Exploration of k-Edge-Deficient Temporal Graphs

Thomas Erlebach[0000−0002−4470−5868] and Jakob T. Spooner[0000−0003−3816−6308]

School of Informatics, University of Leicester, England
{te17,jts21}@le.ac.uk

Abstract. An *always-connected temporal graph* $\mathcal{G} = \langle G_1, ..., G_L \rangle$ with underlying graph $G = (V, E)$ is a sequence of graphs $G_t \subseteq G$ such that $V(G_t) = V$ and G_t is connected for all t. This paper considers the property of *k-edge-deficiency* for temporal graphs; such graphs satisfy $G_t = (V, E - X_t)$ for all t, where $X_t \subseteq E$ and $|X_t| \leq k$. We study the Temporal Exploration problem (compute a temporal walk that visits all vertices $v \in V$ at least once and finishes as early as possible) restricted to always-connected, k-edge-deficient temporal graphs and give constructive proofs that show that k-edge-deficient and 1-edge-deficient temporal graphs can be explored in $O(kn \log n)$ and $O(n)$ timesteps, respectively. We also give a lower-bound construction of an infinite family of always-connected k-edge-deficient temporal graphs for which any exploration schedule requires at least $\Omega(n \log k)$ timesteps.

Keywords: Graph algorithms · Temporal graphs · Graph exploration.

1 Introduction

Given a simple, connected, undirected graph G and a *start vertex* $s \in V(G)$, the task of *exploring* G, i.e., computing a sequence of consecutively crossed edges $e \in E(G)$ that begins at s and visits every vertex $v \in V(G)$ at least once, is both natural and well-understood. A closely related problem was initially considered by Shannon [19], who designed a mechanical maze-solving machine which implemented a depth first search-type technique in order to locate, within a given maze, a prespecified goal. This 'searching' problem is indeed related to graph exploration: if our task is to simply complete an exploration of G, then a solution can be straightforwardly found by performing a DFS starting from s and stopping once all vertices have been visited at least once – clearly this requires $\Theta(n)$ edge-traversals in total.

The graph exploration problem in the context of temporal graphs (i.e., graphs whose edge set can change over time) has also received significant attention in recent years. This problem, known as Temporal Exploration (TEXP), but restricted to *k-edge-deficient* temporal graphs (which we define formally later) is the focus of this paper. Given a temporal graph $\mathcal{G}$, the problem asks that we compute a temporal walk, starting at some prespecified vertex $s \in V(\mathcal{G})$, that makes at most a single edge-traversal in each timestep, and that visits all vertices at least once by the earliest time possible. We formally define the problem and temporal graph model in Section 2, but refer the interested reader to [5, 16] for more on temporal graphs in general, or [6, 17] for more on TEXP. In the most general setting, TEXP makes no assumptions about the input temporal graph, aside from the assumption that the input temporal graph is connected in each timestep (i.e., *always-connected*), which ensures exploration is always possible. This allows an arbitrary number of edges from the underlying graph to be missing in each timestep, and thus the graphs in different timesteps can differ substantially, which leads to pessimistic bounds on the worst-case exploration time. It is therefore interesting to study the question whether better exploration times can be guaranteed if the number of missing edges in each time step is small. To study this question, we also consider always-connected temporal graphs but, in contrast to previous work, we consider k-edge-deficient temporal graphs, whose structure in each step is 'close' to that of its underlying graph, in the sense that at most k edges are missing. Such graphs were defined by Gotoh et al., in [11], where they were considered in a distributed setting. We assume that the temporal structure of an input temporal graph is known in full to an algorithm prior to it computing a solution, as opposed to a setting in which the structure of the graph in each step is revealed online and over time.

Contribution. We introduce the temporal graph property of *k-edge-deficiency*, and consider Temporal Exploration on always-connected temporal graphs that are k-edge-deficient for some $k \in \mathbb{N}$. We define the property formally in Section 2, but essentially these are temporal graphs $\mathcal{G}$ with *underlying graph* G, such that, during each timestep t of $\mathcal{G}$'s lifetime, there are at most k edges $e \in E$ in the underlying graph that are untraversable in (or 'missing' from) $\mathcal{G}$. Let $n = |V(G)|$. In Section 3 we prove for arbitrary $k \in \mathbb{N}$ that k-edge-deficient always-connected temporal graphs can be explored in $O(kn \log n)$ timesteps. In Section 4 we additionally show that 1-edge-deficient graphs can always be explored in $O(n)$ timesteps, giving a recursive exploration algorithm that exploits a number of existing structural/algorithmic results originating from traditional graph theory. Finally, in Section 5, we sketch a modification of an existing $\Omega(n \log n)$ lower bound on the number of timesteps required to explore always-connected temporal graphs with planar underlying graph of maximum degree ≤ 4, presented in [6], that allows us to obtain an $\Omega(n \log k)$ bound on the worst-case time required to explore arbitrary always-connected k-edge-deficient temporal graphs.

Related work. Brodén et al. [3] consider the Temporal Travelling Salesperson Problem on a complete graph with n vertices, with edge costs that can differ between 1 and 2 in each timestep. They show that when an edge's cost changes at most k times over the input graph's lifetime, the problem is NP-complete, but provide a $(2-\frac{2}{3k})$-approximation; for the same problem, Michail and Spirakis [17] prove APX-hardness and provide a $(1.7+\epsilon)$-approximation. Bui-Xuan et al. [4] propose multiple objectives for optimisation when computing temporal walks/paths: e.g., *fastest* (fewest steps used) and *foremost* (arriving at the destination at the earliest time possible). The decision version of the Temporal Exploration problem, which asks whether or not a given temporal graph admits a temporal walk that visits all vertices at least once, is also considered in [17]. They show that the problem is NP-complete when no restrictions are placed on the input; they also propose considering the problem under the *always-connected* assumption, which ensures that exploration is possible provided the lifetime of the input graph is sufficiently long [17]. Erlebach et al. [6] further consider the optimisation variant of the Temporal Exploration problem under the always-connected assumption. They prove an $\Omega(n^2)$ lower bound on the time needed to explore general always-connected temporal graphs, and provide a proof that temporal graphs within this class can be explored in n^2 steps. They also prove a number of bounds on the number of timesteps required to explore various restricted temporal graph classes. Bodlaender and van der Zanden [2] examine TEXP when restricted to graphs of pathwidth at most 2 in each timestep, showing the problem to be NP-complete even under these limiting restrictions. In [14] and [13], Ilcinkas et al. respectively consider TEXP restricted to temporal graphs with underlying cycle or cactus graphs. Akrida et al. [1] consider Return-To-Base TEXP in which a candidate solution must return to the vertex from which it initially departed. In [7], Erlebach et al. prove an $O(dn^{1.75})$ bound on the number of time steps required to explore any temporal graph with degree bounded by d in each step, a considerable improvement over the previously best known $O(\frac{n^2 \log d}{\log n})$ bound [8]. In [9], a *non-strict* variant of TEXP is studied – here, a computed walk may make an unlimited number of edge-traversals in each given timestep. Notions of strict/non-strict paths which respectively allow for a single edge/unlimited number of edge(s) to be crossed in any timestep have been considered before, notably by Kempe et al. in [15] and Zschoche et al. in [20]. In this paper, we only consider strict temporal walks. Gotoh et al. in [12] consider TEXP on temporal graphs with underlying cycle under the so-called (H, S)-view, in which only the availability of edges at most H hops away for at most the next S timesteps is known to an algorithm. Casteigts et al. examined the fixed-parameter tractability of the problem of finding temporal paths between a source and destination that wait no longer than Δ consecutive timesteps at any vertex they visit. Temporal graph exploration has also been studied in a distributed setting: in [11], Gotoh et al. consider a variant in which a collection of cooperating mobile agents construct a map of a temporal graph. In the same paper, they defined the class of k-edge-deficient graphs (under a different name), proving bounds on the number of cooperating agents required to ensure that exploration is possible under a variety of different distributed settings.

2 Preliminaries

We denote by $[n]$ the set $\{1, ..., n\}$. Let $G = (V, E)$ and G' be simple, undirected graphs. We write $G' \subseteq G$ if G' is a (not necessarily induced) subgraph of G. $|V|$ is the *order* of G; $|E|$ is G's *size*. If $X \subseteq V$ is a subset of G's vertex set, we denote by $G - X$ the subgraph of G induced by $V(G) - X$.

Definition 2.1 (Temporal graph). *A temporal graph $\mathcal{G} = \langle G_1, G_2, ..., G_L \rangle$ with underlying graph $G = (V, E)$, order $n = |V|$ and lifetime L is an ordered sequence of subgraphs $G_t = (V, E_t)$ of G, indexed by the timesteps $t \in [L]$. In particular, we have that $V(G_t) = V = V(\mathcal{G})$ and $E_t \subseteq E$ for all $t \in [L]$.*

Let $\mathcal{G} = \langle G_1, G_2, ..., G_L \rangle$ be an arbitrary temporal graph. An edge $e \in E$ that satisfies $e \in E_t$ is *present* during timestep t. If $e \in E$ satisfies $e \notin E_t$, we say that e is *missing* in timestep t. A temporal graph $\mathcal{G} = \langle G_1, G_2, ..., G_L \rangle$ is said to be *always-connected* if it is such that G_t is connected for all $t \in [L]$.

Definition 2.2 (Temporal walk). *A temporal walk $\mathcal{W}$ in a temporal graph $\mathcal{G}$ is an alternating sequence of vertices and edge-time pairs,*

$$\mathcal{W} = v_1, (e_1, t_1), v_2, ..., v_{k-1}, (e_{k-1}, t_{k-1}), v_k.$$

Each edge-time pair (e_j, t_j) denotes the traversal of edge $e_j = \{v_j, v_{j+1}\}$ at timestep t_j, which implies that $e_j \in E_{t_j}$. We require that $t_0 < t_1 < ... < t_{k-1}$, i.e., that the timesteps at which the consecutive edges of $\mathcal{W}$ are traversed are strictly increasing. We say that the walk starts at vertex v_0, and for all $i \in [k]$, we say that $\mathcal{W}$ visits *$v_i \in V(G)$.*

$\mathcal{W}$ is an *exploration schedule* of $\mathcal{G}$ with start vertex $s \in V(\mathcal{G})$ if $\mathcal{W}$ is a temporal walk in $\mathcal{G}$ that starts at s and visits all vertices $v \in V(G)$. Let $\mathcal{W}$ be an exploration schedule in a temporal graph $\mathcal{G}$ with underlying graph G. We denote by $a(\mathcal{W})$ the timestep at which $\mathcal{W}$ first visits the n-th unique vertex $v \in V(G)$; this is the *arrival time* of $\mathcal{W}$. If $\mathcal{W}$ satisfies $a(\mathcal{W}) \leq a(\mathcal{W}')$ for any other exploration schedule $\mathcal{W}'$ with the same start vertex in $\mathcal{G}$, then we say that $\mathcal{W}$ is *foremost*.

Definition 2.3 (Temporal Exploration). *An instance of the* Temporal Exploration *(TEXP) problem is given as a pair $(\mathcal{G}, s)$, where $\mathcal{G} = \langle G_1, G_2, ..., G_L \rangle$ is an arbitrary temporal graph on n vertices with lifetime $L \geq |V(\mathcal{G})|^2 = n^2$, and $s \in V(\mathcal{G})$ is a start vertex. The problem asks that we compute an exploration schedule $\mathcal{W}$ such that $\mathcal{W}$ is foremost and starts at vertex s. It is assumed that G_t ($t \in [L]$) is known to an algorithm prior to it computing a solution.*

It was proven in [6] that arbitrary always-connected temporal graphs admit at least one exploration schedule $\mathcal{W}$ such that $\alpha(\mathcal{W}) \leq n^2$. Hence having $L \geq |V(\mathcal{G})|^2$ ensures that an exploration schedule exists.

Definition 2.4 (k-edge-deficient). *Let $\mathcal{G} = \langle G_1, ..., G_L \rangle$ be a temporal graph with underlying graph $G = (V, E)$ and order $n = |V|$. Then $\mathcal{G}$ is k-edge-deficient (for $k \in \mathbb{N}$) if, for all $t \in [L]$, we have $G_t = (V, E - X_t)$ for some $X_t \subseteq E$ with $|X_t| \leq k$.*

When constructing a walk in a k-edge-deficient temporal graph $\mathcal{G}$, we may speak of an agent *following* a walk W in the underlying graph G. By this, we mean that the agent traverses in $\mathcal{G}$ the edges in the same order as they are traversed by W, and does this whenever it is possible to do so, i.e., whenever the next edge e traversed by W is present in the current timestep t. If that edge is not present, the agent is *blocked on e in step t*. For always-connected k-edge-deficient temporal graphs we require that $G_t = (V, E - X_t)$ is connected for all $t \in [L]$. We consider only always-connected, k-edge-deficient temporal graphs with finite lifetime $L \geq n^2$ – as such, any temporal graph we refer to (unless stated otherwise) is assumed to hold these properties. The following lemma from [6] will be useful.

Lemma 2.5 (Reachability lemma; Erlebach et al. [6]). *Let $\mathcal{G}$ be an arbitrary always-connected temporal graph with vertex set V and lifetime L. Then an agent situated at any vertex $u \in V$ at any time $t \leq L - n$ can reach any other vertex $v \in V$ in at most $|V| - 1 = n - 1$ steps, i.e., by time step $t + n - 1$.*

3 $O(kn \log n)$-Time Exploration of k-Edge-Deficient Temporal Graphs

We present an algorithm that proceeds in rounds. In each round, it considers a forest consisting of $k+1$ edge-disjoint subtrees of a spanning tree of the underlying graph and ensures that all edges of one of these trees can be traversed in the round. The following lemma allows us to split a tree T into a pair of edge-disjoint subtrees (whose union covers $E(T)$) in a balanced way:

Lemma 3.1. *Let T be a tree with $m \geq 2$ edges. Then one can compute two edge-disjoint subtrees T' and T'' such that $|E(T')|, |E(T'')| \in [m/3, 2m/3]$, and such that $E(T') \cup E(T'') = E(T)$.*

Say that a set S of edge-disjoint subtrees $T' \subseteq F$ is a *subtree-cover* of a forest F if, for every $e \in E(F)$ we have $e \in E(T')$ for some $T' \in S$. Call such a subtree-cover S *balanced* if it satisfies the additional property that the tree of largest size in S contains at most three times the number of edges contained by the smallest. By applying Lemma 3.1 to the largest tree in a balanced sub-tree cover, we can show the following lemma:

Lemma 3.2. *Let S be a balanced subtree-cover of some forest F such that $|S| = x$ and $|E(F)| \geq x + 1$ hold. Then one can obtain a balanced subtree cover S' of F such that $|S'| = x + 1$.*

Theorem 3.3. *Let $\mathcal{G} = \langle G_1, ..., G_L \rangle$ be an always-connected, k-edge-deficient temporal graph (for some $k \in \mathbb{N}$) with underlying graph G, and let $|V(G)| = n$. Then, for any start vertex s, there is an exploration schedule $\mathcal{W}$ of $\mathcal{G}$ with $a(\mathcal{W}) = O(kn \log n)$. Moreover, such a schedule can be computed in polynomial time.*

Proof. For $k \geq n - 1$ the result clearly holds as every always-connected temporal graph can be explored in $\leq n(n-1)$ time steps (by repeated application of Lemma 2.5 [6]), so we assume $k < n - 1$ for the rest of the proof.

Compute an arbitrary spanning tree T of G, and let $m = |E(T)|$ – assume w.l.o.g. that $m > k + 1$, otherwise G can be explored in $O(kn)$ steps via $O(k)$

applications of Lemma 2.5. Let $S = \{T\}$ and note that S is a balanced subtree-cover of T. Now apply Lemma 3.2 to S k times to obtain a balanced subtree-cover S^* of size $k + 1$ (possible since $k \leq n - 2$). Let F denote a forest containing all subtrees induced by edges of T that may not yet have been traversed, initially $F = T$.

We now specify our algorithm in terms of an agent that explores the graph in consecutive *rounds*. We denote by t the first step of a given round, and by v the vertex at which the agent is positioned at the beginning of timestep t. Let $m' = \sum_{T_i \in S^*} |E(T_i)|$. At the beginning of the first round $t = 1$, $v = s$, $F = \{T\}$, S^* is a balanced subtree-cover of F (with size $k + 1$), and $m' = m$. While F contains more than $k + 1$ edges, execute a round as follows: Consider the graph from step $t + n$ onward, and place a single virtual agent at an arbitrary vertex v_i in each of the $k + 1$ subtrees $T_i \in S^*$. For each $i \in [k + 1]$, compute an Euler tour of T_i starting from vertex v_i, then let the agents follow the Euler tours of their respective trees for the following $6m'$ steps. Since there are $k + 1$ virtual agents following tours in edge-disjoint subtrees, and since $\mathcal{G}$ is k-edge-deficient, it follows that there are no edges missing from at least one subtree $T' \in S^*$ in every step. Let T_{i^*} be the subtree that had no edges missing during the largest number of steps in the considered $6m'$-step period. Then T_{i^*} had no edge missing for $\geq \frac{6m'}{k+1}$ steps. Since $|S^*| = k + 1$, the smallest tree in S^* cannot contain $> \frac{m'}{k+1}$ edges, so because S^* is balanced the largest tree in S^* contains $\leq \frac{3m'}{k+1}$ edges. Therefore, the $\geq \frac{6m'}{k+1}$ steps in which the virtual agent positioned in T_{i^*} is able to traverse an edge are enough for that agent to complete their Euler tour of T_{i^*} and arrive back at v_{i^*}. Using the steps in the interval $[t, t + n - 1]$, move the real agent, using Lemma 2.5, from v to the vertex v_{i^*} at which the virtual agent began their tour of T_{i^*}. Let W^* be the tour followed by the virtual agent positioned in T_{i^*}; from step $t + n$ to step $t' = t + n + 6m' - 1$, let the real agent complete W^*. Once completed, check if $> k + 1$ edges remain untraversed; if so, consider the set $S' = S^* - \{T_{i^*}\}$ and note that $|S'| = k$. Observe that S' is balanced since S^* was balanced and removing a tree cannot violate this property. Since we have $S' = S^* - \{T_{i^*}\}$, and since S^* covered T, we have that S' covers the forest F' obtained from F by removing the edges of T_{i^*}. Apply Lemma 3.2 to S' to obtain a balanced subtree-cover S'' of F' such that $|S''| = k + 1$ – note that doing so is valid since $|E(F')| > k + 1 = |S'| + 1$, as is required by Lemma 3.2. Now, set $S^* = S''$, $F = F'$, $v = v_{i^*}$ and $t = t' + 1$ and start the next round as above. Once a round is completed and at most $k + 1$ edges remain, stop and use $O(n)$ steps to explore up to $2k + 2$ remaining unexplored vertices one by one using Lemma 2.5.

Note that every vertex v in $V(T) = V(G)$ either (1) belongs to an edge of T that was traversed by the algorithm, or (2) was visited via an application of Lemma 2.5. Hence, the computed walk is an exploration schedule and it remains only to bound its arrival time. In each round, a subtree containing at least a $\frac{1}{3(k+1)}$ fraction of the edges of F is traversed in its entirety. To see this, observe that $|S^*| = k + 1$, so the largest tree in S^* must contain $\geq \frac{m'}{k+1}$ edges; because S^* is balanced, it follows that all trees in S^* have size $\geq \frac{m'}{3(k+1)}$.

Hence, after x rounds, the total number of edges in T that have not yet been removed from F is $\leq m(1-\frac{1}{3(k+1)})^x$. Thus, after $x = 3(k+1)\ln(\frac{m}{k+1}) = O(k \log m) = O(k \log n)$ (recall that $m = |E(T)| = n-1$) rounds there are $\leq m(1-\frac{1}{3(k+1)})^{3(k+1)\ln(\frac{m}{k+1})} \leq k+1$ unexplored edges remaining in F. As each round takes $n + 6m' \leq n + 6m = O(n)$ steps, the total number of steps after $O(k \log n)$ rounds is $O(kn \log m) = O(kn \log n)$. A further at most $(2k+2)n$ steps are needed to explore up to $2k+2$ remaining unvisited vertices. Hence, the entire exploration takes $O(kn \log n) + (2k+2)n = O(kn \log n)$, as required.

Finally, it is easy to see that the algorithm for determining the exploration schedule can be implemented to run in polynomial time. □

4 Linear-Time Exploration of 1-Edge-Deficient Temporal Graphs

A graph $G = (V, E)$ is *k-vertex-connected* (or simply k-connected) if, for any subset $X \subseteq V(G)$ such that $|X| < k$, the subgraph of G induced by $V - X$ is connected. Let $G = (V, E)$ be a connected graph. An edge $e \in E$ is a *bridge* if $G' = (V, E - \{e\})$ is disconnected. A graph $G = (V, E)$ is *2-edge-connected* if it is connected and does not contain a bridge. A 2-edge-connected component (abbreviated 2-ecc) of a graph G is a vertex-maximal induced subgraph $C \subseteq G$ such that C is 2-edge-connected. Note that a 2-ecc can also be a single vertex. We say that a spanning subgraph G'' of G *preserves 2-edge-connectivity* if it contains all bridges of G and, for every 2-ecc C of G, the subgraph of G'' induced by $V(C)$ is 2-edge-connected. In order to show that every connected graph G has a spanning subgraph that preserves 2-edge-connectivity and has only a linear number of edges, we make use of the following result by Nagamochi and Ibaraki.

Theorem 4.1 (Nagamochi and Ibaraki, [18]). *Every k-connected graph $G = (V, E)$ admits a k-connected spanning subgraph $G' = (V', E')$ such that $|E'| \leq k|V|$. Moreover, G' can be computed in $O(|E|)$-time.*

By applying Theorem 4.1 to each biconnected component of a given connected graph G, we can show the following:

Lemma 4.2. *Let G be an arbitrary connected graph and let $\mathcal{C}$ be the set of all 2-eccs of G. Then, G admits a spanning subgraph G^* such that (1) the vertices of each 2-ecc $C \in \mathcal{C}$ form a 2-ecc C^* in G^* with $|E(C^*)| \leq 5|V(C^*)|$; (2) $|E(G^*)| \leq 5|V(G)|$; and (3) $V(G^*) = V(G)$.*

If $\mathcal{G}$ is a 1-edge-deficient, always-connected temporal graph with underlying graph G and G^* is a spanning subgraph of G that preserves 2-edge-connectivity, then the temporal graph $\mathcal{G}^*$ with underlying graph G^* that is obtained from $\mathcal{G}$ by removing all edges that are not in G^* is also always-connected and 1-edge-deficient. This also implies that every cycle C of G^* induces a connected subgraph in every timestep of $\mathcal{G}^*$.

A *circuit* C in a graph G is a closed walk in G that does not repeat edges. In 1-deficient temporal graphs, a circuit behaves like an always-connected temporal graph with underlying cycle, as at most one edge of the circuit can be missing in each step. Thus, we get the following theorem, which was shown in [6] for always-connected cycles.

Theorem 4.3 (Erlebach, Hoffmann and Kammer, [6]). *For every* 1*-edge deficient temporal graph* $\mathcal{G}$ *with underlying circuit* C*, there exists a start vertex from which the graph can be explored in at most* $|E(C)|-1$ *steps.*

The following theorem by Fan allows us to reduce the exploration of a 2-ecc to the exploration of at most three circuits.

Theorem 4.4 (Fan, [10]). *The edges of any 2-edge-connected graph* $G=(V,E)$ *can be covered by at most* 3 *circuits. Moreover, such a cover can be computed in* $O(|E|\cdot|V|)$*-time.*

The edges which belong to no 2-ecc of an arbitrary connected graph G are precisely the bridges of G. Hence, one can represent the structure of G as a tree T, called the 2*-ecc tree* of G, by identifying each 2-ecc with a vertex, and joining two vertices by an edge in T if and only if their corresponding 2-eccs are connected by a bridge in G. In the proof of Theorem 4.6, we will therefore re-use standard terminology for trees: We choose a 2-ecc C as the *root component*. If C' and C'' are 2-eccs such that C' lies on the path from C to C'' in T, then C'' is a *descendant* of C'. If C' and C'' correspond to neighbouring nodes in T and C'' is a descendant of C', then C'' is a *child* of C' and C' is the *parent* of C''. The *subtree* rooted at a 2-ecc C' consists of all 2-eccs that are descendants of C', and the subgraph of G consisting of all those 2-eccs and the bridge edges between them is said to *correspond* to that child subtree. For any child C' of the root C of the 2-ecc tree, we call the subgraph of G that corresponds to the subtree rooted at C' a *child subgraph.*

Lemma 4.5. *Let* G *be an arbitrary connected graph on* n *vertices. Then, there is a* 2*-ecc* C^* *of* G *such that rooting the* 2*-ecc tree of* G *at* C^* *ensures that the child subgraphs (i.e., the subgraphs of* G *corresponding to the subtrees rooted at children of* C^**) each contain at most* $n/2$ *vertices.*

Proof. Consider the tree T obtained by identifying each 2-edge-connected component C of G with a vertex v_C. Root T at an arbitrary node $v_{C'}$, then process the vertices in a bottom up manner, labelling a vertex v_C with the integer $x_C=|\{u\in V(G): u\in V(C') \text{ for a descendant } C' \text{ of } C \text{ in } T\}|$. Select a vertex v_{C^*} such that $x_{C^*}\geq n/2$ and such that v_{C^*} has largest depth in T amongst all such vertices. If v_{C^*} is already the root of T, we are done. Otherwise, let $v_{C'}$ be the parent of v_{C^*} and reroot T at v_{C^*} to form a 2-ecc tree T^*, in which $v_{C'}$ is a child of v_{C^*}. We have that for every child $v_C\neq v_{C'}$ of v_{C^*} in T^* we have $x_C<n/2$, because otherwise the algorithm would have picked v_C rather than v_{C^*}. Furthermore, we have $x_{C^*}\geq n/2$, and so the total number of vertices in all components C'' such that $v_{C''}$ is a descendant of $v_{C'}$ in T^* must be $\leq n/2$. □

Theorem 4.6. *Let $\mathcal{G} = \langle G_1, ..., G_L \rangle$ be an always-connected, 1-edge-deficient temporal graph with arbitrary underlying graph G, and let $|V(G)| = n$. Then, for any start vertex s, there is an exploration schedule $\mathcal{W}$ of $\mathcal{G}$ with $a(\mathcal{W}) = O(n)$. Moreover, such a schedule can be computed in polynomial time.*

Proof. Apply Lemma 4.2 to G in order to obtain a spanning subgraph $G^* \subseteq G$ (with $|E(G^*)| \leq 5n$) such that each 2-ecc C of G forms a 2-ecc C^* in G^* with $|E(C^*)| \leq 5|V(C^*)|$. Apply Lemma 4.5 to G^* to obtain a 2-ecc tree T of G^* with a root component C_1 such that the child subgraphs $G_i \subseteq G^*$ satisfy $|V(G_i)| \leq n/2$. Let k denote the number of 2-eccs in G^*. Let $T(n, k)$ denote the maximum number of timesteps required to explore an arbitrary 1-edge-deficient, always-connected temporal graph on n vertices whose underlying graph has k 2-eccs, at most $5n$ edges, and is such that every 2-ecc C^* satisfies $|E(C^*)| \leq 5|V(C^*)|$, starting from an arbitrary vertex s in the graph at timestep 1. We now specify our exploration algorithm and prove by induction on k that $T(n, k) \leq 164n$.

Base case (Arbitrary n, $k = 1$): G^* consists of a single 2-ecc C_1; without loss of generality assume that $|V(C_1)| \geq 3$. Apply Theorem 4.4 to C_1, obtaining a circuit cover $\{X_1, ..., X_c\}$ of C_1 containing c circuits, where $1 \leq c \leq 3$. Consider now the following 3 time intervals, noting that $|E(X_i)| \leq |E(C_1)| \leq 5n$ for all $i \in [3]$: $I_1 = [n+1, 6n], I_2 = [7n+1, 12n]$ and $I_3 = [13n+1, 18n]$. During the steps of I_i apply Theorem 4.3 to X_i to determine a vertex $v_i \in X_i$ such that an exploration schedule of X_i using at most $|E(X_i)| - 1 \leq 5n - 1$ timesteps begins at v_i at the first step of I_i. Beginning at the start vertex $s \in V(G)$ in timestep 1, employ Lemma 2.5 to move in at most n steps to vertex v_1, wait until the first step of interval I_1, then follow the walk obtained by the application of Theorem 4.3 during interval I_1. If $c > 1$, repeat these steps for all remaining circuits X_i in the computed circuit cover of C_1. Once Theorem 4.3 has been applied to X_c, notice that, for all $i \in [c]$, all vertices of X_i have been visited. Since $\{X_1, ..., X_c\}$ covers all edges of C_1 (and also all edges of G^* since G^* consists only of C_1), it follows that all vertices of G^* have been visited at least once. The number of timesteps taken to achieve this is at most $c(n + 5n) \leq 18n$.

Inductive step (Arbitrary n, $k > 1$): Assume that $T(n, j) \leq 164n$ for all $j < k$ and consider the root component C_1 of G^*. We now distinguish two cases:

Case 1: $|C_1| \geq 2$. Apply Theorem 4.4 to C_1 and obtain a circuit cover $X^* = \{X_1, ..., X_c\}$ of C_1 containing c circuits, where $1 \leq c \leq 3$. Let $V' = \{v \in V(C_1) : v \in e \text{ for some bridge } e\}$. Construct a function $\alpha : V' \to X^*$ by arbitrarily mapping each vertex $v \in V'$ to some circuit $X_i \in X^*$ such that $v \in X_i$. Recall that we root the 2-ecc tree T of G^* at C_1. For each child C_i of C_1 in T, we denote by G_i the child subgraph of G^* corresponding to the subtree of T rooted at C_i. Let $\mathsf{Br} = \{e \in E(G^*) : e \text{ is a bridge and } e \cap V' \neq \emptyset\}$ and, for any $v \in V'$, let $\beta(v) = \{G_i : \{v, u\} \in \mathsf{Br} \text{ for some } u \in G_i\}$. For $i \in [3]$, let $F_i = \bigcup_{\{v \in V' : \alpha(v) = X_i\}} \beta(v)$.

Let $G_{X_i} \subseteq G^*$ be the subgraph of G^* induced by $V(X_i \cup F_i)$ ($i \in [c]$). For each $i \in [c]$, we construct a closed walk in G_{X_i} that will be followed (in opposite directions) by two virtual agents. Both agents start at some arbitrary vertex $s_i \in V(X_i)$ and follow the walk in opposite directions whenever possible, i.e., whenever they are not blocked on the next edge they need to cross. Starting at some timestep t_i, let the agents do the following: Move along the edges of X_i, one in the clockwise direction (agent CW) and the other in the counter-clockwise direction (agent CCW). Whenever either agent reaches for the first time a vertex $v \in V'$ such that $\alpha(v) = X_i$ the agent descends into each $G_j \in \beta(v)$ via the bridge connecting it and vertex $v \in X_i$, and explores G_j via a depth-first search. The only exception is the vertex s_i: If $s_i \in V'$ and $\alpha(s_i) = X_i$, then agent CW descends into each $G_j \in \beta(s_i)$ immediately at the start of the walk (before traversing any edge of X_i), while agent CCW does so only when it returns to s_i after having traversed all edges of X_i. Agent CW processes the subgraphs in $\beta(v)$ in increasing order of their indices, whilst agent CCW processes them in decreasing order of their indices. Once an agent has explored all subgraphs $G_j \in \beta(v)$, then that agent attempts to cross the next edge in X_i. Both agents continue this until the first timestep in which both agents are blocked on the same edge e. If every edge of G^* were to be present in every timestep, it would take each agent at most $\mathsf{Exp}(X_i) = |E(X_i)| + \sum_{G_j \in F_i} 2|V(G_j)|$ steps to carry out their respective walks in G_{X_i}: 1 step to traverse each of the edges of X_i, $2|V(G_j)| - 2$ steps spent exploring G_j via a DFS, and 2 steps spent traversing the bridge edges connecting X_i and each $G_j \in F_i$. Since G^* is 1-edge-deficient, it is possible for the agents to both be blocked on the same edge during the same timestep. We distinguish three subcases as follows. Recall that t_i denotes the timestep in which the exploration of G_{X_i} begins. We use t_i' to denote an upper bound on the timestep by which the exploration of G_{X_i} (possibly except one subgraph, see below for details) is completed by at least one of the two agents.

Case 1.1: If the agents are never blocked on the same edge e during any step t in $[t_i, t_i']$ for $t_i' = t_i + 2\mathsf{Exp}(X_i)$, then, in each timestep $t \in [t_i, t_i']$, at least one of the two agents is able to cross the next edge of their respective walk. In this case, we have that by the end of timestep t_i', the agent that was blocked on an edge in the least number of timesteps $t \in [t_i, t_i']$ will have not been blocked in $\geq \mathsf{Exp}(X_i)$ timesteps and, as such, will have completed their exploration of G_{X_i}.

It remains to consider the situation that the agents are blocked on the same edge of G^* during some timestep in $[t_i, t_i']$, where $t_i' = t_i + 3\mathsf{Exp}(X_i)$ in Case 1.2 and $t_i' = t_i + 4\mathsf{Exp}(X_i)$ in Case 1.3. Consider the timestep t in which the agents are first both blocked on the same edge e.

Case 1.2: $e \in X_i$. Check whether or not e is present during any step $t' \in [t+1, t+|E(X_i)|]$. If yes, wait until that step, then let both agents cross e. If not, let both agents apply Lemma 2.5 in X_1, using at most $|E(X_1)| - 1$ timesteps to move to the opposite endpoint of e, then continue attempting to traverse the next edge of their walk whenever possible. Notice that, during any step $t' \in [t_i, t-1]$, at least one of the two agents was able to cross the next edge in their respective walk, since t is the first timestep in which both agents are blocked on the same edge. When the agents are blocked on e during step t, they either wait at their current vertex for at most $|E(X_i)| - 1$ steps until e is present again, or spend $\leq |E(X_i)| - 1$ steps reaching the opposite endpoint of e by applying Lemma 2.5 in X_i. In either case, it takes at most $|E(X_i)| - 1$ steps for them to reach the opposite endpoint of e. At this point, observe that the vertices $x \in V(X_i)$ and the $G_j \in F_i$ that remain to be explored/processed by agent CW are exactly those that have already been explored/processed by agent CCW (and vice versa). Hence, it follows that the two agents will not be blocked on the same edge again for the remainder of their walks. In all remaining steps, since the sets of vertices unexplored by the walks of the two agents are disjoint, we again have that at least one of the two agents will be able to cross the next edge of their respective walk in all steps $t' \in [t+|E(X_i)|, t'_i]$. Concluding, during the entire time interval $[t_i, t'_i]$, there are $\leq |E(X_i)|$ steps in which neither of the agents can cross the next edge of their respective walk, and by step $t'_i \leq t_i + 2\mathsf{Exp}(X_i) + |E(X_i)| \leq t_i + 3\mathsf{Exp}(X_i)$, it is ensured that the agent who was blocked during the least number of steps since the start of step t_i has completed their exploration of G_{X_i} in at most $3\mathsf{Exp}(X_i)$ steps.

Case 1.3: $e \in G_j$ for some $G_j \in F_i$. Let $b = \{v, u\}$, where $\{v, u\} \in \mathsf{Br}$, $v \in X_i$ satisfies $\alpha(v) = X_i$, and $u \in V(G_j)$. Consider the timestep $t \in [t_i, t'_i]$, during which the two agents are first blocked on e. Let $t_1^*, t_2^* \in [t_i, t'_i]$ denote respectively the timesteps at which the first agent (say agent A_1) and second agent (agent A_2) traverse the edge b from v toward u – clearly we have $t_1^* \leq t_2^* < t$, since $e \in E(G_j)$ and any vertex in $V(G_j)$ can only be reached from X_i by traversing b. We now retrospectively alter the walks of both agents: First, change the walk of A_1 so that, during the interval $[t_1^*, t_2^* - 1]$, A_1 waits at vertex v. Now, change the walks of both A_1 and A_2 during the steps $[t_2^*, t'_i]$, so that they both do not process subgraph G_j, but continue their exploration of X_i and all $G_{j'} \in F_i$ such that $G_{j'} \neq G_j$. We claim that $t_2^* \leq t_i + 2\mathsf{Exp}(X_i)$. To see this, observe that $t \leq t_i + 2\mathsf{Exp}(X_i)$ since, if $t > t_i + 2\mathsf{Exp}(X_i)$, the two agents will not have been blocked on the same edge during any of the steps $[t_i, t_i + 2\mathsf{Exp}(X_i)]$, and so the agent who was blocked on an edge in the least amount of steps during this interval would have traversed an edge of their walk in $\geq \mathsf{Exp}(X_i)$ timesteps – enough to have finished the entire exploration of G_{X_i}. Hence we have $t_2^* \leq t < t_i + 2\mathsf{Exp}(X_i)$, as required. Both agents can then continue following their respective walks during the interval $[t_2^*, t'_i]$ without the possibility of being blocked on the same edge again; by our earlier reasoning this requires of the agent that is blocked during the least number of steps in this period another $\leq 2\mathsf{Exp}(X_i)$ steps. Concluding, one of the two agents will have visited all vertices in $G_{X_i} \setminus V(G_j)$ by the end of step $t_2^* + 2\mathsf{Exp}(X_i) \leq t_i + 4\mathsf{Exp}(X_i)$.

In all three subcases, one of the two agents has explored all vertices of G_{X_i}, except possibly those of a single subgraph G_j of G_{X_i}, in at most $4\mathsf{Exp}(X_i)$ timesteps, and we will let the real agent follow that agent's walk.

After processing all c circuits X_i in this way, there will be at most c subgraphs that have not yet been explored. We next reduce those unexplored subgraphs to at most one: While there are two or more unexplored subgraphs, we repeatedly (1) choose a circuit X in C_1 that contains two vertices of V' that have a bridge to an unexplored subgraph (note that a circuit X such that $|E(X)| \leq 2|V(C_1)|$ must exist), and then (2) process X and the two unexplored subgraphs in the same way as we processed X_i for $1 \leq i \leq c$ above.

After this, there will be at most a single subgraph G_j corresponding to a child subtree rooted at a child of C_1 in the 2-ecc tree that is not yet explored. That subgraph has at most $n/2$ vertices (by choice of C_1) and has at most $k-1$ 2-eccs (because it does not contain the 2-ecc C_1). We now apply the inductive hypothesis to explore G_j recursively in at most $164 \cdot n/2 = 82n$ steps.

To bound the overall number of timesteps, we assume that $c = 3$, that 3 subgraphs remain unexplored after processing G_{X_i} for $i \in [3]$, that two iterations of the procedure for reducing the number of unexplored subgraphs are needed, and that a recursive call needs to be made to explore the final unexplored subgraph. We omit the details, but one can straightforwardly show (via a case analysis) that this is the worst case for the total number of steps needed to complete the exploration.

The whole exploration then consists of the following parts: At most n steps to move from s to a vertex v_1 in X_1; at most $4\mathsf{Exp}(X_1)$ steps to explore G_{X_1} apart from at most one child subgraph G_j. Another at most $n + 4\mathsf{Exp}(X_2)$ steps to do the same for G_{X_2} (where the first n steps allow the agent to move from the vertex where the exploration of G_{X_1} ends to a vertex in X_2), and another at most $n + 4\mathsf{Exp}(X_3)$ steps to do the same for G_{X_3}. Then, at most twice: n steps to move to a vertex in a circuit X (recall that $|E(X)| \le 2|V(C_0)|$) and $4\mathsf{Exp}(X)$ steps to explore it and at least one of the two subgraphs attached to it. Finally, $\le n$ steps are needed to move to a vertex in the last unexplored subgraph G_j, and another $\le 82n$ steps are required to explore that subgraph recursively.

As $\mathsf{Exp}(X_i) = |E(X_i)| + \sum_{G_j \in F_i} 2|V(G_j)|$, we have $\sum_{i=1}^{3} \mathsf{Exp}(X_i) \le 3|E(C_1)| + \sum_{i=1}^{3} \sum_{G_j \in F_i} 2|V(G_j)| \le 15|V(C_1)| + 2\sum_{i=1}^{3} \sum_{G_j \in F_i} |V(G_j)| \le 15n$. Furthermore, for any circuit X in C_1 with two subgraphs G_1 and G_2 attached via bridges, we have $\mathsf{Exp}(X) \le 2|V(C_1)| + 2|V(G_1)| + 2|V(G_2)| \le 2n$. Thus, the total exploration time is at most $6n + 4 \cdot 15n + 8 \cdot 2n + 82n = 82n + 82n = 164n$.

Case 2: $|C_1| = 1$. In this case we apply a similar technique to that used in Case 1, but this case is simpler as the root component consists of a single vertex and all child subgraphs are attached to that same vertex via bridges.

Finally, we remark that all steps in the construction of the exploration schedule can be implemented in polynomial time. □

5 Lower Bound

To complement the upper bounds from Sections 3 and 4, we also present a lower bound on the worst-case exploration time of k-edge-deficient temporal graphs.

Theorem 5.1. *For arbitrarily large n and every k with $2 \le k \le \frac{n}{2} - 1$, there is a k-edge-deficient temporal graph with n vertices for which an optimal exploration takes $\Omega(n \log k)$ steps.*

The theorem can be shown by adapting the construction of a lower bound of $\Omega(n \log n)$ on the exploration time of temporal graphs with underlying planar graphs of maximum degree 4 from [6, Theorem 2]. That construction has a time-varying part (in which $n/2$ edges are missing in each step) and a fixed part (a static path of $n/2$ edges). By reducing the size of the time-varying part and increasing the size of the static part, we obtain Theorem 5.1

6 Conclusion

We have shown that always-connected k-edge-deficient temporal graphs admit an exploration schedule $\mathcal{W}$ with arrival time $O(kn \log n)$; if $k = 1$, the arrival time improves to $O(n)$. The provided proofs are both constructive, yielding polynomial-time algorithms for computing such exploration schedules. As $n - 1$ steps are necessary to explore any graph, our results also yield $O(k \log n)$ and $O(1)$-approximation algorithms for TEXP for the $k \in \mathbb{N}$ and $k = 1$ cases, respectively, as well as an $O(\log n)$-approximation if $k = O(1)$. Furthermore, we gave an infinite family of k-edge-deficient temporal graphs that require $\Omega(n \log k)$ timesteps to be explored. It would be interesting to close the gap between the lower and upper bounds. In particular, an interesting question is whether always-connected k-edge-deficient graphs for $k = O(1)$ can be explored in $O(n)$ steps.

References

1. Akrida, E.C., Mertzios, G.B., Spirakis, P.G.: The temporal explorer who returns to the base. In: Heggernes, P. (ed.) 11th International Conference on Algorithms and Complexity (CIAC 2019). Lecture Notes in Computer Science, vol. 11485, pp. 13–24. Springer (2019). https://doi.org/10.1007/978-3-030-17402-6_2
2. Bodlaender, H.L., van der Zanden, T.C.: On exploring always-connected temporal graphs of small pathwidth. Information Processing Letters **142**, 68–71 (2019). https://doi.org/10.1016/j.ipl.2018.10.016
3. Brodén, B., Hammar, M., Nilsson, B.J.: Online and Offline Algorithms for the Time-Dependent TSP with Time Zones. Algorithmica **39**(4), 299–319 (2004). https://doi.org/10.1007/s00453-004-1088-z
4. Bui-Xuan, B., Ferreira, A., Jarry, A.: Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks. Int. J. Found. Comput. Sci. **14**(2), 267–285 (2003). https://doi.org/10.1142/S0129054103001728
5. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. Int. J. Parallel Emergent Distributed Syst. **27**(5), 387–408 (2012). https://doi.org/10.1080/17445760.2012.668546
6. Erlebach, T., Hoffmann, M., Kammer, F.: On temporal graph exploration. Journal of Computer and System Sciences **119**, 1–18 (2021). https://doi.org/10.1016/j.jcss.2021.01.005
7. Erlebach, T., Kammer, F., Luo, K., Sajenko, A., Spooner, J.T.: Two Moves per Time Step Make a Difference. In: Baier, C., Chatzigiannakis, I., Flocchini, P., Leonardi, S. (eds.) 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 141:1–141:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.141

8. Erlebach, T., Spooner, J.T.: Faster Exploration of Degree-Bounded Temporal Graphs. In: Potapov, I., Spirakis, P., Worrell, J. (eds.) 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 117, pp. 36:1–36:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.MFCS.2018.36
9. Erlebach, T., Spooner, J.T.: Non-strict Temporal Exploration. In: Richa, A.W., Scheideler, C. (eds.) 27th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2020). Lecture Notes in Computer Science, vol. 12156, pp. 129–145. Springer (2020). https://doi.org/10.1007/978-3-030-54921-3_8
10. Fan, G.: Covering graphs by cycles. SIAM Journal on Discrete Mathematics **5**(4), 491–496 (1992). https://doi.org/10.1137/0405039
11. Gotoh, T., Flocchini, P., Masuzawa, T., Santoro, N.: Tight Bounds on Distributed Exploration of Temporal Graphs. In: Felber, P., Friedman, R., Gilbert, S., Miller, A. (eds.) 23rd International Conference on Principles of Distributed Systems (OPODIS 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 153, pp. 22:1–22:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2020). https://doi.org/10.4230/LIPIcs.OPODIS.2019.22
12. Gotoh, T., Sudo, Y., Ooshita, F., Masuzawa, T.: Dynamic ring exploration with (H,S) view. Algorithms **13**(6) (2020). https://doi.org/10.3390/a13060141
13. Ilcinkas, D., Klasing, R., Wade, A.M.: Exploration of Constantly Connected Dynamic Graphs Based on Cactuses. In: Halldórsson, M.M. (ed.) 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO 2014). Lecture Notes in Computer Science, vol. 8576, pp. 250–262. Springer (2014). https://doi.org/10.1007/978-3-319-09620-9_20
14. Ilcinkas, D., Wade, A.M.: Exploration of the T-interval-connected dynamic graphs: The case of the ring. In: Moscibroda, T., Rescigno, A.A. (eds.) 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2013). Lecture Notes in Computer Science, vol. 8179, pp. 13–23. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03578-9_2
15. Kempe, D., Kleinberg, J.M., Kumar, A.: Connectivity and Inference Problems for Temporal Networks. J. Comput. Syst. Sci. **64**(4), 820–842 (2002). https://doi.org/10.1006/jcss.2002.1829
16. Michail, O.: An introduction to temporal graphs: An algorithmic perspective. In: Zaroliagis, C., Pantziou, G., Kontogiannis, S. (eds.) Algorithms, Probability, Networks, and Games: Scientific Papers and Essays Dedicated to Paul G. Spirakis on the Occasion of His 60th Birthday, pp. 308–343. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-24024-4_18
17. Michail, O., Spirakis, P.G.: Traveling salesman problems in temporal graphs. Theor. Comput. Sci. **634**, 1–23 (2016). https://doi.org/10.1016/j.tcs.2016.04.006
18. Nagamochi, H., Ibaraki, T.: A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. Algorithmica **7**(5&6), 583–596 (1992). https://doi.org/10.1007/BF01758778
19. Shannon, C.E.: Presentation of a maze-solving machine. In: Sloane, N.J.A., Wyner, A.D. (eds.) Claude Elwood Shannon – Collected Papers, pp. 681–687. IEEE Press (1993)
20. Zschoche, P., Fluschnik, T., Molter, H., Niedermeier, R.: The complexity of finding small separators in temporal graphs. Journal of Computer and System Sciences **107**, 72–92 (2020). https://doi.org/10.1016/j.jcss.2019.07.006