Robustness of Distances and Diameter in a Fragile Network

Arnaud Casteigts
LaBRI, CNRS, Université de Bordeaux, Bordeaux INP, France

Timothée Corsini
LaBRI, CNRS, Université de Bordeaux, Bordeaux INP, France

Hervé Hocquard
LaBRI, CNRS, Université de Bordeaux, Bordeaux INP, France

Arnaud Labourel
Aix Marseille Univ, CNRS, LIS, Marseille, France

Abstract

A property of a graph G is *robust* if it remains unchanged in all connected spanning subgraphs of G. This form of robustness is motivated by networking contexts where some links eventually fail permanently, and the network keeps being used so long as it is connected. It is then natural to ask how certain properties of the network may be impacted as the network deteriorates. In this paper, we focus on two particular properties, which are the diameter, and pairwise distances among nodes. Surprisingly, the complexities of deciding whether these properties are robust are quite different: deciding the robustness of the diameter is coNP-complete, whereas deciding the robustness of the distance between two given nodes has a linear time complexity. This is counterintuitive, because the diameter consists of the maximum distance over all pairs of nodes, thus one may expect that the robustness of the diameter reduces to testing the robustness of pairwise distances. On the technical side, the difficulty of the diameter is established through a reduction from hamiltonian paths. The linear time algorithm for deciding robustness of the distance relies on a new characterization of two-terminal series-parallel graphs (TTSPs) in terms of excluded rooted minor, which may be of independent interest.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Networks → Network dynamics; Theory of computation → Complexity classes

Keywords and phrases Dynamic networks, Longest path, Series-parallel graphs, Rooted minors

Digital Object Identifier 10.4230/LIPIcs.SAND.2022.9

Funding ANR project ESTATE (ANR-16-CE25-0009-03) and DREAMY (ANR-21-CE48-0003)

1 Introduction

The diameter of a network is the maximum distance between any pair of nodes. This concept plays an important role in various fields of network science. For example, in communication networks and distributed algorithms, the diameter is a key parameter involved in the complexity of basic tasks such as leader election, spanning tree construction, and broadcast. Indeed, both the execution time and number of messages may depend on this parameter. Similarly, distances between nodes play a role in nearly all networking phenomena.

In a physical network, the links may deteriorate and eventually become subject to permanent failure. In this case, either the network is maintained (repaired), or it is used despite the failures so long as communication remains possible, i.e., so long as it remains *connected*. A natural question, is then to what extent the properties of the network could change as the network deteriorates. In graph theoretical terms, the connectivity assumption imposes that the communication graph always remains a *connected spanning subgraph* of the original graph, although one does not know in advance which such subgraph will occur. A notion of robustness accounting for the preservation of a property in *all* these subgraphs was investigated in [3] in the context of covering problems. Although it can be formulated in classical graph theoretical terms, the notion of robustness was initially motivated by a temporal context. Namely, if the lifetime is infinite, then some edges may be recurrent (i.e., always reappear), and some may eventually disappear forever. If the network is guaranteed to have a recurrent temporal connectivity (class $\mathcal{TC}^{\mathcal{R}}$), then a certain connected spanning subset of the edges *must* be recurrent, although one does not know in advance which subset. (In technical terms, the *eventual footprint* is any connected spanning subgraph of the *footprint*.)

In this paper, we investigate the robustness of distances and diameter in the classical (i.e. non-temporal) setting, where deletions are definitive. Relations between edge deletions and distances in a graph have been studied over decades in some fields, such as that of graph spanners. A spanner of a graph is a subgraph that preserves, to some extent, the properties of the input graph – typically, the distances – while retaining as few edges as possible. For example, we know since [1] that a tradeoff exists between the size of the spanner and the deterioration of distances, in proportional terms (called *stretch factor*). Namely, there always exists a spanner of size $O(n^{1+1/k})$ whose stretch factor is at most $2k-1$, where n is the number of nodes in the graph. The reader is referred to [6] for background in graph spanners. A significant difference between this topic and questions about robustness is that, in the case of spanners, the deletions of edges are *chosen* by the algorithm, whereas in the case of robustness, they are imposed by the environment. Thus, it makes sense to think of robustness in terms of adversarial edge deletions. The question is then whether and how the distances are preserved under all possible choices of deletions by an adversary, up to preserving connectivity.

As a warm-up, observe that, for any path P connecting two nodes u and v in a graph G, the adversary can always delete enough edges that this path becomes the only path between u and v (for example, by choosing a spanning tree $G' \subseteq G$ that contains this path). Thus, deciding whether the distance between u and v is robust comes to decide if there exists a path between u and v that is longer than their original distance $d(u, v)$ in G. A similar question was considered recently in the case of *induced* paths [2] and was shown to be solvable in polynomial time, albeit with a time complexity of $O(|G|^{18})$ (as of the current analysis, which the authors of [2] do not consider tight). The non-induced case is arguably simpler. In fact, the question for non-induced paths reduces (without being equivalent) to a question known as the *next-to-shortest* path, which consists of finding a path between u and v that is the shortest among all paths of cost strictly greater than $d(u, v)$. Clearly, $d(u, v)$ is robust if and only if no such path exist. A number of algorithms were introduced for this problem, both in directed [11] and undirected [10, 12, 9] graphs. The best known algorithm, in the undirected case, is that of [9], with a time complexity of $O(n^2)$. It is not known whether this algorithm is optimal for general graphs (in particular, sparse graphs), the quadratic term being independent from the number of edges.

1.1 Contributions

The first set of contributions of this paper is a structural investigation of robustness, which results in a linear time algorithm for deciding whether the distance between two vertices u and v is robust. To do so, we identify and exploit a connection between robust distances and *two-terminal series-parallel* graphs (TTSPs), whose recognition is known to be solvable in linear time [16]. Precisely, we introduce a new class of TTSP graphs, referred to as *TTSPs of fixed length*. Then, we show that the distance between u and v is robust in a graph G if and only if the subgraph of G induced by the union of all paths from u and v is a TTSP of fixed length. The main contribution is the characterization itself. First, we show that general

TTSPs correspond to the graphs that exclude a certain *rooted minor*, namely, a diamond rooted at u and v. This point of view clarifies earlier characterizations of TTSPs that essentially arrived at the same conclusion without relying explicitly on a forbidden pattern. (In the case of series-parallel graphs which are not two-terminal, such a characterization was already known in terms of forbidden K_4 [4, 5].) Then, we establish the correspondence between these forbidden rooted diamonds and the robustness of the distance between u and v. The natural consequence of our characterization is that robustness can be tested in linear time by adapting the recognition algorithm from Valdes, Tarjan, and Lawler [16] to the special case of fixed length TTSPs, with the same running time of $O(n + m)$ operations (where m is the number of edges).

The second set of contributions is the computational complexity of deciding whether the diameter of a graph is robust. Clearly, the concept of diameter is strongly related to the one of distance. However, and quite surprisingly, computing the *robustness* of the diameter turns out to be a much different (and much more difficult) problem than computing the robustness of pairwise distances. Precisely, we show that Robust-Diameter is coNP-complete (in other words, proving that the diameter of certain networks are robust is difficult). This is done through a reduction from Hamiltonian-Path, where one must decide if a path of length $n - 1$ exists in a graph.

1.2 Organization of the document

The document is organized as follows. Section 2 provides the main definitions and some basic observations. Then, we establish in Section 3 that deciding whether the diameter is robust is a difficult (coNP-complete) problem. In Section 4, we investigate several aspects of the robustness of distances, in relation to two-terminal series-parallel graphs (TTSPs). The main results of this section are a new characterization of TTSPs (Section 4.2) and a linear time decision algorithm for robustness of distances (Section 4.4). Finally, we conclude with some remarks and open questions in Section 5.

2 Definitions and preliminary observations

2.1 Basic definitions

An *undirected graph* G is a pair $(V(G), E(G))$ where $V(G)$ and $E(G)$ are two disjoint sets, the set of vertices (or nodes) and the set of edges respectively. Each edge is associated with two vertices called its *endpoints*. A *loop* is an edge whose endpoints are the same vertex. If there are several edges with the same endpoints, these edges are called a *multi-edge*. An undirected graph is *simple* if it does not have loops nor multiple edges. A graph that can have multi-edges is called a *multigraph*. A graph without loops is called *loopless*. Unless otherwise mentioned, all the graphs in this paper are simple. The *order* of a graph G is $|V(G)|$ and its *size* is $|E(G)|$. Two vertices u and v are *adjacent* if there exist an edge uv in $E(G)$. In a loopless graph, the *degree* of a vertex u in G is the number of edges incident with u. In a simple graph, this corresponds to the number of vertices with which u shares an edge, called its *neighbors*. A *complete graph* is a simple graph such that every pair of vertices are neighbors, we denote the complete graph of order n by K_n.

Let H and G be two graphs. We say that H is a *subgraph* of G if and only if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. If $V(H) = V(G)$ and $E(H) \subseteq E(G)$ then H is a *spanning subgraph* of G. Let $X \subseteq V(G)$, the *induced subgraph* $G[X]$ is the subgraph of G on vertex set X and where, for every two vertices u and v of X, $uv \in E(G[X])$ if and only if $uv \in E(G)$. Finally, we use $G - X$ as a shorthand for $G[V - X]$ if X is a set of vertices, and for $(V, E - X)$ if X is a set of edges.

A *path* $\mathcal{P}$ from v_0 to v_k is a sequence of edges $(e_1, e_2, \dots e_k)$ of length k for which there is a sequence of vertices $(v_0, v_1, \dots, v_k)$ such that the endpoints of e_i are v_{i-1} and v_i for $i = 1, \dots, k$. All vertices in a path must be distinct, except possibly for the first and last, in which case the path is a *cycle*. In a simple graph, the sequence of vertices identifies uniquely the corresponding path. The *length* of a path is the number of edges in it. The *distance* between two vertices u and v in a graph G, denoted by $d_G(u, v)$ (or simply $d(u, v)$ when the context of G is clear), is the minimum length of a path from u to v. The *diameter* of a graph G, denoted by $\mathrm{diam}(G)$, is the maximum distance between any pair of vertices. A graph is *connected* if for every pair of vertices u and v, there exists a path from u to v, and it is *biconnected* if for any $v \in V$, the graph $G - \{v\}$ is connected. A *tree* is a connected graph without cycle.

A *connected component* is a maximal connected subgraph. A *block* or *biconnected component* is a maximal biconnected subgraph. A *separator* of a connected graph G is a set of vertices whose removal renders G disconnected. An *articulation point* of a connected graph G is a separator of size 1 (a single vertex). The structure of blocks and separators of a connected graph can be described by a tree called the *block-cut tree* [7]. This tree has a vertex for each block and for each articulation point of the given graph. There is an edge in the block-cut tree for each block and for each articulation point that belongs to that block. For a graph G and two vertices u and v, we say that G is a *block-cut (u, v)-path* if the block-cut tree of G is a path from s to t, such that u (resp. v) is only contained in the block associated to s (resp. t). Observe that by definition, u and v are not articulation points.

Let us now define the notion of robustness that we consider in this work.

▶ **Definition 1** (Robustness). *A property P of a connected graph G is called* robust *[3] if P is satisfied in every connected spanning subgraph of G (including G itself). By extension, if P denotes a quantity rather than a predicate (such as, here, a distance), then it is called robust if its value is the same in all connected spanning subgraphs of G.*

We are now ready to state the main two problems that we address in this paper, namely Robust-Diameter and Robust-Distance.

▶ **Definition 2** (Robust-Diameter).
Input: A graph G.
Output: Whether the diameter of G is robust.

▶ **Definition 3** (Robust-Distance).
Input: A graph G, two vertices u and v of G.
Output: Whether the distance between u and v is robust in G.

2.2 Preliminary observations

In this section, we establish a number of basic facts, most of which are used in the rest of the paper.

▶ **Lemma 4.** *Let G be a connected graph and $P \subseteq G$ a path between distinct vertices u and v, then there exist a spanning tree $S \subseteq G$ such that $P \subseteq S$.*

Proof. Observe that P is a (possibly non-spanning) tree. As long as it is not spanning, one can extend it by adding a node that is not in it, but that has a neighbor in it. Such a node always exists because G is connected. ◀

▶ **Lemma 5.** *Let G be a connected graph and u, v two vertices of G. The distance between u and v is robust if and only if there is no path between u and v longer than $d_G(u, v)$.*

Proof. Suppose there is a longer path in G between u and v. By Lemma 4, it is thus possible for the adversary to reduce G to a spanning tree $S \subseteq G$ such that $d_S(u, v) > d_G(u, v)$. ◀

▶ **Lemma 6.** *Let G be a biconnected graph and u, v and w three vertices of G. There is a path from u to w passing through v.*

Proof. Consider a graph G' obtained from G by adding an extra vertex z adjacent to both u and w. G' is also biconnected since no articulation point was added. By Menger's theorem [13], there exist two vertex-disjoint paths from z to v. Since z has only two neighbors v and w, there are two subpaths: one from u to v and another from v to w. By composing these subpaths, one can thus obtain a path from u to w passing through v. ◀

The following lemma allows us to subsequently restrict our attention to a limited part of the graph, when investigating the robustness of distance between two given vertices.

▶ **Lemma 7.** *Let G be a connected graph, u, v two distincts vertices of G and H the graph induced by the paths from u to v. The distance from u to v is robust in G if and only if it is robust in H.*

Proof. $\implies$ If the distance is robust in G, since H is a subgraph of G, the distance must be robust in H.

$\impliedby$ By definition of H, all paths from u to v in G are also in H, thus, if all those paths have length $d(u, v)$ (equivalent to the robustness of the distance), then all the paths in G from u to v have the same length and the distance is robust as well in G. ◀

Figure 1 shows a graph where the distance between u and v is not robust, this can be verified by only considering the graph induced by all paths from u to v thanks to Lemma 7. Observe that H, the graph induced by the paths from u to v, is always a block-cut (u, v)-path.

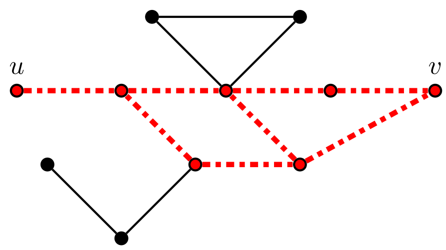

Figure 1 A graph and its subgraph induced by the paths from u to v (in dashed red). This subgraph is a block-cut (u, v)-path.

▶ **Lemma 8.** *Let G be a graph and u, v two vertices of G. Then, the graph H induced by the paths from u to v is a block-cut (u, v)-path.*

Proof. By definition of H, all vertices in $V(H)$ are part of a path between u and v and H is connected. Let T be the block-cut tree of H, by definition, T has a vertex for each block and each articulation point of H. The leaves of T (vertices of degree 1) are blocks, the neighbors of blocks are articulation points, and the neighbors of articulation points are blocks. Suppose that H is not a block-cut (u, v)-path, it means one of the following cases:

- u or v are not part of the leaves in T. Consider a path in T $(x_0, \ldots, x_k)$ that contains both u and v (or their blocks), $u \in x_i$ and $v \in x_j$ such that $i > 0$ or $j < k$. If $i > 0$ (the same could be done with v if $j < k$), then there is a path $(x_0, \ldots, x_i)$ crossing at least one articulation point of H (v_1). It can be deduced that any vertex in v_0 cannot be part of a path between u and v which leads to a contradiction in H.
- T is a tree with at least one vertex of degree 3 or more. Since u and v must be in the leaves of T, then there is a path between u and v that crosses a vertex x of degree at least 3. Consider a neighbor y of x not part of such path. Either y is a block, and x is an articulation point and all paths from u to v in H cannot reach a vertex in y, or y is an articulation point, and there must be another block z that cannot be crossed by paths between u and v (which we assume is not the case). ◀

The following two lemmas are not used in this paper. However, they establish some connections between the robustness of distances and that of the diameter, which is of general interest in the present study.

▶ **Lemma 9.** *All the distances are robust in G if and only if G is a tree.*

Proof. If G is a tree, then the adversary cannot remove any edge, so all the distances are trivially robust. If it is not a tree, then at least one edge uv can be removed, and the distance between u and v thus increases from one to something strictly larger. ◀

▶ **Lemma 10.** *Let G be a connected graph, if $\mathrm{diam}(G)$ is robust, then for every pair of vertices u, v in G such that $d(u, v) = \mathrm{diam}(G)$, their distance is robust.*

Proof. By contradiction, if their distance is not robust, then there must exist a path of length greater than $d_G(u, v)$. By Lemma 4, the adversary can obtain a spanning tree containing this path, whose diameter is thus also greater than $d_G(u, v) = diam(G)$. ◀

3 Robustness of the diameter is hard

In this section, we prove that the problem of deciding whether the diameter of a graph G is robust is coNP-complete. We start with two basic facts that will be used only in this section.

▶ **Lemma 11.** *If H is a connected spanning subgraph of G, then $diam(H) \geq diam(G)$.*

Proof. Let H be a connected spanning subgraph of G. If $diam(H) < diam(G)$, then there must exist two vertices u, v such that $d_G(u, v) = diam(G)$ and $d_H(u, v) < diam(G)$. Let P be a path of length $d_H(u, v)$ in H. Since $H \subseteq G$, P must also exist in G, which contradicts the fact that $d_G(u, v) > d_H(u, v)$. ◀

▶ **Lemma 12.** *The diameter of a connected graph G is robust if and only if it is equal to the length of the longest path of G.*

Proof. $\Longleftarrow$ Let G be a graph of diameter d that is also the longest path in G. Any connected spanning subgraph H of G has, by Lemma 11, $diam(H) \geq d$. If $diam(H) > d$, then there is a path in H (and in G) that is longer than d, which is impossible, thus $diam(H) = diam(G) = d$ for any of these graphs and the diameter of G is robust.

$\Longrightarrow$ By contradiction, let G be a graph whose diameter d is robust even though a longest path of length $l > diam(G)$ exists between some vertices u and v. By Lemma 4, the adversary can obtain a spanning tree T of G containing this path, whose diameter must be strictly larger than that of G. ◀

▶ **Theorem 13.** Robust-Diameter *is* coNP-complete.

Proof. To prove this statement, we will show that the problem is in coNP and that the Hamiltonian-Path problem reduces to it in polynomial time. (Hamiltonian-Path consists of deciding whether a given graph G admits a path of length $n-1$.) The fact that Robust-Diameter is in coNP is direct, using any path of length longer than the diameter as (negative) certificate.

Now, let G be an input graph for Hamiltonian-Path. Without loss of generality, we suppose that G is connected and that it is not itself a path, as otherwise the answer is trivially positive. From G, we can construct a graph H_u as follows: Let P be a graph that consists of a single path of length $2n-2$ on vertices $\{v_i\}$, $i \in [1, 2n-1]$. The graph H_u is built by picking a vertex u in $V(G)$ and adding an edge between u and v_n, the middle vertex of P_{2n-1}. See Figure 2 for an illustration.

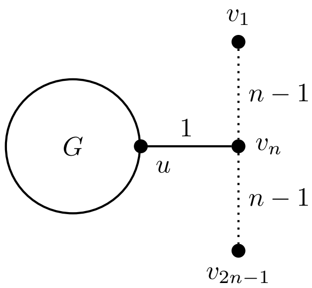

Figure 2 The graph H_u.

We will now prove that G admits a path of length $n-1$ if and only if the diameter of H_u is not robust, for some choice of u. Since G is not itself a path, the diameter of H_u must be $2n-2$. If the diameter of H_u is not robust for some u, then there must exist a path of length at least $2n-1$ in some connected spanning subgraph of H_u. The only way this can happen is that $n-1$ vertices on this path are in G, which implies that G admits a hamiltonian path (starting at u). Conversely, if G admits such a path, then there exists a choice of u such that this path will cause the diameter of H_u to be non-robust. Clearly, the above construction can be made in polynomial time, and guessing u will only contribute an additional factor of n to the complexity. ◀

4 Robustness of pairwise distances

In this section, we investigate the problem of deciding whether the distance between two vertices u and v is robust in a given graph G (Robust-Distance problem). It turns out that the positive instances to this problem can be characterized in terms of two-terminal series-parallel graphs of a certain type. Thus, we start by defining, in Section 4.1, some basic concepts related to two-terminal series-parallel graphs (TTSPs). Our main technical contribution, described in Section 4.2, is an original characterization of TTSPs in terms of excluded rooted diamonds whose "roots" (endpoints) are u and v. This characterization may be of independent interest. In the context of Robust-Distance, it allows us to formulate a necessary condition for the positive instances of the problem, in terms of excluding (u, v)-rooted diamonds (Section 4.3). This condition is however not sufficient, as some TTSPs with respect to u and v may admit paths of different length. We show that existing TTSP recognition algorithms can be adapted at essentially no cost in order to test for the special case of fixed length TTSPs, which capture exactly the properties that should be tested (Section 4.4).

4.1 Two-terminal series-parallel graphs (TTSPs)

The concept of a two-terminal series-parallel graph seems to have been introduced by Riordan and Shannon in [14] (1942). It is now classically defined as follows.

▶ **Definition 14** (Two-terminal series–parallel graph). *Let G be a connected multigraph, s and t two distincts vertices of G called source and sink respectively. G is two-terminal series–parallel (TTSP) if it can be turned into K_2 by a sequence of the following operations:*

- *S: Delete a vertex of degree 2 (other than s or t) and connect its neighbors with an edge.*
- *P: Replace a pair of parallel edges with a single edge connecting the same endpoints.*

Symmetrically, TTSPs can be seen as the graphs which can be obtained from K_2 through the reverse operations of P and S.

These operations S and P are illustrated in Figure 3 and an example of TTSP graph is given in Figure 4. In this example, the distance between s and t is not robust. Note that the fact that s and t are fixed is an important aspect of TTSP graph. For example, if s and t were chosen differently in the graph of Figure 4, the graph would not be a TTSP. The class of graphs that admit a valid pair (s, t) resulting in a TTSP is called SP (for series-parallel). We do not use it in this paper.

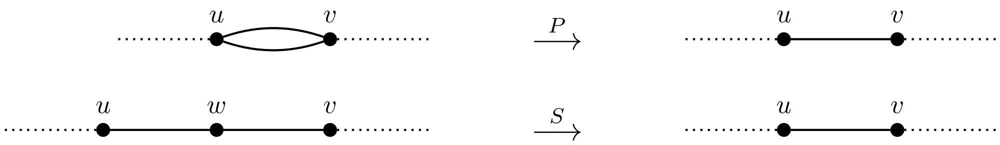

Figure 3 The operations P and S to define TTSP.

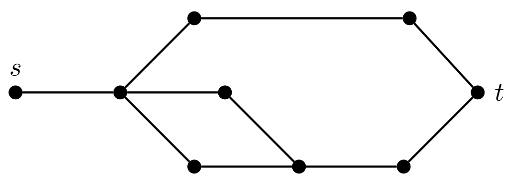

Figure 4 A TTSP between s and t.

When a graph (together with a pair (s, t)) is not a TTSP, then the repeated application of rules S and P eventually fails and one is left with an irreducible graph.

▶ **Definition 15** (TTSP-irreducible). *Let G be a graph and u, v two vertices, then G is TTSP(u, v)-irreducible if G has at least three vertices and the operations S and P cannot be applied relative to u and v.*

The following lemma makes a connection between a TTSP(s, t) and a block-cut (s, t)-path.

▶ **Lemma 16** (Lemma 8 in [5]). *Let G be a TTSP graph with respect to (s, t), then G is a block-cut (s, t)-path.*

However, the converse is not true, and some graphs that are block-cut (s,t)-paths are not a TTSP(s,t). These graphs have special properties characterized through the following lemma (which will be used later).

▶ **Lemma 17.** *Let G be a multigraph that is a block-cut (u,v) path, G is TTSP(u,v)-irreducible if and only if G is simple, with at least 4 vertices, and such that for all $w \in V(G) - \{u,v\}$, $deg(w) \geq 3$.*

Proof. Let G be such a graph.

- P cannot be applied, unless G has multiple edges, so a TTSP(u,v)-irreducible multigraph must be simple.
- S cannot be applied to G, unless there exists a vertex of degree two (other than u and v). Thus, no vertex in an irreducible graph can have degree 2, and since G is a block-cut (u,v)-path, every vertex except u and v must have degree at least 3.
- If G is TTSP(u,v)-irreducible, then it has at least 3 vertices. But since one of them has degree 3 and G is simple, then G actually has at least 4 vertices.

Conversely, if G is simple, with degree at least 3, and has at least four vertices, then (respectively), P cannot be applied, S cannot be applied, and G has at least three vertices. ◀

4.2 Characterization of TTSPs in terms of excluded rooted minor

In this section, we characterize graphs that are TTSP via an excluded rooted minor that corresponds to a complete graph of order four minus one edge called diamond. A similar characterization was mentioned in [16] in which it is stated that a directed graph is not TTSP if only if it has as a subgraph a subdivision of a directed diamond. This result was given as an easy deduction from the classical result of [4] that states that graph is not series-parallel if and only if it has as a subgraph a subdivision of K_4. This characterization of TTSP graphs is not sufficient to directly show that the distance between the two terminal is not robust if the graph between u and v is not a TTSP for which one needs to root the terminal vertices in the minor. Moreover, the setting was different since it considers directed graph. For all these reasons, it seems worth characterizing TTSPs in terms of a clear excluded pattern, which is the purpose of this section. Let us start with basic definitions.

▶ **Definition 18** (Minor). *Let G and H be two graphs, G has a minor H if there is a graph isomorphic to H from G after a succession of the following operations:*

- *deleting a vertex v;*
- *deleting an edge e;*
- *contracting an edge xy into the vertex x: removing y and adding a new edge xz for every z such that $yz \in E(G)$.*

The notion of minor is not precise enough to guarantee the non-robustness of the distance between two vertices u and v, because the position of u and v within the minor matters. Therefore, we use the finer concept of rooted minors, where some vertices can be distinguished in the minor. The difference to "normal" minors is that we want to keep a set $X \subseteq V(G)$ of root vertices alive in the minors. An X-*legal* minor operation is either the deletion of a vertex $y \notin X$, the deletion of any edge, or the contraction of an edge xy into x with $y \notin X$.

▶ **Definition 19** (Rooted minor). *Let G and H be two graphs, $X \subseteq V(G)$ with $|X| \leq |V(H)|$, $\pi : X \to V(H)$ an injection. The pair (G,X) is said to have a π-rooted-minor if G has a minor H such that each vertex $x \in X$ corresponds to the vertex $\pi(x)$ in H obtained with X-legal minor operations.*

We are now ready to show the main technical part of this section. Observe that our definition of rooted minors differs from the definition found in [15] and [17], since all the vertices of the minor are not necessarily rooted. Let H_d be the complete graph of order four minus one edge between x and y ($K_4 \setminus \{x, y\}$). For a TTSP(u, v)-irreducible graph G, we define the bijection $\pi_d : \{u, v\} \to \{x, y\}$ by $\pi_d(u) = x$ and $\pi_d(v) = y$.

▶ **Lemma 20.** *If G is a TTSP(u, v)-irreducible graph and a block-cut (u, v)-path for some $u, v \in V(G)$, then G has a π_d-rooted-minor H_d.*

Proof. We will show the property by induction on the order and the size of the graph. Consider a block-cut (u, v)-path G, such graph is connected. If the order of G is less or equal to 3, then by Lemma 17, G is not a TTSP(u, v)-irreducible graph and the property is satisfied. For $n = 4$, by Lemma 17, G must have two vertices distinct from u and v with degree 3 in order to be TTSP(u, v)-irreducible, thus G has a diamond subgraph and a π_d-rooted-minor H_d. For $n > 4$, consider that G is a TTSP(u, v)-irreducible graph and a block-cut (u, v)-path of order n and size m. Assume by induction that the property is verified for all graphs of order less or equal to $n - 1$ or graphs of order n with a size less or equal to $m - 1$.

If G is not biconnected, then let $c_1, c_2, \ldots c_k$ be the articulation points from u to v. Since G is a block-cut (u, v)-path, any path from u to v crosses these articulation points in order. Let $u = c_0$ and $v = c_{k+1}$. Observe that, for any $0 \leq i \leq k$, the block B between c_i and c_{i+1} must be a TTSP(c_i, c_{i+1})-irreducible graph since otherwise G would not be a TTSP(u, v)-irreducible graph. B is a TTSP(c_i, c_{i+1})-irreducible graph and a block-cut (c_i, c_{i+1})-path of order less than n. By induction, B has a π'_d-rooted-minor H_d with $\pi'_d : \{c_i, c_{i+1}\} \to \{x, y\}$. One can find two disjoint paths: one from u to c_i and another one from c_{i+1} to v that do not contain edges of B. It follows that G has a π_d-rooted-minor H_d. Hence, for the remainder of the proof, one can assume that G is a biconnected graph.

We now consider several cases depending on the neighborhood of u and v. Observe that the degrees of u and v must be at least two since otherwise G would not be biconnected.

- **Case 1:** u and v are adjacent.
  In this case, we consider the graph G' which is G minus the edge uv. First, we show that G' is a block-cut (u, v)-path. By Lemma 6, for any vertex $w \in V(G)$ there is a path from u to v passing through w since G is biconnected. This path also exists in G' since it does not use the edge uv in G. It follows that there is no articulation point separating w from both u and v and so G' is a block-cut (u, v)-path. Assume, by way of contradiction, that G' is a TTSP(u, v) graph. It means that there is a sequence of operations P and S such that G' can be turned into K_2 while preserving u and v. Using the same sequence of operations, G can be turned into a multigraph of two vertices u and v with two edges linking u and v. By applying an operation P, we obtain a K_2 and thus there is a contradiction with the fact that G is a TTSP(u, v)-irreducible graph. Hence, G' is a TTSP(u, v)-irreducible graph and a block-cut (u, v)-path. By induction, since G' is of order n and size $m - 1$, G' has a π_d-rooted-minor H_d and so has G.
- **Case 2**: u is adjacent to $w \neq v$ such that w is not adjacent to other neighbors of u.
  In this case, one can contract edge uw into u. Observe that if $\{u, w\}$ is a separator of G then u is the only articulation point in the connected new graph. One only keeps the block containing v to obtain the graph G'. G' is a biconnected simple graph and all of its vertices are of degree at least 3 except u and v which have degree at least 2 since G' is biconnected. By Lemma 17, G' is a TTSP(u, v)-irreducible graph. Since its order is less than n, it has a π_d-rooted-minor H_d and so has G.

- **Case 3:** u is adjacent to two vertices $w \neq v$ and $z \neq v$ that are adjacent.
  We remove u from G obtaining graph G'. Since G is biconnected, G' is connected. Hence, there is a path P_{wv} from w to v in G' and a path P_{zv} from z to v. Consider a path P_{wv} without z and a path P_{zv} without w if such paths exist. If P_{wv} does not contain z and P_{zv} does not contain w then u, w, z, v define a π_d-rooted-minor H_d. If P_{wv} contains z then its subpath from z to v is a path not containing w. Since the same can be said for P_{zv} and w, it follows that either P_{wv} does not contain z or P_{zv} does not contain w. Assume, without loss of generality, that all paths between w and v contain z. It means that z is an articulation point of G' separating w and v and $\{u, z\}$ is a separator of G. Consider the subgraph G'' obtained by removing from G all vertices that are cut from v by removing $\{u, z\}$ (including w). Observe that, since G'' is biconnected, u has degree at least 2 in G''. If z has degree 2, one contracts edge uz into u. Each other vertex of G'' has the same degree in G'' and G. It follows that all vertices of G'' have degree at least 3 except u and v that have degree at least 2. By Lemma 17, G'' is a TTSP(u, v)-irreducible graph. Since its order is less than n, it has a π_d-rooted-minor H_d and so has G. ◀

▶ **Proposition 21.** *TTSP(u, v) graphs correspond exactly to the block-cut (u, v)-paths which have no π_d-rooted-minor H_d.*

Proof. We show the equivalent proposition that states that G is not a TTSP(u, v) graph if and only if G is not a block-cut (u, v)-path or admits a π_d-rooted-minor H_d.

$\Longleftarrow$ By Lemma 16, if G is not a block-cut (u, v)-path then it is not a TTSP(u, v) graph. Hence, one can assume that G is a block-cut (u, v)-path and admits a π_d-rooted-minor H_d. Consider H, the graph obtained after a succession of operations S and P on G such that no more of these operations can be applied, H is either K_2, or, by Lemma 17, a TTSP(u, v)-irreducible graph. However, S and P are $\{u, v\}$-legal minor operation (P being an edge deletion and S being an edge contraction preserving u and v). Since H_d could not be reduced with S or P, it means that H must have a π_d-rooted-minor H_d. Therefore, H cannot be K_2 and G is not TTSP between u and v.

$\Longrightarrow$ Suppose G is not TTSP between u and v. One can assume that G is a block-cut (u, v)-path since otherwise the property is satisfied. By Lemma 17, G can be reduced to a TTSP(u, v)-irreducible graph H. By Lemma 20, H admits a π_d-rooted-minor H_d. Since S and P are particular minor operations, G also admits a π_d-rooted-minor H_d. ◀

4.3 Robust distance in terms of rooted diamonds

With Proposition 21, we have established that any block-cut (u, v)-path that is not a TTSP must have a rooted diamond. With that characterization, it is easier to characterize the graphs in which $d(u, v)$ is not robust.

▶ **Lemma 22.** *Let G be a connected graph and u, v two vertices of G. If G has a π_d-rooted diamond minor in u, v, then the distance between u and v is not robust.*

Proof. Suppose G admits a π_d-rooted diamond minor in u, v, where x and y are the other two vertices (of degree 3 in the minor). It means there are four paths from u to v in G:
- c_1 that crosses x but not y;
- c_2 that crosses y but not x;
- c_3 that is the same path as c_1 until x, then crosses y from x before crosses the same vertices from y to v as c_2;
- c_4 that crosses y then x and finally v by crossing the same vertices as c_2 (until y) then c_3 (until x) then c_1 (until v).

Consider the subpaths ux, xv, xy, uy and yv crossed by the previous paths, these subpaths are all disjoints. If the distance between u and v was robust, then it would mean that $l(c_1) = l(c_2) = l(c_3) = l(c_4)$. Hence, we have $l(c_3) + l(c_4) = l(c_1) + l(c_2)$ which implies that $2l(xy) = 0$. Since x and y are distincts, a path between the two vertices must have a length of at least 1. Therefore the distance between u and v cannot be robust. ◀

▶ **Lemma 23.** *Let G be a block-cut (u, v) path. If G is not TTSP between u and v, then the distance between u and v is not robust.*

Proof. By Proposition 21, a block-cut (u, v) path which is not a TTSP(u, v) must have a rooted diamond minor in u, v. By Lemma 22, such a graph cannot have a robust distance between u and v. ◀

4.4 An efficient recognition algorithm for distance-preserving TTSPs

We are now ready to exploit the above characterizations in order to test efficiently (indeed, in linear time) whether the distance between two given vertices is robust in a given graph.

▶ **Definition 24.** *A graph G is a TTSP of fixed length (TTSPf) between s and t if, starting with weights of 1, it is turned into K_2 by a sequence of the following operations (see Figure 5):*

- *P_f: Replace a pair of parallel edges of weight i with a single edge of weight i connecting their common endpoints.*
- *S_f: Replace a pair of edges of weight i and j, incident to a vertex of degree 2 other than s or t with a single edge of weight $i + j$.*

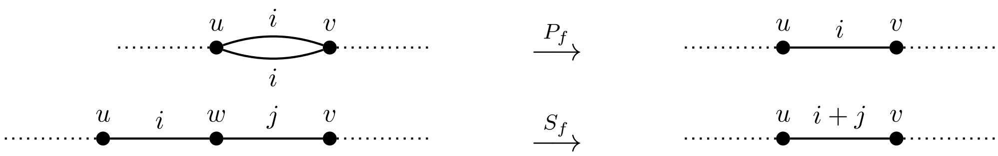

Figure 5 The operations P_f and S_f associated with TTSP of fixed length.

Note that a TTSP of fixed length remains *de facto* a TTSP, because the new operations are only more restricted. In the following, we say that the *length* of a weighted path corresponds to the sum of weights of its edges.

▶ **Lemma 25.** *Let G be a connected edge-weighted multigraph and let s and t be two distinct vertices in G. Consider the edge-weighted multigraph H that results from applying P_f and S_f exhaustively on G. For any length d, there is a path c from s to t of length d in G if and only if there is a path c' from s to t of length d in H.*

Proof. Consider two cases on H:

- If H is the result of the operation P_f on two parallel edges e_1, e_2 of weight i into an edge e' of weight i, then:
  - If c does not cross e_1 or e_2, then c is the same in H;
  - If e_1 or e_2 is crossed by c (but not both), then there is a path c' in H that is the same as c but crossing e' instead, c' has the same length as c. On the contrary, considering c' in H crossing e', it means that there is a path c in G which crosses e_1 or e_2 of the same length.
- If H is the result of the operation S_f on a pair of edges e_1, e_2 of weight i and j incident to a vertex v of degree 2, into an edge e' of weight $i + j$, then:

- If $v \notin c$, then the path c is the same in H;
- If $v \in c$, then c crosses e_1 and e_2 (since v has degree 2 and is neither s nor t). In this case, there is a path c' in H that is the same as c but that crosses e' instead of e_1 and e_2. The length of this path is $d - l(e_1) - l(e_2) + l(e') = d - i - j + (i + j) = d$. Conversely, a path c' that crosses e' in H implies the existence of a path c in G which crosses the contracted vertex v. These paths have the same length. ◀

Lemma 25 guarantees that the length of the paths from s to t are preserved no matter how many times the operations P_f and S_f are applied. Therefore, if there is a longer path in G, it will be possible to find a path of the same length in H after applying a succession of P_f and S_f operations.

▶ **Lemma 26.** *Let G be a TTSP graph between s and t, the distance between s and t is robust if and only if G is a TTSP of fixed length.*

Proof. $\Longleftarrow$ Lemma 25 shows that if G is turned into K_2 with a succession of operations P_f and S_f, then all paths from s to t in G have length $d(s,t)$, meaning the distance between s and t is robust.

$\Longrightarrow$ Let G be a TTSP graph between s and t such that the distance between s and t is robust. Suppose that G can not be reduced to K_2 with a succession of operations P_f and S_f, that is, there is a graph H obtained from G by these operations that cannot be reduced any further and is not K_2:

- If S_f cannot be applied to H, then H does not have any vertex of degree 2 (except s and t), else it would be possible to sum the weight of the edges with the application of S_f;
- If P_f cannot be applied to H, then one of the following must hold:
  - H does not have any multiple edges and with S_f impossible, that would means that G is not TTSP;
  - H has a pair of parallel edges e, f of distinct weights, thus there are in H two paths of different length between u and v, and same in G (by Lemma 25). By Lemma 5, it would mean that the distance is not robust. ◀

Finally the following theorem can be proved:

▶ **Theorem 27.** *Let G be a connected graph, u, v two vertices of G and H the graph induced by the paths from u to v. The distance between u and v is robust if and only if H is a TTSP of fixed length between u and v.*

Proof. The proof combines several previous results:

- by Lemma 7, the distance is robust in $G \iff$ it is robust in H;
- by Lemma 23, the distance is robust in $H \implies H$ is TTSP;
- by Lemma 26, if H is TTSP, then the distance is robust $\iff H$ is TTSP of fixed length.

It can be deduced that if the distance between u and v is robust in G, then H is a TTSP of fixed length between u and v. Reciprocally, if H is a TTSP of fixed length between u and v, then the distance between the two vertices is robust in G. ◀

This theorem means that determining the robustness of the distance between two vertices s, t in a graph G can be done efficiently by performing Algorithm 1 (see below). Our algorithm is heavily based on the recognition of TTSP by applying the operations S and P from [16]. Here, instead, we apply the operations P_f and S_f designed for TTSPf from Definition 24. The original algorithm that uses the TTSP operations runs in $O(n + m)$ time. In order to prove that our algorithm runs in linear time, we will describe the main differences from the TTSP-recognition algorithm. Our algorithm performs the following steps:

1. Extract the graph H from G induced by all paths from s to t (by Lemma 7, the robustness of the distance in H is equivalent to the property in G). Extracting H is similar to finding every biconnected component crossed by a path from s to t. Finding the block decomposition is in $O(n + m)$ time [8]. Finding any path from s to t can be done in $O(n + m)$ time as well in an unweighted graph by doing a Breadth-First-Search. The time complexity of this step is $O(n + m)$;
2. If H is unweighted, we initiate a weight of 1 on each edge of H, this is done in $O(m)$;
3. Check if H is a TTSPf by applying the algorithm from [16]. The only operations added are when applying P_f and S_f instead of P and S. Considering that P and S are applied once per edge (as it is done in the TTSP-recognition algorithm), we only need to verify that we add a constant number of operations for each use of P_f and S_f. First of all, with S_f, applied in Lines 22-33, in the original algorithm, the vertex v of degree 2 is removed with its two edges e_1, e_2 and a new edge e is added to connect its neighbors v_1, v_2, creating a potential multiple edge. Here, the newly created edge e has a weight equal to the sum of the deleted edges e_1, e_2 as shown in Line 26, adding two integers is a constant operation performed once per S_f operations. With the P_f operation, instead of checking every edge of the adjacency list, the original algorithm checks the first edge in the adjacency list of v. After removing the invalid edges in Line 12 that were virtually removed in Lines 17 and 28, the algorithm then checks if a pair of edges that share the same endpoints, in which case it applies P. Here, we first make sure that both edges have the same weight as shown in Line 15, if their weight is different, then the graph is not a TTSPf since P_f would not be applicable. The same kind of verification is done near the end of the algorithm between the remaining edges between s and t in Line 40. Since this verification is only a comparison of integers, it can be done in constant time, as stated before, P_f and S_f being done once per edge, this step of our algorithm adds a complexity of $O(m)$ to the complexity of the original algorithm, therefore this step runs in $O(n+m)$.

5 Concluding remarks and open questions

In this paper, we have shown that the concept of a robust diameter is quite different from the one of robust pairwise distances, so much so that the corresponding decision problems have very different complexities. In the case of the distance, we have identified and exploited a strong connection between TTSP graphs and robust distances, which allowed us to design a linear time algorithm for testing if a given distance is robust. It would be interesting to consider more relaxed versions of the robustness, where one does not ask only whether the distance (or diameter) remains exactly the same, but also whether the deterioration may preserve some comparative quality (this information would have a more practical use). For example, how difficult is it to decide if the distance between u and v may deteriorate up to $d(u, v) + k$ for a fixed k? Similarly, can the robustness of diameter be approximated in the sense of deciding whether the diameter may deteriorate beyond a certain factor of its original value? Beyond the particular case of robust distances and diameter, the study of robust properties in general is in its infancy, and it would be interesting to see if some meta-theorems can be obtained for robust properties in general.

Algorithm 1 Determination of the robustness of the distance.

```
Data: G = (V, E), s, t ∈ V(G)
Result: True iff dist(s, t) is robust
H ← inducedPathsGraph(G, s, t);
(order, size) ← (|V(H)|, |E(H)|);
for e ∈ E(H) do
    e.weight ← 1;
    e.valid ← True ;                                  /* Presence of edges in H */
X ← V(H) \ {s, t};
while X ≠ ∅ do
    v ← X.removefirst();
    while v.degree() > 2 do
        (e1, e2, e3) ← (v.edges()[0], v.edges()[1], v.edges()[2]);
        if ∃e ∈ {e1, e2, e3}, e.valid = False then
            v.edges().delete(e);
        else if ∃(e, f) ∈ {e1, e2, e3}, e, f have the same endpoints then
            if e.weight ≠ f.weight then
                return False ;                        /* Pf cannot be applied */
            else
                f.valid ← False;
                size ← size − 1;
                v.edges().delete(f);
        else
            exit while;
    if v.degree() = 2 then
        (e1, e2) ← v.edges() ;                        /* Application of Sf */
        (v1, v2) ← v.neighbors();
        e ← NewEdge(v1, v2);
        e.weight ← e1.weight + e2.weight;
        H.addEdge(e);
        (e1.valid, e2.valid) ← (False, False);
        (order, size) ← (order − 1, size − 1);
        if v1 ≠ s and v1 ≠ t then
            X.add(v1);
        if v2 ≠ s and v2 ≠ t then
            X.add(v2);
if order > 2 then
    return False;
if size > 1 then
    w ← E(H)[0].weight ;                  /* Final application of Pf on st edges */
    for e ∈ E(H) do
        if e.weight ≠ w then
            return False;
return True;
```

References

1 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

2 Eli Berger, Paul Seymour, and Sophie Spirkl. Finding an induced path that is not a shortest path. *Discrete Mathematics*, 344(7):112398, 2021.

3 Arnaud Casteigts, Swan Dubois, Franck Petit, and John M Robson. Robustness: A new form of heredity motivated by dynamic networks. *Theoretical Computer Science*, 806:429–445, 2020.

4 R.J Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965.

5 David Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992.

6 Cyril Gavoille, Quentin Godfroy, and Laurent Viennot. Node-disjoint multipath spanners and their relationship with fault-tolerant spanners. In *International Conference On Principles Of Distributed Systems*, pages 143–158, 2011.

7 Frank Harary. *Graph Theory.* Addison-Wesley Publishing Company, 1st edition, 1969.

8 John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

9 Kuo-Hua Kao, Jou-Ming Chang, Yue-Li Wang, and Justie Su-Tzu Juan. A quadratic algorithm for finding next-to-shortest paths in graphs. *Algorithmica*, 61(2):402–418, 2011.

10 Ilia Krasikov and Steven D Noble. Finding next-to-shortest paths in a graph. *Information processing letters*, 92(3):117–119, 2004.

11 Kumar N Lalgudi and Marios C Papaefthymiou. Computing strictly-second shortest paths. *Information processing letters*, 63(4):177–181, 1997.

12 Shisheng Li, Guangzhong Sun, and Guoliang Chen. Improved algorithm for finding next-to-shortest paths. *Information processing letters*, 99(5):192–194, 2006.

13 Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

14 John Riordan and Claude E Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physics*, 21(1-4):83–93, 1942.

15 Neil Robertson and Paul D Seymour. Graph minors. xiii. the disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.

16 Jacobo Valdes, Robert E Tarjan, and Eugene L Lawler. The recognition of series parallel digraphs. In *Proc. of the 11th ACM Symposium on Theory of Computing*, pages 1–12, 1979.

17 Paul Wollan. *Extremal functions for graph linkages and rooted minors.* Georgia Institute of Technology, 2005.