Rate of convergence of increasing path-vector routing protocols

Matthew L. Daggitt
Department of Computer Science & Technology
University of Cambridge
mld46@cam.ac.uk

Timothy G. Griffin
Department of Computer Science & Technology
University of Cambridge
tgg22@cam.ac.uk

***Abstract*—A good measure of the rate of convergence of path-vector protocols is the number of synchronous iterations required for convergence in the worst case. From an algebraic perspective, the rate of convergence depends on the expressive power of the routing algebra associated with the protocol. For example in a network of n nodes, shortest-path protocols are guaranteed to converge in $O(n)$ iterations. In contrast the algebra underlying the Border Gateway Protocol (BGP) is in some sense too expressive and the protocol is not guaranteed to converge. There is significant interest in finding well-behaved algebras that still have enough expressive power to satisfy network operators.**

Recent theoretical results have shown that by constraining routing algebras to those that are "strictly increasing" we can guarantee the convergence of path-vector protocols. Currently the best theoretical worst-case upper bound for the convergence of such algebras is $O(n!)$ iterations. However in practice it is difficult to find examples that do not converge in n iterations. In this paper we close this gap. We first present a family of network configurations that converges in $\Theta(n^2)$ iterations, demonstrating that the worst case is $\Omega(n^2)$ iterations. We then prove that path-vector protocols with a strictly increasing algebra are guaranteed to converge in $O(n^2)$ iterations. Together these results establish a tight $\Theta(n^2)$ bound. This is another piece of the puzzle in showing that "strictly increasing" is, at least on a technical level, a reasonable constraint for practical policy-rich protocols.

In memory of Abha Ahuja.

I. Introduction

Complex path-vector protocols such as the Border Gateway Protocol (BGP) [1] provide extremely powerful tools to fine-tune routing configurations. However the very power of the policy language gives rise to anomalies such as non-convergence, non-deterministic convergence and other problems [2], [3], [4].

One of the more successful frameworks for reasoning about these problems is the algebraic approach [5], [6]. This framework untangles the routing problem being solved from the algorithm being used to solve it. A protocol can therefore be decomposed as $protocol = algebra + algorithm$. For instance all distance and path-vector protocols use some variant of distributed version of the Bellman-Ford algorithm. However each protocol instantiates the algorithm with a different algebra. For instance RIP's algebra measures shortest paths, whereas BGP's algebra is far more complicated and includes complex conditional policy.

The main advantage of the algebraic approach is that, by considering an abstract algebra, the convergence of all distance/path-vector protocols can be reasoned about within a single framework. Hence, although much of the more recent work in the area is motivated by BGP, it equally applies to many other path-vector protocols. The model does not yet include protocols that route around congestion or other transient traffic metrics. However protocols such as BGP do not route around congestion.

Recent work has demonstrated that if the underlying algebra is *strictly increasing* then distance/path-vector protocols are guaranteed to converge in a well-behaved manner [5], [7].

In this paper we address the open question of how quickly convergence occurs for path-vectors protocols with strictly increasing algebras. Existing results [5], [7], [8] have focused on convergence rather than the rate of convergence. When examined closely the convergence proofs suggest a worst-case bound of $O(n!)$ synchronous iterations. Clearly there is potentially a problem here. If increasing algebras can take an exponential number of iterations to converge, then for all practical purposes they might as well not converge at all for networks of any reasonable size. However in practice it appears non-trivial to construct examples that take more than n iterations to converge.

We entirely close this gap by demonstrating an example of a strictly increasing path-vector algebra and a family of graphs Q_n that require $\Theta(n^2)$ synchronous iterations to converge. We then prove that in the worst case strictly increasing path algebras converge in $O(n^2)$ iterations. Together these results show that the worst-case is $\Theta(n^2)$.

The $O(n^2)$ proof of convergence has been verified in the theorem prover Agda [9]. Our library, available online [10], also contains the formalised results of [7]. We hope others will be able to use and extend this library to reason about specific routing protocols.

II. Previous work on convergence

A. Algebraic work on convergence

The origins of the algebraic approach stems from early work on semiring best-path problems [6]. This approach successfully modelled many simple algebras such as hop-count and shortest-path. However the proofs of correctness of the best-path algorithms all relied on the fact that these algebras were *distributive*. However work investigating the non-convergence of BGP showed that BGP's algebra violated this "distributivity" property [11]. Furthermore as distributivity equates to all participants in the network sharing a common preference order over routes, it is unlikely that BGP's algebra could ever be made distributive.

An alternative algebraic property "strict increasingness"[1] was proposed by Sobrinho [5]. BGP does not obey such a property, but it is much more natural to suppose that it could as the property only says that the extension of a route must be strictly less preferred than the route itself.

In the original paper Sobrinho showed that if the algebra is strictly increasing then path-vector protocols are guaranteed to converge and reconverge even after arbitrary changes to the network. A more recent paper by Daggitt *et al.* [7] built upon this and showed strictly increasing algebras are guaranteed to converge to a *unique* solution even under considerably weaker asynchronous assumptions.

The strictly increasing property therefore looks to be a promising algebraic theory on which to build the next generation of safe-by-design routing protocols. However the rate of convergence of these algebras is still an open question.

B. Rate of convergence

Measuring the rate of convergence of asynchronous processes is inherently tricky. One obvious approach might be to measure the number of "events" that occur (messages sent/received, table entries updated etc.), however this runs into several problems. Firstly, many of these events happen in parallel and therefore it is not clear that this is an accurate proxy for the actual time required. Secondly, it has been shown that even a simple shortest-paths path-vector protocol may require an exponential number of events to occur in the worst case [12].

Instead we will measure the number of iterations required for the convergence of a synchronous iterative model of path-vector protocols (see Section III-E). This assumes that all nodes propagate their changes instantaneously to each other at the same time. Furthermore given upper bounds on the number of synchronous iterations, the link latency and the loss rate, an upper bound for the time taken for asynchronous convergence can be recovered using the notion of pseudoperiods from [13].

Note that when we talk about the convergence time with respect to the number of synchronous iterations, we are ignoring the $O(n^3)$ work required to compute each iteration. However this work is in some sense uninteresting as it is constant with respect to the algebra and is the portion of work parallelised in the asynchronous version of the protocol. In contrast the iterative component of the work cannot be parallelised and indeed may be adversely affected by the parallelisation.

Labovitz *et al.* [14] demonstrated through a combination of theory and experimentation that is possible in real life to get BGP to explore all $n!$ paths. We should note that BGP does not use an increasing algebra, and therefore this does not affect the theoretical bounds we seek.

The worst-case convergence time for semiring algebras with paths over networks with n nodes has long been known to be $\Theta(n)$ iterations [15]. As mentioned in Section I, current proofs [5], [8], [7] only suggest a worst-case upper bound of $O(n!)$ for the convergence of strictly increasing algebras with paths. These bounds are derived from the number of simple paths in the network.

[1]Sobrinho refers to this property as "monotonicity".

III. An algebraic model of routing

In this section we present our abstract algebraic model of distance-vector routing. The model we describe is taken from [7], which in turn is a refinement of the model proposed by [5]. Section IV presents concrete examples to help the reader to understand the scope and flexibility of this model.

A. Routing algebras

Definition 1. *A* routing algebra *is a tuple* $(S,\ \oplus,\ F,\ \overline{0},\ \overline{\infty})$ *where:*

- *S is the set of weights with $\overline{0} \in S$ and $\overline{\infty} \in S$,*
- *$\oplus : S \times S \to S$ is the choice operator,*
- *F is a set of functions from $S \to S$.*

such that:

R1) $\oplus$ is associative – $\forall x, y, z : x \oplus (y \oplus z) = (x \oplus y) \oplus z$.
R2) $\oplus$ is commutative – $\forall x, y : x \oplus y = y \oplus x$.
R3) $\oplus$ is selective – $\forall x, y : x \oplus y \in \{x,\ y\}$.
R4) $\overline{0}$ is an annihilator for $\oplus$ – $\forall x : x \oplus \overline{0} = \overline{0} \oplus x = \overline{0}$.
R5) $\overline{\infty}$ is an identity for $\oplus$ – $\forall x : x \oplus \overline{\infty} = \overline{\infty} \oplus x = x$.
R6) $\overline{\infty}$ is a fixed point for all f – $\forall f \in F : f(\overline{\infty}) = \overline{\infty}$.

The operator $\oplus$ is the decision procedure used by the protocol to choose between two possible weights. Selectivity implies $\oplus$ always returns one of its two arguments. Hence it is possible to define a total order, $\leq$, over the weights as follows:

$$x \leq y \triangleq x \oplus y = x \qquad x < y \triangleq x \leq y \ \wedge \ x \neq y$$

i.e. $x \leq y$ if and only if x is chosen over y.

The weights $\overline{0}$ and $\overline{\infty}$ represent the weight of the empty path and the weight of an invalid path respectively. Assumptions R4) and R5) imply that $\forall x \in S : \overline{0} \leq x \leq \overline{\infty}$.

Each function in F represents a procedure to transform a route's weight when that route is extended along an edge. Hence elements of F will be used to label edges in the network (see the following section). Readers may be more familiar with arc weights rather than arc functions. However functions are a strictly more expressive model and can capture conditional routing policies such as BGP route maps. Such complex policies are not easily expressible as edge weights. See Section IV for concrete examples of F.

We additionally define the three additional properties. The algebra is *distributive* if:

$$\forall f, x, y : f(x \oplus y) = f(x) \oplus f(y) \tag{1}$$

and is *increasing* if:

$$\forall f, x : x \leq f(x) \tag{2}$$

and is *strictly increasing* if:

$$\forall f, x : x \neq \overline{\infty} \Rightarrow x < f(x) \tag{3}$$

These properties are optional and may or may not be satisfied by a particular routing algebra.

B. Network configuration, paths, and routes

The network is modelled as a directed graph $G = (V,\ E)$, where V is a set of n nodes $V = \{0,\ 1, \cdots, n-1\}$ and E is a set of arcs. A *configuration* of G with respect to a routing algebra $(S,\ \oplus,\ F,\ \overline{0},\ \overline{\infty})$ is a mapping from E to F. Such mappings will be represented by an $n \times n$ adjacency matrix $\mathbf{A}$ where $\mathbf{A}_{ij} \in F$. We assume there exists the constant function $f_{\overline{\infty}} \in F$ that always returns the invalid weight (i.e. $\forall x \in S : f_{\overline{\infty}}(x) = \overline{\infty}$). This function can be used to represent missing edges.

After changes to the network topology (see Section III-E), nodes may be exchanging routes that contain paths that no longer exist in the current configuration. Therefore our definition of a path must be independent of the current graph G and hence is somewhat non-standard.

A *path* $p = [(v_1,\ v'_1),\ \cdots,\ (v_m,\ v'_m)]$ is a (possibly empty) sequence of arcs such that $v'_i = v_{i+1}$ for all $0 \leq i < m$. However, for the reasons outlined above, the arcs are not members of E but are instead arbitrary members of $\mathbb{N} \times \mathbb{N}$.

For notational convenience we will write p as $[\,]$ when $m = 0$ and as $(v_1, v'_1) :: q$ when $0 \leq i < m$ where $q = [(v_2,\ v'_2),\ \cdots,\ (v_k,\ v'_k)]$. A path is *simple* if it never visits a vertex more than once. We also consider a special additional path $\perp$ which represents the invalid path. We will refer to the set of paths as $\mathcal{P}$ and the length of a path p as $|p|$.

C. Path algebras

Our definition of a routing algebra so far models general distance-vector protocols. However we are interested in the strict subset of path-vector protocols, which track the path that routes are generated along and remove any looping paths.

Given an routing algebra $\mathcal{A} = (S,\ \oplus,\ F,\ \overline{0},\ \overline{\infty})$ one way to add paths to $\mathcal{A}$ is as follows:

$$\mathbb{PA}(\mathcal{A}) = (((S - \{\infty\}) \times \mathcal{P}) \cup \{\infty'\},\ \oplus',\ F',\ (\overline{0},\ [\,]),\ \infty').$$

In the augmented algebra, $\mathbb{PA}(\mathcal{A})$, weights are of the form ∞' or $(s,\ p)$ where $s \in S - \{\infty\}$ and p is a path. For all $(s,\ p)$ we have $(s,\ p) \oplus' \infty' = \infty' \oplus' (s,\ p) = (s,\ p)$. Non-$\infty'$ routes are compared lexicographically:

$$(s_1,\ p_1) \oplus (s_2\ p_2) \triangleq \begin{cases} (s_1,\ p_1) & \text{if } s_1 = (s_1 \oplus s_2) \neq s_2 \\ (s_2,\ p_2) & \text{if } s_1 \neq (s_1 \oplus s_2) = s_2 \\ (s_1,\ p_1 \hat{\oplus} p_2) & \text{if } s_1 = s_2 \end{cases}$$

where

$$p_1 \hat{\oplus} p_2 \triangleq \begin{cases} p_1 & \text{if } |p_1| < |p_2| \\ p_2 & \text{if } |p_2| < |p_1| \\ \text{dict}(p_1, p_2) & \text{otherwise} \end{cases}$$

where $\text{dict}(p_1, p_2)$ returns the smallest path in dictionary order.

We define the concatenation operator $\hat{::}$ that takes an arc and a path. The result of $(i, j) \hat{::} p$ is the invalid path $\perp$ if i is already in the path or if the first node of p is not j, otherwise it returns $(i, j) :: p$. For example:

$$\begin{array}{lcl} (5,\ 3) \hat{::} [(3,\ 4),\ (4,\ 5)] & = & \perp \\ (2,\ 1) \hat{::} [(3,\ 4),\ (4,\ 5)] & = & \perp \\ (2,\ 3) \hat{::} [(3,\ 4),\ (4,\ 5)] & = & [(2,\ 3),\ (3,\ 4),\ (4,\ 5)] \end{array}$$

The F' for the path-algebra is the collection of the policy functions $g_{f,u,v}$ which are defined as:

$$g_{f,u,v}(\infty') = \infty'$$

$$g_{f,u,v}(s,p) \triangleq \begin{cases} \infty' & \text{if } f(s) = \infty \\ \infty' & \text{if } (u,\ v) \hat{::} p = \perp \\ (f(s),\ (u,\ v) \hat{::} p) & \text{otherwise} \end{cases}$$

However $\mathbb{PA}$ is just one of many possible methods of adding paths to a routing algebra. For example in BGP's algebra, the path is not the last attribute inspected in a lengthy lexicographic best route selection process. Therefore in order to reason about *all* path-vector protocols, we need to abstract away the exact method of tracking and removing paths.

Definition 2. *A* path algebra *is a tuple* $(S,\ \oplus,\ F,\ \overline{0},\ \overline{\infty},\ path)$ *where:*

- $(S,\ \oplus,\ F,\ \overline{0},\ \overline{\infty})$ *is a routing algebra*
- $path : S \rightarrow \mathcal{P}$ *is a function that returns the path the weight was generated along.*

such that:

P1) $path(x) = \perp\ \Leftrightarrow x = \overline{\infty}$
P2) $path(x) = [\,] \Leftarrow x = \overline{0}$
P3) $path(\mathbf{A}_{ij}(x)) = \begin{cases} \perp & \textit{if } i \in path(x) \\ \perp & \textit{if } j \neq src(path(x)) \\ (i, j) :: path(x) & \textit{otherwise} \end{cases}$

Note that P3) implies that if a path algebra is increasing it is automatically strictly increasing. As expected $\mathbb{PA}(\mathcal{A})$ is a path algebra for any $\mathcal{A}$. Its $path$ function is defined as $path(\infty') = \perp$ and $path((s, p)) = p$.

D. What problem are we solving?

Let $\mathbb{M}_n(S)$ be the set of $n \times n$ matrices over S. Each matrix $\mathbf{X} \in \mathbb{M}_n(S)$ represents one possible global state of the routing protocol. The row $\mathbf{X}_i$ represents the current routing table of node i and $\mathbf{X}_{ij}$ is node i's current route to node j.

Going forwards we will often refer to $\mathbf{X}_{ij}$ as a *route* from i to j. This is a minor abuse of terminology, as $\mathbf{X}_{ij}$ is a weight in S rather than a path in G. The intuition behind this terminology is that every valid route $\mathbf{X}_{ij}$ in a path algebra will contain the path it was generated along.

The aim of distance-vector routing protocols is to find an assignment of routes such that each node's route is the best possible extension of the routes offered to it by each of its neighbours. Concretely, in our algebraic model, we are looking for a state $\mathbf{X}$ such that:

$$\forall i, j : \mathbf{X}_{ij} = \begin{cases} \overline{0} & \text{if } i = j \\ \bigoplus_k \mathbf{A}_{ik}(\mathbf{X}_{kj}) & \text{otherwise} \end{cases} \tag{4}$$

We will now represent this series of equations as a single matrix equation by defining some analogues to traditional matrix addition and multiplication in the standard way [11].

If $\mathbf{X},\ \mathbf{Y} \in \mathbb{M}_n(S)$ we define their sum as

$$(\mathbf{X} \oplus \mathbf{Y})_{ij} \triangleq \mathbf{X}_{ij} \oplus \mathbf{Y}_{ij}.$$

If $\mathbf{A}$ is an adjacency matrix and $\mathbf{X} \in \mathbb{M}_n(S)$ we define the application of $\mathbf{A}$ to $\mathbf{X}$ as:

$$\mathbf{A}(\mathbf{X})_{ij} \triangleq \bigoplus_{0 \leq k < n} \mathbf{A}_{ik}(\mathbf{X}_{kj})$$

and the identity matrix $\mathbf{I}$ as:

$$\mathbf{I}_{ij} = \begin{cases} \overline{0} & \text{if } i = j, \\ \overline{\infty} & \text{otherwise.} \end{cases}$$

Using this notation it is easy to verify that:

$$\mathbf{X} = \mathbf{A}(\mathbf{X}) \oplus \mathbf{I}$$

is equivalent to Equation 4.

E. An iterative solution to the routing problem

The operation of distance-vector and path-vector protocols is closely modelled by the right-hand side of Equation 4. Each node chooses the best route broadcast from routes to all of its neighbours. We therefore define the matrix function σ as:

$$\sigma(\mathbf{X}) \triangleq \mathbf{A}(\mathbf{X}) \oplus \mathbf{I}$$

The result of $\sigma(\mathbf{X})$ is the state resulting from every node synchronously choosing the best extension of its neighbours' routes in state $\mathbf{X}$. A synchronous version of distance-vector protocols can therefore be modelled as repeatedly applying σ to the current state. We use $\sigma^t(\mathbf{X})$ to represent σ applied t times to $\mathbf{X}$. That is, $\sigma^0(\mathbf{X}) = \mathbf{X}$ and $\sigma^{t+1}(\mathbf{X}) = \sigma(\sigma^t(\mathbf{X}))$. The state $\sigma^t(\mathbf{X})$ is therefore the state at time t.

See [7] for an asynchronous version of σ as well as a proof that if the algebra is strictly increasing, then the synchronous and asynchronous version of σ compute the same unique routing solution.

We say σ *converges from* $\mathbf{X}$ if there exists a k such that:

$$\sigma^k(\mathbf{X}) = \sigma^{k+1}(\mathbf{X}) \tag{5}$$

and we say σ *converges* if it converges from every $\mathbf{X}$. If Equation 5 holds, then $\sigma^k(\mathbf{X})$ is a routing solution since:

$$\sigma^k(\mathbf{X}) = \sigma^{k+1}(\mathbf{X}) = \sigma(\sigma^k(\mathbf{X})) = \mathbf{A}(\sigma^k(\mathbf{X})) \oplus \mathbf{I}.$$

Why does our model allow arbitrary starting states? The network topology, as described by $\mathbf{A}$, is fixed in the model just described. However in real networks $\mathbf{A}$ changes over time as new nodes/links are added, old nodes/links are removed, and policies on links are changed. Obviously if such events continue to occur, then talking about convergence is meaningless. The protocol will only converge if there is a sufficiently long period with no changes to the network. Whenever there is a change to the network, our theory will treat that as an entirely new instance of the routing problem. However old information from the previous configuration is still present in the network and therefore we need to talk about convergence from *arbitrary states* rather than the initial state, the identity matrix $\mathbf{I}$. These arbitrary states may contain routes that are inconsistent with the new network topology.

IV. Routing algebra examples

We now present some example routing algebras to demonstrate the flexibility of the model and for use in Section V.

A. Shortest-path algebras

The most familiar examples can be constructed around shortest-path algebras of the form

$$(\mathbb{N} \cup \{\infty\},\ \min, F, \infty, 0)$$

where F is a collection of additive functions. We will use the notation f_n to denote the function $f_n(x) = x + n$. Both $\min$ and $+$ are extended in the natural way handle ∞, so for example $f_\infty(x) = x + \infty = \infty$. Here are few sets of functions that result in useful routing algebras:

$$\begin{array}{lll} F_1 & \triangleq & \{f_n \mid n \in \mathbb{N} \cup \{\infty\}\} & \text{(increasing)} \\ F_2 & \triangleq & \{f_n \mid n \in \mathbb{N} \cup \{\infty\},\ 0 < n\} & \text{(strictly increasing)} \\ F_3 & \triangleq & \{f_1,\ f_\infty\} & \text{(hop-count)} \end{array}$$

Suppose that P is some predicate over the weights in $\mathbb{N} \cup \{\infty\}$. Then g_n will denote the function $g_n(x) =$ if $P(x)$ then $f_n(x)$ else ∞.

$$F_4 \quad \triangleq \quad \{g_1,\ f_\infty\} \quad \text{(hop-count with filtering)}$$

Such conditional policies violate distributivity.

Any of these algebras could be transformed into a path-algebra using the $\mathbb{PA}$ construction.

B. Shortest-widest path algebra

Conditional policies (such as F_4 above) are not the only way to violate distributivity. This is illustrated by the shortest-widest paths algebra, $\mathbb{SWP}$.

The weights of $\mathbb{SWP}$ are of the form $(b,\ d)$ where b is bandwidth and d is length. Thus, $\overline{0} \triangleq (\infty, 0)$ is the best possible route, while $\overline{\infty} \triangleq (0, \infty)$ is the worst possible route (i.e. the weight of the non-existent paths). Weights are compared lexicographically:

$$(a,\ b) \oplus (c,\ d) \triangleq \begin{cases} (a, b) & \text{if } a = \max(a, c) \neq c \\ (c, d) & \text{if } a \neq \max(a, c) = c \\ (a, \min(b, d)) & \text{else if } a = c \end{cases}$$

That is, routes with higher bandwidth are preferred. If two routes have equal bandwidth, then ties are broken by distance.

Policy functions on arcs, $f_{c,w}$, are defined as

$$f_{c,w}(b,\ d) = (\min(c,\ b),\ w + d).$$

Here c can be interpreted as the capacity of an arc and w its length. Note that the first component $\min(c,\ b)$ represents the bandwidth of the bottleneck link.

To see that this algebra is not distributive, suppose $1 < k$. Note that $f_{1,1}((1,\ 1) \oplus (2,\ k)) \neq f_{1,1}(1,\ 1) \oplus f_{1,1}(2,\ k)$ since

$$f_{1,1}((1,\ 1) \oplus (2,\ k)) = f_{1,1}(2,\ k) = (1,\ k+1)$$
$$f_{1,1}(1,\ 1) \oplus f_{1,1}(2,\ k) = (1,\ 2) \oplus (1,\ k+1) = (1,\ 2).$$

C. *Stratified shortest paths*

We now describe a variant of the Stratified Shortest Path (SSP) algebra (see [16]). We start with the increasing routing algebra:

$$\mathbb{INC} \triangleq (\mathbb{N} \cup \{\infty\},\ \min,\ \{f \mid \forall x : x \leq f(x)\},\ 0,\ \infty),$$

The weights of $\mathbb{INC}$ are referred to as levels and policy functions by definition cannot decrease the level.

In this paper we will only use policy functions that can be represented as vectors over $\mathbb{N} \cup \{\infty\}$. The application of the function $f = \langle s_0,\ s_1,\ \ldots,\ s_k \rangle$ is defined as:

$$f(i) \triangleq \begin{cases} s_i & \text{if } i \leq k \\ \infty & \text{if } i > k \\ \infty & \text{if } i = \infty \end{cases}$$

For example:

$$\langle 0, \infty, 17, 3 \rangle(i) = \begin{cases} 0 & \text{if } i = 0 \\ \infty & \text{if } i = 1 \\ 17 & \text{if } i = 2 \\ 3 & \text{if } i = 3 \\ \infty & \text{if } i > 3 \\ \infty & \text{if } i = \infty \end{cases}$$

This algebra can implement the customer/provider/peer policies described in [17] & [18]. Imagine that the level 0 represents routes from customers, 1 represents routes from peers, and 2 represents routes from providers. Then [5] shows that the standard customer/provider/peer policy functions can be represented as:

$$\begin{array}{ll} \langle 0,\ \infty,\ \infty \rangle & \text{(towards customers)} \\ \langle 1,\ \infty,\ \infty \rangle & \text{(towards peers)} \\ \langle 2,\ \ \ 2,\ \ \ 2 \rangle & \text{(towards providers)} \end{array}$$

We can then use $\mathbb{PA}$ to add paths to $\mathbb{INC}$:

$$\mathbb{SPP} \triangleq \mathbb{PA}(\mathbb{INC}).$$

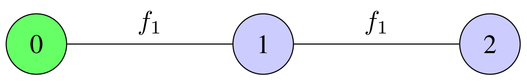
We will use $\mathbb{SPP}$ to construct an example of quadratic convergence time in Section V-C.

V. A FAMILY OF NON-LINEAR CONFIGURATIONS

We will now explore a strictly increasing path algebra and a family of network configurations Q_n that take $\Theta(n^2)$ iterations to converge. This demonstrates that the worst-case is $\Omega(n^2)$. In this section we only consider routing towards a fixed destination, node 0, and therefore ignore all other parts of the global routing state. Also, for notational brevity, we represent paths as sequences of nodes, rather the sequences of edges used elsewhere in the paper.

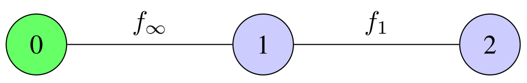
A. *Review: the classic count-to-convergence problem*

The count-to-convergence/infinity problem is a well-known problem that afflicts distance-vector routing protocols [19]. It arises when the weight on a link increases or the link fails. Consider the example in Figure 1 where nodes 1 and 2 seek to find a shortest path route to node 0. Functions f_n are defined as $f_n(x) = n + x$. In the top part of the figure the network first converges, then after link $(1,0)$ goes down nodes 1 and 2 still have what is now a "junk route" to 0 and they continually swap this route between them. If the domain is finite (RIP [19] defines ∞ as 16), then this example will converge.



	Nodes		
Time	0	1	2
0	**0**	∞	∞
1	-	**1**	∞
2	-	-	**2**

$\Downarrow$ – link $(1,0)$ goes down



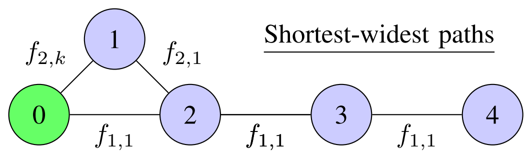
	Nodes		
Time	0	1	2
0	**0**	1	2
1	-	3	2
2	-	3	4
3	-	5	4
4	-	5	6
⋮	⋮	⋮	⋮

Fig. 1. Count-to-convergence with the shortest-paths algebra from an arbitrary state. Green cells indicate that the route has improved. Red cells indicate that the route has got worse. Bold indicates that the node has converged.

B. *Count-to-convergence induced by non-distributivity*

The underlying cause of the count-to-convergence is that nodes are exchanging "junk routes" that are not consistent with the current network configuration. Using the non-distributive shortest-widest paths algebra ($\mathbb{SWP}$) described in Section IV-B, we now show that junk routes can also arise simply from violations of distributivity (Equation 1).

Figure 2 illustrates how the violation of distributivity described in Section IV-B can generate junk routes and hence count-to-convergence, even in the absence of changes to the network topology. We assume that k is some large even number. Nodes 1 and 2 form what we call a *latch*. At time 1 node 1 *opens* the latch by adopting the route generated along the path $[1, 0]$. At the same time node 2 *loads* the route generated along path $[2, 0]$. Then at time 2 node 2 *closes* the latch by adopting the higher bandwidth route along path



Time	0	1	2	3	4
0	$(\infty, \mathbf{0})$	$(0, \infty)$	$(0, \infty)$	$(0, \infty)$	$(0, \infty)$
1	-	$(\mathbf{2}, \mathbf{k})$	$(1, 1)$	$(0, \infty)$	$(0, \infty)$
2	-	-	$(\mathbf{2}, \mathbf{k{+}1})$	$(1, 2)$	$(0, \infty)$
3	-	-	-	$(1, k{+}2)$	$(1, 3)$
4	-	-	-	$(1, 4)$	$(1, k{+}3)$
5	-	-	-	$(1, k{+}2)$	$(1, 5)$
6	-	-	-	$(1, 6)$	$(1, k{+}3)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	-	-	-	$(1, k)$	$(1, k{+}3)$
$k{+}1$	-	-	-	$(\mathbf{1}, \mathbf{k{+}2})$	$(1, k{+}1)$
$k{+}2$	-	-	-	$(1, k{+}2)$	$(\mathbf{1}, \mathbf{k{+}3})$

Fig. 2. Count-to-convergence with the shortest-widest-paths algebra from the initial state, using a distributivity violation. Red cells indicate that the route has got worse. Bold indicates that the node has converged.

$[2, 1, 0]$. But at the same time node 3 adopts the junk route through $[3, 2, 0]$ that it prefers due to the previously described distributivity violation. After time 2 the junk route (and its extensions) are exchanged back-and-forth between nodes 3 and 4. Finally, convergence is reached at time $k + 2$.

Path-vector protocols are designed to solve the count-to-convergence problem by prohibiting nodes from adopting routes with a path containing themselves. For example at time 4 the algebra $\mathbb{PA}(\mathbb{SWP})$ would prevent node 3 from extending node 4's junk route as the resulting path $[3, 4, 3, 2, 1, 0]$ would have a loop in it. Therefore whereas $\mathbb{SWP}$ requires $k + 2$ iterations to converge in this example, $\mathbb{PA}(\mathbb{SWP})$ would require only 4 iterations.

C. An example of $O(n^2)$ convergence

We now describe a family of network configurations Q_n for $n \geq 3$ designed specifically to repeatedly generate and circulate junk routes as long as possible. Q_n has n nodes, labelled 0 through $n - 1$. The network Q_n has m latches and $m - 1$ loops, where $m = \lfloor \frac{n-1}{2} \rfloor$. The i^{th} latch contains the nodes $\{2i - 1, 2i\}$ and the i^{th} loop contains the nodes $\{2i + 1, \cdots, n - 1\}$. Figure 3 illustrates the i^{th} gadget comprised of latch i and loop i.

Why does Q_n require $\Theta(n^2)$ iterations to converge? The distributivity violation of latch i generates junk routes which propagate around loop i for $O(n)$ iterations. When these junk routes are finally flushed from loop i, they trigger latch $i + 1$. As there are $O(n)$ gadgets we then obtain the required bounds.

We will now describe the operation of gadget i in detail. Let time t be the activation time of gadget i. At this time all nodes in latches 1 through $i - 1$ have converged and latch i has just been opened and loaded. That is node $2i - 1$ has opened latch i by accepting the route generated along the path

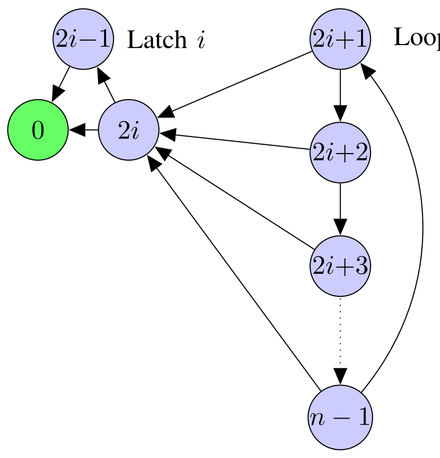


Fig. 3. The i^{th} gadget in Q_n. Junk routes are generated by the latch and then continue to circulate for $n - 2i - 1$ iterations in the loop.

$[2i - 1, 0]$ and node $2i$ has loaded the latch by accepting the route generated along the path $[2i, 0]$. Epoch i then continues:

- At time $t + 1$ node $2i$ closes the latch by adopting the path $[2i, 2i - 1, 0]$. At the same time all the nodes in loop i extend node $2i$'s route with the path $[2i, 0]$.
- At time $t + 2$ latch i is closed. Node $2i$ is offering the route generated along the path $[2i, 2i - 1, 0]$ but due to a distributivity violation, all the nodes in loop i prefer the extensions of the junk route of the path $[2i, 0]$ that was advertised at time $t + 1$. Each node in loop i will accept the extension of the junk route through its counter-clockwise neighbour. For example, node $2i + 1$ takes the path $[2i + 1, 2i + 2, 2i, 0]$, while node $n - 1$ takes the path $[n - 1, 2i + 1, 2k, 0]$.
- This process continues with each node in loop i repeatedly accepting the junk routes offered by its counter-clockwise neighbour.
- At time $t + n - 2i - 1$, it is no longer possible for this to continue as the path of the junk routes will contain every node in loop i. For example, node $2i + 1$ takes the path $[2i + 1, 2i + 2, 2i + 3, \ \cdots, \ n - 1, \ 2k, 0]$, while node $n - 1$ takes the path $[n - 1, 2i + 1, 2i + 2, \cdots n - 2, 2k, 0]$.
- Therefore at time $t + n - 2i$, these junk routes will be discarded and $i + 1^{th}$ gadget will be initialised.

Let $T(n)$ be the convergence time of Q_n. Consider the general case for Q_{n+2}, and the time taken for the 1st gadget to finish executing. It takes 1 iteration for the junk route to escape the 1st latch, and $n - 1$ iterations for the junk route to then be flushed from the 1st loop, giving a total of n iterations. Nodes 1 & 2 will then have converged. If we take these nodes out of the graph, and relabel then we are left with exactly Q_n. Hence we can formulate a recurrence relation for $T(n)$ as

$$T(3) = 2, \qquad T(4) = 3, \qquad T(n + 2) = n + T(n).$$

This is clearly quadratic by inspection. Due to parity issues, solving it is messy, but it can be simplified to:

$$T(n) = \frac{1}{4}n^2 - \frac{1}{2}n + c$$

where $c = 1$ if n is even and $c = \frac{5}{4}$ when n is odd.

i	j	Policy	Implements
1	0	$\langle 0 \rangle$	latch 1 open
2	0	$\langle 1 \rangle$	latch 1 load
2	1	$\langle 0 \rangle$	latch 1 close
3	0	$\langle 2 \rangle$	latch 2 open
3	2	$\langle \infty, 1 \rangle$	violation 1
3	4	$\langle \infty, 1 \rangle$	loop 1
4	0	$\langle 3 \rangle$	latch 2 load
4	2	$\langle \infty, 1 \rangle$	violation 1
4	3	$\langle \infty, \infty, 2 \rangle$	latch 2 close
4	5	$\langle \infty, 1 \rangle$	loop 1
5	0	$\langle 4 \rangle$	latch 3 open
5	2	$\langle \infty, 1 \rangle$	violation 1
5	4	$\langle \infty, \infty, \infty, 3 \rangle$	violation 2
5	6	$\langle \infty, 1, \infty, 3 \rangle$	loops 1, 2
6	0	$\langle 5 \rangle$	latch 3 load
6	2	$\langle \infty, 1 \rangle$	violation 1
6	4	$\langle \infty, \infty, \infty, 3 \rangle$	violation 2
6	5	$\langle \infty, \infty, \infty, \infty, 4 \rangle$	latch 3 close
6	7	$\langle \infty, 1, \infty, 3 \rangle$	loops 1, 2
7	0	$\langle 6 \rangle$	latch 4 open
7	2	$\langle \infty, 1 \rangle$	violation 1
7	4	$\langle \infty, \infty, \infty, 3 \rangle$	violation 2
7	6	$\langle \infty, \infty, \infty, \infty, \infty, 5 \rangle$	violation 3
7	8	$\langle \infty, 1, \infty, 3, \infty, 5 \rangle$	loops 1,2,3
8	0	$\langle 7 \rangle$	latch 4 load
8	2	$\langle \infty, 1 \rangle$	violation 1
8	3	$\langle \infty, 1 \rangle$	loop 1
8	4	$\langle \infty, \infty, \infty, 3 \rangle$	violation 2
8	5	$\langle \infty, \infty, \infty, 3 \rangle$	loop 2
8	6	$\langle \infty, \infty, \infty, \infty, \infty, 5 \rangle$	violation 3
8	7	$\langle \infty, \infty, \infty, \infty, \infty, 5, 6 \rangle$	loop 3 and latch 4 close

Fig. 4. Constructing the adjacency matrix of Q_9. This lists the entries of $\mathbf{B}_{ij}$. If $\mathbf{B}_{ij}$ is not listed, then $\mathbf{A}_{ij} = f_{\infty'}$. Otherwise, if $f = \mathbf{B}_{ij}$ then $\mathbf{A}_{ij} = g_{f,i,j}$ as defined in Section IV-C. The annotation "violation i" means that the policy implements one of the distributivity violations in gadget i and "loop i" means that the policy implements the loop in gadget i.

To implement this scheme, we will use the strictly increasing path algebra $\mathbb{SSP}$ defined in Section IV-C.

We now describe the construction of an adjacency matrix that implements the execution described above. From Figure 4 we can construct the adjacency matrix $\mathbf{A}_{ij}$ of Q_9. Other instances of Q_n are constructed in a similar way.

The figure lists the entries of a matrix $\mathbf{B}_{ij}$. If Figure 4 does not contain an entry for $\mathbf{B}_{ij}$, then $\mathbf{A}_{ij} = f_{\infty'}$. Otherwise, if $f = \mathbf{B}_{ij}$ then $\mathbf{A}_{ij} = g_{f,i,j}$ as defined in Section IV-C. Note that for all edges $(i, 0)$ the policy $\mathbf{B}_{i0} = \langle i-1 \rangle$ ensures that the latch of the next gadget cannot activate before the previous gadget has finished executing.

Figure 5 presents an execution trace of the 17 iterations required for the convergence of Q_9.

VI. Tight upper bounds

In the previous section we explored an example of a strictly increasing path algebra which converged in $\Theta(n^2)$ iterations. Do there exist even more pathological algebras?

We will now answer in the negative by proving that every strictly increasing path algebra converges in at most n^2 iterations. The proof technique is as follows. Consider the nodes that have already converged. For any node with the best route into this set, we show that this node will itself converge after a further $O(n)$ iterations.

In most proofs of convergence for distributive algebras [20] this next node only takes a single additional iteration to converge. However in an increasing, non-distributive algebra this may take significantly longer due to junk routes generated by distributivity violations (as described in Figure 2). The key insight is that junk routes capable of interfering with the convergence of this next node can persist for at most n iterations before necessarily containing a loop and hence being flushed.

A. Some core definitions and lemmas

We now rigorously define some of the terms used informally in previous sections. Every claim in this section has been formalised in the theorem-proving language Agda [9], but we have not included some of the simpler proofs here due to space constraints. Interested readers may consult the Agda code [10].

Let $(S, \oplus, F, \overline{0}, \overline{\infty}, path)$ be a strictly increasing path algebra and consider a network of n nodes represented by an adjacency matrix $\mathbf{A}$. Let $\mathbf{X}$ be the initial state and without loss of generality let j be the destination node.

Lemma 1. $\forall i : \sigma(\mathbf{X})_{ii} = \mathbf{I}_{ii}$. *Omitted.*

First we need a notion of an individual node converging. Our initial attempt is as follows:

Definition 3. *A node i is* fixed *at time t iff*

$$\forall s : t \leq s \Rightarrow \sigma^s(\mathbf{X})_{ij} = \sigma^t(\mathbf{X})_{ij}$$

with $\mathcal{F}_t$ being the set of fixed nodes at time t.

Lemma 2. $s \leq t \Rightarrow \mathcal{F}_s \subseteq \mathcal{F}_t$. *Omitted.*

Lemma 3. $j \in \mathcal{F}_1$. *Omitted.*

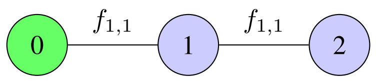
Lemma 4. *If $path(\sigma^t(\mathbf{X})_{ij}) = (i, k) :: p$ and $i \in \mathcal{F}_t$ then $\sigma^t(\mathbf{X})_{ij} = \mathbf{A}_{ik}((\sigma^t(\mathbf{X})_{kj})$ and $p = path(\sigma^t(\mathbf{X})_{kj})$.*

Proof. As $i \in \mathcal{F}_t$ we have that $\sigma^t(\mathbf{X})_{ij} = \sigma^{t+1}(\mathbf{X})_{ij}$ and so $path(\sigma^{t+1}(\mathbf{X})_{ij}) = (i, k) :: p$. The path algebra assumption P3) therefore implies that $\sigma^{t+1}(\mathbf{X})_{ij} = \mathbf{A}_{ik}(\sigma^t(\mathbf{X})_{kj})$ and that $p = path(\sigma^t(\mathbf{X})_{kj})$. Again as $i \in \mathcal{F}_t$ we have that $\sigma^t(\mathbf{X})_{ij} = \mathbf{A}_{ik}(\sigma^t(\mathbf{X})_{kj})$. ☐

Time	Node	Level	Path	Event
1	1	**0**	**1, 0**	open latch 1
1	2	1	2, 0	load latch 1
1	3	2	3, 0	
1	4	3	4, 0	
1	5	4	5, 0	
1	6	5	6, 0	
1	7	6	7, 0	
1	8	7	8, 0	
2	2	**0**	**2, 1, 0**	close latch 1
2	3	1	3, 2, 0	violation 1
2	4	1	4, 2, 0	violation 1
2	5	1	5, 2, 0	violation 1
2	6	1	6, 2, 0	violation 1
2	7	1	7, 2, 0	violation 1
2	8	1	8, 2, 0	violation 1
3	3	1	3, 4, 2, 0	
3	4	1	4, 5, 2, 0	
3	5	1	5, 6, 2, 0	junk circulates
3	6	1	6, 7, 2, 0	in loop 1
3	7	1	7, 8, 2, 0	
3	8	1	8, 3, 2, 0	
4	3	1	3, 4, 5, 2, 0	
4	4	1	4, 5, 6, 2, 0	
4	5	1	5, 6, 7, 2, 0	junk circulates
4	6	1	6, 7, 8, 2, 0	in loop 1
4	7	1	7, 8, 3, 2, 0	
4	8	1	8, 3, 4, 2, 0	
5	3	1	3, 4, 5, 6, 2, 0	
5	4	1	4, 5, 6, 7, 2, 0	
5	5	1	5, 6, 7, 8, 2, 0	junk circulates
5	6	1	6, 7, 8, 3, 2, 0	in loop 1
5	7	1	7, 8, 3, 4, 2, 0	
5	8	1	8, 3, 4, 5, 2, 0	
6	3	1	3, 4, 5, 6, 7, 2, 0	
6	4	1	4, 5, 6, 7, 8, 2, 0	
6	5	1	5, 6, 7, 8, 3, 2, 0	junk circulates
6	6	1	6, 7, 8, 3, 4, 2, 0	in loop 1
6	7	1	7, 8, 3, 4, 5, 2, 0	
6	8	1	8, 3, 4, 5, 6, 2, 0	
7	3	1	3, 4, 5, 6, 7, 8, 2, 0	
7	4	1	4, 5, 6, 7, 8, 3, 2, 0	
7	5	1	5, 6, 7, 8, 3, 4, 2, 0	junk circulates
7	6	1	6, 7, 8, 3, 4, 5, 2, 0	in loop 1
7	7	1	7, 8, 3, 4, 5, 6, 2, 0	
7	8	1	8, 3, 4, 5, 6, 7, 2, 0	
8	3	**2**	**3, 0**	open latch 2
8	4	3	4, 0	load latch 2
8	5	4	5, 0	
8	6	5	6, 0	
8	7	6	7, 0	
8	8	7	8, 0	
9	4	**2**	**4, 3, 0**	close latch 2
9	5	3	5, 4, 0	violation 2
9	6	3	6, 4, 0	violation 2
9	7	3	7, 4, 0	violation 2
9	8	3	8, 4, 0	violation 2
10	5	3	5, 6, 4, 0	
10	6	3	6, 7, 4, 0	junk circulates
10	7	3	7, 8, 4, 0	in loop 2
10	8	3	8, 5, 4, 0	
11	5	3	5, 6, 7, 4, 0	
11	6	3	6, 7, 8, 4, 0	junk circulates
11	7	3	7, 8, 5, 4, 0	in loop 2
11	8	3	8, 5, 6, 4, 0	
12	5	3	5, 6, 7, 8, 4, 0	
12	6	3	6, 7, 8, 5, 4, 0	junk circulates
12	7	3	7, 8, 5, 6, 4, 0	in loop 2
12	8	3	8, 5, 6, 7, 4, 0	
13	5	**4**	**5, 0**	open latch 3
13	6	5	6, 0	load latch 3
13	7	6	7, 0	
13	8	7	8, 0	
14	6	**4**	**6, 5, 0**	close latch 3
14	7	5	7, 6, 0	violation 3
14	8	5	8, 6, 0	violation 3
15	7	5	7, 8, 6, 0	junk circulates
15	8	5	8, 7, 6, 0	in loop 3
16	7	**6**	**7, 0**	open latch 4
16	8	7	8, 0	load latch 4
17	8	**6**	**8, 7, 0**	close latch 4

Fig. 5. A trace for the iteration of Q_9. Green cells indicate that the route has improved. Red cells indicate the route has gotten worse. The step in which a node has converged is shown in **bold**, and after that the node's state is no longer listed. In the initial state at time 0 (not listed) only node 0 has converged and all other nodes have routes ∞'. The notation "violation" stands for distributivity violation.

Perhaps counter-intuitively, if $i \in \mathcal{F}_t$ and $l \in path(\sigma^t(\mathbf{X})_{ij})$ then it is not guaranteed that $l \in \mathcal{F}_t$. This is a direct result of starting the algorithm from an arbitrary state $\mathbf{X}$ rather than from the initial state $\mathbf{I}$.



Time	Node 0	Node 1	Node 2
0	$(\infty, 0, [\,])$	$(2, 1, [(1,0)])$	$(1, 2, [(2,1),(1,0)])$
1	$(\infty, 0, [\,])$	$(1, 1, [(1,0)])$	$(1, 2, [(2,1),(1,0)])$
2	$(\infty, 0, [\,])$	$(1, 1, [(1,0)])$	$(1, 2, [(2,1),(1,0)])$

Fig. 6. Example of an unfixed node on the path of a fixed node for the shortest-widest-paths algebra. The weight of the $(1,0)$ arc has just changed from $f_{2,1}$ to $f_{1,1}$.

For example consider Fig. 6 where the bandwidth of arc $(1,0)$ has decreased from 2 to 1. Node 1 initially contains an inconsistent route, but then adopts the new valid route at time 1. Hence node 1 is not in $\mathcal{F}_0$. However the bandwidth of the arc $(2,1)$ acts as a bottleneck, and therefore node 2 never notices the change in the bandwidth of node 1's route. Hence node 2 is in $\mathcal{F}_0$ but node 1 is not.

Consequently the set $\mathcal{F}_t$ does not necessarily form a tree rooted at j and therefore to make the proof to go through (in particular Lemma 10) we need a stronger notion of a node having converged.

Definition 4. *A node i has* converged *at time t iff:*

$$\forall k \in path(\sigma^t(\mathbf{X})_{ij}) : k \in \mathcal{F}_t$$

with $\mathcal{C}_t$ being the set of converged nodes at time t.

Lemma 5. *$s \leq t \Rightarrow \mathcal{C}_s \subseteq \mathcal{C}_t$. Omitted.*

Lemma 6. *$j \in \mathcal{C}_1$. Omitted.*

Lemma 7. *If $i \in \mathcal{C}_t$ and $l \in path(\sigma^t\mathbf{X}_{ij})$ then $l \in \mathcal{C}_t$. That is, the set $\mathcal{C}_t$ forms a tree routed at j.*

Proof. We prove this by induction over the path of i's route at time t.

Cases 1 & 2: $path(\sigma^t(\mathbf{X})_{ij}) = \bot$ or $path(\sigma^t(\mathbf{X})_{ij}) = [\,]$
Then there cannot exist a node $l \in path(\sigma^t\mathbf{X}_{ij})$ and so the statement is vacuously true.

Case 3: $path(\sigma^t(\mathbf{X})_{ij}) = (i,k) :: p$
If $l \in path(\sigma^t(\mathbf{X})_{ij})$ then either $l = i$ or $l \in p$. In the former case we know $i \in \mathcal{C}_t$ and so $l \in \mathcal{C}_t$. In the latter case we use Lemma 4 to get that $path(\sigma^t(\mathbf{X})_{kj}) = p$. Therefore as $i \in \mathcal{C}_t$ we have that all nodes in p are in $\mathcal{F}_t$ and therefore we have $k \in \mathcal{C}_t$. Hence, as $l \in path(\sigma^t(\mathbf{X})_{kj})$, we can apply the induction hypothesis to prove $l \in \mathcal{C}_t$. □

In Section V we talked informally about *junk* routes. These are routes that exist in the graph but conflict with the current routing choices made by the nodes along the route's path. Consider latch 1 in the trace of Q_9 in Figure 5. At time 2, node 3's route is junk because node 3 thinks that its message will travel along the path $(3,2,0)$. However this conflicts with the routing choice of node 2 and in reality next-hop forwarding will ensure that the message travels along the path $(3,2,1,0)$.

In practice it's easier to define when a route is not junk. Intuitively a route is *real* at time t if all the routing decisions of nodes along its path are consistent with one another.

Definition 5. *A node i is* real *at time t if*

$$\forall (k,l) \in path(\sigma^t(\mathbf{X})_{ij}) : \sigma^t(\mathbf{X})_{kj} = \mathbf{A}_{kl}(\sigma^t(\mathbf{X})_{lj})$$

with $\mathcal{R}_t$ being the set of real nodes at time t.

Lemma 8. *$j \in \mathcal{R}_1$. Omitted.*

Lemma 9. *$i \in \mathcal{R}_t$ and $l \in path(\sigma^t(\mathbf{X})_{ij})$ implies $l \in \mathcal{R}_t$. Omitted.*

Lemma 10. *$\mathcal{C}_t \subseteq \mathcal{R}_t$.*

Proof. Assume $i \in \mathcal{C}_t$, then prove $i \in \mathcal{R}_t$ by induction over the path of $\sigma^t(\mathbf{X})_{ij}$ (using Lemma 4). □

If we started in the initial state $\mathbf{I}$ then we could prove a stronger version of this lemma: $\mathcal{F}_t \subseteq \mathcal{R}_t$. However this is not true when starting in arbitrary states. Figure 6 provides a counterexample as at time 2, node 2 is fixed but it is not real.

Unlike $\mathcal{F}$ and $\mathcal{C}$, the family of sets $\mathcal{R}$ does not necessarily grow monotonically over time as distributivity violations can repeatedly generate fresh junk routes.

To assist with notation we define a time-indexed family of orderings over arcs as follows:

$$(i,k) \preceq^t (l,m) \triangleq \mathbf{A}_{ik}(\sigma^t(\mathbf{X})_{kj}) \leq \mathbf{A}_{lm}(\sigma^t(\mathbf{X})_{mj})$$

Later on in Lemma 15 we will need to prove that some node l permanently adopts the route through one of its neighbour m. Intuitively if $(i,k) \preceq^t (l,m)$ then the route $\mathbf{A}_{ik}(\sigma^t\mathbf{X}_{kj})$ and some of its extensions may be capable of threatening l's adoption of the route through m at some point in the future.

B. The inductive step of the proof

The thrust of the proof is to keep track of $P(t)$, the set of nodes that we can *prove* have converged at time t. This is a (possibly strict) subset of $\mathcal{C}_t$ the nodes that have *actually* converged at time t. We will then show that while there exist nodes not in $P(t)$, then we can prove that at least one of those nodes will have converged by time $t+n$. This node can then be added to $P(t+n)$, the set of nodes proven to have converged at time $t+n$. The whole process can then be repeated and this forms the crux of the inductive argument for convergence.

Consider a time $\tau \geq 1$ and assume $j \in P(\tau)$. Denote $P(\tau)$'s complement as $\overline{P(\tau)}$. Let $E(\tau)$ be the set of all arcs going from $\overline{P(\tau)}$ to $P(\tau)$ (i.e. the in-cutset of $P(\tau)$) and $e_{\min}(\tau) = (i_{\min}(\tau), k_{\min}(\tau))$ be a minimal arc in $E(\tau)$ with respect to $\preceq^\tau$ (see Figure 7). We will now prove that $i_{\min}(\tau) \in \mathcal{C}_{\tau+n}$ and therefore $i_{\min}(\tau)$ can added as a member of $P(\tau+n)$.

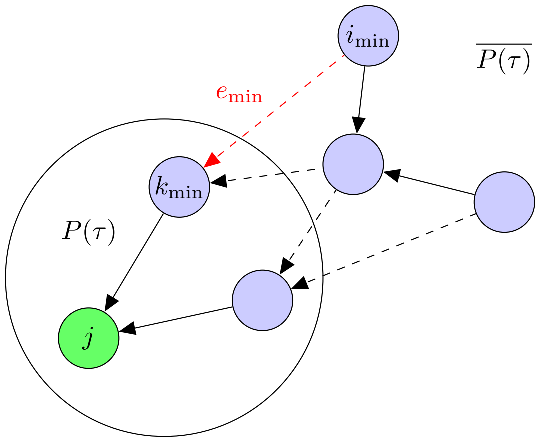


Fig. 7. Construction of the arc $e_{\min}(\tau) = (i_{\min}, k_{\min})$. $P(\tau)$ is the set of nodes we can prove have converged at time τ. Dashed arcs are members of $E(\tau)$, the cutset of $P(\tau)$. The arc $e_{\min}$ is a minimal member of $E(\tau)$ with respect to $\preceq^{\tau}$.

In order to improve readability, we will now drop $e_{\min}(\tau)$, $i_{\min}(\tau)$ and $k_{\min}(\tau)$'s dependencies on τ and just write $e_{\min}$, $i_{\min}$ and $k_{\min}$.

What can stop node $i_{\min}$ from routing through $k_{\min}$? In the distributive world nothing. The node $i_{\min}$ uses a route from either $P(\tau)$ or $\overline{P(\tau)}$. The route through $k_{\min}$ is by definition the best route from $P(\tau)$, and any route from $\overline{P(\tau)}$ must go through $P(\tau)$. Hence distributivity ensures that the route through $\overline{P(\tau)}$ cannot be better than the route through $k_{\min}$.

However non-distributive algebras can generate junk routes before time τ, that can continue to persist in $\overline{P(\tau)}$ after time τ. These junk routes can prevent the adoption of $e_{\min}$. We now prove that a real node in $\overline{P(\tau)}$ can never threaten $e_{\min}$.

Lemma 11. *For all $t \geq \tau$ if $k \in \mathcal{R}_t$ and $k \in \overline{P(\tau)}$ then for all nodes i we have $e_{\min} \preceq^t (i,k)$.*

Proof. We proceed by induction over the path of $\sigma^t(\mathbf{X})_{kj}$.

Case 1: $path(\sigma^t(\mathbf{X})_{kj}) = [\,]$
If the path is empty then we must have that $k = j$, and so we have $j \notin P(\tau)$. This contradicts the assumption at the start of Section VI-B that $j \in P(\tau)$.

Case 2: $path(\sigma^t(\mathbf{X})_{kj}) = \bot$
Then we have the required inequality as follows:

$$\begin{array}{lll} \mathbf{A}_{i_{\min}k_{\min}}(\sigma^t(\mathbf{X})_{k_{\min}j}) & \leq & \text{(by R5))} \\ \infty & = & \text{(by R6))} \\ \mathbf{A}_{ik}(\infty) & = & \text{(by P1) and case assumption)} \\ \mathbf{A}_{ik}(\sigma^t(\mathbf{X})_{kj}) & & \end{array}$$

Case 3: $path(\sigma^t(\mathbf{X})_{kj}) = (k,l) :: p$
If $l \in P(\tau)$ then (k,l) is an element of the cutset E and so we have that $e_{\min} \preceq^t (k,l)$ as $e_{\min}$ is a minimal member of E. If $l \notin P(\tau)$ then we have that $e_{\min} \preceq^t (k,l)$ by applying the inductive hypothesis as $l \in \mathcal{R}_t$ (from Lemma 9). Therefore either way $e_{\min} \preceq^t (k,l)$.

Using this we can then show that the required inequality holds as follows:

$$\begin{array}{lll} \mathbf{A}_{i_{\min}k_{\min}}(\sigma^t(\mathbf{X})_{k_{\min}j}) & \leq & \text{(by } e_{\min} \preceq^t (k,l)) \\ \mathbf{A}_{kl}(\sigma^t(\mathbf{X})_{lj}) & = & \text{(by } k \in \mathcal{R}_t) \\ \sigma^t(\mathbf{X})_{kj} & \leq & \text{(by } F \text{ increasing)} \\ \mathbf{A}_{ik}(\sigma^t(\mathbf{X})_{kj}) & & \end{array}$$

□

Therefore the nodes that are capable of blocking $i_{\min}$ from routing through $e_{\min}$, are those that are not real and that have an extension that threatens $e_{\min}$.

Definition 6. *A node k's route is* dangerous *at time $t \geq \tau$ if:*

$$k \notin \mathcal{R}_t \wedge \exists i : (i,k) \prec^t e_{\min}$$

where $\mathcal{D}^{\tau}_t$ is the set of dangerous nodes at time $t \geq \tau$.

It is important to note that routes are labelled as dangerous *relative to the current minimal arc*. We will now prove that after time τ all dangerous routes must be an extension of some other dangerous route. Hence new dangerous routes cannot be generated after time τ.

Lemma 12. *For all $t \geq \tau$ and $k \in \mathcal{D}^{\tau}_{t+1}$ then there exists l such that $\sigma^{t+1}(\mathbf{X})_{kj} = \mathbf{A}_{kl}(\sigma^t(\mathbf{X})_{lj})$ and $l \in \mathcal{D}^{\tau}_t$.*

Proof. From the definition of $\mathcal{D}^{\tau}_{t+1}$ we know that $k \notin \mathcal{R}_{t+1}$. As the routes $\overline{\infty}$ and $\overline{0}$ are trivially real, k's route cannot be either $\overline{\infty}$ or $\overline{0}$. Therefore $\sigma^{t+1}(\mathbf{X})_{kj} = \mathbf{A}_{kl}(\sigma^t(\mathbf{X})_{lj})$ for some l. We now need to show that $l \in \mathcal{D}^{\tau}_t$ by proving $(k,l) \prec^t e_{\min}$ and $l \notin \mathcal{R}_t$.

As $k \in \mathcal{D}^{\tau}_{t+1}$ we know there exists a node i such that $(i,k) \prec^{t+1} e_{\min}$. We therefore have the following chain of reasoning to prove that $(k,l) \prec^t e_{\min}$:

$$\begin{array}{lll} \mathbf{A}_{kl}(\sigma^t(\mathbf{X})_{lj}) & = & \text{(by prev. reasoning)} \\ \sigma^{t+1}(\mathbf{X})_{kj} & \leq & \text{(by } F \text{ increasing)} \\ \mathbf{A}_{ik}(\sigma^{t+1}(\mathbf{X})_{kj}) & < & \text{(by } (i,k) \prec^{t+1} e_{\min}) \\ \mathbf{A}_{i_{\min}k_{\min}}(\sigma^{t+1}(\mathbf{X})_{k_{\min}j}) & = & \text{(by } k_{\min} \in \mathcal{C}_t) \\ \mathbf{A}_{i_{\min}k_{\min}}(\sigma^t(\mathbf{X})_{k_{\min}j}) & & \end{array}$$

It remains to show that $l \notin \mathcal{R}_t$. Clearly $k \notin P(\tau)$ as that would contradict $k \notin \mathcal{R}_{t+1}$ by Lemma 10. If $l \in P(\tau)$ then (k,l) is an element of the cutset E and so we have that $e_{\min} \preceq^{t+1} (k,l)$ as $e_{\min}$ is a minimal member of E. If $l \notin P(\tau)$ then we have that $e_{\min} \preceq^{t+1} (k,l)$ by applying Lemma 11. Therefore either way $e_{\min} \preceq^{t+1} (k,l)$.

Assume $l \in \mathcal{R}_t$. Then we have a contradiction as we have assumed that $(i,k) \prec^{t+1} e_{\min}$ but we can show $e_{\min} \preceq^{t+1} (i,k)$ as follows:

$$\begin{array}{lll} \mathbf{A}_{i_{\min}k_{\min}}(\sigma^{t+1}(\mathbf{X})_{k_{\min}j}) & = & \text{(by } k_{\min} \in \mathcal{C}_t) \\ \mathbf{A}_{i_{\min}k_{\min}}(\sigma^t(\mathbf{X})_{k_{\min}j}) & \leq & \text{(by } e_{\min} \preceq^{t+1} (k,l)) \\ \mathbf{A}_{kl}(\sigma^t(\mathbf{X})_{lj}) & = & \text{(by prev. reasoning)} \\ \sigma^{t+1}(\mathbf{X})_{kj} & \leq & \text{(by } F \text{ increasing)} \\ \mathbf{A}_{ik}(\sigma^{t+1}(\mathbf{X})_{kj}) & & \end{array}$$

Hence $l \notin \mathcal{R}_t$ and so $l \in \mathcal{D}^{\tau}_t$. □

We now prove that all dangerous routes must have been flushed by time $t + n$.

Lemma 13. $\mathcal{D}^{\tau}_{\tau+n} = \{\}$

Proof. By Lemma 12 we have that the length of the paths of the dangerous routes must increase by 1 each iteration. As routes formed along looping paths are removed, then after n iterations all such dangerous routes must have been eliminated from the routing state. Hence there can no longer be any dangerous junk routes at time $\tau + n$. □

We now prove that at any time after $\tau+n$ the route through $k_{\min}$ is guaranteed to be the best route available to $i_{\min}$.

Lemma 14. *For every node k and time $s \geq \tau + n$ then:*

$$(i_{\min}, k_{\min}) \preceq^s (i_{\min}, k)$$

Proof. Consider an arbitrary k. If $k \in P(\tau)$ then we have the required inequality by the fact that $e_{\min}$ is a minimal arc into $P(\tau)$. If $k \notin \mathcal{R}_s$ then we must have the required inequality otherwise there would exist a dangerous route which would contradict Lemma 13.

Finally if $k \notin P(\tau)$ and $k \in \mathcal{R}_s$ then we have the required inequality by Lemma 11. □

Hence we can now show that at any time $s \geq \tau + n$ then $i_{\min}$ routes through $k_{\min}$.

Lemma 15. *For every time $s \geq \tau + n$ then :*

$$\sigma^s(\mathbf{X})_{i_{\min}j} = \mathbf{A}_{i_{\min}k_{\min}}(\sigma^{s-1}(\mathbf{X})_{k_{\min}j})$$

Proof. As $i_{\min} \notin P(\tau)$ and $j \in P(\tau)$ then clearly $i_{\min} \neq j$. Hence we have as follows:

$$\begin{array}{lcl}
\sigma^s(\mathbf{X})_{i_{\min}j} & = & \text{(by definition of } \sigma) \\
\bigoplus_k \mathbf{A}_{i_{\min}k}(\sigma^{s-1}(\mathbf{X})_{kj}) \oplus \mathbf{I}_{i_{\min}j} & = & \text{(by definition of } \mathbf{I}) \\
\bigoplus_k \mathbf{A}_{i_{\min}k}(\sigma^{s-1}(\mathbf{X})_{kj}) \oplus \overline{\infty} & = & \text{(by R5))} \\
\bigoplus_k \mathbf{A}_{i_{\min}k}(\sigma^{s-1}(\mathbf{X})_{kj}) & = & \text{(by Lemma 14)} \\
\mathbf{A}_{i_{\min}k_{\min}}(\sigma^{s-1}(\mathbf{X})_{k_{\min}j}) & &
\end{array}$$

□

Lemma 16. $i_{\min} \in \mathcal{C}_{\tau+n}$

Proof. We first prove that $i_{\min} \in \mathcal{F}_{\tau+n}$. Consider an arbitrary $s \geq \tau + n$. then:

$$\begin{array}{lcl}
\sigma^{\tau+n}(\mathbf{X})_{i_{\min}j} & = & \text{(by Lemma 15)} \\
\mathbf{A}_{i_{\min}k_{\min}}(\sigma^{\tau+n-1}(\mathbf{X})_{k_{\min}j}) & = & \text{(by } k_{\min} \in \mathcal{C}_\tau) \\
\mathbf{A}_{i_{\min}k_{\min}}(\sigma^{s-1}(\mathbf{X})_{k_{\min}j}) & = & \text{(by Lemma 15)} \\
\sigma^s(\mathbf{X})_{i_{\min}j} & &
\end{array}$$

For any $s \geq \tau + n$ we know that $i_{\min}$ routes through k_{min} which is in C. As $C \subseteq \mathcal{C}_\tau \subseteq \mathcal{C}_{\tau+n} \subseteq \mathcal{F}_{\tau+n}$, we therefore know that all nodes in the path of k are in $\mathcal{F}_{\tau+n}$. Hence all nodes in the path of i are in $\mathcal{F}_{\tau+n}$, and so $i \in \mathcal{C}_{\tau+n}$. □

Theorem 1. *For any strictly increasing path algebra $(S, \oplus, F, \overline{0}, \overline{\infty})$ and network of n nodes then σ converges in at most n^2 iterations from every starting state $\mathbf{X}$.*

Proof. Let $P(t)$ be the set of nodes we can *prove* have converged at time t. Clearly $P(0) = \emptyset$. At time 1 we know by Lemma 6 that j has converged and so let $P(1) = \{j\}$.

By fixing the current time as τ, we can repeatedly generate a new node $i_{\min}(\tau)$ not yet in $P(\tau)$ via the process described in Section VI-B. Lemma 16 proves that $i_{\min}(\tau)$ will have converged at time $\tau + n$, thus $P(\tau + n) = P(\tau) \cup \{i_{\min}(\tau)\}$.

By repeating the process above $n-1$ times we get that after time then $P(n(n-1)+1)$ will contain every node and so the network is guaranteed to have converged. □

VII. Open questions

Not all strictly increasing path algebras converge in $\Theta(n^2)$. For example the shortest-widest paths algebra is not distributive, yet it appears to always to converge in $2n-2$ iterations. It seems likely there are sub-families of strictly increasing path algebras that have distinct worst-case convergence bounds. In some cases it might be feasible to further constrain the policies in order to ensure faster convergence.

Acknowledgment

Matthew Daggitt is supported by an EPSRC Doctoral Training grant. The authors thank the anonymous reviewers and our shepherd, Lachlan Andrew, for their helpful feedback.

References

[1] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," RFC 4271, 2006.

[2] T. G. Griffin and G. Huston, "BGP wedgies," RFC 4264, 2005.

[3] D. McPherson, V. Gill, D. Walton, and A. Retana, "Border gateway protocol (BGP) persistent route oscillation condition," RFC 3345, 2002.

[4] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," *Comp. Networks*, vol. 32, no. 1, pp. 1–16, 2000.

[5] J. L. Sobrinho, "An algebraic theory of dynamic network routing," *Transactions on Networking (TON)*, vol. 13, no. 5, pp. 1160–1173, 2005.

[6] M. Gondran and M. Minoux, *Graphs, dioids and semirings: new models and algorithms*, vol. 41. Springer Science & Business Media, 2008.

[7] M. L. Daggitt, A. J. T. Gurney, and T. G. Griffin, "Asynchronous convergence of policy-rich distributed bellman-ford routing protocols," in *SIGCOMM proceedings*, ACM, 2018.

[8] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking (ToN)*, vol. 10, no. 2, pp. 232–243, 2002.

[9] A. Bove, P. Dybjer, and U. Norell, "A brief overview of agda – a functional language with dependent types," in *Theorem Proving in Higher Order Logics*, pp. 73–78, Springer Berlin Heidelberg, 2009.

[10] M. L. Daggitt, R. Zmigrod, and T. G. Griffin, "Agda routing library," 2018. https://github.com/MatthewDaggitt/agda-routing/tree/sigcomm2018.

[11] J. S. Baras and G. Theodorakopoulos, *Path problems in networks*. Morgan & Claypool Publishers, 2010.

[12] H. Karloff, "On the convergence time of a path-vector protocol," in *ACM-SIAM symposium on Discrete algorithms*, pp. 605–614, 2004.

[13] A. Üresin and M. Dubois, "Parallel asynchronous algorithms for discrete data," *Journal of the ACM (JACM)*, vol. 37, no. 3, pp. 588–606, 1990.

[14] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed internet routing convergence," pp. 175–187, 2000.

[15] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*, vol. 2. Prentice-Hall International New Jersey, 1992.

[16] T. G. Griffin, "Exploring the stratified shortest-paths problem," *Networking Science*, vol. 1, no. 1, pp. 2–14, 2011.

[17] G. Huston, "Interconnection, peering and settlements: Parts I & II," *Internet Protocol Journal (Cisco)*, vol. 2, June 1999.

[18] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 6, pp. 681–692, 2001.

[19] C. Hendrick, "Routing information protocol (RIP)." RFC 1058, 1988.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.