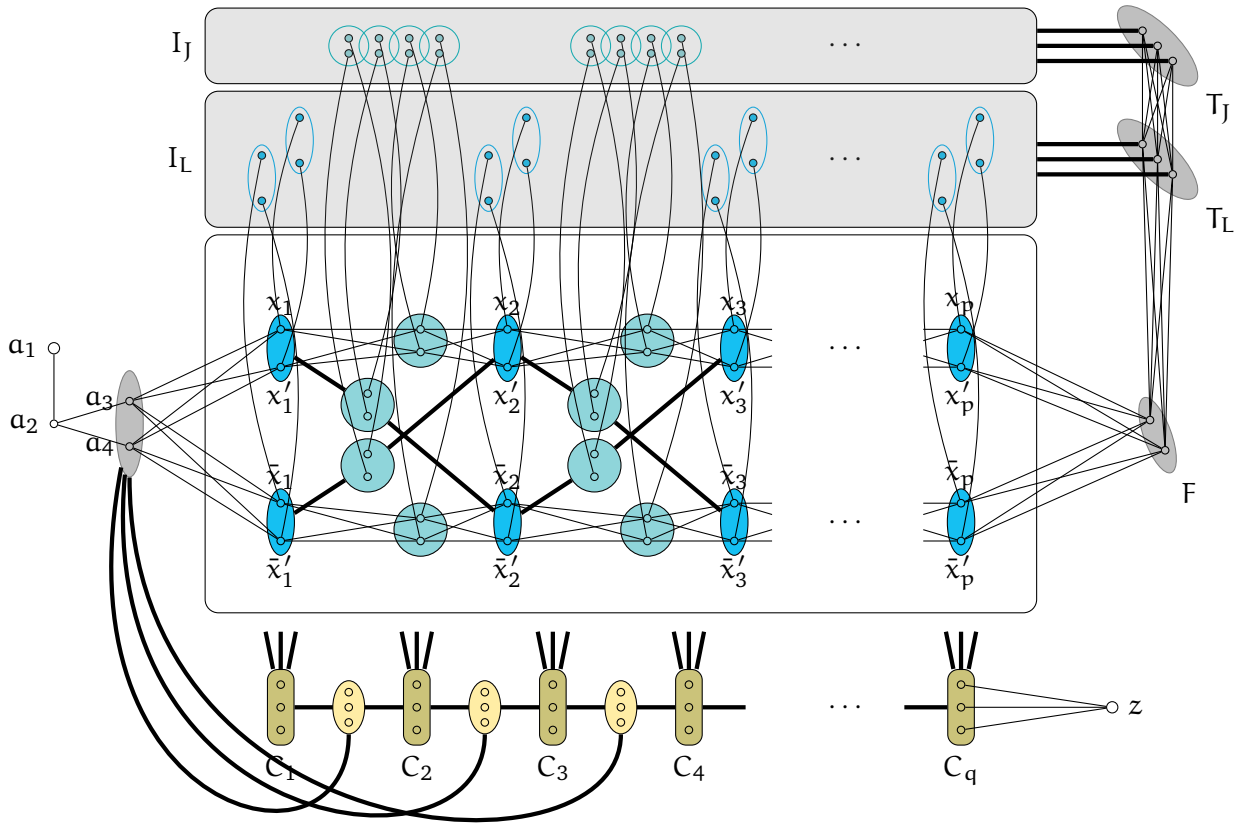

Figure 3: Illustration of the reduction for the MCS end vertex problem. In the main box are the $4p$ literal vertices and their $8p-8$ joint vertices. At the top are the image vertices, the bottom the clause vertices and z, the left the starter, and the right the flash points and transmitters. A thick line means all the edges between the two ends are present.

Lemma 3.2. *If the 3-*SAT *instance $\mathcal{I}$ is satisfiable, then z is an* MCS *end vertex of* G.

Proof. We may assume without loss of generality that setting every variable to be true satisfies the instance; otherwise we may negate all the occurrences of some variables, which does not change the graph. We can construct a corresponding MCS ordering of G that ends at z as follows.

Starting from a_1, an MCS may visit a_2, a_3, and a_4 in order. After that each vertex has at most two visited neighbors, and those having two are the four vertices for literals x_1 and $\bar{x}_1$, and all the clause joint vertices. We choose the two vertices for x_1. In the next, for $i = 2, \ldots, p$, we visit the two vertices in the x_{i-1}–x_i joint, and then the two vertices for x_i. After finishing the two vertices for literal x_p, we visit the two flash points in F, followed by the three transmitters in T_L. Note that before visiting T_L, no unvisited vertex can have three or more visited neighbors. Since every clause contains an unnegated literal, each clause vertex is adjacent to a literal vertex for a negated literal. Thus, each clause vertex has at most two visited neighbors.

After all the three vertices in T_L, every vertex in T_J has three visited neighbors. Thus, we can visit vertices in T_J next, followed by vertices in I_L and I_J. As a result, for each literal vertex and each

literal joint vertex, the number of visited neighbors is increased by one. At this junction there is no unvisited vertex with four or more visited neighbors. Since every literal joint vertex between a positive literal and a negative literal has three visited neighbors, we can visit them, followed by the literal vertices for all p negated literals, and the joint vertices among them. Now that all the literal vertices have been visited, every clause vertex has three visited neighbors. We can visit the vertices in C_1, and then for $i = 2, \ldots, q$, the vertices in the C_{i-1}–C_i clause joint, and then vertices in C_i. We visit z last. This completes the proof. □

For the other direction, we show that whether z can be the last hinges on whether we can reach the flash points without visiting all the three literal vertices adjacent to any clause vertex.

Lemma 3.3. *If z is an* MCS *end vertex of* G*, then* $\mathcal{I}$ *is satisfiable.*

Proof. Let σ be an MCS ordering of G such that $\sigma(z) = n$. Since a_3 and a_4 are false twins, we may assume without loss of generality, $a_3 <_\sigma a_4$; note that we can always switch their positions in σ without changing the validity of this ordering.

We first argue that σ starts from one of the vertices in the starter, and the four vertices in the starter are among the first five vertices in σ; or more specifically, $\{\sigma(a_i) \mid 1 \leqslant i \leqslant 3\} = \{1, 2, 3\}$, and $\sigma(a_4) \in \{4, 5\}$. Suppose that we conduct an MCS on G starting from another vertex. To visit a_1, we have to visit one of a_3 and a_4, and then a_2. By definition, before visiting a_3, at least one neighbor x of a_3 has been visited; note that x is adjacent to a_4. On the other hand, a_2 has to be visited before a_1. Therefore, after visiting a_2, if a_4 has not been visited, then it has two visited neighbors, namely a_2 and x, and should be visited before a_1. However, once both a_3 and a_4 are visited, there is always one vertex that has two or more visited neighbors before all the vertices in $V(G) \setminus \{a_1\}$ are visited. Therefore, a_1 has to be the last vertex of this MCS, contradicting that $\sigma(z) = n$. In the rest, we may assume without loss of generality, $\sigma(a_i) = i$ for $i = 1, 2, 3$. Note that if the fourth vertex of σ is a neighbor of a_3, then the next has to be a_4, because it is the only vertex with two visited neighbors.

Let $R = T_L \cup T_J \cup I_L \cup I_J$. Our second claim is that no vertex in R can be visited before both flash points in F, and the first visited vertex in R is one of T_L, of which the visited neighbors are precisely F. Suppose that a vertex $x \in R$ is visited before we finish both flash points. Then x needs to have two visited neighbors. Thus, $x \notin I_L \cup I_J \cup T_J$ because each vertex in $I_L \cup I_J \cup T_J$ has at most one neighbor out of R. On the other hand, the only neighbors of a vertex in T_L out of R are the flash points.

The third claim is that no clause vertex can be visited before R. After the starter, each vertex in a clause joint has two visited neighbors, a_3 and a_4. Suppose that a vertex in $C_j, j = 1, \ldots, q$ is visited. Then every vertex in the joint(s) for C_j has three visited neighbors. Before all the clause vertices, vertices in the clause joints, and z are visited, there is always a vertex with three visited neighbors. By the second claim, no vertex in R has more than two visited neighbors. Therefore, z has to be visited before all vertices in R, a contradiction.

Now we consider what happens between visiting the starter and the flash points. The fourth claim is that for each $i = 1, \ldots, p-1$, before both the vertices for literal x_i or $\neg x_i$ have been visited, no vertex to the right of them can be visited. If at least one vertex for literal x_i and at least one vertex for $\neg x_i$ have not been visited, then any vertex to the right of them has at most one visited neighbor.

Therefore, for each $i = 1, \ldots, p-1$, both x_i, x_i' or both $\bar{x}_i, \bar{x}_i'$ have been visited before visiting F. The next claim is that before the second vertex of F is visited, both x_p, x_p' or both $\bar{x}_p, \bar{x}_p'$ have been visited. When we visit the first vertex in F, at least two of its neighbors have been visited. We are done if both are vertices for the same literal. Now suppose that one of them is for literal x_p and the

other for its negation. After visiting a vertex in F, both unvisited vertices for literal x_p and $\neg x_p$ have three visited neighbors, and thus should be visited before the unvisited vertex in F.

We are now ready to construct the assignment for $\mathcal{I}$. Let $i = 1, \ldots, p$. We have seen that at the conjunction when the second vertex in F is visited, both vertices for literal x_i or $\neg x_i$ have been visited. If both vertices for literal x_i have been visited, then we set x_i to be true, and false otherwise. (It is possible that all the four vertices x_i, x_i' and $\bar{x}_i, \bar{x}_i'$ have been visited, and in this case it does not really matter what value variable x_i receives). Suppose for contradiction that clause c_j is not satisfied by this assignment. For each literal ℓ in c_j, both vertices for the negation of ℓ have been visited. But then each vertex in C_j has three visited neighbors before visiting F, and they should be visited before vertices in R, contradicting the third claim. This completes the proof. □

Proof of Theorem 1.2. Since the MCS end vertex problem is clearly in NP, its NP-completeness follows from the construction (Proposition 3.1) and Lemmas 3.2 and 3.3. To have the stronger statement in Theorem 1.2 on degree-bounded graphs, we need to modify the construction as follows. First, we replace a_3, a_4 by a $2q-1$ pairs of vertices, and for $i = 1, \ldots, 2q-2$, make the ith pair and $(i+1)$st pair completely adjacent; for $i = 1, \ldots, q-1$, we make the $(2i-1)$st pair adjacent to the clause C_i–C_{i+1} joint; and the last pair (with index $2q-1$) are adjacent to all the literal vertices for literals x_1 and $\neg x_1$. Likewise, we replace $T_L \cup T_J$ by a sequence of vertex triples, and make each odd-indexed triple adjacent to one distinct pair of images of literal vertices, and each even-indexed triple adjacent to one distinct pair of images of literal joint vertices. It is easy to verify that the new graph remains bipartite. The maximum degree is then 15, attained by the literal vertices x_2, $\bar{x}_2$, …, x_{p-1}, $\bar{x}_{p-1}$ (each adjacent to eight joint vertices, one image vertex, and two triples of clause vertices). This concludes the proof. □

References

[1] Arash Ahadi and Ali Dehghan. (2/2/3)-SAT problem and its applications in dominating set problems. *Discrete Mathematics & Theoretical Computer Science*, 21(4):9:1–9:10, 2019. doi:10.23638/DMTCS-21-4-9.

[2] Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaz Krnc, Nevena Pivac, Robert Scheffler, and Martin Strehler. On the end-vertex problem of graph searches. *Discrete Mathematics & Theoretical Computer Science*, 21(1), 2019. URL: http://dmtcs.episciences.org/5572.

[3] Yixin Cao, Zhifeng Wang, Guozhen Rong, and Jianxin Wang. Graph searches and their end vertices. In Pinyan Lu and Guochuan Zhang, editors, *Proceedings of the 30th International Symposium on Algorithms and Computation (ISAAC)*, volume 149 of *LIPIcs*, pages 1:1–1:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.1.

[4] Pierre Charbit, Michel Habib, and Antoine Mamcarz. Influence of the tie-break rule on the end-vertex problem. *Discrete Mathematics & Theoretical Computer Science*, 16(2):57–72, 2014. URL: http://dmtcs.episciences.org/2081.

[5] Derek G. Corneil, Ekkehard Köhler, and Jean-Marc Lanlignel. On end-vertices of lexicographic breadth first searches. *Discrete Applied Mathematics*, 158(5):434–443, 2010. doi:10.1016/j.dam.2009.10.001.

[6] Derek G. Corneil and Richard Krueger. A unified view of graph searching. *SIAM Journal on Discrete Mathematics*, 22(4):1259–1276, 2008. doi:10.1137/050623498.

[7] Jan Gorzny. On end vertices of search algorithms. Master's thesis, University of Victoria, Victoria, British Columbia, Canada, 2015.

[8] Jan Gorzny and Jing Huang. End-vertices of LBFS of (AT-free) bigraphs. *Discrete Applied Mathematics*, 225:87–94, 2017. See also the thesis of the first author. doi:10.1016/j.dam.2017.02.027.

[9] Jan Gorzny and Jing Huang. End-vertices of AT-free bigraphs. In *Proceedings of the 26th Annual International Conference on Computing and Combinatorics (COCOON)*, volume 12273 of *LNCS*, pages 52–63. Springer, 2020. doi:10.1007/978-3-030-58150-3_5.

[10] Donald J. Rose, Robert Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. A preliminary version appeared in STOC 1975. doi:10.1137/0205021.

[11] Robert Endre Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984. With Addendum in the same journal, 14(1):254-255, 1985. doi:10.1137/0213035.