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, Link Failure, Routing

Digital Object Identifier 10.4230/LIPIcs...1

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, 8], 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 [18], 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

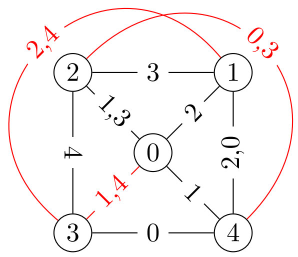

Figure 1 A 2-resilient backup path scheme for K_5 that is not maximally resilient. Numbers on each link are internal nodes of the link's backup path. To each link $\{i, j\}$, the backup path $i, j+1, i+1, j$ is assigned. Assume three links $\{0, 3\}$, $\{2, 4\}$, $\{1, 3\}$ are faulty. Consider a packet initiated at node 0 destined to node 3. Since node 3 is not reachable directly, the packet is forwarded to node 4 to be delivered via the backup path $0, 4, 1, 3$. The packet arrives at node 1 where it hits the failed link $\{1, 3\}$. It is then (recursively) rerouted via the path $1, 4, 2, 3$ on which it hits the failed link $\{4, 2\}$ at node 4. In order to reach node 2, it travels on the path $4, 3, 0, 2$ on which it hits the failed link $\{0, 3\}$ for the second time. Therefore the packet loops through $0, 4, 1, 4, 3, 0$ perpetually. The scheme is not maximally resilient since the graph is 4-connected and a path always exists after any 3 link 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 and waypoint traversal.

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 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 [23]: 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.

1.2 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 review related work in Section 5 and conclude our study in Section 6 with some 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 ℓ. 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 form the context. When a packet arrives at a node and the next link on its path is some failed link ℓ_1, the node (i.e., router) immediately reroutes the packet along the backup path of ℓ_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 sufficiently large subset of simultaneous link failures. Before that, we formalize the actual route that a packet takes under a given 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,*

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

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. $|L| = f$.*

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 ℓ 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 it is not f'-resilient for any $f' > f$.*

Note that maximal resiliency is weaker than "perfect resiliency", where the goal is to reach the destination as long as it is reachable, and a node can decide only the next hop. In our model, a scheme may not be able to provide connectivity even when the destination is reachable under some failure scenario. However, there are graph structures that do not allow perfect resiliency, whereas maximal resiliency is feasible. 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 ℓ includes the link ℓ' 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 $\{(v_{\ell_1}, v_{\ell_2}) \mid \ell_1 \to \ell_2\})$. Hence, BP_G* induce*s the dependency graph $\mathcal{D}(BP_G)$.*

We denote a dependency arc (v_{ℓ_1}, v_{ℓ_2}) by (ℓ_1, ℓ_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. A CoD is trivially a *path of dependencies* (PoD) where the first and the last elements are the same link. 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 a CoD over links incident at $v \in V$. Observe that such CoD always exists. 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 [16] on binary hypercubes (*BHC*). A k-dimensional BHC is in fact 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. 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 $\forall d \in [k] : 0 \leq a_d < n_d$. 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., dth component) which is represented by the '*'.

3 Resiliency Under Cartesian Product

We now introduce a generic 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 into a PoD by removing one of its arcs, which is realizable by removing the head link of an arc from the BP of the tail link of the arc.

▶ **Definition 5.** *An r-resilient backup path scheme BP_G is* well-structured *if and only if for every node v there exists a special link incident at v, denoted by $L^*_{BP_G}(v)$, that satisfies the following conditions.*

1. *Let $:= \bigcup_v L^*_{BP_G}(v)$. There is one CoD $\mathcal{C}^*_{BP_G}$ that consists only of links in $L^*_{BP_G}$.*
2. *The following procedure breaks all CoDs.*
   a. *For every link $\ell \notin L^*_{BP_G}$ s.t. $BP_G(\ell) \cap L^*_{BP_G} \neq \emptyset$, do as follows.*

  i. *Let x_1 and x_2 be the two nodes on $BP(\ell)$, closest to either endpoints of ℓ, s.t. $L^*_{BP_G}(x_1), L^*_{BP_G}(x_2) \in BP(\ell)$*
  ii. *Remove every link of $BP(\ell)$ between x_1 and x_2, i.e. the subpath $BP(\ell)[x_1, x_2]$.*
 b. *Pick one link $\ell^* \in L^*_{BP_G}$ arbitrarily and remove it from the backup path of the (unique) link $\ell \in L^*_{BP_G}$ where $(\ell, \ell^*) \in \mathcal{C}^*_{BP_G}$.*

3. *In every CoD at least r arcs are left, not eliminated by the procedure.*

Intuitively, these conditions mandate a choice of $L^*_{BP_G}$ that for every CoD, the packet that realizes the CoD traverses a link in $L^*_{BP_G}$. These links will be used to break all CoDs open into PoDs, before extending BP_G into a scheme for product graphs for which G is a "base graph". For this reason, we refer to links in $L^*_{BP_G}$ often as *feedback links*, a concise way to indicate they correspond to feedback vertices of the dependency graph that intersect all cycles in that graph that are shorter than $r + 1$ arcs.

Concretely, Definition 5 constrains the set $L^*_{BP_G}$ in a way that for every CoD one of the following two cases must apply. Case 1. The CoD may contain an arc with head in $L^*_{BP_G}$ and removing the head link from the BP of the tail link is sufficient for breaking the CoD (e.g., the case with all $CoD(v)$'s). Case 2. The CoD may not contain any link in $L^*_{BP_G}$ as the tail or head of an arc, but it contains an arc (ℓ_1, ℓ_2) that the packet departing from either endpoints of ℓ_1 (traversing $BP_G(\ell_1)$) has to traverse a link in $L^*_{BP_G}$ before reaching $\ell_2 \notin L^*_{BP_G}$. The procedure 5.2 removes not only the links of $L^*_{BP_G}$ from the BP (at line 5.2(a)ii), but also the link ℓ_2, since it is not anymore reachable from ℓ_2. Note that Case 1 applies to the unique CoD $\mathcal{C}^*_{BP_G}$ which is handled separately at 5.2b.

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

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

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

1. $W_{BP_G} = \emptyset$.
2. Let $w_0 := v$. Initialize the last traversed feedback link $\ell^* = L^*_{g^d}(w_0)$. 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, $L^*_{g^d}(w_t) = \{w_t, u\}$ 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.
4. $W_{BP_G} = W$.

The walk W_{BP_G} begins with the link $L^*_{BP_G}(w_0)$. Then it proceeds to the next link on the backup path of the last traversed link $\ell^* \in L^*_{BP_G}$ 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 $\ell \in L^*_{BP_G}$ is on the backup path of some $\ell' \in L^*_{BP_G}$ and

$(\ell', \ell) \in \mathcal{C}^*_{BP_G}$. Therefore, the loop at Line 3 reaches an iteration where the last traversed $\ell^* \in L^*_{BP_G}$ includes $L^*_{BP_G}(w_0)$ 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 $L^*_{BP_G}(w_t)$ is on the backup path of the last traversed $\ell^* \in L^*_{BP_G}$, then it next traverses $L^*_{BP_G}(w_t)$ 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, $L^*_{BP_G}(w_t)$ is walked immediately after a (`FINISHED`) visit to w_t. At the end, both endpoints of every link in $L^*_{BP_G}$ are marked `FINISHED` and since $\bigcup_{\ell \in L^*_{BP_G}} \ell = V[G]$, 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 one of the dimensions. 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 $0, \ldots, n_d - 1$ 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 feedback vertex set $L^*_{BP_{g^d}} \subseteq E[g^d]$ is given. Let us fix a circular order over base graphs, e.g., $g^1, \ldots, g^d$. A node $v := (a_1, \ldots, a_k) \in \mathcal{G}$ corresponds to the a_dth node in the dth base graph $g^d, d \in [k]$.

Let $inc_d(1, \ldots, a_k)$ denote the (successor) function that takes a node in $\mathcal{G}$, increments the dth digit, applies any carry flag rightward rotating left, and discards any carry back to the dth digit. Observe that for a fixed $d \in [k]$, the function inc_{d+1} defines a total order over all instances of g^d. Hence, we denote the ith 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 ith 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 $P^*(v^d_i) = \{v^d_i, \ldots, v^d_{i+1}\}$, that connects v^d_i and v^d_{i+1} in $\mathcal{G}$ through the sequence of base graphs $g^{d+1}, g^{d+2}, \ldots$. The intermediate nodes are determined by digits that are incremented during the operation $inc_{d+1}(v^d_i)$. Algorithm 1 depicts this procedure.

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

function $P^*(v^d_i)$
  $P = \{v^d_i\}$, $v = v^d_i$, $d' = d + 1$, $carry = 1$ ▷ initialize
  while $carry > 0 \wedge d' \neq d$ **do** ▷ emulating $inc_{d+1}(v)$
    if $a_{d'} < n_{d'} - 1$ **then**
      $v[d'] = v[d'] + 1$, $carry = 0$ ▷ increment the d'th digit
    else
      $v[d'] = 0$, $carry = 1$
      $d' = (d' + 1) \pmod k$ ▷ move to the next digit, rotating left
    $P = P \cup \{v\}$ ▷ append v to P
  return P

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 $BP_{g_i^d}$ into $BP_{\mathcal{G}}$ by extending backup paths of links that contain or traverse a feedback link, i.e., links that are tail of some feedback arc. Consider any feedback arc $(\ell, \ell') \in \mathcal{A}_{BP_{g_i^d}}(\mathcal{C})$. Since $\ell' \in BP_{g_i^d}(\ell)$, we can break $\mathcal{C}$ by extending $BP_{g_i^d}(\ell)$ into a backup path that does not traverse ℓ' (i.e., detours ℓ'). We detour $\ell' = \{x_1, x_2\}$ via a pair of walks through $g_i^{d+1}, g_i^{d+1}, \ldots$ that reaches the next instance of g_i^d, i.e., the instance given by inc_{d+1}. That is, the paths $P^*(v_i^d(x_1))$ and $P^*(v_i^d(x_2))$. By reconnecting $v_{i+1}^d(x_1)$ and $v_{i+1}^d(x_2)$ through g_{i+1}^d, 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 $BP_{g_i^d}$'s into one scheme $BP_{\mathcal{G}}$.

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

```
1:  Initialize BP_G = ∅
2:  for every d ∈ [k] and all instances g_i^d do
3:      BP_{g_i^d} = ForBaseGraph(d, i)
4:  BP_G = ⋃_{d∈[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 ∈ g^d to v_i^d[x] ∈ g_i^d.
7:      Let L_i^d := L*_{BP_{g_i^d}}
8:      for every ℓ ∈ g_i^d, ∉ L_i^d s.t. BP_{g_i^d}(ℓ) ∩ L_i^d ≠ ∅ 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}(ℓ)[x_1, x_2]                          ▷ the part of BP to be removed
11:         S* := inc_{d+1}(S)                  ▷ 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'_ℓ := (P_ℓ \ {S}) ∪ {S*} ∪ P*(x_1) ∪ P*(x_2)
14:         BP_{g_i^d}(ℓ) = P'_ℓ
        return BP_{g_i^d}(ℓ)
```

▶ **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 (ℓ_1, ℓ_2)* traverses *the base graph $g^d, d \neq d'$ if and only if ℓ_1 and ℓ_2 differ in their dth digits. Moreover, if the dth digit from ℓ_1 to ℓ_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 ℓ_1 traveling on the backup path of ℓ_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 dth 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 emphasis 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* ℓ_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* $P := \{(\ell_{first}, \ell_1), \dots, (\ell_s, \ell_{last})\}$ *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* ℓ'_{first} *and ends at* ℓ'_{last} *s.t.* $|P| \geq |\tilde{P}|$*.*

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

▶ **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 dth digit, to one node $s \in \tilde{g}^d$. As a result, endpoints of links $\ell' \in g^{d'}_*, d' \neq d$ merge into one node which transforms ℓ' into a loop link. Let $\mathcal{C}'$ denote the set of arcs in $\mathcal{C}$ after this transformation. Since C is a CoD, the contribution from g^d to C cannot be more than $r_d + 1$ arcs. We argue that it is exactly $r_d + 1$.

If $\mathcal{C}$ includes links only in g^d-instances (i.e., no link in $\mathcal{C}$ has endpoints with equal dth digits), then $\mathcal{C}'$ is already a min-CoD in $\tilde{g}^d$ and $|\mathcal{C}'| \geq r_d + 1$. 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 the contribution of $g^{d'}$. Observe that these arcs are created due to Line 2.12 when a BP in g^d_i is extended into the next g^d-instance via a pair of walks that traverse other dimensions including d', and they correspond to arcs induced by BP_{g^d} that are eliminated by the procedure 5.2. 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 dth digits at their endpoints which shrink into loop links. We remove all arcs $(\ell', \ell'') \in \mathcal{C}'$ where ℓ' or ℓ'' is a

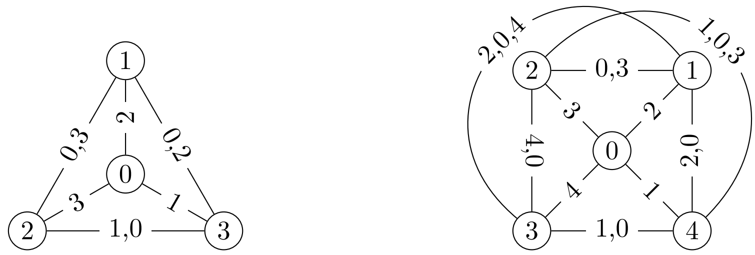

Figure 2 Maximally resilient schemes for K_4 and K_5. The numbers on each link are the internal nodes of the link's backup path.

loop link, as well as loop arcs. As a result, parts of $\mathcal{C}'$ along which the dth 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 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 $V = \{0, \ldots, n-1\}$ 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}

```
for each link ℓ ∈ E[K_n] do
    if 0 ∈ ℓ then                              ▷ i.e. ℓ = {0, i}
        BP_{K_n}(ℓ) = [0, i+1, i]
    else                                       ▷ i.e. ℓ = {i, j}, i, j ≠ 0
        BP_{K_n}(ℓ) = [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: $\{i,j\} \overset{A}{\to} \{i,j+1\}$, $\{0,j\} \overset{B}{\to} \{0,j+1\}$, $\{i,j\} \overset{C}{\to} \{0,j+1\}$ and $\{0,j\} \overset{D}{\to} \{j,j+1\}$. Note that with each type, i and j are interchangeable due to the symmetry in paths of Case 5. In Figure 2 (right), an exemplary cycle of dependencies 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 cycle of dependencies 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 $\mathcal{C} = \{i,j\} \overset{A}{\to} \{i,j+1\} \overset{A}{\to} \{i,j+2\} \ldots \{i,i-1\} \overset{C}{\to} \{i,0\} \overset{D}{\to} \{i,i+1\} \overset{A}{\to} \ldots \{i,j\}$. Obviously $|A[\mathcal{C}]| = n-1$ and therefore we exclude this case from the rest of our proof.

Given a cycle of dependencies $\mathcal{C}$, we construct a non-descending sequence of node ids $\mathcal{S} = (v_0, v_1, \ldots, n-1, \ldots, v_0)$ such that for every $0 \leq t < |\mathcal{S}|$, we have $\mathcal{S}_t \in \mathcal{C}_t$ and $\mathcal{S}_{t+1} \leq \mathcal{S}_t + 1$. That is, $\mathcal{S}$ is *monotonically contiguous*. In words, $\mathcal{S}$ is a circular sequence $1^+, \ldots, (n-1)^+$, and consecutive elements in $\mathcal{S}$ are endpoints of consecutive links in $\mathcal{C}$. Since every arc increments only one endpoint by 1, there must be at least $n-1$ arcs in $\mathcal{C}$. We construct $\mathcal{S}$ 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 during the loop. The next failure is either $\{i+1,j\}$ or $\{i,j+1\}$. Therefore p either is rerouted to the node $i+1$ or it stays at i. The packet eventually leaves the node i, otherwise there is a non-A arc. That is, p visits all nodes in a non-descending order before it arrives back to i. Therefore, $\mathcal{S}$ is a non-descending sequence of all non-zero node ids.
2. All dependencies in $\mathcal{C}$ are of type B. We take the sequence of non-zero endpoints. I.e., $\mathcal{S}[t] = v \in \mathcal{C}_t, v \neq 0$.
3. In this case $\mathcal{C}$ includes multiple dependency 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 $\{i_0, j_0\} \overset{A}{\to}, \ldots$ chosen arbitrarily.

   a. Given a A-PoD, say $\{i,j\} \overset{A}{\to}, \ldots, \overset{A}{\to} \{i',j'\}$, 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 $\{i',j'\} \overset{C}{\to} \{i'+1,0\}$ or $\{i',j'\} \overset{C}{\to} \{0,j'+1\}$. 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., $\{0,j''\} \overset{D}{\to} \{j'',j''+1\}$. 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 monotonically 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 property is preserved. ◀

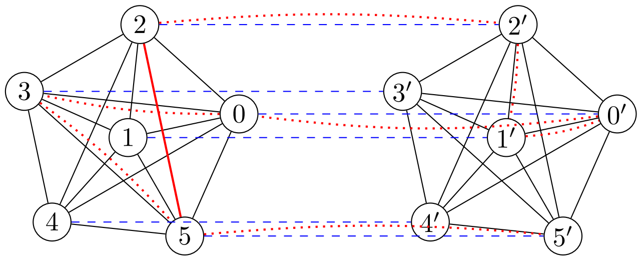

Figure 3 A $(6,2)$-cube. Each dashed blue line is a K_2-instance. They connect the two K_6-instances. They admit (respectively) 0- and 4-resilient schemes. The dotted line traces $BP_{\mathcal{G}}(\{2,5\}) = [2,2',1',0',0,3,5]$. On K_6, Lemma 6 gives the feedback walk $0,1,0,2,1,3,1,4,1,5,1$, if it starts with node 0. 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 $\{2,5\} \xrightarrow{*} \{2',1'\} \to \{0',3'\} \to \{0',4'\} \to \{0',5'\} \to \{1,5\} \to \{2,5\}$ and $\{2,5\} \xrightarrow{*} \{2,2'\} \to \{2,1\} \to \{2,0\} \to \{2,3\} \to \{2,4\} \to \{2,5\}$, 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).

In the following lemmata, we show that this scheme is well-structured. First, we need to determine the feedback 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* $\{\{i,n-1\} \to \{i,1\} \mid i \in \{0,2,3,\ldots,n-2\}\} \cup \{\{1,n-1\} \to \{0,1\}\}$ *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 $v \in \{1,\ldots,n-1\}$, 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 feedback 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.*

Proof. We observe the conditions in Definition 5 as follows. The set of feedback 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 $(a_k,\ldots,a_1), \forall i \in [k] : 0 \le a_i < r_i$ (Figure 3). 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 ith 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 Δ-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 $1 \le d \le k$. Thus, Algorithm 2 integrates individual complete graph's schemes into one scheme BP_{GHC}. See Figure 3 for an example.

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

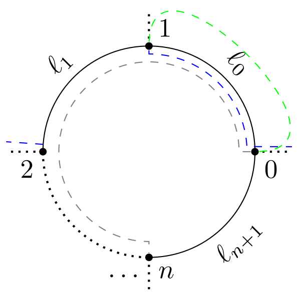

Figure 4 Solid lines are links of the cycle graph C_{n+1}. Dotted lines perpendicular to the cycle represent incident links that belong to a base graph in another dimension. Dashed lines follow backup paths in $BP_{\mathcal{G}}$ where $\mathcal{G}$ is the Cartesian product of C_{n+1} and some other base graphs. The walk constructed in Lemma 6 is $0, 1, 2, \ldots, n-1, n, n-1, n-2, \ldots, 2, 1, 0$. By Lemma 17, in order to break all CoDs, the backup path of ℓ_0 (dashed green) detours every other link in C_{n+1} using the next dimension base graph. The backup path of ℓ_1 (dashed blue) takes ℓ_0, but detours every other link. Similarly, ℓ_2 (not shown here) takes ℓ_0, ℓ_1 on its backup path and detours ℓ_3 to ℓ_{n+1}. This goes on until ℓ_{n+1} which uses only links on the C_{n+1}.

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 Δ 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 feedback links. We choose the set of feedback arcs to be $F := \{(\ell_i, \ell_j) \mid 0 \le i < j \le |n| - 1\}$. Observe that it includes one of the two links in every min-CoD.

▶ **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 4 and Figure 5 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 ℓ_0 in the base graph C_{n_d}, for every $d \in [k]$. Let $\mathcal{B}' = \{P_{n_1}, \ldots, P_{n_k}\}$ be the set of paths where each P_{n_d} is obtained by removing ℓ_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 ℓ_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 ℓ_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 $\mathcal{M}$.*

5 Related Work

Motivation. Resilient routing is a common feature of most modern communications networks, 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 [10]) middleboxes [4], respectively *local protection schemes* in Multiprotocol Label Switching (MPLS) terminology [25], 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 [23], 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 [15], it has been shown that existing solutions for SR fast failover, based on TI-LFA [17], do not work in the presence of two or more failures. However, TI-MFA [15] and non-SR predecessors [20] 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 [7] exist, but they do not provide any formal protection guarantees, except for torus graphs [22]. Beyond a single failure [17] in general and two failures on the torus [22], we are not aware of any approaches that work in the by us considered model, except for a recent work on standard binary hypercubes [16]. However, it is not clear how to extend [16] 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 Feigenbaum et al. [11], Chiesa et al. [5,6] leveraging arc-disjoint network decompositions, also by Elhourani et al. [8], Stephens et al. [27,28], and Schmid et al. [3,12–14,24]. Even though they provide $\Omega(k)$-resilience in k-connected graphs, this guarantee pertains only to reaching the destination, and does not transfer to link protection.

We note that there is furthermore a relatively large set of works that relies on recomputing the routing structure after failures, e.g., [2,9,19,21,26,29,30]. However, such mechanisms do not provide protection *during* convergence and are hence orthogonal to our model.

6 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 et al. Charting the algorithmic complexity of waypoint routing. *CCR*, 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 *OPODIS*, 2013.
4. B. Carpenter and S. Brim. Middleboxes: Taxonomy and issues. RFC 3234, RFC Editor, February 2002. `http://www.rfc-editor.org/rfc/rfc3234.txt`.
5. Marco Chiesa et al. The quest for resilient (static) forwarding tables. In *Proc. IEEE INFOCOM*, 2016.
6. Marco Chiesa et al. On the resiliency of static forwarding tables. *IEEE/ACM Trans. Netw.*, 25(2):1133–1146, 2017.
7. Hongsik Choi, Suresh Subramaniam, and Hyeong-Ah Choi. On double-link failure recovery in WDM optical networks. In *Proc. IEEE INFOCOM*, 2002.
8. Theodore Elhourani, Abishek Gopalan, and Srinivasan Ramasubramanian. Ip fast rerouting for multi-link failures. *IEEE/ACM Trans. Netw*, 24(5):3014–3025, 2016.
9. Gábor Enyedi, Gábor Rétvári, and Tibor Cinkler. A novel loop-free ip fast reroute algorithm. In *EUNICE*, pages 111–119. Springer, 2007.
10. ETSI. Network functions virtualisation. In *White Paper*, 2013.
11. Joan Feigenbaum et al. Ba: On the resilience of routing tables. In *Proc. ACM PODC*, 2012.
12. Klaus-Tycho Foerster et al. Local fast failover routing with low stretch. *ACM SIGCOMM CCR*, 1:35–41, January 2018.
13. Klaus-Tycho Foerster et al. Bonsai: Efficient fast failover routing using small arborescences. In *Proc. IEEE/IFIP DSN*, 2019.
14. Klaus-Tycho Foerster et al. Casa: Congestion and stretch aware static fast rerouting. In *Proc. IEEE INFOCOM*, 2019.
15. Klaus-Tycho Foerster, Mahmoud Parham, Marco Chiesa, and Stefan Schmid. TI-MFA: keep calm and reroute segments fast. In *Global Internet Symposium (GI)*, 2018.
16. Klaus-Tycho Foerster, Mahmoud Parham, Stefan Schmid, and Tao Wen. Local fast segment rerouting on hypercubes. In *Proc. OPODIS*, 2018.
17. 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`.
18. Pierre François et al. Achieving sub-second IGP convergence in large IP networks. *CCR*, 35(3):35–44, 2005.
19. Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM CCR*, volume 41, pages 350–361, 2011.

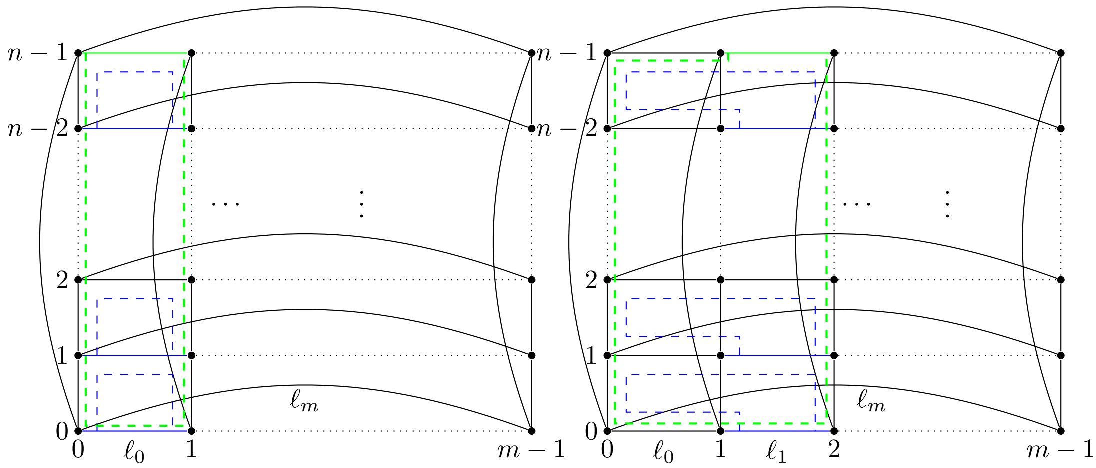

Figure 5 Each solid line is a link of the 2-dimensional $m \times n$ torus $\mathcal{T}$, which is the Cartesian product of C_m and C_n. Horizontal cycles are C_m-instances and vertical cycles are C_n-instances. Dashed lines depict example backup paths in $BP_{\mathcal{T}}$. In the left picture, backup path of four instances of $\ell_0 \in C_m$ are shown. Notice how all instances of ℓ_0 use each other sequentially on their backup paths. The backup path of ℓ_0 in the nth instance (in green, thick) has to detour all the other ℓ_0's in order to use the ℓ_0-instance at row 0. This is imposed by the walk on C_n constructed in Lemma 6 (Figure 4). Also notice backup paths of ℓ_1's on the right picture. The only difference backup paths of $\ell_0's$ is that they use the ℓ_0 in the same instance before proceeding to the next C_m-instance. In a similar fashion, each ℓ_2-instance uses ℓ_0, ℓ_1 in the same C_m-instance and so on, up to ℓ_m which uses only the links on the same C_m-instance.

20 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*, pages 241–252. ACM, 2007.

21 Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Zhi-Li Zhang, and Chen-Nee Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Trans. Netw*, 15(2):359–372, 2007.

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

23 P. Pan, G. Swallow, and A. Atlas. Fast reroute extensions to rsvp-te for lsp tunnels. RFC 4090, RFC Editor, May 2005.

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

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

26 Aman Shaikh, Chris Isett, Albert Greenberg, Matthew Roughan, and Joel Gottlieb. A case study of ospf behavior in a large enterprise network. In *Proc. ACM SIGCOMM Workshop on Internet Measurment*, 2002.

27 Brent Stephens, Alan L. Cox, and Scott Rixner. Plinko: Building provably resilient forwarding tables. In *Proc. 12th 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 Junling Wang and Srihari Nelakuditi. Ip fast reroute with failure inferencing. In *Proc. SIGCOMM INM*, pages 268–273, 2007.

30 Baobao Zhang, Jianping Wu, and Jun Bi. Rpfp: Ip fast reroute with providing complete protection and without using tunnels. In *Proc. IWQoS*, 2013.

Proof of Lemma 11. By assumption, P begins with an arc tailed at $\ell_{first} \in g_i^d$. Let (ℓ_{first}, ℓ') be the feedback arc induced by $BP_{g_i^d}$ that is picked at Line 2.9 and then is handled by detouring a feedback link $\ell' \in L^*_{g_i^d}$ via g_{i+1}^d 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 dth 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 feedback links in $L^*_{g^d}$, as follows. The first arc in $\tilde{P}$ is (ℓ_{first}, ℓ'). With each arc in A, the dth 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 feedback 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 feedback 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 ℓ^* and ends at ℓ^{**}, the feedback link traversed by the last arc in A. we consider two cases.

Case i) ℓ_{last} is a feedback 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 feedback links traversed by A. (1) Due to Line 2.12 and Lemma 6.2, arcs in A traverse feedback links of BP_{g^d} in the same order they appear in $\mathcal{C}^*_{BP_{g^d}}$. (2) The dth digit does not change, from the head of the last arc in A until the arc headed at ℓ_s. Combining (1) and (2) implies that ℓ_{last} succeeds ℓ^{**} 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) ℓ_{last} is not a feedback link, i.e., $\ell_{last} \notin L^*_{g^d}$. Let w_t the value of the dth digit at ℓ_s. The walk $W_{BP_{g^d}}$ from Lemma 6 visits the node $w_t \in g^d$ immediately before traversing the incident feedback link $\ell^{**} := Ł^*_{g^d}(w_t)$ (Line 6.3d). The pair of paths computed at Line 2.12 traverse nodes of g^d (i.e., values of the dth 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) ℓ^{**} before any other link in g^d, in particular, before ℓ_{last}. Therefore $\ell^{**} \in BP_{\mathcal{G}}(\ell_s)$ and (ℓ_s, ℓ^{**}) 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 ℓ^{**}, walks on $BP_{g^d}(\ell^{**})$ until the next feedback link is reached. Hence, ℓ_{last} is on this backup path and (ℓ^{**}, ℓ_{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. ◀