Universität Konstanz

Survivable Networks with Bounded Delay

Serafino Cicerone
Gabriele Di Stefano
Dagmar Handke

Konstanzer Schriften in Mathematik und Informatik
Nr. 81, Februar 1999

ISSN 1430–3558

Survivable Networks with Bounded Delay*

Serafino Cicerone[1], Gabriele Di Stefano[1], and Dagmar Handke[2]

[1] Dipartemento di Ingegneria Elettrica, Università degli Studi di L'Aquila
67040 Monteluco di Roio - L'Aquila, Italy.
{cicerone,gabriele}@infolab.ing.univaq.it

[2] Universität Konstanz, Fakultät für Mathematik und Informatik,
78457 Konstanz, Germany.
Dagmar.Handke@uni-konstanz.de

81/1999

Abstract. We investigate new classes of graphs which guarantee constant delays even in the case of multiple edge failures. This means the following: as long as two vertices remain connected if some edges have failed, then the distance between these vertices in the faulty graph is at most a constant factor k times the original distance.

In a first part, we consider k–*self-spanners*, the class of graphs where the distance constraint holds even for an *unlimited* number of edge faults. In particular, we give (non-algorithmic) strict characterizations of the graph classes and show that the problem of minimizing k such that a given graph is a k–self-spanner is co-$\mathcal{NP}$-complete. For small values of k, characterizations in terms of possible topologies of biconnected components are given.

In the second case, the number of edge failures is bounded by a constant ℓ. These graphs are called (k,ℓ)–*self-spanners*. We prove that the problem of maximizing ℓ for a given graph when $k > 4$ is fixed is $\mathcal{NP}$-complete, whereas the dual problem of minimizing k when ℓ is fixed is solvable in polynomial time. We show how graph operations such as *Cartesian product* and *split composition* affect the self-spanner properties of the composed graph. We also investigate several popular network topologies (like *grids*, *tori*, *hypercubes*, *butterflies*, and *cube-connected cycles*) with respect to their self-spanner properties.

1 Introduction

A major concern in network design is fault-tolerance and reliability. That means that the network to be constructed remains reliable even in the case of (multiple) node or link failures. According to the applications, the term 'reliability' may stand for several different features. Often, it is important to design a network in a way such that distances between nodes remain small even in the case of faulty links or nodes; i.e. the delays that are incurred by the failures have to be bounded.

In [CD98], Cicerone and Di Stefano introduce a class of graphs which guarantees constant delays even in the case of an unlimited number of *vertex failures*. Their results do not carry over to the dual case of edge failures. In this paper, we now investigate the case of *edge failures* and introduce new classes of graphs which guarantee constant delay factors even in the case of multiple edge failures. That means the following: As long as two vertices remain connected, the distance between these vertices in the faulty graph is at most a constant factor times the distance in the non-faulty graph. In the following, this constant factor is called *stretch factor*.

We use the following graph theoretic model for building communication networks: sites of the network are represented by vertices, links between pairs of sites by edges. We assume

* Research partially supported by the MURST project 'Teoria dei Grafi e Applicazioni', and by the *Deutsche Forschungsgemeinschaft* under grant Wa 654/10-1

that the transmission of a message incurs a constant unit cost for each direct link. Thus, we are dealing with unweighted, undirected graphs. Distances between sites are measured by shortest paths in the underlying graph. In the case of edge failures, the distance is given by a shortest path in the subnetwork that is induced by the non-faulty edges.

Our work is a first step towards a characterization of survivable networks with guaranteed bounded delay when edge failures are considered. As opposed to [CD98], we do not only consider the case where the number of edge failures is unlimited. We also follow the more realistic approach of a limited number of edge failures. In both cases, we are interested in characterizational as well as structural aspects of the given graph classes.

In [Han98], Handke follows an opposite approach. She considers the problem of finding sparse subgraphs in a given graph such that distance within this subgraph is at most a (fixed) constant times the original distance, even when an edge or a vertex in the subgraph fails. Thus, that work focuses on finding fault-tolerant, (almost) distance preserving subnetworks, whereas here we concentrate on the construction of new networks, such that fault-tolerance and delay constraints can be guaranteed simultaneously.

Several authors investigated the ability of specific topologies to tolerate faults. For example in [LMS92] and [CMS97], it is shown how arrays and several bounded-degree networks with failures can emulate non-faulty such networks. The general approach used there is that of finding an embedding of the faulty graph into the non-faulty graph such that load, congestion and dilation are small. Thus, the structure of the network is not fixed, but the role of any vertex may vary depending on the actual fault, incurring additional work for re-routing. In our work, we do not allow this restructuring, but keep the identification of each vertex fixed. Thus, we cannot use the framework of graph embedding here.

Our paper contains the following results: After a short introduction into the basic notation in Section 2, we first investigate networks with bounded delay and unlimited fault-tolerance, so-called *k–self-spanners* (Section 3). In particular, we give several different equivalent, strict characterizations for this graph class, and list some major properties. As all these turn out to be non-algorithmic in general, we describe the structure of k–self-spanners for $k \leq 3$ in detail, and show that the problem of finding the minimum k such that a graph is a k–self-spanner is co-$\mathcal{NP}$-complete. Thus we cannot hope for finding an efficient algorithm for deciding whether the stretch factor of a graph is smaller than a given constant value. As a further result, we show a strong relationship between networks with bounded delay and unlimited fault-tolerance w.r.t. edge and vertex failures.

In Section 4, we consider the case of a limited number of edge failures; that is we bound the number of possibly faulty edges by a constant ℓ, called *fault-tolerance value.* The corresponding graphs are called *(k, ℓ)–self-spanners.* After a formal definition and some straightforward observations, we discuss on the structural differences between k–self-spanners and (k, ℓ)–self-spanners. The first main results concern the problems of deciding whether a given a graph is a (k, ℓ)–self-spanner: The problem is $\mathcal{NP}$-complete for the general case where k and ℓ are part of the input and remains $\mathcal{NP}$-complete if $k \geq 5$ is fixed. However, if $k \leq 3$ is fixed, or if $\ell \geq 0$ is fixed, then there are polynomial time algorithms. Thus only the case where $k = 4$ is fixed remains open.

In the subsequent subsection, we examine how two popular graph operations affect the self-spanner properties of a graph. In particular, we prove that a graph that arises by *Cartesian product* or *split composition* inherits the self-spanner properties of the underlying graphs. In the case of *Cartesian product* we also show strong results especially for small stretch factors and fault-tolerance values.

The last part shows how the new graph class of (k, ℓ)–self-spanners fits into the context of some popular network topologies. We show for example that meshlike topologies such as *grids*, *tori*, and *hypercubes* exhibit strong self-spanner properties in particular for small fault-tolerance values. Bounded-degree approximations of the hypercube such as *butterflies* and *cube-connected cycles*, however, result in big stretch factors even in the case of small fault-tolerance values. This serves as a strong justification for the relevance of the graph class introduced here.

2 Basic Notation

In this work, we use standard notation for graphs. Let $G = (V, E)$ be a simple (i.e. without multiple edges or loops), unweighted, and undirected graph. Let n denote the number of vertices, and let m denote the number of edges. The *set of vertices* (and *set of edges*, resp.) of G is denoted by $V(G)$ (and $E(G)$, resp.). $G[R]$ where $R \subseteq V(G)$, denotes the subgraph of G induced by R. $G - e$ where $e \in E(G)$ is the graph obtained from G by deleting edge e. The neighborhood $N_G(v)$ of a vertex v in G is the set of all vertices that are adjacent to v in G.

$d_G(u, v)$ is the *distance* between two vertices u and v in G, i.e. the length of the shortest path. If we consider *cycles*, we always mean *simple* cycles, i.e. cycles in which each vertex appears at most once. The *length of a cycle* is the number of its vertices or its edges, resp. An edge is a *chord* of a cycle C if it connects two non-adjacent vertices of C. A cycle C in G is called *induced* if $G[V(C)] = C$, i.e. if C does not contain a chord.

Let P_n be the *path graph* and C_n the *induced cycle graph* (also called ring), respectively, with n vertices. The graph K_n is the complete graph on n vertices. A *diamond* is a biconnected graph formed by two possibly adjacent vertices u and v which are connected by $K \geq 2$ disjoint paths of length 2.

For a connected graph, an *articulation vertex* is a vertex the deletion of which disconnects the graph. A graph is called *biconnected* (or 2*–vertex–connected*) if it has no articulation vertex. It is called ℓ*–vertex–connected* if there is no subset of vertices S of size $\ell - 1$ such that $G[V \backslash S]$ is disconnected. A graph is ℓ*–edge–connected* if no deletion of a $\ell - 1$ edges disconnects it. An edge e of G is called *bridge* if $G - e$ is disconnected. Observe that an ℓ–edge–connected graph does not contain a bridge if $\ell \geq 2$.

For any fixed rational $k \geq 1$, a k*–spanner* of an unweighted graph G is a spanning subgraph S in G such that the distance between every pair of vertices in S is at most k times their distance in G. The parameter k is called *stretch factor*. We say that an edge e is *covered* if in S there exists a path of length at most k that connects the endpoints of e. Such a path is called a *covering path*. Since in particular each edge has to be covered in a k–spanner, it is clear that in unweighted graphs S is a k–spanner of G if and only if S is a $\lfloor k \rfloor$–spanner of G. Thus it suffices to consider integer stretch factors k.

Remark 1. In [CC95], it is shown that a subgraph $S = (V, E')$ of a graph $G = (V, E)$ is a k–spanner if and only if the distance constraint holds for all edges that do not belong to S:

$$d_S(u, v) \leq k, \text{ for every edge } e = \{u, v\} \in E \backslash E'.$$

The concept of spanners has been introduced by Peleg and Ullman in [PU87], where they used spanners to synchronize asynchronous networks. One of the many other applications for spanners are communication networks, where one is interested in finding a sparse subnetwork that nevertheless guarantees constant delay factor. Further results on k–spanners and variants thereof can be found for example in [ADD$^+$93,Soa92,Cai94,CC95,HPS94].

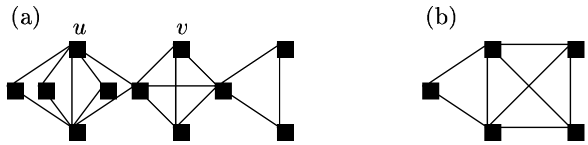

Fig. 1. Examples for k–self-spanners.

3 Networks with bounded delay and unlimited fault-tolerance

In [CD98], Cicerone and Di Stefano introduce a class of graphs which guarantees constant delays even in the case of an unlimited number of *vertex failures*. These graphs are called *k–bounded induced distance graphs*, where k is a real value indicating the delay bound. In this section, we follow the dual approach and examine a class of graphs which guarantees constant delays even in the case of an *unlimited number of edge failures*.

3.1 Definitions and basic observations

Let us first restate the definition of k–bounded induced distance graphs.

Definition 2 (k–bounded induced distance graph, [CD98]).

1. For any fixed real number $k \geq 1$, a graph $G = (V, E)$ is called a *k–bounded induced distance graph* if for every *connected induced* subgraph $G' = (V', E')$ of G, the following holds:
$$d_{G'}(u, v) \leq k \cdot d_G(u, v), \text{ for each } u, v \in V'.$$
Denote the class of all k–bounded induced distance graphs by $BID(k)$.
2. For a graph G, the *stretch number* $s(G)$ denotes the smallest k such that $G \in BID(k)$.

Observe that the definition can be applied to connected and disconnected graphs. Moreover, if k is irrational (for instance, $k = \pi$) then $BID(k')$ could be different from $BID(k)$ for any rational k' such that $k' < k$. In any case, the stretch number $s(G)$ of a graph G is always a rational number.

As an analogue to Definition 2, now taking subgraphs where only edges are deleted, we get the following definition for the dual class:

Definition 3 (k–self-spanner).

1. For any fixed real number $k \geq 1$, a graph $G = (V, E)$ is called a *k–self-spanner* if for every subgraph $G' = (V, E')$ of G, the following distance constraint holds:
$$d_{G'}(u, v) \leq k \cdot d_G(u, v), \text{ for each pair } \{u, v\} \text{ of connected vertices in } G'.$$
Denote the class of all k–self-spanners by $SS(k)$, the parameter k of the class is called *stretch factor*.
2. For a graph G, $minS(G)$ denotes the the smallest k such that $G \in SS(k)$.

The name of the class is motivated by its strict relationship to the concept of k–spanners: all connected components C of G' are k–spanners of $G[V(C)]$. Thus a graph of this class 'spans itself'.

Figure 1 gives examples of the self-spanner properties of two graphs. The graph G in (a) belongs to $SS(3)$, but since $minS(G) = 3$ it does not belong to $SS(2)$. If G' is achieved by adding the edge $\{u, v\}$ then $minS(G') = 6$ and thus G' does not belong to $SS(3)$ anymore. The graph in (b) belongs to $SS(4)$, but not to $SS(3)$.

Observe that the definition works equally well for connected and disconnected graphs; but it is obvious that we can restrict our analysis to connected graphs, since otherwise we can deal with each connected component separately. Thus in the following we only consider *connected* graphs.

The following lemma indicates that we do not have to consider all possible subgraphs if we want to check whether a graph belongs to a class $SS(k)$. In Part 2 of that lemma, we show that we can restrict ourselves to (connected) *spanning* subgraphs. The next important observation is stated in Part 3: the distance constraint has to be fulfilled within each connected subgraph. Thus, k–self-spanners do not only model networks in which the same constant delay is guaranteed with respect to an unlimited number of *edge failures*, but also to *vertex failures*. Thus k–self-spanners model the general case of vertex/edge failures:

Lemma 4. *Let $G = (V, E)$. The following statements are equivalent:*

1. $G \in SS(k)$;
2. *every* spanning *subgraph $G' = (V, E')$ of G is a k–spanner of G;*
3. *every* connected *subgraph $G' = (V', E')$ of G is a k–spanner of $G[V']$.*

Proof.
1. $\Rightarrow$ 2. Trivial.
2. $\Rightarrow$ 3. Assume that every *spanning* subgraph of G (which is surely connected since G is) is a k–spanner of G and there is a connected subgraph $G' = (V', E')$ of G such that $d_{G'}(u, v) > k \cdot d_G(u, v)$ for two vertices $u, v \in V'$. Expand G' to a connected spanning subgraph $G'' = (V, E'')$ by linking missing vertices of G to V' such that these vertices do not lie on a cycle (this is always possible since G is connected). Then, G'' is a spanning subgraph of G and $d_{G''}(u, v) > k \cdot d_G(u, v)$, a contradiction.
3. $\Rightarrow$ 1. Assume that every *connected* subgraph $G' = (V', E')$ of G is a k–spanner of $G[V']$. Thus we can combine these subgraphs to form any arbitrary subgraph of G that fulfills the condition of Definition 3. □

Remark 5.
1. Since we are only dealing with unweighted graphs and we directly use the notion of k–spanners for the characterization of the class $SS(k)$, it is clear that $G \in SS(k)$ if and only if $G \in SS(\lfloor k \rfloor)$ for all real $k \geq 1$.[1] Thus in the following we will only consider integer values for k.
2. Our definition stems from the point of view of network design. Observe, however, that we could have equally well chosen Part 2 of Lemma 4 for the definition. The characterization of $SS(k)$ is then motivated from a graph theoretical point of view.

It is easy to see that the following straightforward properties hold:

Lemma 6.
1. *Let G be a graph with $|V| = n$. Then $G \in SS(k)$ for all $k \geq n - 1$.*
2. *If $1 \leq k \leq k'$ then $SS(k) \subseteq SS(k')$.*

This induces the following natural recognition problem for $SS(k)$:

Problem 7 (Minimum Self-Spanner Problem).

Given: A graph G and an integer $k \geq 1$.
Problem: Does G belong to $SS(k)$, i.e. $minS(G) \leq k$?

[1] Observe that, as mentioned above, this does *not* hold for k–bounded induced distance graphs. Here real stretch numbers really do matter.

3.2 Characterizations of k–self-spanners

In this subsection, we identify some characterizing properties for the class of k–self-spanners in terms of longest simple paths and cycles. On the one hand, this leads to strict (and efficient) characterizations for $SS(k)$ for small k. On the other hand, these results imply that the Minimum Self-Spanner Problem is hard in general. We also show a nice relationship between the classes of k–self-spanners and k–bounded induced distance graphs. The following theorem gives straightforward characterizations of k–self-spanners.

Theorem 8. *Let $G = (V, E)$ be a graph. The following propositions are equivalent:*

1. *$G \in SS(k)$;*
2. *every simple cycle of G has at most $k + 1$ edges;*
3. *for each $e = (u, v) \in E$, a longest simple path between u and v in G has length at most k.*

Proof.

1. ⇒ 2. By contradiction, let us assume that there exists a simple cycle C in G with at least $k + 2$ edges. Let (u, v) be an edge of C, and let G' be the subgraph of G induced by the edges of C except (u, v). Hence, $d_{G'}(u, v) \geq k + 1$. This inequality implies that G' is not a k–spanner of $G[V(G')]$, a contradiction of Part 3 of Lemma 4.

2. ⇒ 3. Trivial.

3. ⇒ 1. By contradiction, let us assume that $G \notin SS(k)$. By Part 3 of Lemma 4, there exists a connected subgraph $G' = (V', E')$ of G such that G' is not a k–spanner of $G[V']$. By Remark 1, there exists an edge $e = (u, v)$ in $G[V']$ that does not belong to E' such that $d_{G'}(u, v) > k$. This results in a simple path of length at least $k + 1$, a contradiction. □

From Part 2, it follows that if want to find the minimal stretch factor $minS(G)$ of a graph G, we have to examine the longest simple cycles of G. As a second consequence of the previous theorem, $SS(k)$ is closed under subgraphs:

Corollary 9. *If G is in $SS(k)$ for some fixed k then also every subgraph of G is in $SS(k)$.*

Proof. This is an immediate consequence of Part 3 of Theorem 8. In fact, the length of a longest simple path in G cannot increase in any subgraph of G. □

Together with Part 2 of Lemma 4, the previous corollary shows that in particular the *spanning trees* play an important role for the determination of the stretch factor of a graph, and thus for the characterization of a graph in terms of $SS(k)$.

As a further consequence of Theorem 8, it is possible to describe the graphs that belong to $SS(k)$ for small stretch factors k.

Lemma 10. *Let G be a graph. Then following characterizations hold:*

1. *$G \in SS(1)$ if and only if every maximal biconnected component of G is a K_2;*
2. *$G \in SS(2)$ if and only if every maximal biconnected component of G is a K_3 or a K_2;*
3. *$G \in SS(3)$ if and only if every maximal biconnected component of G is a diamond, a K_4, a K_3, or a K_2.*

Proof. The characterization of $SS(1)$ directly follows from the definition of k–self-spanners. Now observe that any biconnected graph G with at least four vertices has a stretch factor $minS(G) \geq 3$ and that every diamond D has a stretch factor $minS(D) = 3$. The extension of a diamond to another biconnected graph by adding vertices yields either a diamond or a graph G' with a stretch factor $minS(G') \geq 4$. This proves Parts 2 and 3. □

Thus, by Part 1 of the previous lemma, $SS(1)$ is the set of all trees, or if we consider also disconnected graphs the set of all forests.

Although the recognition problem for the class $SS(k)$ is polynomially solvable for small fixed values of k (as in case $k \leq 3$), we show that the problem is hard for the general case. We do this by showing that the complementary problem of the Minimum Self-Spanner Problem is $\mathcal{NP}$-complete.

Theorem 11. *The Minimum Self-Spanner Problem is co-$\mathcal{NP}$-complete.*

Proof. As mentioned in [GJ79] (ND28), the following Longest Circuit Problem is $\mathcal{NP}$-complete: Given a graph $G = (V, E)$ and a positive integer $q \leq |V|$, is there a simple cycle in G of length q or more? By Part 2 of Theorem 8 this is exactly the complementary problem of the Minimum Self-Spanner Problem. □

The following lemma establishes a strong relationship between the classes of k–self-spanners and k–bounded induced distance graphs:

Lemma 12. *For all integers $k \geq 3$, $SS(k) \subseteq BID(\frac{k-1}{2})$.*

Proof. If a graph G belongs to $SS(k)$ then, by Part 2 of Theorem 8, the longest simple cycle of G has at most k edges. In [CD98], it has been shown that if a longest simple cycle of G has at most $k+1$ edges, then the stretch number $s(G)$ of G is at most $(k-1)/2$. □

Notice that $BID(1)$ is the well-known class of *distance-hereditary graphs*: A graph G is distance-hereditary if for each connected induced subgraph G' of G the following holds: $d_{G'}(u, v) = d_G(u, v)$, for all $u, v \in G'$. See for example in [BM86,How77] for a survey on these graphs. Lemma 10 shows that the classes of k–self-spanners for small stretch factors contain only distance-hereditary graphs. Moreover, for all $k \geq 3$, we can find a distance-hereditary graph that contains a longest cycle with k edges. Thus, $SS(k) \cap BID(1) \neq \emptyset$ for all $k \geq 1$.

To conclude this section, observe the following: Theorem 8 and Lemma 10 show that we have to pay for a large delay k, if we ask for a class $SS(k)$ that contains non-trivial networks. This fact is due to the strong constraint for the fault-tolerance that we have used in the definition of k–self-spanners: A k–self-spanner has to guarantee for fixed bounded delay even in case of an *unlimited* number of edge failures. In the light of applicability, this assumption is overly pessimistic. In the next section, we therefore consider limited fault-tolerance.

4 Networks with bounded delay and limited fault-tolerance

In this section, we introduce a class of graphs which models networks in which the delay is bounded by the constant k when *at most* ℓ edge failures occur. As we will see, this class contains representative topologies for networks.

4.1 Definitions and basic observations

We are interested in graphs that exhibit the following property: If (at most) ℓ edges in a graph G fail, then for all pairs of vertices that remain connected the distance constraint ($d_{G'}(u,v) \leq k \cdot d_G(u,v)$, where G' is the graph arising from G by deleting the failed edges) is fulfilled. Thus as opposed to the previous section, the number of faulty edges is limited to a fixed number. We do not care about cases where two vertices are separated by the failure of edges since then the definition of distance does not apply. This leads to the following definition (cf. Definition 3):

Definition 13 ((k, ℓ)–self-spanner).

1. For any fixed real $k \geq 1$ and fixed integer $\ell \geq 0$, a graph $G = (V, E)$ is called a *(k, ℓ)–self-spanner* if for every subgraph $G' = (V, E')$ of G with $|E'| \geq m - \ell, E' \subseteq E$ the following holds: $d_{G'}(u,v) \leq k \cdot d_G(u,v)$, for each pair $\{u,v\}$ of connected vertices in G'.
   Denote the class of all (k,ℓ)–self-spanners by $SS(k,\ell)$. The parameter k is called *stretch factor*, the parameter ℓ is called *fault-tolerance value* of the class $SS(k,\ell)$.
2. For a graph G, $minS_\ell(G)$ denotes the smallest k such that $G \in SS(k,\ell)$, whereas $maxT_k(G)$ denotes the largest ℓ such that $G \in SS(k,\ell)$.

For examples consider again Figure 1. Let G be the graph in (a). Then $minS_1(G) = 2$, $minS_2(G) = 3$, $maxT_2(G) = 1$, and $maxT_3(G) = 2$. Thus G belongs to $SS(2,1)$ and to $SS(3,2)$, but not to $SS(2,2)$. The "opaque cube" as shown in Figure 2 is in $SS(3,1)$.

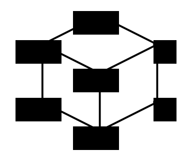

Fig. 2. Opaque cube.

The remarks concerning the choice of the name, non-integer stretch factors, and disconnected graphs apply as in the case of k–self-spanners. Thus, in the following we can restrict ourselves to *positive integer* values for k and to *connected* graphs. In the case of a limited number of edge failures, we can also simplify the procedure to check whether a graph belongs to a class $SS(k,\ell)$: we do not have to consider all (possibly disconnected) subgraphs but only connected subgraphs. We get the following lemma:

Lemma 14. *For any fixed integers $k \geq 1, \ell \geq 0$, $G \in SS(k,\ell)$ if and only if every* connected *and* spanning *subgraph $G' = (V,E')$ with $|E'| \geq m - \ell, E' \subseteq E$, is a k–spanner of G.*

Proof.

$\Rightarrow$ Trivial.

$\Leftarrow$ Suppose that every connected spanning subgraph $G' = (V,E')$ with $|E'| \geq m - \ell, E' \subseteq E$ is a k–spanner of G and, by contradiction, assume that G is *not* a (k,ℓ)–self-spanner. By definition, there is a subgraph $G'' = (V,E'')$ with $|E''| \geq m - \ell, E'' \subseteq E$ (not necessarily connected) such that there is a pair of vertices u and v (within one connected component of G'') and $d_{G''}(u,v) > k \cdot d_G(u,v)$. Since G is connected there is also a connected subgraph $\tilde{G} = (V, \tilde{E})$ with $E'' \subset \tilde{E} \subseteq E$ (and thus $|\tilde{E}| \geq m - \ell$) constructed as follows: Let $\mathcal{C}$ be the set of connected components of G''. Obtain $\tilde{G}$ from G'' by adding $|\mathcal{C}| - 1$ bridge edges such that $\tilde{G}$ is minimally connected. Then $d_{\tilde{G}}(u,v) > k \cdot d_G(u,v)$ and thus $\tilde{G}$ is not a k–spanner of G, a contradiction. □

Note that, as opposed to k–self-spanners, here we cannot directly incorporate vertex failures. Consider for example again the "opaque cube" as shown in Figure 2. As stated above, this graph is in $SS(3,1)$, but the graph G' obtained from removing the internal vertex is not (in fact, it has a stretch factor $minS_1(G) = 5$, and thus is in $SS(5,1)$). Hence, in the case of $SS(k,\ell)$, we purely model edge failures. In the following, we use Lemma 14 as a characterization of the class $SS(k,\ell)$.

Remark 15. Observe that if we want to check whether a given graph belongs to a class $SS(k,\ell)$, by Remark 1, it suffices to check the distance constraint only for faulty edges of each subgraph; i.e. if $G' = (V, E')$ is a subgraph of $G = (V, E)$ with $|E'| \geq |E| - \ell, E' \subseteq E$ we have to check if

$$(*) \quad d_{G'}(u,v) \leq k \text{ for every } e = \{u,v\} \in E \backslash E'.$$

Note that the definition of (k,ℓ)–self-spanners does *not* imply that G is $(\ell+1)$–edge–connected. As stated above, we do not care for pairs of vertices (or edges) that are separated by the edge failures. If we want to take this into account (e.g. to achieve 'true' fault-tolerance, such that we can always guarantee for a 'short' connection between any pair of vertices even in the case of ℓ edge failures) we can restrict our attention to graphs belonging to the intersection of the classes of $(\ell+1)$–edge–connected graphs and (k,ℓ)–self-spanners.

We are interested in characterizational as well as structural aspects of the class of (k,ℓ)–self-spanners. In the next subsection, we consider the problem of recognizing graphs that belong to a given class and investigate characterization problems where we are interested in finding the optimal stretch factor or fault-tolerance value of a given graph. Although the general characterization problem is hard there are several ways of examining the self-spanner properties of certain graphs. We do this in two subsequent subsections: First, we discuss graph operations that allow for efficient construction of self-spanner networks or easy recognition of special cases. In the last subsection, we investigate the self-spanner properties of some popular network topologies.

4.2 Characterization of (k,ℓ)–self-spanners

We are interested in finding (strict) efficient characterizations for the class $SS(k,\ell)$. For this aim we start by stating some (more or less) straightforward results on (k,ℓ)–self-spanners and define the problems to be considered formally. As our main results, we establish an almost complete set of complexity results for these problems.

Trivial cases and straightforward results. Let us first consider some trivial cases:

Lemma 16.

1. *No delay, i.e. $k = 1$:*
   Then, for all $\ell > 0$, $SS(1,\ell)$ is the set of all trees. If we omit the connectivity constraint then $SS(1,\ell)$ is the set of all forests (for all $\ell > 0$). Thus $SS(1)$ and $SS(1,\ell)$ describe the same class of graphs.
2. *No edge failure, i.e. $\ell = 0$:*
   Then, for all $k \geq 1$, $SS(k,0) = SS(1,0)$ is the set of all (connected) graphs.
3. *Weak delay constraints, i.e. large stretch factors:*
   Let G be a connected graph. Then G belongs to $SS(k,\ell)$ for all $k \geq |V| - 1$ for any $\ell \geq 0$.
4. *Strong fault-tolerance constraints, i.e. large fault-tolerance values:*
   Let G be any connected graph. Then, if G belongs to $SS(k, |E|-|V|+1)$ (for some $k \geq 1$), then G also belongs to $SS(k,\ell)$ for all $\ell \geq |E| - |V| + 1$.
   In particular, if G belongs to $SS(k, |E| - |V| + 1)$ then G also belongs to $SS(k)$.

Proof. Parts 1 and 2 are straightforward. To see Part 3, observe that if we allow for a stretch factor of $k \geq |V| - 1$ then we do not really impose a distance constraint: any vertex of G may be used for a detour. It remains to proof Part 4:

If G is a (k, ℓ)–self-spanner for $\ell = |E| - |V| + 1$ then, in particular, every spanning tree of G is a k–spanner of G. If more than $|E| - |V| + 1$ edges fail then the resulting subgraph is necessarily disconnected and thus (by Lemma 14) no further constraints are imposed. □

Thus in the following, given a graph G we will only consider stretch factors of $2 \leq k \leq |V| - 2$ and fault-tolerance values of $1 \leq \ell \leq |E| - |V| + 1$. The cases for small and large stretch factors and small fault-tolerance values can be considered trivial, whereas the case of large fault-tolerance values coincides with the case of k–self-spanners as discussed in Section 3.

By this lemma, it is clear that for every connected graph G there are some parameters k and ℓ such that G belongs to $SS(k, \ell)$. Analogously, if we fix one of the parameters we can always find a feasible value for the other parameter.It is easy to see that (k, ℓ)–self-spanners have inductive properties with respect to the parameters as stated below.

Lemma 17.
1. *If $k \leq k'$ then $SS(k, \ell) \subseteq SS(k', \ell)$.*
2. *If $\ell \leq \ell'$ then $SS(k, \ell') \subseteq SS(k, \ell)$.*

Remark 18. The class $SS(k, \ell)$ is *not* closed under subgraphs (as opposed to Corollary 9 for $SS(k)$; cf. Figure 2). Also it is not closed under supergraphs in the following sense: If a graph G is in $SS(k, \ell)$ for some fixed parameters k and ℓ then there may be a supergraph of G on the same vertex set (i.e. a graph with additional edges) that does *not* belong to $SS(k, \ell)$. The same still holds if we consider only $(\ell + 1)$–edge–connected graphs.

As a consequence of the previous remark, the self-spanner properties cannot be inferred directly from the self-spanner properties of sub- or supergraphs. For examples of standard graphs that exhibit some particular self-spanner properties, it is easy to see that $P_n \in SS(1, \ell)$ for any $\ell \geq 1$ since P_n is a tree. Furthermore $C_n \in SS(n - 1, \ell)$ but $C_n \notin SS(n - 2, \ell)$ for any $\ell \geq 1$, since $minS_\ell(C_n) = n - 1$ for any $\ell \geq 1$ (i.e. the failure of one edge results in a path of length $n - 1$).

Considered problems. Starting from the above observations, we are interested in finding non-trivial parameters such that a graph is a (k, ℓ)–self-spanner. This includes the problem of deciding for given parameters k and ℓ whether a given graph belongs to $SS(k, \ell)$ as well as the more general recognition problems where we fix one of the parameters and try to optimize the other. This brings up the following optimization, resp. characterization problems:

Problem 19 (Minimum Stretch Factor, MinStretch$_\ell$).

Given: A graph G and an integer $k \geq 1$.
Problem: Does G belong to $SS(k, \ell)$, i.e. $minS_\ell(G) \leq k$?

Problem 20 (Maximum Fault-Tolerance Value, MaxTolerance$_k$).

Given: A graph G and an integer $\ell \geq 0$.
Problem: Does G belong to $SS(k, \ell)$, i.e. $maxT_k(G) \geq \ell$?

Problem 21 (General Self-Spanner Problem).

Given: A graph G and $k \geq 2, \ell \geq 1$.
Problem: Does G belong to $SS(k, \ell)$?

Complexity results. We now turn to analyzing the complexity of the problems mentioned above. Let us first consider the special case where we allow for single edge failures only, i.e. $\ell = 1$.

Lemma 22. *$G \in SS(k, 1)$ if and only if every edge of G is either a bridge or belongs to an induced cycle of length at most $k + 1$.*

Proof.
$\Leftarrow$ Let e be an arbitrary edge of G and consider $G' = G - e$. We have to show property (*) of Remark 15. If e is a bridge in G then G' is disconnected and there is nothing to show. If e is not a bridge then G' remains connected and by assumption G' is a k–spanner of G.
$\Rightarrow$ (by contradiction) Assume $G \in SS(k, 1)$, and there is an edge $e = \{u, v\}$ that is not a bridge and that does not belong to an induced cycle of length at most $k + 1$. Consider $G' = G - e$. Then G' is connected and $d_{G'}(u, v) > k$, a contradiction. □

Considering multiple edge failures, it is clear that bridges again do not contribute to the stretch factor. But unfortunately we cannot extend the characterization in a straightforward way. If we restrict ourselves to $(\ell + 1)$–edge–connected graphs we get the following lemma:

Lemma 23. *Let $G = (V, E)$ be $(\ell + 1)$–edge–connected. Then $G \in SS(k, \ell)$ if and only if for every edge $e = \{u, v\}$ of G there are at least ℓ edge disjoint paths (not involving e) of length at most k connecting u and v.*

Proof.
$\Leftarrow$ Consider a subgraph $G' = (V, E')$ with $E' \subseteq E$ and $|E'| \geq |E| - \ell$, and an edge $e = \{u, v\} \in E \backslash E'$. There are ℓ edge disjoint paths (not involving e) of length at most k connecting u and v. Thus, even if the remaining $\ell - 1$ edge failures happen to appear in one of these paths each, at least one covering path for e in G' remains.
$\Rightarrow$ By contradiction, assume $G \in SS(k, l)$, and there is an edge $e = \{u, v\}$ such that there are at most $j < \ell$ edge disjoint paths (not involving e) of length at most k connecting u and v. Since j is maximal, there are j edges within the edge disjoint paths such that the following holds: the subgraph G' constructed by deleting e and these selected edges remains connected (since G is $(\ell + 1)$–edge–connected) but $d_{G'}(u, v) > k$, a contradiction to $G \in SS(k, \ell)$. □

Observe that we cannot relax on the edge–connectivity constraint in this lemma. Consider for example the diamond consisting of a C_4 and one chord: this graph belongs to $SS(3, 2)$ but it does not fulfill the constraints of Lemma 23.

Now, if we fix the fault-tolerance value ℓ, we can determine the smallest possible stretch factor of a given graph in polynomial time:

Theorem 24. *MinStretch$_\ell$ is in $\mathcal{P}$ for all $\ell \geq 0$.*

Proof. Let $\mathcal{G} = \{G' \mid G' = (V, E'), |E'| \geq |E| - \ell\}$ be the set of all subgraphs of the given graph $G = (V, E)$ in which we have removed at most ℓ edges. Then a straightforward (brute-force, rather naive) algorithm to solve the MinStretch$_\ell$ is as follows:

For any $G' \in \mathcal{G}'$, and for any edge $e = \{u, v\} \in E$, run your favorite shortest-path algorithm on G' for the pair $\{u, v\}$ and determine $stretch(G', e) := d_{G'}(u, v)$ if u and v remain connected in G'. Otherwise let $stretch(G', e) := 0$. Then $minS_\ell(G) = \max_{G'} \max_e \{stretch(G', e)\}$.

This algorithm can be implemented in polynomial time since

$$|\mathcal{G}| = \sum_{i=1}^{\ell} \binom{m}{i} \le \sum_{i=1}^{\ell} m^i \le \ell \cdot m^\ell \le n^{2(\ell+1)},$$

where $n = |V|, m = |E|$ and, for each element of $\mathcal{G}$, we have to repeat a polynomial time shortest-path algorithm at most n^2 times. □

Observe that the running time is bounded by a polynomial in n that contains ℓ as an exponent, but here we consider ℓ fixed. Surely, this algorithm and its analysis are rather naive and the running time may be lowered significantly. But here, we do not elaborate on that. As a consequence of the previous theorem we also have:

Corollary 25. *The problem of deciding whether a graph is a (k,ℓ)–self-spanner for some given parameters $k \ge 1$ and $\ell \ge 0$ is in $\mathcal{P}$.*

Now, we consider the dual problem where we fix the stretch factor and we want to find the largest fault-tolerance value of a given graph. As stated in Lemma 23, to solve this, it is crucial to find edge disjoint paths between any two vertices such that the paths have bounded length. Unfortunately, this problem is hard (cf. [GJ79], (ND41)):

Problem 26 (Maximum Length-Bounded Disjoint Paths).

Given: A graph G, two vertices s and t, positive integers $K, L \le n$.

Problem: Does G contain L or more mutually edge disjoint paths from s to t, which all have length at most K?

As shown in [IPS82], the problem is $\mathcal{NP}$-complete for all fixed $K \ge 5$; it is polynomially solvable for $K \le 3$, and it is open for $K = 4$. The proofs given there work for $(\ell+1)$–edge–connected graphs as well. Observe that the problem of finding the maximum number of edge disjoint paths from s to t, under *no length constraint*, is solvable in polynomial time by standard network flow techniques. But this does not help here, since we need to guarantee the length constraints. Together with the observation that we can decide in polynomial time whether a given graph is $(\ell+1)$–edge–connected, we get the following results:

Theorem 27.

1. *MaxTolerance$_k$ is $\mathcal{NP}$-complete for all $k \ge 5$.*
2. *MaxTolerance$_k$ is in $\mathcal{P}$ for $k = 1, 2, 3$.*
3. *The General Self-Spanner Problem is $\mathcal{NP}$-complete.*

Thus, only MaxTolerance$_4$ remains to be settled.

4.3 Graph operations for constructing (k,ℓ)–self-spanners

We are interested in structural aspects of the class of (k,ℓ)–self-spanners. In particular, we want to find (easy) operations that allow for efficient construction of self-spanner networks or easy recognition of special cases. Here, we consider two well-known graph operations, namely the *Cartesian product* and the *split composition* of graphs, and show how these operations affect the construction of (k,ℓ)–self-spanners.

Cartesian product. The Cartesian product is defined as follows (cf. [Har69]):

Definition 28 (Cartesian product). Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The *Cartesian product* $G := G_1 \times G_2$ is the graph having vertex set V and edge set E as follows:

- $V = \{(x_1, x_2) \mid x_1 \in V_1, x_2 \in V_2\}$
- $E = \{\{(x_1, x_2), (y_1, y_2)\} \mid (x_1 = y_1 \text{ and } \{x_2, y_2\} \in E_2) \text{ or } (x_2 = y_2 \text{ and } \{x_1, y_1\} \in E_1)\}$.

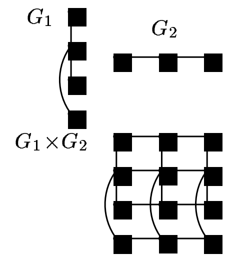

Fig. 3. Example for Cartesian product.

Thus, two vertices of $G_1 \times G_2$ are adjacent if and only if the first components are equal and the second components form an edge in G_2 or vice versa. See Figure 3 for an example. Many graphs that are used for modeling common network topologies can be defined in terms of Cartesian product of simpler graphs. We will use this fact in the following subsection. The following properties of Cartesian product graphs are straightforward.

Lemma 29. *Let $G := G_1 \times G_2$ for two graphs G_1 and G_2. Then*

1. $|V| = |V_1| \cdot |V_2|$, $|E| = |V_2| \cdot |E_1| + |V_1| \cdot |E_2|$.
2. *If G_1 or G_2 are* not *connected then G is* not *connected.*
3. *Let $cc(G)$ denote the number of connected components of G. Then $cc(G) = cc(G_1) \cdot cc(G_2)$.*
4. *If G_1 is c_1–edge–connected and G_2 is c_2–edge–connected then G is $(c_1 + c_2)$–edge–connected.*
5. *For any $x_1 \in V_1$, $G[\{(x_1, x_2) \mid x_2 \in V_2\}]$ is isomorphic to G_2, and analogously for any $x_2 \in V_2$, $G[\{(x_1, x_2) \mid x_1 \in V_1\}]$ is isomorphic to G_1.*

Since we are again only interested in connected components of G it suffices to consider *connected* graphs G_1 and G_2. The next lemma shows that graphs that arise from the Cartesian product of two graphs have strong self-spanner properties. In particular, it indicates that a stretch factor of 3 plays an important role.

Theorem 30. *Let G_1, G_2 be two arbitrary connected graphs. Then the following holds:*

1. *If $G_1 \in SS(k_1, \ell_1)$ and G_1 $(\ell_1 + 1)$–edge–connected, and $G_2 \in SS(k_2, \ell_2)$ and G_2 $(\ell_2 + 1)$–edge–connected then $G_1 \times G_2 \in SS(\max\{k_1, k_2\}, \min\{\ell_1, \ell_2\})$.*
2. *Let δ be the minimum vertex degree of both G_1 and G_2. Then $G_1 \times G_2 \in SS(3, \delta)$. In particular, $G_1 \times G_2 \in SS(3, 1)$.*
3. *$G_1 \times G_2 \in SS(2, \ell)$ if and only if every edge in G_1 (and G_2, resp.) belongs to at least ℓ disjoint triangles in G_1 (and G_2, resp.).*
4. *If G_1 or G_2 contains a bridge then $maxT_2(G) = 0$, i.e. there is no $\ell > 0$ such that $G_1 \times G_2 \in SS(2, \ell)$. In particular, if G_1 or G_2 contains a bridge and $G_1 \times G_2 \in SS(k, \ell)$ for some $\ell > 0$ then $k \geq 3$.*

Proof. Let $G = (V, E) = G_1 \times G_2$, $G_1 = (V_1, E_1)$, and $G_2 = (V_2, E_2)$.

1. Consider edge $e = \{(x_1, x_2), (y_1, y_2)\}$ in G. It suffices to show that $d_{G'}((x_1, x_2), (y_1, y_2)) \leq \max\{k_1, k_2\}$, where G' is obtained from G by removing the edge e and $\min\{\ell_1, \ell_2\} - 1$ other arbitrary edges. By Part 5 of Lemma 29, $e \in E$ belongs to an induced subgraph G'' isomorphic to G_1 or to G_2. By assumption, $G_1 \in SS(k_1, \ell_1)$ and $G_2 \in SS(k_2, \ell_2)$. Hence, after the removal of at most $\min\{\ell_1, \ell_2\}$ edges (including e) from G'', G'' remains connected and the distance between (x_1, x_2) and (y_1, y_2) is at most $\max\{k_1, k_2\}$.

2. Consider edge $e = \{(x_1, x_2), (y_1, y_2)\}$ in G. We have to show that $d_{G'}((x_1, x_2), (y_1, y_2)) \leq 3$, where G' is obtained from G by removing e and $\delta - 1$ other arbitrary edges. By definition of the Cartesian product, either $x_1 = y_1$ and $\{x_2, y_2\} \in E_2$, or $x_2 = y_2$ and $\{x_1, y_1\} \in E_1$. W.l.o.g. assume the first case. Since G_1 has minimum vertex degree δ there exist δ distinct vertices x_1^j in $V_1, 1 \leq j \leq \delta$, such that $\{x_1, x_1^j\} \in E_1$. The existence of these edges in G_1 imply that $\{(x_1, x_2), (x_1^j, x_2)\}, \{(x_1^j, x_2), (x_1^j, y_2)\}$, and $\{(x_1^j, y_2), (x_1, y_2)\}$ are disjoint edges in G for $1 \leq j \leq \delta$. These three edges each form disjoint paths in G from vertex (x_1, x_2) to vertex (y_1, y_2) (since by assumption $x_1 = y_1$), and hence $d_{G'}(\{(x_1, x_2), (y_1, y_2)\}) \leq 3$.
3. $\Leftarrow$ Trivial.
   $\Rightarrow$ Consider edge $e = \{(x_1, x_2), (y_1, y_2)\}$ in G and w.l.o.g. assume that $x_1 = y_1$ and $\{x_2, y_2\} \in E_2$. Since $G \in SS(2, \ell)$, there are ℓ edge disjoint paths from (x_1, x_2) to (y_1, y_2) of length at most 2 in G not using e. According to the proof of Part 2, any path from (x_1, x_2) to $(x_1 \equiv y_1,\ y_2)$ via a vertex (v, w) with $v \neq x_1$ has length at least 3. Thus, there are vertices $z_j \in V_2$ such that $\{(x_1, x_2), (x_1, z_j)\}, \{(x_1, z_j), (x_1, y_2)\} \in E$, and hence $\{x_2, z_j\}, \{z_j, y_2\} \in E_2$ for $1 \leq j \leq \ell$. Thus e belongs to ℓ disjoint triangles in G_2. The same arguments also hold for G_1.
4. Part 4 is a special case of Part 3. □

Observe that, for Part 1 of the previous theorem, it is really necessary to claim the respective edge connectivity. Otherwise we cannot guarantee that the graph considered in the proof remains connected. Also, for Part 3 of that theorem, it does not suffice to claim that $G_1 \in SS(2, \ell)$ (and $G_2 \in SS(2, \ell)$, resp.): we again need that both graphs are $(\ell + 1)$–edge–connected. For smaller stretch factors, i.e. $k = 1$, we already know that $G_1 \times G_2$ has a stretch factor smaller than 2 if and only if it is a tree.

Remark 31. Part 2 of Lemma 30 is tight in the following sense: If $G_i \notin SS(2, 1)$ and G_i has minimum vertex degree $\delta (i = 1, 2)$, then $minS_\delta(G_1 \times G_2) = 3$ and $maxT_3(G_1 \times G_2) = \delta$. Thus $G_1 \times G_2 \in SS(3, \delta)$, but $G_1 \times G_2 \notin SS(2, \delta)$ and $G_1 \times G_2 \notin SS(3, \delta + 1)$.

Split composition. The split composition is defined as follows:

Definition 32 (split composition). Let $G_1 = (V_1 \cup \{m_1\}, E_1)$ and $G_2 = (V_2 \cup \{m_2\}, E_2)$, where $V_1 \cap V_2 = \emptyset$. The vertices m_1 and m_2 are called *marked vertices*. The *split composition* $G := G_1 * G_2$ is the graph having vertex set V and edge set E as follows:

– $V = V_1 \cup V_2$,
– $E = E(G_1[V_1]) \cup E(G_2[V_2]) \cup \{\{x, y\} | x \in N_{G_1}(m_1), y \in N_{G_2}(m_2)\}$, where $N_G(v)$ denotes the neighborhood of vertex v in G.

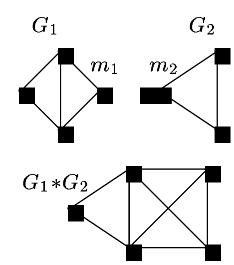

Fig. 4. Example for split composition.

That means that $G := G_1 * G_2$ is obtained from G_1 and G_2 by taking all edges that lie within V_1 and V_2, and all neighbors of m_2 in G_2 are connected to all neighbors of m_1 in G_1. See Figure 4 for an example. Observe that for all $y \in N_{G_2}(m_2)$, $G[V_1 \cup \{y\}]$ is isomorphic to G_1. The same holds analogously for G_2.

The split composition is the inverse of the decomposition operation introduced in [Cun82]. In [Bra93], split composition has been used for example to build distance-hereditary graphs, complete graphs, complete bipartite graphs, trees, and cographs. In [CD98] it is shown how split composition can be used to construct k–bounded induced distance graphs. The following lemma shows how self spanner properties can be pertained when using the split composition.

Theorem 33. *Let $G_i \in SS(k_i, \ell_i)$, $i = 1, 2$. Then $G_1 * G_2 \in SS(\max\{k_1, k_2\}, \min\{\ell_1, \ell_2\})$.*

Proof. Any edge $\{x, y\}$ in $G_1 * G_2$ belongs to an induced subgraph isomorphic to G_1 or to G_2. The rest of the proof follows the same lines as the proof of Part 1 of Lemma 30. □

It is clear that the stretch factor and fault-tolerance value of the previous lemma are tight: If k_1 and k_2 are minimum stretch factors for fault-tolerance values ℓ_1 and ℓ_2, then $G_1 * G_2$ cannot have a stretch factor less than $\max\{k_1, k_2\}$ for the fault-tolerance value $\min\{\ell_1, \ell_2\}$.

4.4 Self-Spanner properties of some popular network topologies

As we have seen in the previous subsection, we can construct graphs that exhibit certain self-spanner properties by using the Cartesian product or split composition. We now follow the opposite approach and examine some network topologies that are used widely with respect to their self-spanner properties. In particular, we consider meshlike networks like *grid*, *torus*, and *hypercube*. As examples for hypercube derived networks we investigate *cube connected cycles* and *butterfly* networks. Especially the last two topologies, which are bounded-degree approximations of the hypercube, are widely studied (and used) in the interconnection network community (see for example [Lei92,ABR90] and the references therein).

Grids, tori, and hypercubes. The topologies which are easiest in structure are the meshlike networks: A *grid* $G_{n,m}$ (with $n, m \geq 2$) has $m \cdot n$ vertices labeled with distinct pairs (i, j), where $1 \leq i \leq n$ and $1 \leq j \leq m$. Two vertices are adjacent if their labels differ by 1 in exactly one coordinate. A grid $G_{n,m}$ with wrap-around edges is called a *torus*, and it is denoted by $T_{n,m}$. The *d-dimensional binary hypercube*, H_d, $d \geq 1$ has 2^d vertices which are labeled with the binary strings of length d. Two vertices in H_d are adjacent if their labels differ in exactly one bit. These three topologies can be constructed by (multiple) application of the Cartesian product:

- The hypercube H_d is recursively defined from P_2 by $H_d = P_2 \times H_{d-1} = \underbrace{P_2 \times \ldots \times P_2}_{d \text{ times}}$;
- the grid $G_{n,m}$ is the Cartesian product $P_n \times P_m$ for $n, m \geq 2$;
- the torus $T_{n,m}$ is the Cartesian product $C_n \times C_m$ for $n, m \geq 3$.

The following lemma indicates the self-spanner properties of these topologies. Observe that the fault-tolerance value of the torus is higher than that of the grid due to the additional wrap-around connections which make the topology symmetric. But it is clear from Remark 18, that the addition of edges does not result in higher fault-tolerance values in general.

Theorem 34.

1. *$G_{n,m}$ belongs to $SS(3, 1)$, but not to $SS(2, 1)$.*
   If $n > 2$ or $m > 2$ then $G_{n,m}$ does not belong to $SS(3, 2)$.
   If $n, m > 2$ then $G_{n,m}$ belongs to $SS(5, 2)$, but not to $SS(4, 2)$ or $SS(5, 3)$.
2. *$T_{n,m}$ belongs to $SS(3, 2)$, but not to $SS(2, 2)$.*
   If $n > 3$ or $m > 3$ then $T_{n,m}$ does not belong to $SS(3, 3)$.
   $T_{n,m}$ belongs to $SS(\min\{5, \max\{n, m\} - 1\}, 3)$.
   If $n, m \geq 5$ then $T_{n,m}$ belongs to $SS(5, 4)$, but not to $SS(4, 4)$.
   If $n, m > 5$ then $T_{n,m}$ does not belong to $SS(5, 5)$.
3. *H_d belongs to $SS(3, d - 1)$, but not to $SS(3, d)$ or to $SS(2, 1)$.*

Proof.

1. $G_{n,m} \in SS(3,1)$ and $G_{n,m} \notin SS(2,1)$ are immediate consequences of Parts 2 and 4 of Lemma 30. To see the other self-spanner properties, observe that for any edge on the border of the grid there is only one path of length 3 connecting the endpoints of that edge, all other paths have length 5 or longer. This 3-path (and the edge itself) may be broken by a double edge failure such that the endpoints still remain connected (if n, m are large enough). Thus, $G_{n,m} \in SS(5,2)$ and if $G_{n,m} \neq C_4$ then $G_{n,m} \notin SS(4,2)$ and if $n, m > 2$, $G_{n,m} \notin SS(5,3)$.
2. Parts 2 and 3 of Lemma 30 imply that $T_{n,m} \in SS(3,2)$ and $T_{n,m} \notin SS(2,2)$. From Remark 31 it follows that $T_{n,m} \notin SS(3,3)$, if $m > 3$ or $n > 3$. Observe that $T_{3,3} \in SS(3,3)$. For every edge $\{x, y\}$ in $T_{n,m}$ there are two edge disjoint paths of length 3 connecting x and y and one (also disjoint) path of length at most $\max\{n, m\}-1$. If n and m are at least 5 then there are six different paths of length 5 connecting x and y, but only two of length at most 4. It is easy to see that at least one of these paths of length 5 remains complete if $\{x, y\}$ and three further edges are removed. If n and m are at least 6, consider the case of failure of five direct parallel edges in $T_{n,m}$: $T_{n,m}$ remains connected and the middle failing edge has a stretch factor that is greater than 5. Thus, $T_{n,m} \in SS(\min\{\max\{n, m\} - 1, 5\}, 3)$. For m, n large enough, $T_{n,m} \in SS(5,4)$, but $T_{n,m} \notin SS(4,4)$ and $T_{n,m} \notin SS(5,5)$.
3. To show that H_d belongs to $SS(3, d-1)$, but not to $SS(3, d)$, it is sufficient to observe that every edge e of H_d belongs to $d-1$ induced cycles of length 4 which are edge disjoint apart from e. By Part 4 of Lemma 30, H_d does not belong to $SS(2,1)$. □

Hypercube derived networks. We now consider two different types of bounded-degree approximations of the hypercube. We (mostly) follow the notation as in [HPS94].

The *cube-connected cycles graph* of dimension d, denoted CCC_d, is derived from H_d by replacing each vertex of H_d by a *fundamental cycle* of length d. Each vertex of such a cycle is labeled by a tuple $(i, x), 0 \leq i \leq d-1$, and i is called the *level* of the vertex. Apart from the *cycle edges* of the fundamental cycles, each vertex (i, x) is connected to vertex $(i, x(i))$, where $x(i)$ denotes the vertex of H_d that is labeled by the same string as vertex x but with bit i flipped. These edges are called *hypercube edges*.

The *butterfly graph* (with wrap-around) of dimension d, denoted B_d, is derived from H_d similarly as CCC_d: B_d consists of the same vertices $(i, x), 0 \leq i \leq d-1$, as CCC_d and the same *fundamental cycles* of length d. But now each vertex (i, x) is connected by two *hypercube edges* to vertices $(i+1, x(i))$ and $(i-1, x(i-1))$.

CCC_d can be obtained from B_d by replacing each pair of edges $\{(i, x), (i+1, x(i))\}$ and $\{(i, x), (i-1, x(i-1))\}$ by a single edge $\{(i, x), (i, x(i))\}$. Thus, CCC_d can be viewed as a spanning subgraph of B_d. In [ABR90], it is shown that different hypercube-derived topologies can be embedded within other such topologies with small slowdown. Results on the existence of cycles and the construction of k–spanners can be found in [Ros91] and [HPS94], respectively. But all these results do not imply on the self-spanner properties of the topologies studied here. We get the following results concerning the self-spanner properties of the topologies above:

Theorem 35.

1. *B_d belongs to $SS(3,1)$ and to $SS(d+1,2)$, but not to $SS(2,1)$, $SS(d,2)$, or $SS(d+1,3)$.*
2. *CCC_d belongs to $SS(7,1)$ and to $SS(\max\{7, d-1\}, 2)$, but not to $SS(6,1)$.*

Proof.

1. Any edge of B_d belongs to exactly one induced cycle of length 4 consisting of two cycle edges and two hypercube edges. Thus $B_d \in SS(3,1)$. From [Ros91], we know that B_d does not contain a cycle of length 3 if $d > 3$. For smaller d it is clear that there is no cycle of length 3 containing a hypercube edge. Thus $B_d \notin SS(2,1)$.
   Now consider the case when two edges fail: If two edges of the same fundamental cycle fail there still remains a path of length 3 connecting the endpoints of the faulty edges each. If both cycle edges of a 4-cycle as mentioned above fail then there always remains a path of length $d-1$ via a fundamental cycle, but no shorter one. If a cycle edge and a hypercube edge within such a 4-cycle fail then a shortest path of length $d+1$ remains but not two such paths.
2. CCC_d consists of the same fundamental cycles as B_d but contains only half of the hypercube edges. This results in longer cycles: For every *hypercube edge*, there are two (shortest) edge disjoint paths of length 7 which connect the endpoints. For every *cycle edge*, there is a path of length $d-1$ (via the fundamental cycle) and another (disjoint) path of length 7 using hypercube edges. Thus $CCC_d \in SS(7,1)$, $CCC_d \in SS(\max\{7, d-1\}, 2)$, and $CCC_d \notin SS(6,1)$. □

The previous lemma shows that bounded-degree approximations of the hypercube like CCC_d and B_d perform poorly with respect to their self-spanner properties: In the case of single edge failures the stretch factor is still a constant (though much larger than for the hypercube), but for double edge failures the stretch factor grows linearly with the dimension d. Thus the guarantees for delays in case of failures are really weak for these kinds of topologies.

5 Conclusion and future work

In this paper, we have introduced the classes of k–self-spanners and (k,ℓ)–self-spanners which model survivable networks with bounded delay. Thus, these networks guarantee constant delay factors even in the case of multiple edges failures. We have considered both the cases of unlimited and limited number of edge failures. We have given characterizational as well as structural results, and we have shown that some popular network topologies exhibit (more or less) strong self-spanner properties.

We consider this work as a first step towards a more general approach to the design of survivable networks with bounded delay. On one hand, we are interested in further investigating the self-spanner properties of other popular topologies. On the other hand, we would like to design sparse (k,ℓ)–self-spanner network for given parameters k and ℓ and specific connectivity requirements.

References

[ABR90] Fred S. Annexstein, Marc Baumslag, and Arnold L. Rosenberg. Group action graphs and parallel architectures. *SIAM J. Comput.*, 19(3):544–569, 1990.

[ADD+93] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9:81–100, 1993.

[BM86] H.J. Bandelt and H.M. Mulder. Distance–hereditary graphs. *J. of Combinatorial Theory, Series B*, 41:182–208, 1986.

[Bra93] Andreas Brandstädt. Special graph classes — a survey. Technical Report SM–DU–199, Universität Gesamthochschule Duisburg, 1993.

[Cai94] Leizhen Cai. $\mathcal{NP}$-completeness of minimum spanner problems. *Discrete Applied Math.*, 48:187–194, 1994.

[CC95] Leizhen Cai and D.G. Corneil. Tree spanners. *SIAM J. Discrete Math.*, 8(3):359–387, 1995.

[CD98] Serafino Cicerone and Gabriele Di Stefano. Graphs with bounded induced distance. In *Proceedings 24rd International Workshop on Graph-Theoretic Concepts in Computer Science, WG'98*, pages 177–191. Springer-Verlag, Lecture Notes in Computer Science, vol. 1517, 1998.

[CMS97] R. Cole, Bruce M. Maggs, and Ramesh K. Sitaraman. Reconfiguring arrays with faults part I: Worst-case faults. *SIAM J. Comput.*, 26(6):1581–1611, 1997.

[Cun82] W.H. Cunningham. Decomposition of directed graphs. *SIAM J. Alg. Disc. Meth.*, 3:214–228, 1982.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$–Completeness.* W H Freeman & Co Ltd, 1979.

[Han98] Dagmar Handke. Independent tree spanners: Fault-tolerant spanning trees with constant distance guarantees (extended abstract). In *Proceedings 24rd International Workshop on Graph-Theoretic Concepts in Computer Science, WG'98*, pages 203–214. Springer-Verlag, Lecture Notes in Computer Science, vol. 1517, 1998.

[Har69] Frank Harary. *Graph theory.* Addison–Wesley Publishing Company, 1969.

[How77] Edward Howorka. A characterization of distance-hereditary graphs. *Quart. J. Math., Oxford*, 2(28):417–420, 1977.

[HPS94] Marie-Claude Heydemann, Joseph G. Peters, and Dominique Sotteau. Spanners of hypercube-derived networks. Technical Report CMPT TR 94-02, School of Computer Science, Simon Fraser University, Burnaby, Canada, April 1994.

[IPS82] Alon Itai, Y. Perl, and Yossi Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12:277–286, 1982.

[Lei92] Frank Thomson Leighton. *Introduction to parallel algorithms and architectures: arrays, trees, hypercubes.* Morgan Kaufman Publishers, 1992.

[LMS92] Frank Thomson Leighton, Bruce M. Maggs, and Ramesh K. Sitaraman. On the fault tolerance of some popular bounded-degree networks. Technical Report CS-TR-385-92, Dept. of Computer Science, Princeton University, New Jersey, 1992.

[PU87] David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. In *Proceedings 1987 6th ACM Symposium on Principles of Dist. Comp., Vancouver*, pages 77–85, 1987.

[Ros91] Arnold L. Rosenberg. Cycles in networks. Technical Report UM-CS-1991-020, Dept. of Computer and Information Science, Univ. of Massachusetts, 1991.

[Soa92] José Soares. Graph spanners: a survey. *Congressus Numerantium*, 89:225–238, 1992.