Local Certification of Reachability

Oliver Bachtler
TU Kaiserslautern
Kaiserslautern, Germany
bachtler@mathematik.uni-kl.de

Tim Bergner
TU Kaiserslautern
Kaiserslautern, Germany
bergner@mathematik.uni-kl.de

Sven O. Krumke
TU Kaiserslautern
Kaiserslautern, Germany
krumke@mathematik.uni-kl.de

ABSTRACT

Motivated by self-stabilising algorithms, the objective of local certification is to verify whether a (global) property holds while being restricted to a local view of the graph. Roughly speaking, this works as follows: a prover assigns a certificate to every vertex of a graph. Subsequently a verifier checks, for every vertex, whether its local view of the graph is consistent with the property we wish to verify. If this is the case at all vertices, then it accepts and it rejects otherwise. For a prover-verifier pair to locally certify a graph property, the verifier must accept any graph with this property that received certificates from the prover and it must reject any proof on a graph that does not have the desired property. The quality of such a pair is measured by the prover's size which is the length of the longest certificate it uses.

One common model for local certification is that of proof labelling schemes. This is a prover-verifier pair in which the verifier at a vertex v has access to the information at v as well as the certificates of all its neighbours. Determining whether an undirected graph has an st-path (for specified nodes s and t) can be done with a prover of size 1 and in the directed case it is known that $O(\log \Delta(G))$ bits suffice, where $\Delta(G)$ denotes the maximum degree of G. We prove a matching lower bound, in particular, showing that a constant amount of bits does not suffice.

KEYWORDS

Local Certification, Proof Labelling Schemes, Reachability

1 INTRODUCTION

The objective of local certification is to locally verify (global) properties in a distributed system. It is motivated by self-stabilising algorithms [4]. These algorithms are used in distributed systems which are subject to faults and have the property that they converge to a solution for a given problem. A possible way of designing such an algorithm is to first move to a solution and then to maintain it for as long as it remains correct. This so called local detection paradigm was introduced in [1]. Local certification describes this last step, where the algorithm needs to detect whether the solution is correct or not.

In local certification, a global prover has to convince a verifier that a graph has a specific property. To do so, the prover first presents a proof by assigning certificates to the vertices. Afterwards, the verifier decides at every vertex whether to accept or reject the proof provided. The decision at a vertex v is solely based on the local view of the graph around v, including certificates. If a graph has the designated property, the prover must be able to choose certificates in a way that makes the verifier accept at every vertex. Otherwise, the verifier must reject at some vertex of the graph, regardless of which certificates are given. What exactly the term local view means differs amongst the various local certification concepts.

To illustrate the concept, we sketch how local certification of bipartiteness works [10]. As local view, we assume that every vertex has access to its own certificate as well as those of its neighbours. In the case of a bipartite graph, the prover could specify a bipartition using the certificates 0 and 1. The verifier then only needs to check that the certificates of its neighbours differ from its own. If this is the case everywhere, then the graph is bipartite, so the verifier never accepts a non-bipartite graph. Furthermore, the bipartition specified by the prover makes the verifier accept. Hence, this prover-verifier pair locally certifies bipartiteness using a single bit.

Local certification does not restrict the computational power of the prover or verifier. Instead the quality is measured by the certificate lengths needed. The prover's size is the length of the longest certificate it assigns to a vertex. Given some property, a natural question to consider is what size a prover needs to have and what size suffices to locally certify this property.

The answer to this question may depend on the precise concept used since a "larger" local view of the verifier may result in smaller certificate lengths being sufficient. In this paper, we are interested mainly in two such concepts: proof labelling schemes and locally checkable proofs. These were introduced in [12] and [10], respectively. A formal definition is given in Section 2, but here it suffices to know that the verifiers in locally checkable proofs are more powerful that the ones in proof labelling schemes. Thus, ideally, one determines lower bounds on the prover's size using locally checkable proofs, and upper bounds using proof labelling schemes (since then they hold for both concepts).

Many results of this type are known, for example, certifying acyclicity [12] and planarity [6] requires $\Theta(\log n)$ bits, whilst minimum spanning trees need $\Theta(\log n \log W)$ bits [11], where n is the order of the graph and W is the largest weight of an edge. These bounds are for proof labelling schemes, though the planarity result also works for locally checkable proofs. That $\Theta(\log n)$ bits are required for acyclicity is also true in the locally checkable proof setting is was open until recently and is, in fact, also shown in [6]. For more results we refer to [5], a recent survey of local certification.

A very basic problem is that of certifying whether an st-path exists, for two specified vertices s and t, which we refer to as the st-reachability problem. In [10] it is shown how to solve this for undirected graphs using a single bit. In the directed case, a prover using $O(\log \Delta(G))$ bits, where $\Delta(G)$ in the maximum degree of the graph G, is described. Whether a constant size proof exists is posed

as an open question. We recapitulate how these locally checkable proofs work at the start of Section 3 and why the directed case cannot be solved analogously.

Another paper that looks at locally certifying st-reachability is [7]. Here the authors use a significantly more restrictive model in which the verifier sees less information, making it weaker. To obtain a logarithmic lower bound for st-reachability, they additionally restrict to a one-way communication model in which vertices only see their predecessors. The proof heavily relies on this restriction and a lower bound for the standard two-way communication is left open.

This suggests that the directed version is harder than the undirected one, which is a common occurrence: In the k disjoint paths problem the task to find vertex-disjoint paths between k pairs of specified vertices. It can be solved in polynomial time on undirected graphs for any fixed k [13] but it is NP-complete in the directed case, even for $k = 2$ [8]. Similarly, the feedback arc set problem is NP-complete [9] while its undirected counterpart is solved by computing a spanning tree. Another result [2] demonstrates that undirected st-reachability can be expressed by existential monadic second order logic and directed reachability cannot.

We show that locally certifying st-reachability (in the proof labelling scheme setting) is another such example. More precisely, we prove that the upper bound of $O(\log \Delta(G))$ is tight. In particular, this covers the missing lower bound in [7] and shows that no constant amount of bits suffices. Hence, this yields a new very basic problem where we now have tight bounds for proof labelling schemes, but whose lower bounds do not extend to locally checkable proofs.

Outline. In Section 2 we introduce the notation we need and formally define proof labelling schemes and locally checkable proofs. The proof of the lower bound for the directed st-reachability problem is presented in Section 3 and in Section 4 we discuss the difficulties that occur when extending our result to locally checkable proofs.

2 PRELIMINARIES

Basic notation. The notation for this paper is based on [3], but we briefly summarise what we need here. All graphs are non-empty and finite. We write uv for an edge from u to v and $E(v)$ denotes the set of edges incident to the vertex v. The set of neighbours of v in G is $N_G(v)$ or $N(v)$. We denote the maximum degree of a graph G by $\Delta(G)$ and its diameter by $\operatorname{diam}(G)$.

A *path* is a graph P with vertex set $V(P) = \{v_1, \dots, v_n\}$ and edge set $E(P) = \{v_1v_2, \dots, v_{n-1}v_n\}$, for which we write $P = v_1 \dots v_n$. If u and v are distinct vertices of a path P, then uPv is the unique path in P joining u and v. If $|uPv| \geq 3$, then we set $\mathring{u}P\mathring{v} \coloneqq uPv - \{u, v\}$. Similarly, for a uv-path P, we write $\mathring{P}$ for $P - \{u, v\}$.

Local certification. Let $\mathcal{G}$ be a class of (directed or undirected) graphs, for example, $\mathcal{G}$ could be the class of all connected undirected graphs. Further, let $\mathcal{F} \subseteq \mathcal{G}$ be the graphs in $\mathcal{G}$ that satisfy a certain property (acyclicity, for example). In this context we specify that graphs $G \in \mathcal{G}$ have an *identity* for every vertex $v \in V(G)$, which is denoted by $\mathrm{id}(v)$ and can be encoded in $O(\log |V(G)|)$ bits. All identities in a graph are distinct. Furthermore, vertices also have *labels*, $\mathrm{L}(v)$, that can contain further information (but may be empty). For example, these can be used to indicate colours of vertices or to select a certain subset of the edges.

A *proof* for a graph G is a function $\mathbf{P}\colon V(G) \to \{0,1\}^*$ that assigns a binary *certificate* (a bit string) to each vertex of G. The *size* $|\mathbf{P}|$ of a proof is the maximum number of bits in any of its certificates and the set of all proofs for a graph G is denoted by $\mathbb{P}(G)$. For the empty certificate and the empty label we write ε. Note that the label $L(v)$ of a vertex v is part of the graph G whereas the certificate $\mathbf{P}(v)$ at v is provided by the prover.

A *verifier* for a class $\mathcal{G}$ is a function $\mathcal{V}$ defined for all triples $(G, \mathbf{P}, v)$ with $G \in \mathcal{G}$, $\mathbf{P} \in \mathbb{P}(G)$, and $v \in V(G)$. Its output is a single bit, that is,

$$\mathcal{V}\colon \bigcup_{G \in \mathcal{G}} (\{G\} \times \mathbb{P}(G) \times V(G)) \to \{0,1\}.$$

A verifier $\mathcal{V}$ *accepts* (a proof $\mathbf{P}$) at a vertex v if $\mathcal{V}(G, \mathbf{P}, v) = 1$. It accepts (a proof $\mathbf{P}$ for) a graph G if it accepts at all $v \in V(G)$ and it *rejects* the proof otherwise. The verifier is *sound* for a subset $\mathcal{F}$ of $\mathcal{G}$ if it rejects any graph not in $\mathcal{F}$, regardless of the proof provided, that is, for all $G \in \mathcal{G} \setminus \mathcal{F}$ and all proofs $\mathbf{P} \in \mathbb{P}(G)$ there exists a vertex $v \in V(G)$ such that $\mathcal{V}(G, \mathbf{P}, v) = 0$.

A *prover* is a function $\mathcal{P}$ that maps every $G \in \mathcal{F}$ to a proof $\mathcal{P}(G) \in \mathbb{P}(G)$. The *size* $s(\mathcal{P})$ of a prover is the maximum size of any proof it assigns to a graph in $\mathcal{F}$, often expressed as a function of $|V(G)|$. A prover $\mathcal{P}$ is *complete* for a verifier $\mathcal{V}$ if $\mathcal{V}$ accepts $\mathcal{P}(G)$ for all $G \in \mathcal{F}$.

A pair $\pi = (\mathcal{P}, \mathcal{V})$, consisting of a prover and verifier, locally certifies $\mathcal{F} \subseteq \mathcal{G}$ if it is both *complete* and *sound*. The pair is complete if $\mathcal{P}$ is complete for $\mathcal{V}$ and it is sound if the verifier is sound for $\mathcal{F}$. The *size* $s(\pi)$ of π is just the size of its prover.

Restricting the verifier. So far the definition does not include locality and this is where the various concepts that are used in the literature differ. We now describe restrictions on the verifier that make its computation local and lead to the definition of proof labelling schemes and locally checkable proofs.

Let G be a graph and $v \in V(G)$. The set of vertices in G with distance at most r (with respect to number of edges) to v is denoted by $B^r_G(v)$ or just $B^r(v)$. For directed graphs, we always use the underlying undirected graph to determine balls. A verifier is *r-local* if it satisfies that

$$\mathcal{V}(G, \mathbf{P}, v) = \mathcal{V}(G^r_v, \mathbf{P}^r_v, v) \text{ for all } G, \mathbf{P}, v$$

where $G^r_v = G[B^r(v)]$ and $\mathbf{P}^r_v$ is the restriction of $\mathbf{P}$ to $B^r(v)$. We say a verifier is *neighbourhood-local* if

$$\mathcal{V}(G, \mathbf{P}, v) = \mathcal{V}(G^N_v, \mathbf{P}^1_v, v) \text{ for all } G, \mathbf{P}, v$$

where $G^N_v = (B^1(v), E(v))$. If, additionally, the verifier does not depend on the labels or identities of any vertex other than v when faced with $(G, \mathbf{P}, v)$, then we call the verifier *label-* or *identity-restricted*, respectively. We say it is *restricted*, if it is both label- and identity-restricted and satisfies that

$$\mathcal{V}(G, \mathbf{P}, v) = \mathcal{V}(G^-_v, \mathbf{P}^-_v, v) \text{ for all } G, \mathbf{P}, v$$

where G^-_v is the star with v at its centre and an edge vx_e (respectively x_ev) for each edge $e = vu$ ($e = uv$) in G. The proof $\mathbf{P}^-_v$ maps v to $\mathbf{P}(v)$ and x_e to the certificate $\mathbf{P}(u)$ if $e = vu$ or $e = uv$.

With this notation we can now define proof labelling schemes and locally checkable proofs.

Definition 2.1. A *proof labelling scheme* is a prover-verifier pair $\pi = (\mathcal{P}, \mathcal{V})$ in which $\mathcal{V}$ is restricted.

Definition 2.2. A *locally checkable proof* is a prover-verifier pair $\pi = (\mathcal{P}, \mathcal{V})$ in which $\mathcal{V}$ is r-local for some constant $r \geq 1$.

Note that, in a proof labelling scheme, the verifier at a vertex v may only use v's identity and label, as well as the certificates assigned to v and its neighbours. In contrast, the verifier in a locally checkable proof has access to much more information. Even if we restrict ourselves to a 1-local verifier, it can still use the identities and labels of the neighbours as well as the graph structure of the neighbourhood, meaning it knows which of its neighbours are the same and whether or not they are adjacent.

We introduced the term neighbourhood-local, which comes between the two. In contrast to proof labelling schemes, the verifier here can detect parallels and anti-parallels, but, unlike locally checkable proofs, it is still limited to direct neighbours and does not know which of these are adjacent. We actually prove that a pair $(\mathcal{P}, \mathcal{V})$ that locally certifies st-reachability needs $\Omega(\Delta(G))$ bits if the verifier is neighbourhood-local and identity-restricted in the next section. This yields the results for proof labelling schemes as a corollary.

An example. To wrap up the preliminaries, we illustrate these concepts on an example. We have already seen how to certify bipartiteness and we now sketch how to verify acyclicity using a proof labelling scheme as a slightly more involved example. As mentioned in the introduction, $\Theta(\log n)$ bits are needed for this and the lower bound does not extend to locally checkable proofs. We illustrate why and refer to [12] for a more detailed proof.

The upper bound is obtained by rooting a tree: An arbitrary vertex is chosen as the root and the certificate at each vertex is its distance to this root. To locally verify this, a vertex with certificate d need only check that it has exactly one neighbour with distance $d-1$ and all other neighbours have distance $d+1$ (unless $d = 0$ in which case all neighbours must have distance 1). This is a proof labelling scheme since the graph is a tree if the verifier accepts: in any cycle in the graph, the vertex with largest certificate has at least two neighbours whose certificates are not larger, which the verifier can detect.

To obtain the lower bound, assume that a proof labelling scheme exists that uses $o(\log n)$ bits. Then, for large enough n, we can assume it has fewer than $c \log n$ bits for some appropriate constant c, chosen such that any path of length n contains two disjoint pairs that are labelled identically. By connecting the second vertex of the first pair to the first vertex of the second pair, a cycle is obtained in which all local behaviour is identical. Since the path must be accepted, the verifier also accepts the cycle, contradicting soundness.

The reason this no longer works for locally checkable proofs is because, in this concept, the verifier can see the identities of its neighbours even if the size of the prover is too small to simply add these to the certificates. As a result, the verifier would notice that its neighbour has changed in the transition to the cycle, since the ends of the new edge now have a neighbour whose identity differs from the path.

3 VERIFYING REACHABILITY

The st-reachability problem. The *(directed) st-reachability problem* starts with a graph class $\mathcal{G}$ of (directed) graphs in which there is a unique vertex labelled s and one labelled t (and all other vertices have empty labels). The subclass $\mathcal{F}$ that we wish to verify contains those graphs in which t is reachable from s, that is, those graphs in $\mathcal{G}$ that have an st-path. For the remainder of this section, $\mathcal{G}$ and $\mathcal{F}$ will be used to denote these graph classes.

In the undirected case, a single bit is sufficient to verify reachability (as described in [10]): by fixing some shortest st-path P and setting $\mathbf{P}(v) = 1$ if $v \in P$ (and leaving the certificate empty otherwise), a vertex v can check that either $\mathbf{P}(v) = \varepsilon$ or exactly two of its neighbours are also assigned a non-empty certificate. The vertices s and t are exceptions, they must receive certificate 1 and require exactly one neighbour with this property.

This breaks down in the directed case, since the analogous requirement of asking for exactly one predecessor and one successor is not true. Even on shortest paths, the existence of back-edges may lead to larger quantities of both. This can be fixed by additionally specifying the distance from s (as in the verification of acyclicity), letting us detect back-edges. Alternatively, one can specify which of the edges incident to a vertex leads to the successor on the path. These yield upper bounds of $O(\log \operatorname{diam}(G))$ and $O(\log \Delta(G))$, both of which are potentially $O(\log n)$.

We now show that the second approach is best possible in the sense that $o(\Delta(G))$ bits are insufficient to obtain a proof labelling scheme in the directed case. To achieve this, we assume that a proof labelling scheme $(\mathcal{P}, \mathcal{V})$ exists that only uses x bits and therefore uses at most $c := 2^{x+1} - 1$ distinct certificates. We then construct a graph $G \in \mathcal{G}$ that the verifier would falsely accept and check that its maximum degree is polynomial in c. This yields that $\log \Delta(G)$ is some multiple of x, and $x \in \Omega(\log \Delta(G))$.

As we already mentioned at the end of the last section, our construction works even if we can detect parallels and anti-parallels and can see the labels of neighbouring nodes. It also does not use the relation between c and x. Therefore, we suppose that $(\mathcal{P}, \mathcal{V})$ is a prover-verifier pair with neighbourhood-local and identity-restricted verifier that locally verifies directed st-reachability using c distinct certificates. To facilitate our argumentation, we now provide some notation.

Split paths and their properties. Let $\overleftarrow{P}$ be an st-path with back-edges, that is, $P = sv_1 \dots v_k t$, $\overleftarrow{A} = \{v_i v_j : i > j\}$ is the set of back-edges of P, and $P \subseteq \overleftarrow{P} \subseteq P + \overleftarrow{A}$ (for some k). Note that $\overleftarrow{P} \in \mathcal{F}$. Let uv be an edge of P and ba be a back-edge in $\overleftarrow{P}$ with $uv \in \mathring{a}P\mathring{b}$. The *split-path* (of $\overleftarrow{P}$) at uv using ba is the graph $\overleftarrow{P} - \{uv, ba\} + \{bv, ua\}$, which is in $\mathcal{G} \setminus \mathcal{F}$. This operation is illustrated in Figure 1.

We now show that assigning certain certificates to the vertices of a path with back-edges would make $\mathcal{V}$ accept a split-path, and hence these assignments may not occur. For simplicity, we write $\mathbf{P}(xy)$ for $(\mathbf{P}(x), \mathbf{P}(y))$, where $\mathbf{P}$ is some proof and xy is an edge.

Lemma 3.1. *Let $\overleftarrow{P}$ be an st-path with back-edges, $\mathbf{P} = \mathcal{P}(\overleftarrow{P})$, uv be an edge of P, and ba be a back-edge in $\overleftarrow{P}$ with $uv \in \mathring{a}P\mathring{b}$. If $v \notin N(b)$, $u \notin N(a)$, and $vu \notin \overleftarrow{P}$, then $\mathbf{P}(uv) \neq \mathbf{P}(ba)$.*

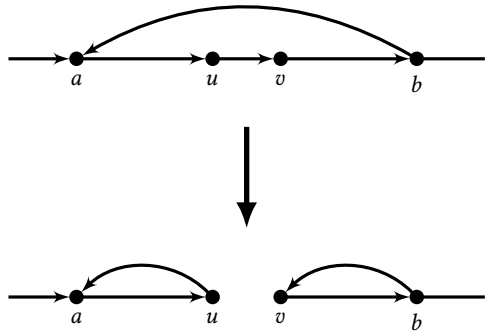

Figure 1: An illustration of the split-path operation.

Proof. Suppose $\mathbf{P}(uv) = \mathbf{P}(ba)$. Let S be the split path of $\overleftarrow{P}$ at uv using ba. We show that $\mathcal{V}$ accepts $\mathbf{P}$ for S, which contradicts $S \in \mathcal{G} \setminus \mathcal{F}$.

Notice that for all vertices $x \notin \{a, b, u, v\}$ the ball $B^1(x)$ remains unchanged and thus $\mathcal{V}$ accepts at all these vertices. Also, in the remaining four balls we only exchange the vertices a and v or b and u. Since these are assigned the same certificate by assumption, the verifier is faced with the same certificates. Moreover, none of these vertices can be s or t, so they are all unlabelled. Finally note that the remaining assumptions of the lemma ensure that none of the edges uv, ba, bv, or ua have a parallel or an anti-parallel in $\overleftarrow{P}$ or S, hence the graphs $\overleftarrow{P}^N_x$ and S^N_x also coincide for $x \in \{a, b, u, v\}$. □

Constructing a counterexample. We are now ready to construct a path with back-edges $\overleftarrow{P}$ in which any proof that the verifier accepts leads to a split path which is accepted as well. Since the verifier accepts $\mathcal{P}(\overleftarrow{P})$, this is a contradiction.

We let $r = \binom{c}{2} + c$ and define a graph G_k for $0 \le k \le r$, each of which is an st-path with back-edges. For a copy H of a graph G_k, we write $P(H)$ for the copy of the st-path of G_k in H and write $s(H), t(H)$ for its start and end, respectively. The graph G_0 is simply the path su_0v_0t. For $k \ge 1$, the graph G_k is the disjoint union of k copies $H_1, \dots, H_k$ of G_{k-1} which are combined to an st-path with back-edges as follows: copies H_i and H_{i+1} are connected by a path $t(H_i)u_iv_is(H_{i+1})$ introducing new vertices u_i and v_i. Additionally, we prepend the path $su_0v_0s(H_1)$ and append the path $t(H_k)u_kv_kt$. Finally, all possible back-edges between vertices $\{u_i, v_i\}$ and $\{u_j, v_j\}$ for $i > j$ are added.

This construction is visualised in Figure 2 and a formal definition of the edge and vertex set of the graph G_k is given below:

$$V(G_k) = \bigcup_{i=1}^{k} V(H_i) \cup \{u_i, v_i : 0 \le i \le k\} \cup \{s, t\},$$

$$E(G_k) = \bigcup_{i=1}^{k} E(H_i) \cup \{u_iv_i : 0 \le i \le k\} \cup \{v_{i-1}s(H_i), t(H_i)u_i : 1 \le i \le k\} \cup \{su_0, v_kt\}.$$

We say that a pair of certificates (c_1, c_2) is *missing* on a path with back-edges if no edge on the path has certificate (c_1, c_2). We extend this definition to sets $\{c_1, c_2\}$, where c_1 and c_2 need not be distinct, and call such a set missing if the tuples (c_1, c_2) and (c_2, c_1) are. Note that r is exactly the number of such sets. With these definitions at hand we can now prove the following lemma.

Lemma 3.2. *For every $k \le r$, the graph G_r contains a copy H of G_k in which at least $r - k$ sets are missing on the path $P(\mathring{H})$.*

Proof. We prove this by induction on k, where in the case $k = r$ there is nothing to show. For $k < r$ let H' be the copy of G_{k+1} given by the induction hypothesis and let $P' = P(\mathring{H'})$. Then $r - (k+1)$ sets are missing on P' and every vertex on this path is assigned a certificate whose corresponding set is amongst the remaining $k+1$ many. Since G_{k+1} has the $k+2$ edges $u_0v_0, \dots, u_{k+1}v_{k+1}$, at least two of the corresponding edges in P' are assigned certificates that give rise to the same set $\{c_1, c_2\}$. For simplicity, we assume these edges are u_0v_0 and u_1v_1. This set is not missing in P', but we show that it is missing on the path $P = P(\mathring{H})$ where H is the copy of G_k between u_0v_0 and u_1v_1 in H'.

To see that this is indeed the case, we note that for any edge uv on the path P we can apply Lemma 3.1 to the edges uv and $ba = u_1u_0$ in G_r: since the only edges with an end in $V(H') \setminus V(H)$ and the other in $V(H)$ are $v_0s(H)$ and $t(H)u_1$, we get that $v \notin N(u_1)$ and $u \notin N(u_0)$. Moreover, $vu \notin G_r$ since G_r has no anti-parallels by construction. Therefore, the assumptions of Lemma 3.1 are satisfied, and $\mathbf{P}(uv) \ne \mathbf{P}(u_1u_0)$. The same holds for the back-edges u_1v_0, v_1u_0, and v_1u_1.

Since we assumed that both u_0v_0 and u_1v_1 are assigned a certificate corresponding to the set $\{c_1, c_2\}$, one of the four back-edges has certificate (c_1, c_2) and another has (c_2, c_1). Thus, we have ensured that both of these are missing, giving us the new missing set $\{c_1, c_2\}$ (in addition to the $r - (k+1)$ many provided by the induction hypothesis), which completes the proof. □

By Lemma 3.2 for $k = 0$, G_r has a copy H of G_0 in which r pairs are missing on the path $P = P(\mathring{H})$. But since these are all possible pairs, the single edge in P has no valid assignment, which is a contradiction. To complete this section, we only need to see how c relates to $\Delta(G_r)$.

Observation 3.3. The maximum degree of G_r is $2r + 2 = c^2 + c + 2$.

Proof. We prove by induction that the maximum degree of G_k is $2k + 2$. Since G_0 is a path, this holds initially. For $k > 0$ let v be a vertex in G_k. If v is in some copy H of G_{k-1}, then its degree is at most $2k$: in H only the vertices $s(H)$ and $t(H)$ have incident edges to vertices outside of this copy of H, and these have degree 2 in G_k.

Any other vertex of G_k is s, t, or in $\{u_0, v_0, \dots, u_k, v_k\}$. The first two have degree 1 and any u_i or v_i has two neighbours on the path and $2k$ further neighbours in G_k, namely all u_j and v_j for $j \in \{0, \dots, k\} \setminus \{i\}$. Hence, the maximum degree of G_k is $2k + 2$.

The missing equality follows from the definition of r. □

By Observation 3.3, $\Delta(G_r)$ is indeed polynomial in c. Simple computations yield that at least $\log_2(c+2) - 1$ bits are required to obtain $c + 1$ distinct certificates and $\log_2(\Delta(G_r)) \le 2\log_2(c) + 2$. Consequently, since we need at least $c + 1$ certificates to correctly verify G_r, at least $\frac{1}{2}\log_2(\Delta(G_r)) - 2$ bits are necessary. We have thus arrived at the desired result.

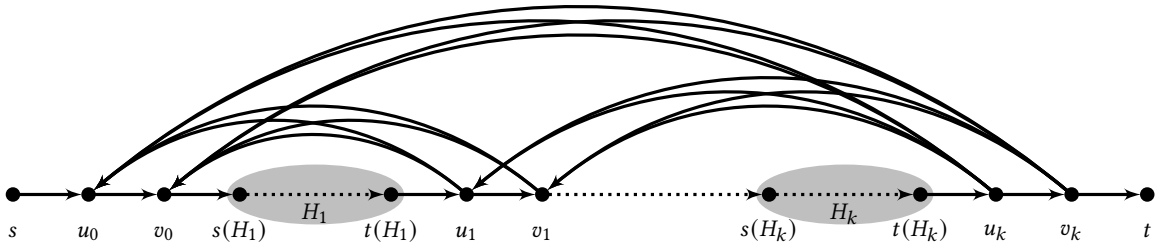

Figure 2: A visualisation of the graph G_k constructed for the proof of Theorem 3.4.

Theorem 3.4. *The st-reachability problem cannot be locally certified by a prover-verifier pair $(\mathcal{P}, \mathcal{V})$ of size $o(\Delta(G))$ if the verifier is neighbourhood-local and identity-restricted.*

As a corollary, we get the same result for proof labelling schemes.

Corollary 3.5. *There exists no proof labelling scheme of size $o(\log \Delta(G))$ for directed st-reachability.*

4 CONCLUSION AND FURTHER RESEARCH

We have shown that $\Omega(\log \Delta(G))$ bits are necessary in any proof labelling scheme for directed st-connectivity. This implies the only missing bound in [7], but does not answer the open question in [10], since locally checkable proofs are not covered. It would be desirable to obtain an example that fools the stronger verifiers allowed in locally checkable proofs. We now describe why this additional verification power is problematic for the example we constructed.

Note that we are able to deal with parallels and anti-parallels, but already had to exclude being able to see which neighbours are adjacent and which ones are not. The reason for this is that the proof of Lemma 3.1 no longer works in this case: if we look at u's local view of the graph in Figure 1, then it replaces v by a. But if v was adjacent to other neighbours of u and a is not, then the verifier can now detect this. Indeed, this happens in the graph G_r. In the graph G_k, the vertices $u_0, \ldots, u_k, v_0, \ldots, v_k$ form a complete graph (in the undirected sense). But if we transition to a split path at one of the edges $u_i v_i$, using a back-edge ba from outside this G_k, then we replace the neighbour v_i that is part of this complete graph by some other vertex a that neighbours none of these vertices.

Similarly, larger radii are problematic. It is not implausible to generalise Lemma 3.1 to paths (by replacing back-edges by "back-paths" and forbidding paths with the same certificate sequence below them). However, this is not helpful. The reason is that it does not allow us to generate new forbidden paths, which is easiest exemplified by assigning all intermediate vertices on back-paths a unique certificate, distinguishing them from the rest.

The last obstacle introduced by locally checkable proofs is that the identities of the neighbours are now visible, for which tools similar to the ones used in [10] seem to be required: multiple graphs in $\mathcal{F}$ with disjoint identities need to pieced together to obtain one that is not in $\mathcal{F}$, but locally appears to be.

REFERENCES

[1] Yehuda Afek, Shay Kutten, and Moti Yung. 1997. The local detection paradigm and its applications to self-stabilization. *Theoretical Computer Science* 186, 1 (1997), 199–229. https://www.sciencedirect.com/science/article/pii/S0304397596002861

[2] Miklos Ajtai and Ronald Fagin. 1990. Reachability is harder for directed than for undirected finite graphs. *Journal of Symbolic Logic* 55, 1 (1990), 113–150. https://doi.org/10.2307/2274958

[3] Reinhard Diestel. 2010. *Graph Theory* (fourth ed.). Graduate Texts in Mathematics, Vol. 173. Springer, Heidelberg; New York. https://doi.org/10.1007/978-3-662-53622-3

[4] Shlomi Dolev. 2000. *Self-stabilization.* MIT press.

[5] Laurent Feuilloley. 2021. Introduction to local certification. *Discrete Mathematics & Theoretical Computer Science* 23, 3 (Sep 2021). https://doi.org/10.46298/dmtcs.6280

[6] Laurent Feuilloley, Pierre Fraigniaud, Ivan Rapaport, Éric Rémila, Pedro Montealegre, and Ioan Todinca. 2020. Compact Distributed Certification of Planar Graphs. arXiv:2005.05863 [cs.DC]

[7] Klaus-Tycho Foerster, Thomas Luedi, Jochen Seidel, and Roger Wattenhofer. 2018. Local checkability, no strings attached: (A)cyclicity, reachability, loop free updates in SDNs. *Theoretical Computer Science* 709 (2018), 48–63. https://doi.org/10.1016/j.tcs.2016.11.018 Special Issue of ICDCN 2016 (Distributed Computing Track).

[8] Steven Fortune, John Hopcroft, and James Wyllie. 1980. The directed subgraph homeomorphism problem. *Theoretical Computer Science* 10, 2 (1980), 111 – 121. https://doi.org/10.1016/0304-3975(80)90009-2

[9] Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA.

[10] Mika Göös and Jukka Suomela. 2016. Locally Checkable Proofs in Distributed Computing. *Theory of Computing* 12, 19 (2016), 1–33. https://doi.org/10.4086/toc.2016.v012a019

[11] Amos Korman and Shay Kutten. 2006. Distributed Verification of Minimum Spanning Trees. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing* (Denver, Colorado, USA) *(PODC '06)*. Association for Computing Machinery, New York, NY, USA, 26–34. https://doi.org/10.1145/1146381.1146389

[12] A. Korman, S. Kutten, and D. Peleg. 2010. Proof labeling schemes. *Distributed Computing* 22 (2010), 215–233. https://doi.org/10.1007/s00446-010-0095-3

[13] N. Robertson and P.D. Seymour. 1995. Graph Minors .XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B* 63, 1 (1995), 65 – 110. https://doi.org/10.1006/jctb.1995.1006