# Maximally Resilient Replacement Paths for a Family of Product Graphs

**Mahmoud Parham**
University of Vienna, Faculty of Computer Science, Vienna, Austria
mahmoud.parham@univie.ac.at

**Klaus-Tycho Foerster**
University of Vienna, Faculty of Computer Science, Vienna, Austria
klaus-tycho.foerster@univie.ac.at

**Petar Kosic**
University of Vienna, Faculty of Computer Science, Vienna, Austria
petar.kosic@univie.ac.at

**Stefan Schmid**
University of Vienna, Faculty of Computer Science, Vienna, Austria
stefan_schmid@univie.ac.at

## Abstract

Modern communication networks support fast path restoration mechanisms which allow to reroute traffic in case of (possibly multiple) link failures, in a completely *decentralized* manner and without requiring global route reconvergence. However, devising resilient path restoration algorithms is challenging as these algorithms need to be inherently *local*. Furthermore, the resulting failover paths often have to fulfill additional requirements related to the policy and function implemented by the network, such as the traversal of certain waypoints (e.g., a firewall).

This paper presents local algorithms which ensure a maximally resilient path restoration for a large family of product graphs, including the widely used tori and generalized hypercube topologies. Our algorithms provably ensure that even under multiple link failures, traffic is rerouted to the other endpoint of every failed link whenever possible (i.e. *detouring* failed links), enforcing waypoints and hence accounting for the network policy. The algorithms are particularly well-suited for emerging segment routing networks based on label stacks.

**2012 ACM Subject Classification** Networks → Routing protocols; Computer systems organization → Dependable and fault-tolerant systems and networks; Mathematics of computing → Graph algorithms

**Keywords and phrases** Product Graphs, Resilience, Failures, Routing

**Digital Object Identifier** 10.4230/LIPIcs.OPODIS.2020.26

**Funding** Research (in part) supported by the Vienna Science and Technology Fund (WWTF) project WHATIF, ICT19-045, 2020-2024.

## 1 Introduction

Communication networks have become a critical infrastructure of our society. With the increasing size of these networks, however, link failures are more common [2, 10], which emphasizes the need for networks that provide a reliable connectivity even in failure scenarios, by quickly rerouting traffic. As a global re-computation (and distribution) of routes after failures is slow [21], most modern communication networks come with fast *local* path restoration mechanisms: conditional failover rules are *pre-computed*, and take effect in case of link failures *incident* to a given router.

Devising algorithms for such path restoration mechanisms is challenging, as the failover rules need to be *(statically) pre-defined* and can only depend on the *local* failures; at the same time, the mechanism should tolerate multiple or ideally, a *maximal* number of failures (as long as the underlying network is still connected), no matter where these failures may occur. Furthermore, besides merely re-establishing connectivity, reliable networks often must also account for additional network properties when rerouting traffic: unintended failover routes may disrupt network services or even violate network policies. In particular, it is often important that a flow, along its route from $s$ to $t$, visits certain policy and network function critical "waypoints", e.g., a firewall or an intrusion detection system, even if failures occur.

Today, little is known about how to provably ensure a high resiliency under multiple failures while preserving visits to waypoints. This paper is motivated by this gap. In particular, we investigate local path restoration algorithms which do not only provide a maximal resilience to link failures, but also never "skip" nodes: rather, traffic is rerouted around failed links individually, hence *enforcing waypoints* [1].

## 1.1 Related Work

**Motivation.** Resilient routing is a common feature of most modern communications networks [6], and the topic has already received much interest in the literature. However, most prior research on static fast rerouting aims at restoring connectivity to the final destination, without considering waypoint properties as in our work. Such waypoint preservation is motivated by the advent of (virtualized [11]) middleboxes [4], respectively *local protection schemes* in Multiprotocol Label Switching (MPLS) terminology [26], and by the recent emergence of Segment Routing (SR), where routing is based off label stacks – more precisely by the label on top of the stack [24], which is treated as the next routing destination.

**Path restoration.** Only little is known today about static fast rerouting under multiple failures, while preserving waypoints. In TI-MFA [16], it has been shown that existing solutions for SR fast failover, based on TI-LFA [20], do not work in the presence of two or more failures. However, TI-MFA [16] and non-SR predecessors [22] rely on failure-carrying packets, which is undesirable as discussed before and we overcome in the current paper.

For the case of two failures, heuristics [9] exist, but they do not provide any formal protection guarantees, except for torus graphs [23]. Beyond a single failure [20] in general and two failures on the torus [23], we are not aware of any approaches that work in our model, except for a recent work on binary hypercubes [17]. However, it is not clear how to extend [17] to e.g. generalized hypercubes, and the approach followed in this paper presents a more generic scheme for the Cartesian product of *any* set of base graphs, as long as "well-structured" base graph schemes are provided.

**Connectivity restoration without waypoints.** Static fast failover mechanisms without waypoints are investigated by Chiesa et al. [5,7,8] leveraging arc-disjoint network decompositions, also by Elhourani et al. [10], Stephens et al. [27,28], and Schmid et al. [3,14,15,18,19,25]. Beyond that, the concept of perfect resilience (any number of failures) is investigated in [12,13,29]. Even though it is possible to provide $\Omega(k)$-resilience in $k$-connected graphs, this guarantee pertains only to reaching the destination, and does not transfer to link protection.

## 1.2 Contributions

We initiate the study of local (i.e., *immediate*) path restoration algorithms on product graphs, an important class of network topologies. More specifically, our algorithms are 1) resilient to a maximum number of failures (i.e., are *maximally robust*), 2) respect the (waypoint) path traversal of the original route (by detouring failed links), and 3) are compatible with current technologies, and in particular with emerging segment routing networks [24]: our algorithms do not require packets to carry failure information, routing tables are static, and forwarding just depends on the packet's top-of-the-stack destination label and the incident link failures.

Our main result is an efficient scheme that can provide maximally resilient backup paths for arbitrary Cartesian product of given base graphs, as long as "well-structured" schemes are provided for the base graphs. Using complete graphs, paths, and cycles as base graphs, we can generate maximally resilient schemes for additional important network topologies such as grids, tori, and generalized hypercubes.

## 1.3 Organization

The remainder of this paper is organized as follows. We first introduce necessary model preliminaries in Section 2, followed by our main result in Section 3, where we provide a general scheme to compute maximally resilient path restoration schemes for product graphs. We then show how our scheme can be leveraged for specific graph classes in Section 4, for the selected examples of complete graphs, generalized hypercubes, grids, and torus graphs. We conclude our study in Section 5 with a few open questions.

## 2 Preliminaries

We consider undirected graphs $G = (V, E)$ where $V$ is the set of *nodes* and $E$ is the set of *links* connecting nodes.

▶ **Definition 1.** *A* backup path *(a.k.a. replacement path) for a link $\ell \in E$ is a simple path that connects the endpoint of the link $\ell$. Let $\mathcal{P}$ be the set of all backup paths in a graph. An injective function $BP_G : E \to \mathcal{P}$ that maps each link to one of its backup paths is a* backup path scheme.

We may drop the subscript when the graph $G$ is clear from the context. When a packet arrives at a node and the next link on its path is some failed link $\ell_1$, the node (i.e., router) immediately reroutes the packet along the backup path of $\ell_1$, given by $BP(\ell_1)$. The packet may encounter a second failed link $\ell_2 \in BP(\ell_1)$. Now assume $\ell_1 \in BP(\ell_2)$. The packet loops between the two links indefinitely as one link lies on the BP of the other. To this end, we need to characterize backup paths that do not induce such infinite forwarding loops under any subset of simultaneous link failures restricted only in cardinality. Before that, we formalize the actual route that a packet takes under the "failure scenario" $L$.

▶ **Definition 2.** *Given any subset of links $L \subset E$, a* detour route *around a link $\ell \in L$, denoted by $R_G(\ell, L)$, is obtained by recursively replacing each link in $BP_G(\ell) \cap L$ with its respective detour route. Precisely,*

$$R_G(\ell, L) = (BP_G(\ell) \setminus L) \cup \bigcup_{\ell' \in BP_G(\ell) \cap L} R_G(\ell', L). \quad (1)$$

*Moreover, 1) $BP_G$ is* resilient *under the* failure scenario *$L$ if and only if $\forall \ell \in L$, the detour $R_G(\ell, L)$ exists, i.e., the recursion terminates, and*
*2) $BP_G$ is* $f$-resilient *if and only if it is resilient under every $L \subset E$ s.t. $|L| \leq f$.*

In words, when a packet's next hop is across the failed link $\ell \in L$, it gets rerouted along the route $R_G(\ell, L)$ which ends at the other endpoint of $\ell$ hence evading all failed links. A BP scheme is $f$-resilient if for every subset of up to $f$ failed links, replacing each failed link with its backup path produces a route that excludes failed links. The replacement process from a packet's perspective occurs recursively as in (1). A packet ends up in a loop permanently when it encounters a failed link for which the detour (1) does not exist. Then, the scheme is $f$-resilient if a packet that encounters a failed link reaches the other endpoint of the link by traversing the BP of that link and the BP of any consequent failed link that it encounters along the way.

Definition 2 implies that we cannot have a resiliency higher than graph connectivity, since $L$ may simply consist of all links incident to one node which makes a detour impossible.

▶ **Definition 3.** *An $f$-resilient backup path scheme $BP_G$ is* maximally resilient *if and only if there is no $(f+1)$-resilient scheme.*

Next, we introduce the notion of "dependency" on which we establish some key definitions used widely in the analysis of resiliency in our proofs.

▶ **Definition 4.** *We say there is a* dependency *relation $\ell \to \ell'$ if and only if the link $\ell$ includes the link $\ell'$ on its backup path, i.e., $\ell' \in BP_G(\ell)$. We represent all dependency relations as a directed* dependency graph *$\mathcal{D}(BP_G)$ with vertices $\{v_\ell \mid \ell \in G\}$ and arcs $\{(v_{\ell_1}, v_{\ell_2}) \mid \ell_1 \to \ell_2\}$). $BP_G$* induces *the dependency graph $\mathcal{D}(BP_G)$.*

We denote a dependency arc $(v_{\ell_1}, v_{\ell_2})$ by $(\ell_1, \ell_2)$ for simplicity. Any backup path scheme $BP_G$ induces cycles in $\mathcal{D}(BP_G)$, as otherwise there is a link without any BP assigned to it. We refer to one such cycle as *cycle of dependencies* or CoD for short. Similarly, we define *path of dependency* or PoD for short.

Observe that a CoD captures a failure scenario that leads to a permanent loop. Rewording Definition 2, $BP_G$ is $f$-resilient if and only if every CoD is longer than $f$, i.e., it consists of at least $f+1$ dependency arcs. Hence, CoDs with the shortest length determine the resiliency and we refer to them as *min-CoD*s.

Next, we introduce some additional notations and definitions based on Definition 4. Let $CoD(v)$ denote the CoD over links incident to $v \in V$. We consider maximally resilient schemes for special regular graphs which implies $CoD(v)$ is unique. Note that non-incident links may induce (min-)CoDs as well. We focus on special regular graphs and resiliency thresholds that are maximal for the connectivity (or the degree) of the those graphs. Then, a min-CoD cannot be shorter that the degree of the respective regular graph, which implies $CoD(v)$ is unique for every node $v$.

In Section 3, we present a backup path scheme for certain $k$-dimensional *product graphs*, by generalizing the solution presented in [17] on binary hypercubes (*BHC*). A $k$-dimensional BHC is the Cartesian product of any set of BHCs where dimensions add up to $k$. A product graph $\mathcal{G}$ is the Cartesian product of *base graphs* in $\{g^1, \ldots, g^k\}$. That is, $\mathcal{G} = \prod_{d \in [k]} g^d$ where $\prod$ denotes the Cartesian product and each $g^d$ is the base graph *in dimension $d$*. Let $n_d := |V[g^d]|, d \in [k]$ denote the order of $g^d$. Nodes in a product graph are represented as $k$-tuples $(a_k, \ldots, a_1)$ where $\forall d \in [k] : 0 \le a_d < n_d$. Likewise, we assume labels $(a_k, \ldots, a_{d-1}, *, a_{d+1}, \ldots, a_1)$ for links where their endpoint nodes differ in their *dth digit* (i.e., $d$th component) which is represented by the '*'.

## 3 Resiliency for Cartesian Product

We now introduce an algorithm to compute a maximally resilient scheme for special product graphs. More specifically, the algorithm takes the scheme of each base graph and combines them in a way that yields a scheme for the Cartesian product of those base graphs. However, it requires each individual scheme to possess some structural properties. We begin with the characterization of these properties.

We can *break* a CoD open into a PoD by removing one of its arcs, which is achieved by removing the head link of an arc from the BP of its tail link.

▶ **Definition 5.** *An $r$-resilient backup path scheme $BP_G$ is* well-structured *if and only if there is a set of* boundary links *$L^*$ that for every node $v$ contains a unique link incident to $v$, satisfying the following conditions.*

1. *There is a unique CoD $\mathcal{C}^*$ that consists only of links in $L^*$.*
2. *The following procedure breaks all CoDs.*
   a. *For every link $\ell \notin L^*$ s.t. $BP_G(\ell) \cap L^* \neq \emptyset$;*
      i. *There are exactly two nodes $x_1$ and $x_2$ on $BP(\ell)$, s.t. $L^*(x_1), L^*(x_2) \in BP(\ell)$.*
      ii. *Remove every link of $BP(\ell)$ between $x_1$ and $x_2$, i.e. the subpath $BP(\ell)[x_1, x_2]$.*
   b. *To break $\mathcal{C}^*$, pick one arc $(\ell', \ell^*) \in \mathcal{C}^*$ arbitrarily and remove $\ell^*$ from $BP_G(\ell')$.*
3. *At least $r$ arcs are left in every CoD (not removed at 2(a)ii).*

Intuitively, these conditions mandate a choice of $L^*$ that for every CoD, the packet that realizes the CoD traverses a boundary link. Removing the arc headed at such link breaks the CoD open into a PoD. We refer to such arc as a *feedback arc.* Later, we close the PoD into a new CoD that is induced by the scheme of a product graph for which $G$ is a “base graph”.

Concretely, Definition 5 constrains the set $L^*$ in a way that for every CoD one of the following two cases must apply. Case 1. The CoD may contain an arc headed to a link in $L^*$. Then removing the head link from the BP of the tail link is sufficient to break the CoD. Case 2. The CoD may not contain any link in $L^*$ as the tail or head of some arc, but it contains an arc $(\ell_1, \ell_2), \ell_2 \notin L^*$ that the packet departing from either endpoints of $\ell_1$, traversing $BP_G(\ell_1)$, has to traverse some link in $L^*$ before reaching $\ell_2$. The procedure (at line 5.2(a)ii), removes not only links of $L^*$ from the BP but also the link $\ell_2$, since it lies between $x_1$ and $x_2$. Note that Case 1 applies also to the unique CoD $\mathcal{C}^*$ which is handled separately at 5.2b.

Next, we establish a lemma that constructs a walk on all nodes of $G$, using a given BP scheme and the corresponding set of boundary links.

▶ **Lemma 6.** *Assume a well-structured scheme $BP_G$ and a set of links $L^*$ satisfying Definition 5 are given. There exists a closed walk $W$ on all nodes of $G$ that 1) visits each node $v \in G$ immediately before traversing the link $L^*(v)$, and 2) links in $L^*$ are traversed in the same order they are traversed by $\mathcal{C}^*$.*

**Proof.** The following procedure marks every node in $G$ with `FINISHED` as soon as a visit to $v$ is followed by walking the link $L^*(v)$.

1. $W = \emptyset$.
2. Let $w_0 := v$. Initialize with the last traversed boundary link $\ell^* = L^*(w_0)$. Let $\{w_0, w_1\} := \ell^*$, then initialize the walk $W = [w_0, w_1]$.
3. Repeat:
   a. Assume $W = [w_0, w_1, \ldots, w_t]$ is the current walk, $L^*(w_t) = \{w_t, u\}$ and let $\ell'_{w_t} := \{w_t, u'\} \in BP_G(\ell^*), u' \neq w_{t-1}$.

**b.** If $w_{t-1} = u \wedge w_t \neq w_{t-2}$ then $w_{t+1} = u$.
**c.** Else, $w_{t+1} = u'$.
**d.** If $w_{t+1} = u$ then $\ell^* = \ell_{w_t}$ and mark $w_t$ with `FINISHED`.
**e.** If $w_t = w_0 \wedge \{w_0, w_1\} \in BP_G(\ell^*)$ then Break.

The walk $W$ begins with the link $L^*(w_0)$. Then it proceeds to the next link on the backup path of the last traversed link $\ell^* \in L^*$ at Line 3c (initially $\ell^* = \ell_{w_0}$), or it traverses the recently walked link $\{w_{t-1}, w_t\}$ in the opposite direction at Line 3b (i.e., from $w_t$ to $w_{t-1}$). By assumption, any $\ell \in L^*$ is on the backup path of some $\ell' \in L^*$ and $(\ell', \ell) \in \mathcal{C}^*_{BP_G}$. Therefore, the loop at Line 3 reaches an iteration where the last traversed $\ell^* \in L^*$ includes $L^*(w_0)$ on its backup path, which breaks the loop at Line 3e. The last visited node must be $w_0$ implying $W$ is a closed walk. Whenever $W$ reaches a node $w_t$ and $L^*(w_t)$ is on the backup path of the last traversed $\ell^* \in L^*$, then it next traverses $L^*(w_t)$ for the first time at Line 3c in one direction, or for the second time at Line 3b in the reverse direction. In either case, $L^*(w_t)$ is walked immediately after a (`FINISHED`) visit to $w_t$. At the end, both endpoints of every link in $L^*$ are marked `FINISHED` and since $\bigcup_{\ell \in L^*} \ell = V[G]$, all nodes are marked `FINISHED`. ◀

We will use the walk in the construction of the scheme for a multi-dimensional graph where $G$ is the base graph in some dimension. The walk is used to guide backup paths of links in other dimensions when they need to traverse the dimension of $G$.

## 3.1 The Construction

For every base graph $g^d$, we assign node labels $0, \dots, n_d - 1$ such that nodes are ordered as they are `FINISHED` in Lemma 6. I.e., the first node `FINISHED` gets 0, the second one gets 1 and so on. Assume, for each $g^d \in \mathcal{G}$, a well-structured, $r_d$-resilient backup path scheme $BP_{g^d}$ together with a boundary set $L^*_{BP_{g^d}} \subseteq E[g^d]$ is given. Let us fix a circular order over base graphs, e.g., $g^1, \dots, g^d$. A node $v := (a_1, \dots, a_k) \in \mathcal{G}$ corresponds to the $a_d$th node in the $d$th base graph $g^d, d \in [k]$.

Let $inc_d(1, \dots, a_k)$ denote the (successor) function that takes a node in $\mathcal{G}$, increments the $d$th digit, applies any carry flag rightward rotating left, and discards any carry back to the $d$th digit. Observe that for a fixed $d \in [k]$, the function $inc_{d+1}$ defines a total order over all instances of $g^d$. We denote the $i$th instance by $g^d_i$. We write $g^d_i$ (instead of $g^d$) only when we refer to a specific *$g^d$-instance.* similarly, $\ell \in \mathcal{G}$ is a *$g^d$-link* if it is an instance of a link in $g^d$.

Let $v^d_i(x)$ denote the mapping $V[g^d] \mapsto V[g^d_i] \subseteq V[\mathcal{G}]$, where $v^d_i(x)$ is the $i$th instance of the node $x \in g^d$. Then, $v^d_{i+1}(x) = inc_{d+1}(v^d_i(x))$. Similarly, for a path (i.e., subset) of nodes $P$, we have $v^d_i(P) = \cup_{v \in P} v^d_i(v)$. We use $v^d_i$ whenever the node $x$ is not relevant to the context. Next, we compute a path $P^*(v^d_i) = \{v^d_i, \dots, v^d_{i+1}\}$, that connects $v^d_i$ and $v^d_{i+1}$ in $\mathcal{G}$ through the sequence of base graphs $g^{d+1}, g^{d+2}, \dots$. The intermediate nodes are determined by digits incremented during the operation $inc_{d+1}(v^d_i)$. Algorithm 1 depicts this procedure.

We initialize the scheme for every $g^d$-instance with a copy of $BP_{g^d}$, i.e., $\forall i : BP_{g^d_i} = BP_{g^d}$. Then, we integrate $BP_{g^d_i}$ into $BP_{\mathcal{G}}$ by extending backup paths of links that contain or traverse a boundary link, i.e., links that are tail of some feedback arc. Consider any feedback arc $(\ell, \ell') \in \mathcal{A}_{BP_{g^d_i}}(\mathcal{C})$. Since $\ell' \in BP_{g^d_i}(\ell)$, we can break $\mathcal{C}$ by extending $BP_{g^d_i}(\ell)$ into a backup path that does not traverse $\ell'$ (i.e., detours $\ell'$). We detour $\ell' = \{x_1, x_2\}$ via a pair of walks through $g^{d+1}_i, g^{d+1}_i, \dots$ that reaches the next instance of $g^d_i$, i.e., the instance given by $inc_{d+1}$. That is, the paths $P^*(v^d_i(x_1))$ and $P^*(v^d_i(x_2))$. By reconnecting $v^d_{i+1}(x_1)$ and $v^d_{i+1}(x_2)$ through $g^d_{i+1}$, we finish the construction of the extended backup path. In Algorithm 2, we use notations and constructions defined so far to describe the integration of all $BP_{g^d_i}$'s into one scheme $BP_{\mathcal{G}}$.

**Algorithm 1** Construction of $P^*(v_i^d), v_i^d = (a_0, \ldots, a_{k-1})$.

```
1: function $P^*(v_i^d)$
2:     $P = \{v_i^d\}$, $v = v_i^d$ , $d' = d + 1$, $carry = 1$                ▷ initialize
3:     while $carry > 0 \land d' \neq d$ do                              ▷ emulating $inc_{d+1}(v)$
4:         if $a_{d'} < n_{d'} - 1$ then
5:             $v[d'] = v[d'] + 1$, $carry = 0$                          ▷ increment the $d'$th digit
6:         else
7:             $v[d'] = 0$, $carry = 1$
8:             $d' = (d' + 1) \pmod k$                        ▷ move to the next digit, rotating left
9:         $P = P \cup \{v\}$                                                ▷ append $v$ to $P$
       return $P$
```

**Algorithm 2** Construction of $BP_{\mathcal{G}}$.

```
1: Initialize $BP_{\mathcal{G}} = \emptyset$
2: for every $d \in [k]$ and all instances $g_i^d$ do
3:     $BP_{g_i^d} = \text{ForBaseGraph}(d, i)$
4: $BP_{\mathcal{G}} = \bigcup_{d \in [k], i} BP_{g_i^d}$

5: function ForBaseGraph$(d, i)$
6:     Initialize $BP_{g_i^d} = BP_{g^d}$, relabel all nodes from $x \in g^d$ to $v_i^d[x] \in g_i^d$.
7:     Let $L_i^d := L^*$ of $BP_{g_i^d}$ (Definition 5)
8:     for every $\ell \in g_i^d, \notin L_i^d$ s.t. $BP_{g_i^d}(\ell) \cap L_i^d \neq \emptyset$ do          ▷ Definition 5.2a
9:         Let $x_1$ and $x_2$ be nodes as specified in Definition 5.2(a)i.          ▷ detour points
10:        $S := BP_{g_i^d}(\ell)[x_1, x_2]$                            ▷ the part of BP to be removed
11:        $S^* := inc_{d+1}(S)$                    ▷ the copy of $S$ in the next $g^d$-instance $g_{i+1}^d$
12:        Compute $P^*(x_1)$ and $P^*(x_2)$                                       ▷ Algorithm 1
13:        $P'_\ell := (P_\ell \setminus \{S\}) \cup \{S^*\} \cup P^*(x_1) \cup P^*(x_2)$
14:        $BP_{g_i^d}(\ell) = P'_\ell$
       return $BP_{g_i^d}(\ell)$
```

▶ **Definition 7.** *Let $\ell_1 := \{u, v\} \in g_i^{d'}, \ell_2 := \{u', v'\} \in g_j^{d'}, j \neq i$. We say that the dependency arc $(\ell_1, \ell_2)$* traverses *the base graph $g^d, d \neq d'$ if and only if $\ell_1$ and $\ell_2$ differ in their dth digits. Moreover, if the dth digit from $\ell_1$ to $\ell_2$ increases by 1 then we say the arc traverses $g^d$ in* uphill direction*. Otherwise the dth digits resets to zero and the arc traverses $g^d$ in* downhill direction*.*

Restating Definition 7, two packets departing from the two endpoints of $\ell_1$ traveling on the backup path of $\ell_1$ together traverse a pair of links in two $g^d$-instances (symmetrically), before reaching $\ell_2 \in BP_{g_i^d}(\ell_1)$. The pair of $g^d$-links are distinct instances of the same link in $g^d$ and they are traversed in the same direction due to the symmetric construction of the pair of paths at Line 2.12. That is, either towards their higher endpoint (i.e. larger $d$th digit), which we refer to as the uphill direction, or the opposite (downhill) direction.

▶ **Definition 8.** *We say an arc $(\ell_1, \ell_2), \ell_1 \in g_i^{d'} \ell_2 \in g_j^d$* crosses *$g^d$ if the two links belong to different base graphs, i.e. $d' \neq d$, or both are in the same $g^d$-instance, i.e. $d = d'$ and $i = j$.*

Similarly, we say a PoD (CoD) traverses or crosses $g^d$ if it includes an arc that, respectively, traverses or crosses $g^d$. Therefore, if a PoD does not cross $g^d$-link then it means it does not contain any $g^d$-link as the head of an arc. We emphasize that by construction, an arc either crosses or traverses a base graph $g^d$.

▶ **Definition 9.** *An arc $(\ell_1, \ell_2) \in \mathcal{C}$ is the* contribution *of $g^d$ in one these cases: it crosses $g^d$, it traverses $g^d$ in the uphill direction, or $\ell_2$ is a $g^d$-link and the arc traverses all other dimensions in the downhill direction.*

By Definition 9 every arc is the contribution of a unique base graph.

## 3.2 Analysis of Resiliency

We begin with a series of lemmas that show each base graph contributes its resiliency to the resiliency of $BP_{\mathcal{G}}$.

▶ **Lemma 10.** *Let $P$ be a PoD induced by $BP_{\mathcal{G}}$ that traverses $g^d$ in the uphill direction at least once and it does not cross $g^d$. Then, there exists a PoD $\tilde{P}$ induced by $BP_{g^d}$ that consists of the links in $L^*_{BP_{g^d}}$ that are traversed by $P$ s.t. $|P| \geq |\tilde{P}|$.*

We defer the proof to the appendix due to space constraint.

**Proof.** We have $|C| \geq |\tilde{C}|$ by applying Lemma 10. Then the claim follows because of the assumption that $BP_{g^d}$ is $r_d$-resilient, which directly implies $|\tilde{C}| \geq r_d + 1$. ◀

▶ **Lemma 11.** *Let $P := \{(\ell_{first}, \ell_1), \ldots, (\ell_s, \ell_{last})\}$ be a PoD induced by $BP_{\mathcal{G}}$. Assume $\ell_{first} \in g^d_i$ and $\ell_{last} \in g^d_j$ are the only $g^d$-links on $P$ for some $i$ and $j$. Let $\ell'_{first}, \ell'_{last} \in g^d$ be the corresponding links in $g^d$. Then there exists a PoD $\tilde{P}$ induced by $BP_{g^d}$ that begins with $\ell'_{first}$ and ends at $\ell'_{last}$ s.t. $|P| \geq |\tilde{P}|$.*

**Proof.** By assumption, $P$ begins with an arc tailed at $\ell_{first} \in g^d_i$. Let $(\ell_{first}, \ell')$ be the feedback arc induced by $BP_{g^d_i}$ that is picked at Line 2.9 and then is handled by detouring a boundary link $\ell' \in L^*_{g^d_i}$ via $g^d_{i+1}$ at Lines 2.9 to 2.14. Let $A \subseteq P$ be the set of arcs in $P$ that traverse $g^d$ in the uphill direction. Note the $d$th digit changes only along arcs in $A$ and remains unchanged along arcs $P \setminus A$. We construct a PoD $\tilde{P}$ over a subset of boundary links in $L^*_{g^d}$, as follows. The first arc in $\tilde{P}$ is $(\ell_{first}, \ell')$. With each arc in $A$, the $d$th digit increases by 1 from its tail to its head. Recall that the value of this digit is a node label in $g^d$, and an increment by 1 corresponds to traversing a boundary link of $g^d$. Consider arcs in $A$ sorted in the order they appear in $P$. Let $\ell^* \in L^*_{BP_{g^d}}$ be the boundary link traversed by the first arc in $A$ (possibly, $\ell^* = \ell'$). Let $P' := P \setminus \{(\ell_{first}, \ell_1), (\ell_s, \ell_{last}\}$. By assumption, $P'$ does not cross $g^d$ and therefore it begins at $\ell^*$ and ends at $\ell^{**}$, the boundary link traversed by the last arc in $A$. we consider two cases.

Case i) $\ell_{last}$ is a boundary link, i.e., $\ell_{last} \in L^*_{g^d}$, then we apply Lemma 10 to $P'$ and we obtain a PoD $P'', |P''| \leq |P'|$, over the boundary links traversed by $A$. (1) Due to Line 2.12 and Lemma 6.2, arcs in $A$ traverse boundary links of $BP_{g^d}$ in the same order they appear in $\mathcal{C}^*_{BP_{g^d}}$. (2) The $d$th digit does not change, from the head of the last arc in $A$ until the arc headed at $\ell_s$. Combining (1) and (2) implies that $\ell_{last}$ succeeds $\ell^{**}$ in this ordering and therefore $(\ell^{**}, \ell_{last}) \in \mathcal{C}^*_{BP_{g^d}}$ is an arc induced by $BP_{g^d}$. Thus, $\tilde{P} := \{(\ell_{first}, \ell^*)\} \cup P'' \cup \{(\ell^{**}, \ell_{last})\}$ is a PoD (induced by $BP_{g^d}$) and $|P| = |P'| + 2 \geq |P''| + 2 = |\tilde{P}|$, which satisfies the lemma.

Case ii) $\ell_{last}$ is not a boundary link, i.e., $\ell_{last} \notin L^*_{g^d}$. Let $w_t$ the value of the $d$th digit at $\ell_s$. The walk $W_{BP_{g^d}}$ from Lemma 6 visits the node $w_t \in g^d$ immediately before traversing the incident boundary link $\ell^{**} := L^*_{g^d}(w_t)$ (Line 6.3d). The pair of paths computed at Line 2.12 traverse nodes of $g^d$ (i.e., values of the $d$th digits along the paths) in the same order as they are walked on by $W_{BP_{g^d}}$. This means that $BP_{\mathcal{G}}(\ell_s)$ traverses (some two instances of) $\ell^{**}$ before any other link in $g^d$, in particular, before $\ell_{last}$. Therefore $\ell^{**} \in BP_{\mathcal{G}}(\ell_s)$ and $(\ell_s, \ell^{**})$ is an arc induced by $BP_{\mathcal{G}}$. Then, $P' := P \setminus \{(\ell_s, \ell_{last})\} \cup \{(\ell_s, \ell^{**})\}$ is a PoD as well. By Lemma 6.3, the walk $W_{BP_{g^d}}$, after traversing $\ell^{**}$, walks on $BP_{g^d}(\ell^{**})$ until the next boundary link is reached. Hence, $\ell_{last}$ is on this backup path and $(\ell^{**}, \ell_{last})$ is an arc induced by $BP_{g^d}$. is a PoD induced by $g^d$. Now, similarly to the case (i), we remove the first and the last arcs in $P'$ and obtain a PoD $P''$ that does not cross $g^d$. By applying Lemma 10 to $P''$, we obtain a PoD $P^*$ induced by $g^d$ s.t. $|P^*| \leq |P''|$. Thus, $\tilde{P} := \{(\ell_{first}, \ell^*)\} \cup P^* \cup \{(\ell^{**}, \ell_{last})\}$ is a PoD induced by $g^d$ and $|P| = |P'| = |P''| + 2 \geq |P^*| + 2 = |\tilde{P}|$, which concludes the lemma. ◀

▶ **Theorem 12.** *The backup path scheme $BP_{\mathcal{G}}$ is $(\Delta - 1)$-resilient where $\Delta = \sum_{d \in [k]} (r_d + 1)$.*

**Proof of Theorem 12.** Consider any CoD $\mathcal{C}$ induced by $BP_{\mathcal{G}}$. We shrink $\mathcal{G}$ down to a single instance of $g^d$ denoted by $\tilde{g}^d$. To this end, we map all nodes in $\mathcal{G}$ with equal $d$th digit, to one node $s \in \tilde{g}^d$. As a result, links between nodes with equal $d$th digits merge into a single node, which transforms them into loop links. We remove all arcs having a loop link as an endpoint and denote the remaining arcs by $\mathcal{C}'$. Since $g^d$ is $r_d$-resilient, $g^d$ contributes up to $r_d + 1$ arcs to $\mathcal{C}$; we argue that the contribution is exactly $r_d + 1$ arcs.

If $\mathcal{C}$ consists of $g^d$-links only (i.e., endpoints of every arc in $\mathcal{C}$ have different $d$th digits), then all arcs in $\mathcal{C}$ are preserved (i.e. not removed) after the transformation, which implies $\mathcal{C}'$ is a CoD in $\tilde{g}^d$ and $|\mathcal{C}| \geq |\mathcal{C}'| \geq r_d + 1$. However, some arcs in $\mathcal{C}'$ are projection of arcs in $\mathcal{C}$ that are not the contribution of $g^d$ (Definition 9). They traverse some $g^{d'}, d' \neq d$ in the uphill direction and hence are exclusively the contribution of $g^{d'}$. These are the same arcs eliminated at Line 5.2(a)ii. Definition 5.3 guarantees at least $r_d$ non-eliminated arcs left which implies at least $r_d + 1$ arcs in $\mathcal{C}'$ cross $g^d$ and are its contribution. There must be one arc that traverses all dimensions except $d$ in the downhill direction, which means in total there are at least $r_d + 1$ arcs contributed from $g^d$.

Else, if $\mathcal{C}$ does not contain cross $g^d$-link, then it only traverses $g^d$. Recall that traversing $g^d$ is guided by the closed walk constructed in Lemma 6 and with each (FINISHED) visit to nodes there is an increment, i.e. an uphill traversal. Hence, $g^d$ in this case contributes a number of arcs equal to the number of FINISHED visits, which in turn is the number of its nodes, or $|V[g^d]| \geq r_d + 1$.

Else, $\mathcal{C}$ both traverses and crosses $g^d$. Then there are links with equal $d$th digits at their endpoints which shrink into loop links. We remove all arcs $(\ell', \ell'') \in \mathcal{C}'$ where $\ell'$ or $\ell''$ is a loop link, as well as loop arcs. As a result, parts of $\mathcal{C}'$ along which the $d$th digit does not change, is eliminated and $\mathcal{C}'$ is segmented into separate PoDs. Let $\mathcal{S} \subset \mathcal{C}'$ denote the set of remaining arcs (tails and heads of which in $\tilde{g}^d$). Notice that arcs in $\mathcal{S}$ form disconnected PoDs. Moreover, for each PoD $P \subseteq \mathcal{S}$, the tail of the first arc and the head of the last arc belongs to $\tilde{g}^d$. The remaining arcs (which do not include any $g^d$-link) are in $\overline{\mathcal{S}} := \mathcal{C} \setminus \mathcal{S}$. Due to the segmentation of $\mathcal{C}$, $\overline{\mathcal{S}}$ forms disconnected PoDs, each beginning with an arc tailed at a link in $\tilde{g}^d$ and ends at an arc headed at link in $\tilde{g}^d$. Since these PoDs cross $g^d$ only at their end links, we apply Lemma 11 to each PoD $P' \subseteq \overline{\mathcal{S}}$ and we obtain a PoD $\tilde{P}$ induced by $BP_{g^d}$. Then, by adding each obtained $\tilde{P}$ to $\mathcal{C}$, we reconnect all consecutive PoDs in $\mathcal{S}$ and join them into a CoD $\tilde{\mathcal{C}}$ induced by $BP_{g^d}$, which means $|\tilde{\mathcal{C}}| \geq r_d + 1$. Due to proof of Lemma

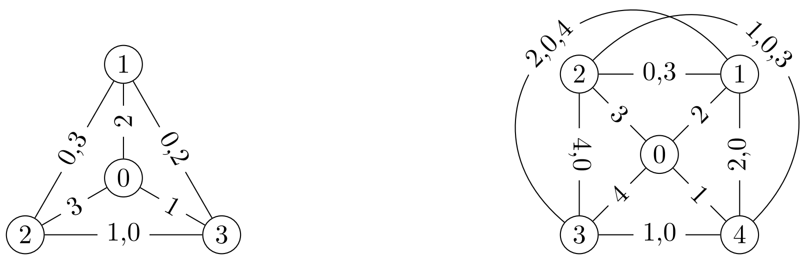

**Figure 1** Maximally resilient schemes for $K_4$ and $K_5$. The numbers on each link are the internal nodes of the link's backup path.

11, every arc in $\tilde{\mathcal{C}}$ is either projected from an arc in $\mathcal{C}$ that has $g^d$-links as endpoints, i.e., crossing $g^d$, or is projected from some arc in $\mathcal{C}$ that traverses $g^d$ in the uphill direction. Thus by definition 9, every arc in $\tilde{\mathcal{C}}$ is the contribution of $g^d$. ◀

## 4 Generalized Hypercubes and Tori

We have described above how to construct a maximally resilient scheme for Cartesian products of given base graphs using their well-structured schemes. In this section, we showcase examples of these base graphs and apply our results to their products. In particular, we will present efficient and robust path restoration schemes for generalized hypercube graphs and tori.

### 4.1 Complete Graphs and Generalized Hypercubes

A complete graph over $n$ nodes is defined as $K_n = (V, E)$ where $V = \{0, \dots, n-1\}$ and the links $E = \{\{i, j\} | i, j \in V, i \neq j\}$. We present a $(n-2)$-resilient scheme for $K_n$ denoted by $BP_{K_n}$, which we later leverage for generalized hypercubes. In the following assume every increment $(+1)$ is performed in modulo $n$ and it skips 0. That is, $i + 1 \equiv i \pmod{n-1} + 1$ We generate all backup paths in two simple cases as described in Algorithm 3.

**Algorithm 3** Construction of $BP_{K_n}$.

```
1: for each link $\ell \in E[K_n]$ do
2:     if $0 \in \ell$ then                                  ▷ i.e. $\ell = \{0, i\}$
3:         $BP_{K_n}(\ell) = [0, i+1, i]$
4:     else                                                  ▷ i.e. $\ell = \{i, j\}, i, j \neq 0$
5:         $BP_{K_n}(\ell) = [i, j+1, 0, i+1, j]$
```

▶ **Theorem 13.** *The backup path scheme $BP_{K_n}$ is $(n-2)$-resilient.*

**Proof.** The dependencies from a link $\{i, j\}$ where $i, j \neq 0$, to other links can be observed in four distinct types: $\{i, j\} \xrightarrow{A} \{i, j+1\}$, $\{0, j\} \xrightarrow{B} \{0, j+1\}$, $\{i, j\} \xrightarrow{C} \{0, j+1\}$ and $\{0, j\} \xrightarrow{D} \{j, j+1\}$. Note that with each type, $i$ and $j$ are interchangeable due to the symmetry of BP produced at Line 3.5. In Figure 1 (right), an exemplary CoD that consists of all the four types can be: $\{1, 2\} \to \{1, 3\} \to \{0, 4\} \to \{0, 1\} \to \{1, 2\}$. Next, we show that any CoD consists of at least $n-1$ arcs, implying $n-2$ resiliency. If $\mathcal{C}$ consists of links all incident to some node $i \neq 0$, then $\mathcal{C} = \{i, j\} \xrightarrow{A} \{i, j+1\} \xrightarrow{A} \{i, j+2\} \dots \{i, n-1\} \xrightarrow{C} \{i, 0\} \xrightarrow{D} \{i, i+1\} \xrightarrow{A} \dots \{i, j\}$. Clearly $\mathcal{C}$ consists of $n-1$ arcs and therefore in the remainder we focus on CoDs over links non-incident to the same node.

Given a CoD $\mathcal{C}$, we construct a sequence of node ids $S = (v_0, v_1, \dots, n-1, \dots, v_0)$ such that for every $0 \le t < |\mathcal{S}|$, it holds $S_{t+1} \le S_t + 1$, and the tail of the $t$-th arc in $\mathcal{C}$ is a link $\{S_t, *\}$. Observe that such sequence implies there are $|S| \ge n-1$ arcs in $\mathcal{C}$. We construct $S$ as follows.

1. All dependencies in $\mathcal{C}$ are of type $A$. Assume the packet $p$ that realizes the CoD is currently at node $i$ and hits the failed link $\{i,j\} \not\ni 0$. Let $\mathcal{S}$ be the sequence of nodes that $p$ visits until it arrives back to $i$. The next failure (by type $A$) is either $\{i+1, j\}$ or $\{i, j+1\}$. Therefore $p$ either is rerouted to the node $i+1$ or it stays at $i$. That is, $p$ visits all nodes contiguously before it arrives back to $i$. After applying type $A$ to either of the outcomes and repeating on each consequent link in a similar way, the packet visits all nodes contiguously.
2. All dependencies in $\mathcal{C}$ are of type $B$. We take the sequence of non-zero endpoints. I.e., $\mathcal{S}[t] = v \in \mathcal{C}_t, v \ne 0$.
3. $\mathcal{C}$ contains multiple arc types. We refer to a path of arcs all in type $X$ as type $X$-PoD. We split $\mathcal{C}$ into maximal dependency paths of types $A$ and $B$, which are concatenated by dependency arcs of type $C$ and $D$. We extract a sub-sequence from each maximal PoDs and patch them into a single sequence $\mathcal{S}$ as follows. Initially, let $\mathcal{S} = \emptyset$ and start with a maximal $A$-PoD $\{i_0, j_0\} \xrightarrow{A}, \dots$ chosen arbitrarily.
   a. Given a $A$-PoD, say $\{i,j\} \xrightarrow{A}, \dots, \xrightarrow{A} \{i', j'\}$, the packet that realizes the PoD visits two sub-sequences depending on whether it starts at $i$ or $j$. Let $S_1$ and $S_2$ be the produced sub-sequences ending with $i'$ and $j'$ respectively. The $A$-PoD is followed by a type $C$ arc, that is $\{i', j'\} \xrightarrow{C} \{i'+1, 0\}$ or $\{i', j'\} \xrightarrow{C} \{0, j'+1\}$. With the first case, pick the sequence $S_1$, otherwise pick $S_2$. Append to $\mathcal{S}$ the chosen sequence and then the incremented node id at the head of the $C$-arc (i.e. $i'+1$ or $j'+1$).
   b. If $\mathcal{C}$ proceeds with a $B$-PoD then append to $\mathcal{S}$ the sequence of non-zero node ids.
   c. After the $C$-arc and possibly a $B$-PoD, there must be a $D$-arc. E.g., $\{0, j''\} \xrightarrow{D} \{j'', j''+1\}$. The $D$-arc is then followed by a $A$-PoD (possibly the first one). If we are back to the first $A$-PoD, i.e., $\{j'', j''+1\} = \{i_0, j_0\}$, then $\mathcal{S}$ is already a circular sequence. Else, we continue the construction by repeating from step (a)

It is easy to see that the current sequence is contiguous after (a), (b) and (d). In particular, after (d), $\mathcal{S}$ ends with $j''$ and any sub-sequence chosen next in (a) begins with $j''$ or $j''+1$. In either case the claim is preserved. ◀

In the following lemmata, we show that this scheme is well-structured. First, we need to determine the boundary links.

▶ **Lemma 14.** *Every CoD induced by the scheme from Theorem 13 includes a link in $B_{K_n} := \{\{1, i\} \mid 0 \le i \le n-1\}$ and the subset of arcs $\{\{i, n-1\} \to \{i, 1\} \mid i \in \{0, 2, 3, \dots, n-2\}\} \cup \{\{1, n-1\} \to \{0, 1\}\}$ are feedback arcs.*

**Proof.** The sequence $\mathcal{S}$ constructed in the Proof 13 contains every non-zero node id regardless of the given CoD. This means that for any node $v \in \{1, \dots, n-1\}$, every CoD includes some link incident to $v$. We pick $v = 1$ w.l.o.g. We identify feedback arcs as those that head to a boundary link which is a unique arc in every CoD except the one induced by $B_{K_n}$. For this case (i.e. $CoD(1)$), we designate $\{1, n-1\} \to \{0, 1\}$ as the feedback arc. ◀

Next, we observe the properties required by Definition 5.

▶ **Lemma 15.** *The scheme $BP_{K_n}$ (Theorem 13) is well-structured.*

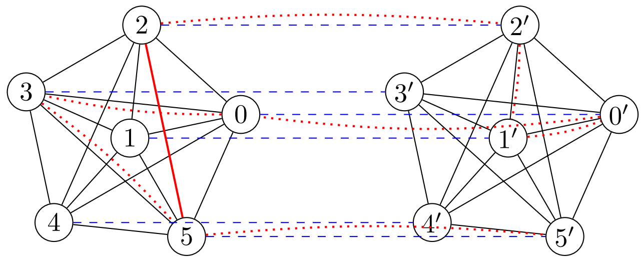

**Figure 2** A $(6,2)$-cube. Each dashed blue line is a $K_2$-instance. They connect the two $K_6$-instances. They admit (respectively) 0- and 4-resilient schemes. The dotted line traces $BP_{\mathcal{G}}(\{2,5\}) = [2,2',1',0',0,3,5]$. On $K_6$, Lemma 6 gives the walk $0,1,0,2,1,3,1,4,1,5,1$ over the boundary links of $K_6$, which are all the links incident to 1. The `FINISHED` order is $0,1,2,3,4,5$. In turn, Algorithm 2 generates backup paths such as $BP_{\mathcal{G}}(\{0,0'\}) = [0,1,1',0']$ and $BP_{\mathcal{G}}(\{1,1'\}) = [1,0,2,2',0',1']$. Hence, $K_2$-instances induce the CoD: $\{0,0'\} \to \{1,1'\} \to \{2,2'\} \to \{3,3'\} \ldots \{0,0'\}$. Observe in example CoDs $\{2,5\} \overset{*}{\to} \{2',1'\} \to \{0',3'\} \to \{0',4'\} \to \{0',5'\} \to \{1,5\} \to \{2,5\}$ and $\{2,5\} \overset{*}{\to} \{2,2'\} \to \{2,1\} \to \{2,0\} \to \{2,3\} \to \{2,4\} \to \{2,5\}$, the starred arcs are counted as the contribution of $K_2$ $(0+1$ arcs), while the rest are the contribution of $K_6$ $(4+1$ arcs).

**Proof.** We observe the conditions in Definition 5 as follows. The set of boundary links in Lemma 14 form a single CoD. Moreover, for every $v \in V[K_n], v \neq 1$, we have $B_{K_n}(v) = \{1, v\}$ and $B_{K_n}(1) = \{1,0\}$, which means every CoD has some link in $L^*_{BP_{K_n}}$ as the endpoint of some arcs. Therefore the procedure 5.2 can break all CoDs. Definition 5.3 can be observed in the proof of Theorem 13. ◀

Next, we formally define the generalized hypercube (GHC) as a special product graph. Given $r_i > 0, i \in [k]$, nodes in $(r_k, \ldots, r_1)$-cube are represented as $k$-tuples $(a_k, \ldots, a_1), \forall i \in [k] : 0 \leq a_i < r_i$ (Figure 2). Therefore there are $\prod_{i\in[k]} r_i$ nodes in a $k$-GHC. Every two nodes $(a_k, \ldots, a_1)$ and $(b_k, \ldots, b_1)$ that differ only at their $i$th digit, say $a_i$ and $b_i$, are connected by an $i$-dim link. The degree of each node is $\Delta = \sum_{i\in[k]}(r_i - 1)$ and the graph is $\Delta$-connected. Observe that $i$-dim links form cliques of $r_i$ nodes. More precisely, there are $\prod_{j\neq d} r_j$ instances of $K_{r_d}$ for every $1 \leq d \leq k$. Thus, Algorithm 2 integrates individual complete graph's schemes into one scheme $BP_{GHC}$. See Figure 2 for an example.

▶ **Corollary 16.** *The backup path scheme $BP_{GHC}$ is $(\Delta - 1)$-resilient.*

**Proof.** By Lemma 15, the scheme from Theorem 13 is well-structured. Due to the fact that a GHC is the Cartesian product of complete graphs, we can apply Theorem 12 which directly implies the claim. ◀

Observe that $\Delta$ failures can disconnect generalized hypercubes, i.e., $(\Delta - 1)$-resiliency is the best we can hope for.

## 4.2 Torus and Grid

Let $\mathcal{B} := \{C_{n_1}, \ldots, C_{n_k}\}$ be a given set of base graphs where each $C_{n_d}, d \in [k]$ is a cycle on $n_d$ nodes. A $k$-dimensional torus $\mathcal{T}$ is the Cartesian Product of $k$ cycles. That is, $\mathcal{T} = \prod_{d\in[k]} C_{n_d}$. Consider a cycle $C_n \in \mathcal{B}$ and its links $\ell_0, \ell_1, \ldots, \ell_{|n|-1}$ as they appear on the cycle. Any cycle is 1-resilient since simply every link includes every other link on its backup path: $\forall \ell \in E[C_n] : BP_{C_n}(\ell) = E[C_n] \setminus \{\ell\}$. Clearly, $BP_{C_n}$ induces $\binom{n}{2}$ CoDs, each on two arcs. The set $B = E[C_n] \setminus \{\ell_0\}$ includes a link from every CoD, therefore it is a (minimal) set of boundary links. We choose the set of feedback arcs to be $F := \{(\ell_i, \ell_j) \mid 0 \leq i < j \leq |n| - 1\}$. Observe that it includes one of the two links in every min-CoD.

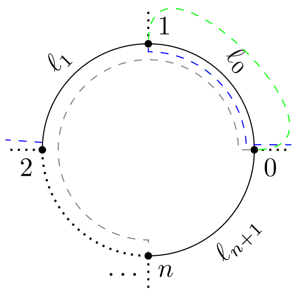

**Figure 3** Solid lines are links of the cycle graph $C_{n+1}$. Dotted lines perpendicular to the cycle represent incident links that belong to a base graph in another dimension. Dashed lines follow backup paths in $BP_{\mathcal{G}}$ where $\mathcal{G}$ is the Cartesian product of $C_{n+1}$ and some other base graphs. The walk constructed in Lemma 6 is $0, 1, 2, \dots, n-1, n, n, n-1, n-2, \dots, 2, 1, 0$. By Lemma 17, in order to break all CoDs, the backup path of $\ell_0$ (dashed green) detours every other link in $C_{n+1}$ using the next dimension base graph. The backup path of $\ell_1$ (dashed blue) takes $\ell_0$, but detours every other link. Similarly, $\ell_2$ (not shown here) takes $\ell_0, \ell_1$ on its backup path and detours $\ell_3$ to $\ell_{n+1}$. This goes on until $\ell_{n+1}$ which uses only links on the $C_{n+1}$.

▶ **Lemma 17.** *The scheme $BP_{C_n}$ is well-structured.*

**Proof.** Every link $\ell_j \in E[C_n]$ has a non-feedback arc to every link $\ell_i \in E[C_n], i < j$ (i.e. $(\ell_j, \ell_i) \notin F$). Any CoD includes at least one arc $(\ell_{j'}, \ell_{i'})$ where $j' > i'$. Hence it includes at least one non-feedback arc, which satisfies Definition 5 trivially. ◀

Now that we know $BP_{C_n}$ is well-structured, we construct $BP_{\mathcal{T}}$ using Algorithm 2 and apply Theorem 12 directly. (See Figure 3 and Figure 4 for an illustration, in the appendix)

▶ **Corollary 18.** *The backup path scheme $BP_{\mathcal{T}}$ is $(2k-1)$-resilient on the $k$-dimensional torus $\mathcal{T}$.*

As a $k$-dimensional torus can be disconnected by $2k$ failures, our scheme is maximally resilient.

Next, we address $k$-dimensional grids via a reduction to torus. By the construction of $BP_{\mathcal{T}}$, only the link $\ell_0 \in C_n$ has a feedback arc to every other link in $C_n$. Let $\ell_0^d \in C_{n_d}$ be the link that corresponds to $\ell_0$ in the base graph $C_{n_d}$, for every $d \in [k]$. Let $\mathcal{B}' = \{P_{n_1}, \dots, P_{n_k}\}$ be the set of paths where each $P_{n_d}$ is obtained by removing $\ell_0^d$ from $C_{n_d} \in \mathcal{B}$ (i.e. $P_{n_d} = C_{n_d} \setminus \ell_0^d$). We construct a scheme for the grid $\mathcal{M} = \prod_{d\in[k]} P_{n_d}$ as follows. Consider the scheme $BP_{\mathcal{T}}$ from Corollary 18. For every $d \in [k]$ and every backup path that uses (an instance of) $\ell_0^d \in C_{n_d}$, we replace $\ell_0^d$ with its backup path. Formally, $\forall d \in [k], \ell \in E[\mathcal{T}], \neq \ell_0^d : BP_{\mathcal{M}}(\ell) = (BP_{\mathcal{T}}(\ell) \setminus \ell_0^d) \cup BP_{\mathcal{T}}(\ell_0^d)$. Since every $\ell \in E[\mathcal{T}], \neq \ell_0^d$ includes $\ell_0^d$ on its backup path, (after short-cutting wherever applies) we have a backup path $BP_{\mathcal{M}}(\ell)$ for every $\ell \in E[\mathcal{M}]$. Each dependency to or from $\ell_0^d, d \in [k]$ is now replaced by a dependency to a link on $BP_{\mathcal{T}}(\ell_0^d)$. Hence, we have replaced PoDs of two arcs with one arc, which in turn reduces the length of some min-CoDs by one. Hence, the $(2k-1)$-resilient scheme is reduced to a $(2k-1-k) = (k-1)$-resilient scheme $BP_{\mathcal{M}}$. As a $k$-dimensional grid can be disconnected by $k$ failures, we obtain a maximally resilient scheme:

▶ **Theorem 19.** *The backup path scheme $BP_{\mathcal{M}}$ is $(k-1)$-resilient on the $k$-dimensional grid $\mathcal{M}$.*

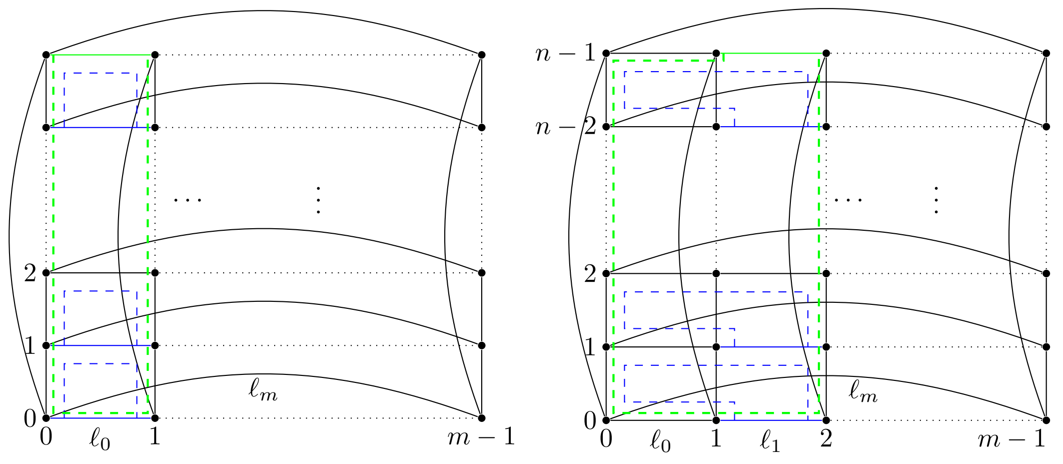

**Figure 4** Each solid line is a link of the 2-dimensional $m \times n$ torus $\mathcal{T}$, which is the Cartesian product of $C_m$ and $C_n$. Horizontal cycles are $C_m$-instances and vertical cycles are $C_n$-instances. Dashed lines depict example backup paths in $BP_{\mathcal{T}}$. In the left picture, backup path of four instances of $\ell_0 \in C_m$ are shown. Notice how all instances of $\ell_0$ use each other sequentially on their backup paths. The backup path of $\ell_0$ in the $n$th instance (in green, thick) has to detour all the other $\ell_0$'s in order to use the $\ell_0$-instance at row 0. This is imposed by the walk on $C_n$ constructed in Lemma 6 (Figure 3). Also notice backup paths of $\ell_1$'s on the right picture. The only difference backup paths of $\ell_0' s$ is that they use the $\ell_0$ in the same instance before proceeding to the next $C_m$-instance. In a similar fashion, each $\ell_2$-instance uses $\ell_0, \ell_1$ in the same $C_m$-instance and so on, up to $\ell_m$ which uses only the links on the same $C_m$-instance.

## 5 Conclusion and Future Work

This paper studied the design of algorithms for local fast failover in the setting that requires guaranteed (policy and function preserving) visits to every waypoint along the original path, under multiple link failures. Our main result is a maximally resilient backup path scheme for the Cartesian product of any set of base graphs, as long as for each base graph a well-structured scheme is provisioned. We showcased applications of this result using complete graphs, cycles, and paths by providing a well-structured scheme for each base graph separately. This allowed us to devise algorithms for important network topologies, such as generalized hypercubes and tori. In general, the result applies to the product of any combination of these base graphs as well.

We see our work as a first step and believe that it opens several promising directions for future research. From a dependability perspective, the main open question is whether $k$-connectivity is always sufficient for $(k-1)$-resiliency w.r.t. backup paths. It might be insightful to understand the logic behind schemes formulated by Definition 5.

## References

1 Saeed Akhoondian Amiri, Klaus-Tycho Foerster, Riko Jacob, and Stefan Schmid. Charting the algorithmic complexity of waypoint routing. *Comput. Commun. Rev.*, 48(1):42–48, 2018.

2 Alia K Atlas and Alex Zinin. Basic specification for IP fast-reroute: loop-free alternates. *IETF RFC 5286*, 2008.

3 Michael Borokhovich and Stefan Schmid. How (not) to shoot in your foot with SDN local fast failover: A load-connectivity tradeoff. In *Proc. OPODIS*, 2013.

**4** B. Carpenter and S. Brim. Middleboxes: Taxonomy and issues. RFC 3234, RFC Editor, February 2002. URL: `http://www.rfc-editor.org/rfc/rfc3234.txt`.

**5** Marco Chiesa, Andrei V. Gurtov, Aleksander Madry, Slobodan Mitrovic, Ilya Nikolaevskiy, Michael Schapira, and Scott Shenker. On the resiliency of randomized routing against multiple edge failures. In *Proc. ICALP*, 2016.

**6** Marco Chiesa, Andrzej Kamisiński, Jacek Rak, Gábor Rétvári, and Stefan Schmid. A Survey of Fast Recovery Mechanisms in the Data Plane. *TechRxiv*, 2020. URL: `https://www.techrxiv.org/articles/preprint/Fast_Recovery_Mechanisms_in_the_Data_Plane/12367508`.

**7** Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrovic, Andrei V. Gurtov, Aleksander Madry, Michael Schapira, and Scott Shenker. On the resiliency of static forwarding tables. *IEEE/ACM Trans. Netw.*, 25(2):1133–1146, 2017.

**8** Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrovic, Aurojit Panda, Andrei V. Gurtov, Aleksander Madry, Michael Schapira, and Scott Shenker. The quest for resilient (static) forwarding tables. In *Proc. IEEE INFOCOM*, 2016.

**9** Hongsik Choi, Suresh Subramaniam, and Hyeong-Ah Choi. On double-link failure recovery in WDM optical networks. In *Proc. IEEE INFOCOM*, 2002.

**10** Theodore Elhourani, Abishek Gopalan, and Srinivasan Ramasubramanian. IP fast rerouting for multi-link failures. *IEEE/ACM Trans. Netw*, 24(5):3014–3025, 2016.

**11** ETSI. Network functions virtualisation. In *White Paper*, 2013.

**12** Joan Feigenbaum, Brighten Godfrey, Aurojit Panda, Michael Schapira, Scott Shenker, and Ankit Singla. Brief announcement: on the resilience of routing tables. In *Proc. ACM PODC*, 2012.

**13** Klaus-Tycho Foerster, Juho Hirvonen, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. On the feasibility of perfect resilience with local fast failover. In *Proc. APOCS*, 2021.

**14** Klaus-Tycho Foerster, Andrzej Kamisinski, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Bonsai: Efficient fast failover routing using small arborescences. In *Proc. IEEE/IFIP DSN*, 2019.

**15** Klaus-Tycho Foerster, Andrzej Kamisinski, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Improved fast rerouting using postprocessing. In *Proc. IEEE SRDS*, 2019.

**16** Klaus-Tycho Foerster, Mahmoud Parham, Marco Chiesa, and Stefan Schmid. TI-MFA: keep calm and reroute segments fast. In *Global Internet Symposium (GI)*, 2018.

**17** Klaus-Tycho Foerster, Mahmoud Parham, Stefan Schmid, and Tao Wen. Local fast segment rerouting on hypercubes. In *Proc. OPODIS*, 2018.

**18** Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Local fast failover routing with low stretch. *Comput. Commun. Rev.*, 48(1):35–41, 2018.

**19** Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Casa: Congestion and stretch aware static fast rerouting. In *Proc. IEEE INFOCOM*, 2019.

**20** Pierre François, Clarence Filsfils, Ahmed Bashandy, and Bruno Decraene. Topology Independent Fast Reroute using Segment Routing. Internet-Draft draft-francois-segment-routing-ti-lfa-00, Internet Engineering Task Force, November 2013. URL: `https://datatracker.ietf.org/doc/html/draft-francois-segment-routing-ti-lfa-00`.

**21** Pierre François, Clarence Filsfils, John Evans, and Olivier Bonaventure. Achieving sub-second IGP convergence in large IP networks. *Comput. Commun. Rev.*, 35(3):35–44, 2005.

**22** Karthik Lakshminarayanan, Matthew Caesar, Murali Rangan, Tom Anderson, Scott Shenker, and Ion Stoica. Achieving convergence-free routing using failure-carrying packets. In *Proc. ACM SIGCOMM*, 2007.

**23** Eunseuk Oh, Hongsik Choi, and Jong-Seok Kim. Double-link failure recovery in WDM optical torus networks. In *Proc. ICOIN*, 2004.

**24** P. Pan, G. Swallow, and A. Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. RFC 4090, RFC Editor, May 2005.

**25** Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. Load-optimal local fast rerouting for resilient networks. In *Proc. IEEE/IFIP DSN*, 2017.

26 Stefan Schmid and Jiri Srba. Polynomial-time what-if analysis for prefix-manipulating mpls networks. In *Proc. IEEE INFOCOM*, 2018.

27 Brent Stephens, Alan L. Cox, and Scott Rixner. Plinko: Building provably resilient forwarding tables. In *Proc. ACM HotNets*, 2013.

28 Brent Stephens, Alan L Cox, and Scott Rixner. Scalable multi-failure fast failover via forwarding table compression. *SOSR. ACM*, 2016.

29 Baohua Yang, Junda Liu, Scott Shenker, Jun Li, and Kai Zheng. Keep forwarding: Towards k-link failure resilient routing. In *Proc. IEEE INFOCOM*, 2014.