Generalized Graph Products for Network Design and Analysis

Eric Parsonage, Hung X. Nguyen, Rhys Bowden, Simon Knight, Nickolas Falkner, Matthew Roughan
University of Adelaide, Australia
{eric.parsonage,hung.nguyen,rhys.bowden,simon.knight,nickolas.falkner,matthew.roughan}@adelaide.edu.au

***Abstract*—Network design, as it is currently practiced, involves putting devices together to create a network. However, a network is more than the sum of its parts, both in terms of the services it provides, and the potential for bugs. Devices are important, but their combination into a network should follow from expression of high-level policy, not the minutiae of network device configuration. Ideally we want to consider the network as a whole object.**

In this paper we develop *generalized graph products* that allow the mathematical design of a network in terms of small subgraphs that directly express business policy. The result is a flexible algebraic description of networks suitable for manipulation and proof.

The approach is more than just design – it allows for analysis of existing networks providing an understanding of the policies used in their construction, something which can be difficult if the original designers no longer work on that network. We apply the approach to several real world networks to demonstrate how it can provide insight, and improve design.

I. Introduction

Graphs of communications networks have received a great deal of interest over the last few decades (see [1]–[8]), for both scientific interest and practical reasons. Network graphs represent many of the properties of the underlying communications network, such as reliability and performance. They are therefore valuable inputs into many network algorithms, and much effort has gone into the measurement and synthesis of network graphs to use in testing algorithms.

Models of graph formation tell us something about how networks are designed. An engineer may be able to describe their methods for network design, but we wish to learn universal laws of network formation that will continue to apply as technology evolves. Through understanding these laws we can learn how best to improve the underlying technology to fit the network design process, rather than providing engineers with features they do not need, or requiring them to work around technological limitations.

In this paper we investigate how structure is incorporated into data communications networks. We see clear evidence of hierarchy, which may not be suprising to someone familiar with network operations. Hierarchy is used in networking to improve scalability, and is taught in introductory networking courses such as the Cisco CCNA certification.

A hierarchical network could be organized into PoPs (Points of Presence), which are sets of routers, often located in the same physical location. We frequently see repeated patterns in PoP level designs. In large networks the design of each PoP may be almost identical. Another example of hierarchy is the organization of a data center, which has a small number of routers at the top level, and a descending tree of switches down to the servers.

Why is this so? Simply, network designers often apply "cookie cutter" methods to design networks, though that term unnecessarily trivializes the importance of repeated patterns. Repetition makes network operations vastly simpler: the management of two PoPs requires the same skills. Equipment can be bulk purchased, debugging is easier, and adding new PoPs is simpler. Finally, networks based on templated design lead to simple design methodologies. For instance, the inter-PoP level network topology can be optimized relatively simply, as details such as redundancy will be supplied by the provision of pairs of redundant routers in each PoP, with redundant links between them. Design often refers to the graph topology of router interconnections, but templated design can extend to other details, such as physical configuration within racks, connections with external networks, or additional servers such as Domain Name Service or Network Management Systems.

While repeated patterns can improve a network in many ways, there are often exceptions. For instance, some PoPs are too insignificant to warrant provision of a redundant pair of routers, though that might be the norm in the rest of the network. Solving the problems created by exceptions to pattern based design is one of the motivations of this work.

This view of repeated patterns does not fit either of the two main network model paradigms. On one hand we have random networks [1], where any structure is ignored (there are exceptions such as hierarchical random models [9] but although these respect hierarchy they ignore repeated patterns in structure). On the other we have the more convincing Heuristically Optimal Topologies networks [10], [11], which make the case that networks arise from engineers optimizing the network (at least heuristically). Optimization is a very broad notion, and can include arbitrary constraints, but typically in the context of network modeling the constraints do not include any pattern elements. This could be remedied by using ideas presented here.

In this paper, we show examples of real networks that illustrate the principles described above. We also go much further and describe an algebraic method for describing, analyzing, and constructing such networks. The method is based on the notion of *graph products*, which we describe in detail below.

The approach we take is analogous to recent develop-

978-1-4577-1394-1/11/$26.00 ©2011 IEEE

ments in routing protocol design, namely *metarouting* and its extensions [12], [13]. The motivation of both metarouting and our work is to express high-level networking policy in mathematical terms leading to formal proofs.

Our technique allows us to express network design policy algebraically as a set of subgraphs and graph products, with the necessary flexibility to build in exceptions to general design rules. From this we can construct a range of real networks and analyze networks to discover the high-level rules used for their construction.

II. Background and Related Work

Throughout we use $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ to define a graph, where $\mathcal{N}$ is the set of nodes or vertices, and $\mathcal{E}$ is the set of edges or links. We define $N(\mathcal{G}) = \mathcal{N}$, i.e., the nodes of $\mathcal{G}$, and $E(\mathcal{G}) = \mathcal{E}$, i.e., the edges.

Define $Adj(\mathcal{G}) = A_G$, the adjacency matrix of $\mathcal{G}$, i.e., $a_{ij} = 1$ indicates that link $(i, j) \in E(\mathcal{G})$, otherwise $a_{ij} = 0$. In this paper we use undirected graphs for ease of exposition (and because most networks are described by symmetrical links) but there is no difficulty in extending results to directed graphs.

There is a standard notion in graph theory, namely that of a *graph product*. The concept is simple. Take the product of two graphs, and this should produce a third, however, it turns out that there are quite a few possible ways to do this, and none of them is obviously the "right" way.

We start by taking a Cartesian product of the nodes of the two graphs, i.e., we take a new set of nodes $N(\mathcal{G} \times \mathcal{H})$, to be

$$N(\mathcal{G}) \times N(\mathcal{H}) = \{(m, k) | m \in N(\mathcal{G}) \text{ and } k \in N(\mathcal{H})\}.$$

The interesting part of a graph product is usually what we do with the edges. Given two graphs $\mathcal{G}$ and $\mathcal{H}$ a short list of definitions and the notation we use for them follows. In each case, we look for the connectivity between two arbitrary vertices (u, u') and (v, v') both in $N(\mathcal{G}) \times N(\mathcal{H})$.

- **The Cartesian product $\mathcal{G} \square \mathcal{H}$:** [14], [15] requires that any two vertices $(u, u'), (v, v') \in \mathcal{G} \square \mathcal{H}$ are adjacent iff $u = v$ and $(u', v') \in E(\mathcal{H})$ or $u' = v'$ and $(u, v) \in E(\mathcal{G})$.
- **The Tensor product $\mathcal{G} \times \mathcal{H}$:** [15], [16] sometimes called the *direct, relational, categorical, cardinal*, or *Kronecker product* is defined by any two vertices/nodes $(u, u'), (v, v') \in \mathcal{G} \times \mathcal{H}$ being adjacent iff $(u, v) \in E(\mathcal{G})$ and $(u', v') \in E(\mathcal{H})$ and $u \neq v$ and $u' \neq v'$. It is equivalent to taking the Kronecker (or tensor) product of the adjacency matrices of $\mathcal{G}$ and $\mathcal{H}$.
- **The Strong product $\mathcal{G} \boxtimes \mathcal{H}$:** [14], [15] also known as the *normal* or *AND product*, has edges which are the union of the Cartesian and Tensor products.
- **The Lexicographic product $\mathcal{G} \bullet \mathcal{H}$:** [15], [17] also known as *graph composition*, is defined by two vertices (u, u') and (v, v') being adjacent iff $(u, v) \in E(\mathcal{G})$ or $u = v$ and $(u', v') \in E(\mathcal{H})$
- **The Rooted product $\mathcal{G} \circ \mathcal{H}$:** [18] requires the extra definition of a *root node* $h \in N(\mathcal{H})$. Given this the rooted product is defined by requiring that two vertices (u, u') and (v, v') are adjacent iff $u' = h$ and $v' = h$ and $(u, v) \in E(\mathcal{G})$ or $(u', v') \in E(\mathcal{H})$ and $u = v$. It can be intuitively explained to be $|N(\mathcal{G})|$ copies of $\mathcal{H}$, each of whose root is associated with a node of $\mathcal{G}$.
- **The Corona product $\mathcal{G} \odot \mathcal{H}$:** [19] is the only product not defined on a Cartesian product of the nodes of the two graphs, so it does not fit neatly into the general pattern described above, but it has interesting properties so we consider it here as well. The Corona product is created by taking a copy of $\mathcal{G}$ and connecting each node $i \in N(\mathcal{G})$ to every node of a copy $\mathcal{H}_i$ of $\mathcal{H}$. The copies $\mathcal{H}_i$ are not connected to each other, except through $\mathcal{G}$.

The definitions of products may not seem intuitive, but the results often are. Figures 1–5 show simple examples of such networks. Many more simple cases exist, and map well to common examples of network designs: hypergraphs, prisms and *book graphs* can be generated using graph products of simple subgraphs.

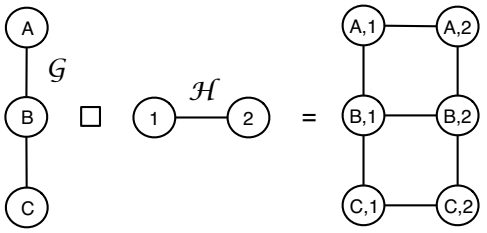

Fig. 1. Cartesian product of a path and a single edge forms a *ladder network*.

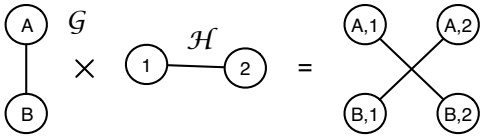

Fig. 2. Tensor product.

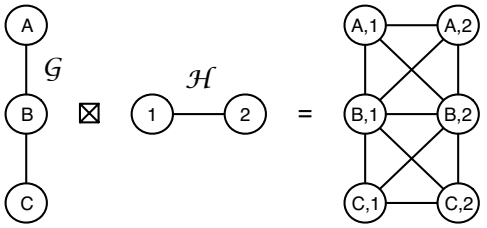

Fig. 3. Strong and Lexicographic products give the same result in this case.

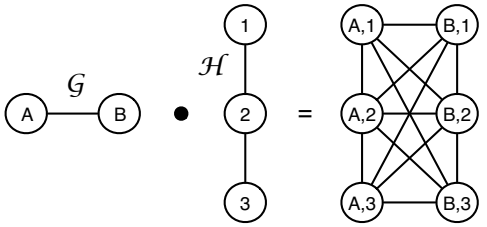

Fig. 4. Lexicographic product showing non-commutativity when compared with Fig. 3.

The advantage of constructing a network from a graph product is the known properties of these products: for instance, the Cartesian, Tensor and Strong products are commutative (up to isomorphisms) whereas the Lexicographic, Rooted and Corona products are not. Other properties follow, for example the Tensor product is connected iff both $\mathcal{G}$ and $\mathcal{H}$

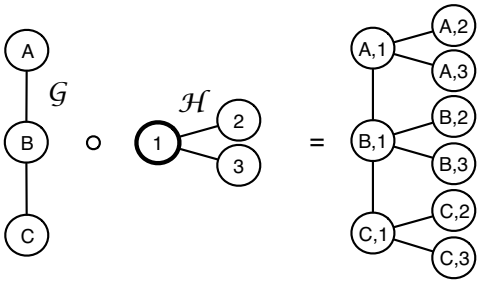

Fig. 5. Rooted product (note the bold node in $\mathcal{H}$ indicates the root).

are connected, and at least one factor is non-bipartite, and we shall discuss such properties relevant to networking in later sections.

For those who have worked with networks, graph products are suggestive of some aspects of network design. For instance, if $\mathcal{G}$ defines a PoP-level network topology, then we can construct a simple router-level topology with redundant pairs of routers, by using a single edge network $\mathcal{H}$ and the Cartesian product. If a more redundant link structure is required, then the Strong product could be used. Likewise, we could build a PoP-level design, with an access distribution tree in each PoP using the rooted product. The interesting thing about the products is the combination of a product and the subgraphs describes a policy for the construction of the network.

Some research has touched on this aspect of design. Most recently [20] has considered the power of the assumptions of structure in networks, though without consideration of graph product based construction. There is also work [21] showing that repeated application of graph products generates graphs that display fractal properties. However, these result in strictly regular, repetitive graphs. As we shall see in the following section we need to be able to loosen this restriction.

The above definitions are standard, but in some ways unsatisfying. It is preferable to be able to express the operations algebraically, in a common format so that we can compare, combine and operate on them. The definitions of graph products can be written in terms of the Kronecker product of adjacency matrices. The Kronecker product of $m \times n$ matrix A and $k \times \ell$ matrix B is defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix},$$

so that $A \otimes B$ has size $km \times \ell n$. Then the graph products can be written as follows (noting that the adjacency matrices A_G and A_H have respective size $n \times n$ and $m \times m$ where $\mathcal{G}$ has n nodes, and $\mathcal{H}$ has m nodes):

$$Adj(G \,\square\, H) = A_G \otimes I_m + I_n \otimes A_H, \quad (1)$$

$$Adj(G \times H) = A_G \otimes A_H, \quad (2)$$

$$Adj(G \boxtimes H) = A_G \otimes I_m + I_n \otimes A_H + A_G \otimes A_H, \quad (3)$$

$$Adj(G \bullet H) = A_G \otimes J_m + I_n \otimes A_H, \quad (4)$$

$$Adj(G \circ H) = A_G \otimes D_{m,k} + I_n \otimes A_H, \quad (5)$$

$$Adj(G \odot H) = A_G \otimes D_{m+1,0} + I_n \otimes \hat{A}_H, \quad (6)$$

where I_n is the $n \times n$ identity matrix, J_n is the $n \times n$ matrix with 1 in every position and $D_{m,k}$ is the $m \times m$ elementary matrix with a 1 on the kth diagonal and zeros elsewhere (k is the row number in A_H that corresponds to the node representing the root of $\mathcal{H}$).

The Corona product requires special augmented matrices, which intuitively arise because the Corona product can equivalently be constructed by adding a *root* node to $\mathcal{H}$, which is adjacent to all the other nodes. We then perform the rooted product with the augmented graph $\hat{\mathcal{H}}$. For simplicity, we number the rooted node zero, and augment the adjacency matrix of $\mathcal{H}$, by adding a row of ones to the first row and column to get the matrix $\hat{A}_H$.

The Kronecker or Tensor product result (2) is widely known [16]. Other relationships are easy to derive, for instance (1), (3) and (4) are given as Exercise 96, without proof in [22], so the proofs are not included here.

These representations are important as they allow us to understand the structure of the adjacency matrix of the product graph. It is this understanding that allows the generalizations found in sections IV, V and VI.

III. Data, High-Level Network Design Principles, and Exceptions

A. Data

A great deal can be learnt about network design from network engineers, either through mailing lists such as the NANOG list, or direct discussion. To justify the technique we present here, we supplement these informal sources with data. The data comes in the form of a series of network maps, published by network operators, and collected in the Internet Topology Zoo [23]. The Zoo contains over 200 networks, including several where structural relationships, such as PoPs, were provided. We use these networks as examples to motivate and illustrate our approach.

B. High-level rules for network design

Networks are not typically built by running a single mathematical optimization routine. The mathematical approach to network management – specifying a cost function and constraints, and then performing a minimization – is optimal, but only if the cost function and constraints, and the input traffic matrix are correct. In practice, the costs are approximate and based on capital costs, not operations costs which are hard to approximate; it is hard to incorporate all of the many details of the real constraints; and traffic matrices are at best predictions (and sometimes outright guesses). The result is that formal optimization does not have the power to justify the complexity it introduces.

Instead, networks are design heuristically. Network operators start with some broad set of objectives, and rules of thumb, and build the network from there, perhaps using mathematical optimization for some relatively simple components of the network, such as weight assignment.

So the question is, "What are the high-level rules for networks design or at the least, what are the ones that

are justifiable?" The question is important, because we are concerned with describing a network as an object, not simply a collection of connected devices. We discussed high-level design rules with network operators and used the Internet Topology Zoo's collection of network maps to elucidate a number of important network properties and design features.

The four main high-level features in the data we examined were:

1) Connectivity: It may seem obvious that the basic requirement for a network is to be connected, but it should still be formally stated in the high-level network description because (i) the network specification should be complete and checkable; (ii) there are multiple types of connectivity – internal connectivity, Internet connectivity, and connectivity at the physical, IP or application layer; and (iii) in the Topology Zoo we have observed several networks which are not connected. That problem may seem strange, but it is possible to design a network that is unachievable through cost, geographic or technological constraints.

2) Redundancy: Most networks are designed to have some type of redundancy. This was a key requirement of the early ARPANET design [24], and has continued to this day. However, there are multiple types of possible redundancy: e.g. edge vs node; multiple layers at which redundancy can appear (e.g., SONET vs IP); multiple degrees of redundancy (say the number of links that can fail before the network partitions); consideration of the amount of capacity needed to avoid overloads under failure scenarios; and consideration of *Shared-Risk Link Groups* (SRLGs) [25] (say fibers that share a conduit). Different network operators have different redundancy requirements, and different parts of a single network may have different requirements.

3) Simplicity: A simple network is not only easy to design and build. It is easier and cheaper to manage [26]. Simplicity is hard to define, so we have a number of sub-headings related to simplicity:

- *Hierarchy* in networks is often associated with scalability, but when used well it can make a network much easier to understand. Breaking a network into regions, and/or PoPs (Points of Presence) allows it to be more easily described, visualized and managed. For instance, locations of devices like route-reflectors and DNS servers, and logical structures such as OSPF areas are often based on the regional or PoP structure of a network.
- *Repeated structures* can be used in networking to simplify network maps, improve re-usability of equipment, and reduce training requirements. Examples are the design of a PoP (up to and including rack placement and wiring diagrams in some cases), the way multiple redundant routers connect PoPs, and the configuration of interconnects.
- *Congruence* is the property that there is a natural mapping between different, but associated network elements or services.

Each of the three features can be seen as a form of encapsulation: a standard software engineering concept where one set of related data and/or procedures is grouped together and given a common interface. As in software, encapsulation aids in re-usability and maintainability by enabling multiple parties to work independently on well defined components of, in this case, a network. However, here these rules are far more structural (in a sense we will describe in the following section).

4) Correctness: A network should be correct: it should function as specified. However, the specification of a network must be both sufficiently detailed to check that it is correct, and it should be possible to verify correctness. The approach presented in this paper allows network design to be done in a way as that makes the specification of the network more precise, and correctness tests easier.

This is obviously a partial list of important features that network operators use in design, but they are at the core of many network designs. We shall show in the following sections how they can be designed into a network through simple application of graph products. However, as common as these features are, there are also exceptions.

Exceptions are often deliberate, and so we must also be able to include these into our approach. Some exceptions may be accidental, and one of the other facilities granted by our method is the ability to analyze a network and discover exceptions, which allows engineers to consider these, and make a decision about whether the exception was warranted. In the next section we provide an example of a network to show how exceptions appear, and what types of exceptions we need to be able to model.

C. Exceptions

For the sake of illustration, in this section we shall concentrate on one example – AARNET – which is simple enough to be easily illustrated, but which also exhibits all of the most common exceptions to a simple graph product formulation of a network. AARNET is the Australian Academic Research Network (see Figure 6). We have discussed their design process with them, and they also publish detailed network diagrams and design information [27]. Australia's population is concentrated on the coast, and AARNET's domestic network is based on a line graph starting in Perth and working its way around the coast to northern Queensland. It has several spurs – to Hobart, Alice Springs and Darwin.

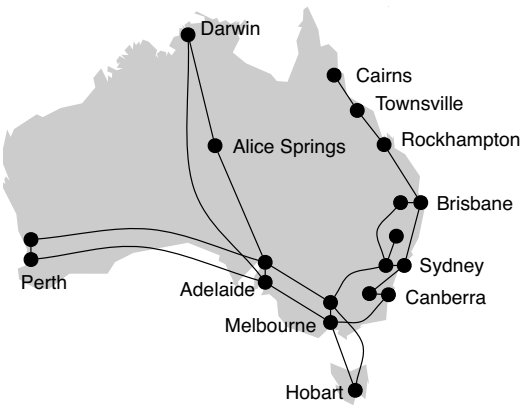

Fig. 6. AARNET router level network 2009.

The network has pairs of routers in each of its major PoPs: Perth, Adelaide, Melbourne, Sydney and Canberra. We can (almost) construct this part of the network with a Cartesian product of two line graphs, producing a ladder graph.

There are several exceptions. There are nodes that have no redundancy: e.g. the links to Northern Queensland. There are also nodes that have link, but not node redundancy, e.g. Alice Springs, Darwin and Hobart each have 2-connectivity to the main network, but only a single router. Finally, Canberra has both node and link redundancy, but not shared-node group level redundancy (discussed later).

The design of the network was deliberate – it is like this for a reason. Redundancy is important, but it comes at a large cost. The distances in Australia are large, and outside of the major cities the population is quite small. Hence, there is node and link redundancy for the major cities, but a reduced level of redundancy for minor PoPs.

The problem for the application of graph products to generation and design of networks is to allow for such exceptions, which are well motivated, while providing for simple structure on the main part of the network. The typical exceptions we have seen across the Topology Zoo are the need to allow for different numbers of (redundant) routers in a PoP, and for different degrees of connectivity between PoPs.

These requirements motivate the following generalizations of the graph products previously described.

IV. First Generalization

It is helpful in what follows to ascribe a role to the subgraphs $\mathcal{G}$ and $\mathcal{H}$ whose product we take, and so for illustrative purposes we consider the problem of PoP based hierarchical network. We shall take $\mathcal{G}$ to specify the inter-PoP network connections (just which PoPs are connected, without reference to which routers provide the actual connections) and $\mathcal{H}$ will describe the interior structure of the PoPs, i.e., the way the routers within a PoP are connected. The graph product will define the router-level connectivity of the network. For instance see Figure 1 which shows an example based on a simple Cartesian product, where $\mathcal{G}$ represents the inter-PoP graph and the $\mathcal{H}$ represents intra-PoP connectivity which is the same for all the PoPs.

The challenges in forming a generalized graph product are

1) PoPs must be able to have a variable number of nodes.
2) We must be able to allow for different graph product operations on each inter-PoP link to allow for different levels of redundancy.
3) Different nodes should be able to take on different roles (e.g. backbone vs access router) in the PoP.
4) In degenerate cases, for instance where all PoPs are identical, the generalized product should collapse down to the appropriate specific product.
5) In the same way that the standard Strong graph product is the disjoint union of the standard Cartesian and Kronecker graph products then the generalised Strong product should be the union of the generalised Cartesian and Kronecker graph products.
6) We desire the ability to carry labels (for instance, distance, link weight, or capacity) onto the new graph (with possible modifications).

In this first generalization, we allow each PoP to have a different internal structure, e.g., for the number of routers in a PoP to vary.

Examination of the relationships (1)-(5) reveals that all but (2) are in the form of two parts:

1) A term of the form $A_G \otimes X$ where X is determined by the type of graph product.
2) A term of the form $Y \otimes A_H$ where Y is determined by the type of graph product.

In most cases the matrices X and Y are obvious, for instance in all cases but (2) Y is just the identity. In the case of (2) we could write the product either by taking X or Y equal to the zero matrix.

Once the products are written in this general form, other new products become obvious, for instance

$$Adj(G \star H) = A_G \otimes A_H + I_n \otimes A_H,$$

which results in a graph with connectivity somewhere between the tensor product and strong product. This has some interesting properties, for instance shortest paths through it have a lower average distance between nodes in a graph than the Cartesian product. We call this the *star product*.

Using the above insight, we can write all of the graph products in the form

$$Adj(\mathcal{G} \otimes_{f,g} \mathcal{H}) = A_G \otimes f(A_H) + I_n \otimes g(A_H) \tag{7}$$

$$= \begin{bmatrix} a_{1,1}f(A_H) & \cdots & a_{1,n}f(A_H) \\ \vdots & \ddots & \vdots \\ a_{n,1}f(A_H) & \cdots & a_{n,n}f(A_H) \end{bmatrix} + \begin{bmatrix} g(A_H) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & g(A_H) \end{bmatrix} \tag{8}$$

where $a_{ij} = (A_G)_{ij}$ and the functions f and g depend on the product as shown in Table I. Note that in some cases the only dependency in f or g on the adjacency matrix comes from the size of that matrix.

Product	Notation	$f(A_H)$	$g(A_H)$
Cartesian product	$\mathcal{G} \ \square \ \mathcal{H}$	I_m	A_H
Tensor product	$\mathcal{G} \times \mathcal{H}$	A_H	$\mathbf{0}_m$
Strong product	$\mathcal{G} \boxtimes \mathcal{H}$	$I_m + A_H$	A_H
Lexicographic product	$\mathcal{G} \bullet \mathcal{H}$	J_m	A_H
Rooted product	$\mathcal{G} \circ \mathcal{H}$	$D_{m,k}$	A_H
Corona product	$\mathcal{G} \odot \mathcal{H}$	$D_{m,1}$	$\hat{A}_H$
Star product	$\mathcal{G} \star \mathcal{H}$	A_H	A_H

TABLE I
FUNCTIONAL BASIS FOR GRAPH PRODUCTS.

The change of notation (7) admits new possible graph products. We can create a new graph product by substituting new choices for $g(\cdot)$ and $f(\cdot)$ but we leave this extension for future work.

Once we make this identification, the first and second components of the product (7) take on distinct meaning. The first part $A_G \otimes f(A_H)$ defines the inter-PoP links, and the second part defines the intra-PoP links. Routers automatically

get a label based on their co-ordinate in the Cartesian product space, i.e., if the PoPs $N(\mathcal{G}) = \{A, B, \ldots, \}$, and the routers $N(\mathcal{H}) = \{1, 2, \ldots, \}$ then the routers in the product graph get names combining the PoP and router number, e.g., $(A, 1)$ and so on. Obviously the above identification is not necessary for the generalized graph product, but we use it to illustrate the power of the approach in what follows.

Many properties of these graph products are well known. For instance, consider edge connectivity: an edge cut of a graph $\mathcal{G}$ is a group of edges whose total removal renders the graph disconnected. The edge-connectivity $\lambda(\mathcal{G})$ is the size of the smallest edge cut. The edge connectivity of a disconnected graph or the trivial graph (with 1 node) is 0.

A lower bound for the edge connectivity of standard Cartesian graphs is known [28]:

$$\lambda(\mathcal{G} \,\square\, \mathcal{H}) \geq \lambda(\mathcal{G}) + \lambda(\mathcal{H}).$$

There are equivalent results for node connectivity [28], and the edge connectivity of strong product is provided in [29], but the connectivity for tensor product is currently unknown.

A. Generalization 1

Given this notational change, out first generalization is to allow for non-identical PoP structures. Imagine that instead of a single $\mathcal{H}$ to describe the same structure inside all PoPs, we have a set of graphs $\{\mathcal{H}_i\}_{i=1}^{n}$ which describe the connectivity of each of the n PoPs in the network. This allows us to change the number of nodes per PoP, and the structure inside the PoP.

In this case, the block sizes of the adjacency matrices of the n PoPs would not fit together in one Kronecker product. However, we can construct a new Kronecker type product using functions of two variables. Similar to (8), the adjacency matrix for this generalized product can be written as

$$\begin{aligned} &Adj(\mathcal{G}, \otimes_{f,g}(H_1, \ldots, H_n)) = \\ &\begin{bmatrix} a_{11}f(A_{H_1}, A_{H_1}) & \cdots & a_{1n}f(A_{H_1}, A_{H_n}) \\ \vdots & \ddots & \vdots \\ a_{n1}f(A_{H_n}, A_{H_1}) & \cdots & a_{nn}f(A_{H_n}, A_{H_n}) \end{bmatrix} \\ &+ \begin{bmatrix} g(A_{H_1}, A_{H_1}) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & g(A_{H_n}, A_{H_n}) \end{bmatrix}, \end{aligned} \tag{9}$$

where the functions $f(A_{H_i}, A_{H_j})$ (for $i \neq j$) and $g(A_{H_i}, A_{H_i})$ are given in Table II, where

- n_i is the number of nodes in $\mathcal{H}_i$
- $I_{n_i \times n_j}$ is a matrix of dimension $n_i \times n_j$ with 1 on the diagonal entries and 0 everywhere else.
- $L_{n_i \times n_j}$ is a matrix of dimension $n_i \times n_j$ with $L(i, j) = 1$ if $A_{H_i}(i, j) = 1$ or $A_{H_j}(i, j) = 1$ and 0 otherwise.

These generalised variants were chosen so that they satisfy the conditions required at the start of this section, e.g., that they collapse back to standard products when all the $\mathcal{H}_i$ are the same.

In Figure 7 we construct our first approximation to the AARNET network using this product. There are two classes of PoPs: those with one node, and those with two connected

Product	Notation	$f(A_{H_i}, A_{H_j})$	$g(A_{H_i}, A_{H_i})$
Cartesian	$\mathcal{G} \;\square\; (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$I_{n_i \times n_j}$	A_{H_i}
Tensor	$\mathcal{G} \;\times\; (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$L_{n_i \times n_j}$	$\mathbf{0}_{n_i}$
Strong	$\mathcal{G} \boxtimes (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$I_{n_i \times n_j}$ + $L_{n_i \times n_j}$	A_{H_i}
Lexical	$\mathcal{G} \bullet (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$J_{n_i \times n_j}$	A_{H_i}
Rooted	$\mathcal{G} \circ (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$D_{m,1}$	A_{H_i}
Corona	$\mathcal{G} \odot (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$D_{m+1,1}$	$\hat{A}_{H_i}$
Star	$\mathcal{G} \star\; (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$L_{n_i \times n_j}$	A_{H_i}

TABLE II
FUNCTIONAL BASIS FOR GRAPH PRODUCTS WITH DIFFERENT $\mathcal{H}_i$, WHERE n_i IS THE NUMBER OF NODES IN $\mathcal{H}_i$.

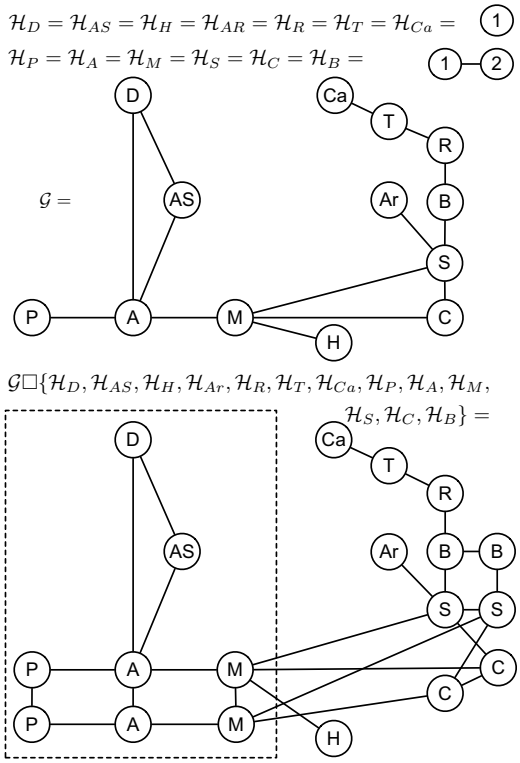

Fig. 7. Modeling the AARNET network with the generalized Cartesian product. PoP names are abbreviated to their first letter.

nodes. The inter-PoP graph is shown as $\mathcal{G}$ in the figure, and the product graph is also shown. Note that the number of routers in each PoP is correct, but the inter-PoP connectivity between these is not correct, except for the simpler parts of the network. There is a requirement that we be able to vary the connectivity model (or product) on a link by link basis to be able to reproduce the AARNET network, and that is what we consider in the following section. However, first we show that we can still work with generalized products, for instance to determine their connectivity.

B. Connectivity of the generalized Cartesian product

The generalized products maintain many properties of the standard products. Due to space constraints, we consider only the edge-connectivity of the generalized Cartesian product. This is closely related to the design requirements for connectivity and redundancy given in sections III-B1 and III-B2.

For simplicity of the argument, we label the nodes of the graph $\mathcal{H}_i$ with n_i nodes by $V(\mathcal{H}_i) = \{1, \ldots, n_i\}$. Similarly,

we label the nodes in $\mathcal{G}$ as $V(\mathcal{G}) = \{1, \ldots, n\}$.

Let i be a vertex of $\mathcal{G}$. The subgraph of $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n)$ induced by $\{i\} \times V(\mathcal{H}_i)$ is isomorphic to $\mathcal{H}_i$. We shall call this graph $\mathcal{H}_i$. Let $m = \max(|V(\mathcal{H}_1)|, \ldots, |V(\mathcal{H}_n)|)$, the maximum size of all the PoP graphs. For a given value of $i \leq m$, the subgraph of $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n)$ induced by $\{(i,j) | i \in V(\mathcal{G}), j \in V(\mathcal{H}_1), \ldots, V(\mathcal{H}_n)\}$ is denoted by $\mathcal{G}_i$. Note that $\mathcal{G}_j, 1 \leq j \leq m$, are subgraphs of $\mathcal{G}$ and that by labelling nodes of the PoP graph $\mathcal{H}_i$ from 1 to $n_i = |V(\mathcal{H}_i)|$, the implicit assumption among $\mathcal{G}_j$ is that $|V(\mathcal{G}_j)|$ is non-decreasing, e.g., $|V(\mathcal{G}_j)|) \geq |V(\mathcal{G}_{j+1})|)$. $\mathcal{G}_i$ and $\mathcal{H}_j$ are called the *G-fibers* and *H-fibers* of the Cartesian product graph [29].

Let

$$\lambda_G = \min\{\lambda(\mathcal{G}_1), \ldots, \lambda(\mathcal{G}_m)\},$$
$$\text{and } \lambda_H = \min\{\lambda(\mathcal{H}_1), \ldots, \lambda(\mathcal{H}_n)\}.$$

When $\mathcal{G}$ and all $\mathcal{H}_i$ are finite, connected graphs with at least one node, the following result provides a lower bound for the connectivity of the generalised Cartesian product $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n)$.

Theorem 1: **(Connectivity – Generalised Cartesian)** The edge connectivity of the network $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n)$ has a lower bound $\lambda_G + \lambda_H$, i.e.,

$$\lambda\left(\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n)\right) \geq \lambda_G + \lambda_H.$$

Proof: Let $\mathcal{S}$ be an edge set of $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n)$, $\mathcal{S} \subseteq \mathcal{E}\left(\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n\}\right)$ with $|\mathcal{S}| < \lambda_G + \lambda_H$. We need to show that $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n) \backslash \mathcal{S}$ is connected.

Let

$$\bar{g}_j = |S \cap E(\mathcal{G}_j)|, \text{ for } j = 1, \ldots, m, \text{ and } \bar{\lambda}_G = \max_j\{\bar{g}_j\};$$
$$\bar{h}_i = |S \cap E(\mathcal{H}_i)|, \text{ for } i = 1, \ldots, n, \text{ and } \bar{\lambda}_H = \max_i\{\bar{h}_i\}.$$

As $\mathcal{H}_i$ and $\mathcal{G}_j$ are edge-disjoint and completely cover $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n)$,

$$\sum_i \bar{h}_i + \sum_j \bar{g}_j = |S|.$$

Furthermore, as $\bar{\lambda}_G + \bar{\lambda}_H \leq \sum_i \bar{h}_i + \sum_j \bar{g}_j < \lambda_G + \lambda_H$, at least one of the two following inequalities is true: $\bar{\lambda}_G < \lambda_G$ or $\bar{\lambda}_H < \lambda_H$. We consider these two cases separately.

1) **Case 1:** $0 \leq \bar{\lambda}_G < \lambda_G$ implying all G-fibers are non-trivial (i.e., they have at least 2 nodes) and are connected in $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n) \backslash \mathcal{S}$. Let k be the number of H-fibers with the maximum number of nodes m. Without loss of generality, denote these H-fibers as $\mathcal{H}_{i_1}, \ldots, \mathcal{H}_{i_k}$. Because at least one H-fibers has m nodes, $k \geq 1$. We only need to prove that one of these H-fibers is connected in $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n) \backslash \mathcal{S}$ as it will connect all the G-fibers and makes the whole graph connected.
   Consider the G-fiber $\mathcal{G}_m$, the number of nodes in $\mathcal{G}_m$ is $|V(\mathcal{G}_m)| = k$. Note that $\lambda_G \leq \lambda(G_m) \leq |V(\mathcal{G}_m)| - 1 = k - 1$. We now consider the following two cases.
   a) If $\lambda_H \geq 1$, then

$$|S| < \lambda_G + \lambda_H \leq k - 1 + \lambda_H \leq k\lambda_H,$$

   for all $\lambda_H \geq 1$, and $k \geq 1$. Thus, at least one of the H-fibers $\mathcal{H}_{i_1}, \ldots, \mathcal{H}_{i_k}$ is connected.
   b) If $\lambda_H = 0$, we again consider two sub-cases:
      i) If $m = 1$, all the PoP graphs have one node. Thus, the product graph is simply a graph of one G-fiber, namely $\mathcal{G}_1$. And as $\bar{\lambda}_G = |S| < \lambda(\mathcal{G}_1)$, this G-fiber is connected.
      ii) If $m > 1$, then all k H-fibers $\mathcal{H}_{i_1}, \ldots, \mathcal{H}_{i_k}$ have $m > 1$ nodes and their connectivity $\lambda(\mathcal{H}_{i_l}) \geq 1$. As

$$|S| < \lambda_G + \lambda_H = \lambda_G \leq k - 1 < \sum_{l=1}^{k} \lambda(\mathcal{H}_{i_l}),$$

      at least one of the k H-fibers $\mathcal{H}_{i_1}, \ldots, \mathcal{H}_{i_k}$ is connected.

2) **Case 2:** $0 \leq \bar{\lambda}_H < \lambda_H$, implying $\lambda_H > 0$, so all H-fibers are connected and are not trivial. Let $q = \min\{|V(\mathcal{H}_1)|, \ldots, |V(\mathcal{H}_n)|\}$, the minimum size of all the H-fibers (PoP graphs). As $\mathcal{H}_i$ are non-trivial, $q \geq 2$. Each of the G-fibers $\mathcal{G}_1, \ldots \mathcal{G}_q$ has n nodes and is isomorphic to $\mathcal{G}$. Thus, $\lambda(G_1) = \ldots = \lambda(G_q) = \lambda(\mathcal{G})$. We only need to show that the one of the G-fibers $\mathcal{G}_1, \ldots \mathcal{G}_q$ is connected in $\mathcal{G} \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_n) \backslash \mathcal{S}$. Indeed, consider the following cases.
   a) If $\lambda_G > 0$, as $\lambda_H = \min\{\lambda(\mathcal{H}_1), \ldots, \lambda(\mathcal{H}_n) \leq q - 1$,

$$|S| < \lambda_G + \lambda_H \leq q - 1 + \lambda_G \leq q\lambda_G.$$

   Thus at least one of $\mathcal{G}_1, \ldots, \mathcal{G}_q$ is connected.
   b) $\lambda_G = 0$, again there are two sub-cases
      i) If $\lambda(\mathcal{G}) = 0$, we have the trivial case of the product graph containing only one H-fiber (PoP graph). This H-fiber is connected as $\bar{\lambda}_H < \lambda_H$.
      ii) If $\lambda(\mathcal{G}) > 0$, then

$$|S| < \lambda_H \leq q - 1 \leq (q-1)\lambda(\mathcal{G}).$$

      Thus one of the fibers $\mathcal{G}_1, \ldots \mathcal{G}_q$ is connected.

■

In many cases, we are only interested in the connectivity of small (but important) parts of the network. The following corollary is useful when analyzing these sub-networks.

Corollary 1: For any subgraph $\mathcal{G}' \subseteq \mathcal{G}$ with $n' = |V(\mathcal{G}')|$ nodes and n' PoP graphs $\{\mathcal{H}_1, \ldots, \mathcal{H}_{n'}\}$, let $\mathcal{G}'_1, \ldots, \mathcal{G}'_m$ be the G-fibers in the product $\mathcal{G}' \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_{n'})$. The connectivity of $\mathcal{G}' \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_{n'})$ has a lower bound

$$\lambda\left(\mathcal{G}' \,\square\, (\mathcal{H}_1, \ldots, \mathcal{H}_{n'})\right) \geq \min\{\lambda(\mathcal{G}'_1), \ldots, \lambda(\mathcal{G}'_m)\} + \min\{\lambda(\mathcal{H}_1), \ldots, \lambda(\mathcal{H}_{n'})\}.$$

V. Second Generalization

We now extend the graph products to allow for different inter-PoP links to have different structure. For instance, one may wish to have different levels of redundancy depending on the importance of a particular edge in the inter-PoP-level graph. We need to generalize the Kronecker product in the

following way. Instead of a matrix of scalars, take A to be a matrix of functions $a_{ij}(\cdot)$, then

$$A \otimes B = \begin{bmatrix} a_{11}(B) & \cdots & a_{1n}(B) \\ \vdots & & \vdots \\ a_{m1}(B) & \cdots & a_{mn}(B) \end{bmatrix}.$$

Given that definition, take

$$\begin{aligned} Adj((\mathcal{G}, F_G, G_G) \otimes \mathcal{H}) &= F_G \otimes A_H + G_G \otimes A_H \\ &= (F_G + G_G) \otimes A_H, \end{aligned} \quad (10)$$

where instead of the adjacency matrix A_G, we create a matrix F_G which contains elements that are functions chosen from a set of functions implied by the choices for $f(\cdot)$ in Table I or Table II for different $\mathcal{H}_i$. The matrix of functions G_G is likewise chosen to match the relevant $g(\cdot)$ function.

Choosing the appropriate function for each block of the Kronecker product allows us to choose the type of connectivity between each pair of PoPs. In Figure 8 we extend our example of AARNET using different functions on different links. On link P-A we use the Cartesian product but the link A-AS uses the tensor product. This generalization allows us to design correctly the component of the network within the dotted box, but some parts outside of this box are still incorrect.

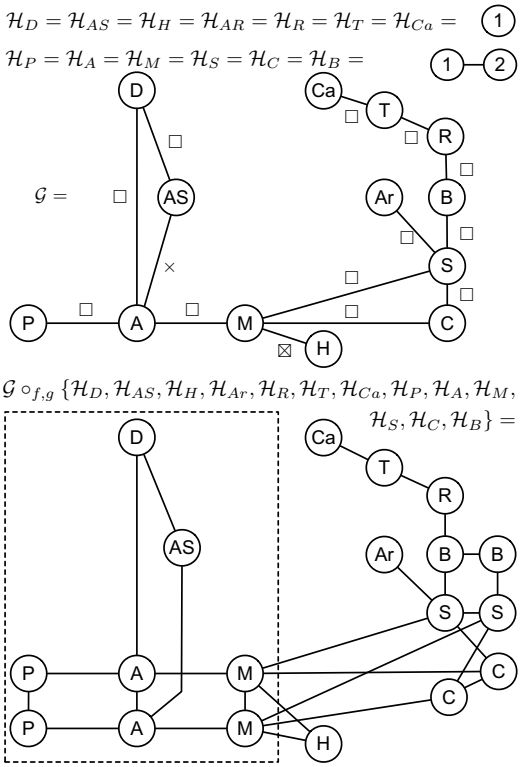

Fig. 8. Modeling the AARNET network with different products on links.

VI. Third Generalization

The functions $f_{ij}(A_{H_i}, A_{H_j})$ in the second generalization required us to define different connections simultaneously for *all* nodes in a pair of PoPs. In many cases, such as sections of the AARNET network outside the dotted box, it is desirable to specify connectivities for only a subset of nodes. This can be achieved by breaking $f_{ij}(A_{H_i}, A_{H_j})$ into several smaller functions. The trivial approach is to specify each link for each pair of routers between the two PoPs separately, which would require us to specify $n_i n_j$ connections for the pair $\mathcal{H}_i$ and $\mathcal{H}_j$.

However, nodes in a network are not connected arbitrarily. They often form structures to provide high level redundancy and reliability requirements. For example, the network might consist a major backbone that connects all the PoPs to provide connectivity, with smaller backbones connecting a chosen subset of PoPs to provide redundancy at important locations.

Let m be the number of these separate network wide connectivity patterns. We can generalize the graph product to include these patterns by first assigning each router in a PoP with a label and then replacing the graph $\mathcal{G}$ with a set of graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$ where each of the graphs $\mathcal{G}_l$ represents the connectivity of nodes with label l in the PoPs. The set of labels assigned to nodes in the PoP graph $\mathcal{H}_i$ is denoted by $\mathcal{L}_i$. Note that $\mathcal{L}_i \subseteq \{1, \ldots, m\}$.

The graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$ are called the *G-fibers* of the network. The links in each G-fiber are called *Cartesian links*. We call links that connect routers with different labels between PoPs *cross links*.

We can define the Cartesian and cross links by functions on the node labels. For a pair of PoPs $\mathcal{H}_i$ and $\mathcal{H}_j$ and two nodes $k \in V(\mathcal{H}_i)$, $l \in V(\mathcal{H}_j)$, let $s(k, \mathcal{H}_i, l, \mathcal{H}_j)$ be the function that takes two labels (k and l) and two PoP graphs and returns a matrix that specify whether the two nodes with these two labels are directly connected by a link. That is,

$$s(k, \mathcal{H}_i, l, \mathcal{H}_j) = D_{k,l}\ , \quad (11)$$

where $D_{k,l}$ is a matrix of dimension $n_i \times n_j$ with 1 at the (k,l)-entry and 0 everywhere else.

The adjacency matrix now takes the form

$$\begin{aligned} &Adj(((\mathcal{G}_1, \ldots, \mathcal{G}_m), F_G, G_G) \otimes (\mathcal{H}_1, \ldots, \mathcal{H}_n)) = \\ &\begin{bmatrix} a_{11}f_{11}(A_{H_1}, A_{H_1}, \mathcal{L}_1, \mathcal{L}_1) \cdots a_{1n}f_{1n}(A_{H_1}, A_{H_n}, \mathcal{L}_1, \mathcal{L}_n) \\ \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ a_{n1}f_{n1}(A_{H_n}, A_{H_1}, \mathcal{L}_n, \mathcal{L}_1) \cdots a_{nn}f_{nn}(A_{H_n}, A_{H_n}, \mathcal{L}_n, \mathcal{L}_n) \end{bmatrix} \\ &+ \begin{bmatrix} g_{11}(A_{H_1}, A_{H_1)} & & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & & g_{nn}(A_{H_n}, A_{H_n}) \end{bmatrix}, \end{aligned}$$

with the $g(\cdot)$ functions are the same functions in (10) and Table II, but the $f(\cdot)$ functions now take node labels in addition to A_{H_i}, A_{H_j} as variables. The functions $f(A_{H_i}, A_{H_j}, \mathcal{L}_i, \mathcal{L}_j)$ (for $i \neq j$) for different products are given in Table III with $f(A_{H_i}, A_{H_i}, \mathcal{L}_i, \mathcal{L}_i) = \mathbf{0}_{n_i \times n_i}$. We omit the lexicographical, rooted, Corona and star products in the table as we do not use them for the networks in the Topology Zoo.

Figure 9 shows the G-fibers for AARNET. These can be used with the generalized product to accurately generate the AARNET network. This generalized product can be used to model most networks in the Topology Zoo, for example the ACOnet network as shown in Figure 10 and the Internode network in Figures 11. We have also applied it to larger

Product	Notation	$f_{ij}(A_{H_i}, A_{H_j}, \mathcal{L}_i, \mathcal{L}_j)$
Cartesian	$(\mathcal{G}_1, \ldots, \mathcal{G}_m) \ \square \ (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$\sum_{k \in \mathcal{L}_i \cap \mathcal{L}_j} s(k, \mathcal{H}_i, k, \mathcal{H}_j)$
Tensor	$(\mathcal{G}_1, \ldots, \mathcal{G}_m) \ \times \ (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$\sum_{k \in \mathcal{L}_i} \sum_{l \in \mathcal{L}_j, l \neq k} s(k, \mathcal{H}_i, l, \mathcal{H}_j) \ (A_{H_i}(k,l) \vee A_{H_j}(k,l))$
Strong	$(\mathcal{G}_1, \ldots, \mathcal{G}_m) \boxtimes (\mathcal{H}_1, \ldots, \mathcal{H}_n)$	$\sum_{k \in \mathcal{L}_i \cap \mathcal{L}_j} s(k, \mathcal{H}_i, k, \mathcal{H}_j) + \sum_{k \in \mathcal{L}_i} \sum_{l \in \mathcal{L}_j, l \neq k} s(k, \mathcal{H}_i, l, \mathcal{H}_j) \ (A_{H_i}(k,l) \vee A_{H_j}(k,l))$

TABLE III
FUNCTIONAL BASIS FOR THIRD GENERALIZATION OF GRAPH PRODUCTS.

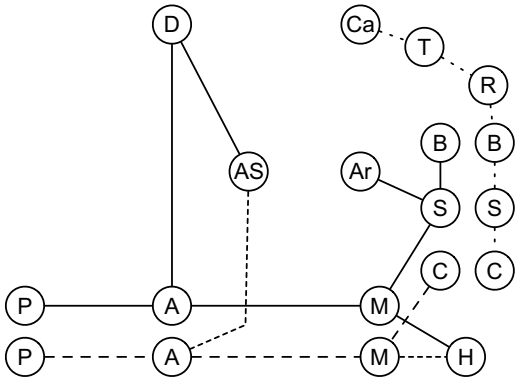

Fig. 9. The AARNET network consists of three G-fibers (indicated by line types) and 13 H-fibers (PoPs). The H-fibers are connected mostly using Cartesian links except the two cross links from A to AS and M to H. For clarity, we omit the intra-PoP links in the graphs.

networks such as SUNET, the Swedish Academic Network, with 25 PoPs and 3 G-fibers, though we omit the plots due to space constraints. This factorization into G and H fibers is closely aligned with the design requirement for operational simplicity in section III-B3. We believe the number of G-fibers in a network is one measure of simplicity. In fact, AARNET have told us that they manage software updates by taking advantage of G-fibers.

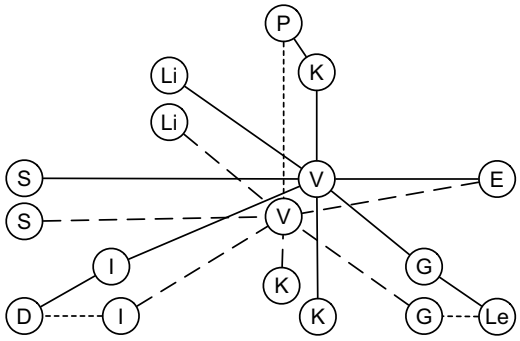

Fig. 10. The ACOnet network has two G-fibers and 11 H-fibers. All of the H-fibers are interconnected using Cartesian links except three cross links at St. Polten (P) to Vienna (V), Leoben (Le) to Graz (G), and Dornbirn (D) to Innsbruck (I).

VII. USEFUL PROPERTIES

The work above may seem to be more of a notational convenience than a real change to the way we approach network modeling and design. However, the advantage of algebraic descriptions is that they can be more easily manipulated. To this end we have developed a prototype graphical user interface

We will consider one particular example: assigning weights to links. We have only been dealing with adjacency matrices, but the algebraic structures we have developed are easily applied to the problem of assigning values to each link. Before we can specify exactly how to assign link weights we need to decide what purpose these link weights should achieve.

In a network constructed using the graph product, there are two types of links: intra-PoP links (links within each H-fiber, on the diagonal blocks of the adjacency matrix) and inter-PoP links (links between H-fibers on the off-diagonal blocks). We shall allow an operator to specify the intra-PoP weights on each H-fiber, and aim to assign weights to the inter-PoP links to achieve high-level management requirements. One such requirement is making one G-fiber the main fiber for traffic between POPs, and the others provide backups. If we are using shortest path routing, then we want to assign a lower weight to the main fiber than to the others.

Another purpose for the link weights is ensuring that local traffic remains local, e.g. that traffic between two routers in the same POP does not go via another POP. To do this it is merely necessary to ensure that the links in G-fibers have a sufficiently high weight compared to those in the H-fibers. We can satisfy both this locality and backup criterion together.

We start with matrices of link weights instead of adjacency matrices. Let each link weight w be a positive real number or 0 for links that do not exist. Define matrices B_{H_i} such that the $(j,k)^{\text{th}}$ entries are the weights of link (j,k) in graph $\mathcal{H}_i$. Assume for now that each G-fiber is identical, and define B_G similarly to B_{H_i}. We also define $\mathbf{v}$, a vector of weights, with one entry for each G-fiber. The purpose of $\mathbf{v}$ is to allow separate weights to be assigned to different G-fibers. Let $\text{diag}(\mathbf{v})$ be the matrix with the entries of $\mathbf{v}$ on its diagonal.

We start with assigning weights to a simple Cartesian product graph. The weight matrix of $\mathcal{G} \ \square \ \mathcal{H}$ is

$$B_{G \ \square \ H} = B_G \otimes \text{diag}(\mathbf{v}) + I_n \otimes B_H.$$

Compare this to equation (1). We have replaced the adjacency matrices A_G, A_H with the weight matrices B_G, B_H, and we have replaced I_m with $\text{diag}(\mathbf{v})$. This formulation gives each G-fiber a distinct, tunable weight.

With appropriate choice of weights, $B_{G \ \square \ H}$ can be used to ensure both locality and make one G-fiber a main fiber, and the other ones backup G-fibers. Let the i^{th} fiber be the main fiber. Let h^+ be any number greater than the sum of the weights in $\mathcal{H}$, and the number of nodes in $\mathcal{G}$ be n and set the $B_G = A_G$. Then set $v_i = h^+$ and $v_j = (n-1) \times h^+$ for $j \neq i$. Locality is ensured as every G-fiber link has greater weight than all the links in a single H-fiber combined, and G-fiber i will be used preferentially.

Another application for such link weights is load-balancing where we simple set $v_j = 1$ for all j so that "parallel" links (links travelling between the same two H-fibers) are all the same weight.

This method of assigning link weights extends naturally to the generalised graph product. We can still use Equation (10), however replace A_{H_i} with B_{H_i} and in the definition of f in Table III replace $s(k, \mathcal{H}_i, k, \mathcal{H}_j)$ with $v_k s(k, \mathcal{H}_i, k, \mathcal{H}_j)$. An example on the Internode network topology is given in figure 11. This uses G-fiber 2 as the main G-fiber, and the other G-fibers as backup. Here $\mathbf{v} = [50, 10, 50]$.

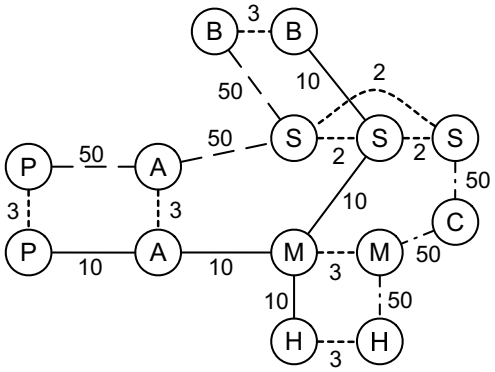

Fig. 11. The Internode network can be modelled as a generalised Cartesian product of 3 G-fibers and 7 H-fibers (PoP graphs). Link weights are shown on the edges. Here the H-fibers labeled P, A, M, H, B each consist of a single link of weight 3. The H-fiber labeled S is 3 links of weight 2. G-fiber 2 has a lower weight than G-fibers 1 and 3 because it's the primary fiber in this weight assigment. Here $\mathbf{v} = [50, 10, 50]$.

VIII. Conclusion and future work

This paper describes several generalized graph products that allow the mathematical design of a network in terms of small subgraphs. We have applied these products to construct and analyze several real world networks and in sections IV-B and VI discuss how these results are related to the high-level design requirements of section III-B. We also give an example of the use of these products to automate tasks such as weight assignment consistent with some high-level requirement. We have created a prototype GUI for creating algebraic descriptions of a network generated using graph products and *vice versa*.

Here the decomposition of existing networks was done manually by removing all intra pop links from a router level graph. The remaining graph generally represents the disjoint G-fibers of the network. However there are exceptions where nodes within a PoP are still connected due to links that cross G-fibers. This leads to cases where the decomposition is not unique. Future work would include the development of algorithms that calculate decompositions based on some optimization criteria. In order to maximize the benefit of this formalization, future work will also include the development of techniques that allow for auto-configuration, network synthesis and optimization.

Acknowledgement

The authors would like to acknowledge the support of the Australian Research Council through grants DP0985063 and DP110103505, and two Australian Postgraduate Awards.

References

[1] B. Waxman, "Routing of multipoint connections," *Selected Areas in Communications, IEEE Journal on*, vol. 6, pp. 1617 –1622, dec 1988.
[2] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *ACM SIGCOMM*, pp. 251–262, 1999.
[3] A. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, 1999.
[4] R. Albert, H.Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, pp. 378–382, 2000.
[5] S.-H. Yook, H.Jeong, and A.-L. Barabási, "Modeling the Internet's large-scale topology," *PNAS*, no. 99, pp. 13382–13386, 2002.
[6] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of IEEE Infocom '96*, (San Francisco, CA), 1996.
[7] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topology generators: degree-based vs. structural," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 147–159, August 2002.
[8] J. C. Doyle, D. L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, and W. Willinger, "The "robust yet fragile" nature of the Internet," *Proceedings of the National Academy of Sciences of the USA (PNAS)*, vol. 102, pp. 14497–502, October 2005.
[9] E. Ravasz and A.-L. Barabási, "Hierarchical organization in complex networks," *Phys. Rev. E*, vol. 67, p. 026112, Feb 2003.
[10] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A first-principles approach to understanding the Internet's router-level topology," in *SIGCOMM '04*, (New York, NY, USA), pp. 3–14, ACM, 2004.
[11] D. Alderson, L. Li, W. Willinger, and J. Doyle, "Understanding internet topology: principles, models, and validation," *Networking, IEEE/ACM Transactions on*, vol. 13, pp. 1205 – 1218, dec. 2005.
[12] T. Griffin and J. Sobrinho, "Metarouting," *ACM SIGCOMM*, vol. 35, no. 4, pp. 1–12, 2005.
[13] T. G. Griffin, "The stratified shortest-paths problem," in *COMSNET*, 2009.
[14] G. Sabidussi, "Graph multiplication," *Math.Zeitschr.*, vol. 72, pp. 446–457, 1960.
[15] F. Gurski, "Graph operations on clique-width bounded graphs." arXiv:cs/0701185v1http://arxiv.org/pdf/cs.DS/0701185, 2007.
[16] P. M. Weichsel, "The Kronecker product of graphs," *Proceedings of the American Mathematical Society*, vol. 13, no. 1, pp. 47–52, 1962.
[17] D. Geller and S. Stahl, "The chromatic number and other functions of the lexicographic product," *Journal of Combinatorial Theory, Series B*, vol. 19, no. 1, pp. 87–95, 1975.
[18] C. Godsil and B. McKay, "A new graph product and its spectrum," *Bulletin of the Australian Mathematical Society*, vol. 18, no. 1, pp. 21–28, 1978.
[19] H. Kwong and S.-M. Lee, "On the integer-magic spectra of the corona of two graphs," *Congressus Numerantium*, pp. 207–222, 2005.
[20] K. Yoshida, Y. Kikuchi, M. Yamamoto, Y. Fujii, K. Nagami, I. Nakagawa, and H. Esaki, "Inferring PoP-level ISP topology through end-to-end delay measurement," in *PAM 2009*, pp. 35–44, 2009.
[21] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, March 2010.
[22] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 0: Introduction to Combinatorial Algorithms and Boolean Functions (Art of Computer Programming)*. Addison-Wesley Professional, 1 ed., 2008.
[23] Topology zoo website: http://www.topology-zoo.org, 1st January 2011.
[24] P. Baran, "On distributed communications: 1. introduction to distributed communications network." RAND Memorandum, August 1964.
[25] P. Sebos, J. Yates, G. Hjalmtysson, and A. Greenberg, "Auto-discovery of shared risk link groups," in *Optical Fiber Communication Conference and Exhibit, 2001. OFC 2001*, vol. 3, pp. WDD3–1 – WDD3–3 vol.3, 2001.
[26] R. Bush and D. Meyer. Request for Comments: 3439: http://tools.ietf.org/html/rfc3439, December 2002.
[27] G. Korporaal, *AARNet — 20 years of the internet in Australia*. http://mirror.aarnet.edu.au/pub/aarnet/AARNet_20YearBook_Full.pdf, 2009.
[28] G. Sabidussi, "Graphs with a given group and given graph-theoretical properties," *Canad. J. Math.*, vol. 9, pp. 515–525, 1957.
[29] B. Bresar and S. Spacapan, "Edge-connectivity of strong products of graphs," *Discussiones Mathematicae, Graph Theory*, vol. 27, pp. 333–343, 2007.