On the Forwarding Paths Produced by Internet Routing Algorithms

Seweryn Dynerowicz
Faculty of Computing Science
University of Namur
Email: seweryn.dynerowicz@unamur.be

Timothy G. Griffin
Computer Laboratory
University of Cambridge
Email: tgg22@cam.ac.uk

***Abstract*—Most Internet routing protocols have one of two algorithms lurking at their core — either Dijkstra's algorithm in the case of *link-state* protocols or a distributed Bellman-Ford algorithm in the case of *distance-vector* or *path-vector* protocols. When computing simple shortest paths these protocols can be modified to utilize *all* best paths with a combination of next-hop sets and Equal Cost Multi-Path (ECMP) forwarding. We show that this picture breaks down even for simple modifications to the shortest path metric. This is illustrated with *widest-shortest paths* where among all shortest paths only those with greatest bandwidth are considered best. In this case Bellman-Ford and Dijkstra may compute different sets of paths and neither can compute all best paths. In addition, some paths computed by Dijkstra's algorithm cannot be implemented with next-hop forwarding. We provide a general algebraic model that helps to clarify such anomalies. This is accomplished by computing paths *within the route metric* rather than with specialized algorithmic extensions. Our results depend on the distinction between global and local optima that has hitherto been applied almost exclusively to more exotic routing protocols such as BGP.**

I. Introduction

Most widely deployed Internet routing protocols have one of two algorithms lurking at their core — either Dijkstra's algorithm in the case of *link-state* protocols or a distributed Bellman-Ford algorithm in the case of *distance-vector* or *path-vector* protocols. When restricted to a shortest-path metric these protocols easily accommodate the computation of *all* optimal paths and each path set can be efficiently implemented with next-hop sets together with Equal Cost Multi-Path (ECMP) forwarding (a data-plane method of distributing traffic among available next-hops).

There has also been considerable interest in enhancing the metrics of intra-domain Internet routing protocols to capture more that just distance (such as delay). On one end of the spectrum are attempts to find heuristics for the general *QoS routing problem* [1], [2], [3] — finding paths that simultaneously satisfy multiple independent constraints — which is known to be NP-complete [4], [5]. At the other end of the spectrum we find proposals for extending existing Internet routing protocols with enhanced *QoS metrics*. [6], [7].

Several papers have addressed the combination of ECMP and QoS metrics [7], [8]. However, in the current paper we focus on the following question which to our knowledge has not been addressed in the literature.

Main questions. If standard Internet routing protocols are extended with enhanced QoS metrics, then which set of optimal paths can they compute? Out of those computed optimal paths which can be implemented with hop-by-hop forwarding?

Our enhanced metrics will be captured in structures, called *bounded semirings*, of the form $(S,\ \oplus,\ \otimes)$ [9], [10], [11], [12], [13]. Informally, $\oplus$ is used to compute best path weights while $\otimes$ is used to compute the weight of a path from the arc weights. For standard shortest path routing, $\oplus = \min$ and $\otimes = +$ (see Section III). Assume that the link weights of a network are represented by an adjacency matrix $\mathbf{A}$, where $\mathbf{A}(i,\ j)$ is the weight of arc $(i,\ j)$. We present version of distributed Bellman-Ford and Dijkstra's algorithm that have been suitably modified to support semiring metrics.

Denote the set of optimally weighted paths in the network by *optimal-paths*$(\mathbf{A})$. For each algorithm denote the largest set of paths that naturally arise from the algorithm as *computed-paths*$(\mathbf{A})$ and by *next-hop-paths*$(\mathbf{A})$ the subset of the computed paths that can be implemented with next-hop forwarding.

Main results. We can now state our main results.

For distributed Bellman-Ford:

$$\begin{aligned} \textit{next-hop-paths}(\mathbf{A}) &= \textit{computed-paths}(\mathbf{A}) \\ &\subseteq \textit{optimal-paths}(\mathbf{A}) \end{aligned}$$

For Dijkstra's algorithm:

$$\begin{aligned} \textit{next-hop-paths}(\mathbf{A}) &\subseteq \textit{computed-paths}(\mathbf{A}) \\ &\subseteq \textit{optimal-paths}(\mathbf{A}) \end{aligned}$$

All of these path sets coincide exactly when the metric is *cancellative*. That is, when $a \otimes b = a \otimes c$ always implies that $b = c$.

Without cancellativity, distributed Bellman-Ford and Dijkstra's algorithm may compute different sets of paths, and neither is guaranteed to yield all best paths. In addition, while the Bellman-Ford paths are next-hop implementable, the paths associated with Dijkstra's algorithm may not be. That is, in order to forward along any path implicit in Dijkstra's algorithm (already not all optimal paths in some cases), we either need to restrict our attention to the subset of those paths that are next-hop implementable or we employ some form of tunneling.

We illustrate these results in Section II using "widest-shortest paths" [3], [7] where among all shortest paths only those paths with maximal bandwidth are considered to be best paths. This composite metric is not cancellative since the $\min$

associated with finding a bottleneck link is not cancellative (for example, $2 \min 3 = 2$ and $2 \min 4 = 2$, but $3 \neq 4$).

Our approach. Given an adjacency matrix $\mathbf{A}$, the *globally optimal* solution to the path problem is a matrix $\mathbf{A}^*$ such that

$$\mathbf{A}^*(i,\ j) = \text{best path weight over all paths } i \text{ to } j. \quad (1)$$

In this framework, $\mathbf{A}^*$ solves for $\mathbf{L}$ the (left) equation

$$\mathbf{L} = (\mathbf{A} \otimes \mathbf{L}) \oplus \mathbf{I} \quad (2)$$

and also solves for $\mathbf{R}$ in the (right) equation

$$\mathbf{R} = (\mathbf{R} \otimes \mathbf{A}) \oplus \mathbf{I}. \quad (3)$$

where $\oplus$ and $\otimes$ have been lifted in a natural way to matrix operations and $\mathbf{I}$ is the multiplicative identity (see Section III). This classical framework depends on the distributivity of $\otimes$ over $\oplus$. That is, $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$. However, recent work has shown that without distributivity it may be that solutions to Equations (1)-(3) exist but are all distinct [14]. Furthermore, in some cases where distributivity fails solutions to Equation (2) can be computed by distributed Bellman-Ford while solutions to Equation (3) can be computed by Dijkstra's algorithm (one row at a time). In fact these two equations can serve as specifications for the solutions which are computed by *distance-vectoring* and *link-state* respectively. Informally, $\mathbf{L}(i,\ j)$ represents the best path to j that i can obtain given the best paths at i's immediate neighbors. Those familiar with BGP [15] will recognize this as the kind of solution implicit in that protocol, which operates in an environment of autonomous systems where global optimality may have no real meaning. On the other hand, $\mathbf{R}(i,\ j)$ represents the best path to j that i can obtain given it's best paths to j's immediate neighbors. In both cases these paths may differ from the globally optimal paths and so we call $\mathbf{L}$ and $\mathbf{R}$ the left- and right-locally optimal solutions.

$$\begin{pmatrix} \text{original metric} \\ + \\ \text{complex algorithm} \end{pmatrix} \rightarrow \begin{pmatrix} \text{modified metric} \\ + \\ \text{generic algorithm} \end{pmatrix}$$

Fig. 1: The Algorithm to Algebra (A2A) method attempts to shift complexity from an algorithm to the metric, which is captured in an algebraic structure such as a semiring.

We next apply what we call the *Algorithm to Algebra* (A2A) method, illustrated in Figure 1. This method attempts to remove elements from an algorithm and fold them into the metric so that a generic algorithm can be used. Using the A2A method it may be easier to reveal the problem being solved by the original metric/algorithm (global optimality, left- or right-local optimality). Among other applications, this method has been used to model eBGP vs. iBGP [16], areas in OSPF and levels in ISIS, route redistribution and administrative distance [17], [18].

In this paper we apply this method to versions of Bellman-Ford and Dijkstra that compute paths or next-hop sets. We fold the computation of paths or next-hop sets into the metric (Section IV). This is accomplished in a simple way using a lexicographic product, and using the results of [16] we show that unless the original metric is cancellative the modified metric will not be distributive and Equations (1)-(3) may have distinct solutions. This application of the A2A method relies on the use of the general semiring model where $\oplus$ is associated with a partial order rather than being restricted to a total order as in [7].

II. Algorithms

In this section we explore the forwarding paths naturally produced by Dijkstra's algorithm and distributed Bellman-Ford.

A. Routing Metrics

Standard shortest-path algorithms rely on two binary operations, $\min$ for comparison of path weights, and $+$ for computing the weight of a path from its arc weights. If we do not consider negative weights, then we have a minimal element 0 and a maximal element ∞ used as a weight when there is no path (addition is extended so that $a + \infty = \infty + a = \infty$). There is a long history [9], [19], [11], [12], [13] of generalizing shortest paths to structures called *semiring*[1], which have the form

$$(S,\ \oplus,\ \otimes,\ \overline{0}\ , \overline{1}).$$

We postpone to the next section a more detailed account of semirings, and present here only a few simple examples that have proven useful in network routing.

name	S	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$	paths
sp	$\mathbb{N}^\infty$	min	$+$	∞	0	shortest
bw	$\mathbb{N}^\infty$	max	min	0	∞	widest
rel	$[0,\ 1]$	max	$\times$	0	1	most reliable

Here $\mathbb{N}^\infty \equiv \mathbb{N} \cup \{\infty\}$ The abstract notation $(S, \oplus, \otimes, \overline{0}, \overline{1})$ is derived from the familiar *ring* of real numbers $(\mathbb{R}, +, \times, 0, 1)$, which might initially lead to some confusion for shortest paths where $\oplus = \min$ and $\otimes = +$. Semirings differ from rings in that the additive operation need not admit inverses. That is, $(S, \oplus)$ is only required to be a monoid, not a group. This allows us to define a non-trivial "natural" order on S:

$$a \leq_\oplus b \equiv a = a \oplus b.$$

We will also use the notation

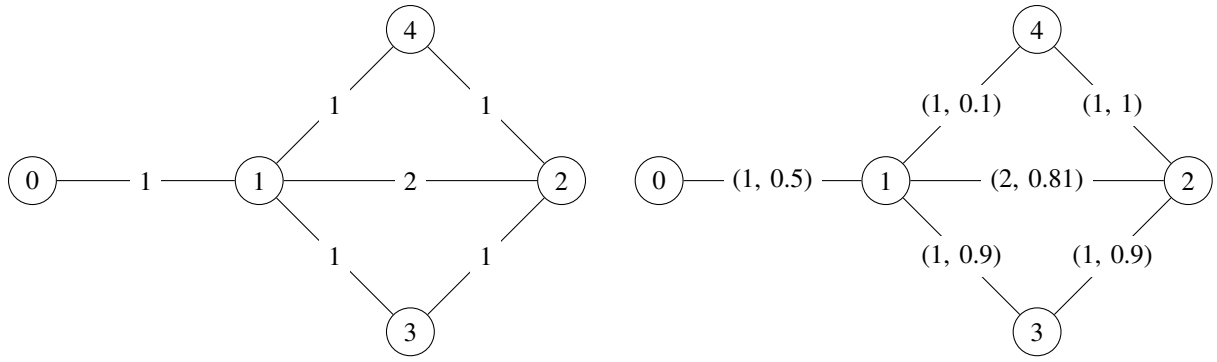
$$a <_\oplus b \equiv a = a \oplus b \neq b.$$

We write $a \leq b$ and $a < b$ when the operation $\oplus$ is clear from context. In the examples listed above this order is *total* — it is always true that either $a \leq b$ or $b \leq a$. It is important to note that this will not be the case for semirings in general, where $\leq$ is only a partial order.

A graph $G = (V, E)$ weighted over a semiring S can be represented by an adjacency matrix $\mathbf{A}$, where $\mathbf{A}(i, j) = \overline{0}$ exactly when $(i, j) \notin E$. The null path ϵ is given the weight $\overline{1}$, while a path

$$p = (v_1, v_2)(v_2, v_3) \cdots (v_k, v_{k+1}) \quad (4)$$

[1] We cite the first edition [11] since later editions sadly omitted the semiring generalization of shortest-paths.



(a) Weights represent distance. **(b)** Weights (d, r) represent distance d and reliability r.

Fig. 2: Simple example networks for (a) shortest paths and (b) most reliable shortest-paths.

is given weight

$$\mathbf{A}(v_1, v_2) \otimes \mathbf{A}(v_2, v_3) \otimes \cdots \otimes \mathbf{A}(v_k, v_{k+1}).$$

Such a path is *elementary* (or *loop free*) when $k + 1 = |\{v_1, v_2, \cdots v_k, v_{k+1}\}|$. Given $\mathbf{A}$, we define the matrix of globally optimal path weights as

$$\mathbf{A}^*(i,\ j) \equiv \bigoplus_{p \in \mathcal{P}(i,j)} w(p), \tag{5}$$

where $\mathcal{P}(i, j)$ represents the set of all paths from i to j. This matrix may not be well-defined for some semirings, but all of the examples presented above are *bounded semirings* and $\mathbf{A}^*$ is guaranteed to exist (see the next section).

Consider the shortest-paths semiring and the weighted graph of Figure 2 (a), which can be represented with the following adjacency matrix.

$$\mathbf{A} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} \infty & 1 & \infty & \infty & \infty \\ 1 & \infty & 2 & 1 & 1 \\ \infty & 2 & \infty & 1 & 1 \\ \infty & 1 & 1 & \infty & \infty \\ \infty & 1 & 1 & \infty & \infty \end{bmatrix} \end{array} \tag{6}$$

Here it is easy to (visually) check that

$$\mathbf{A}^* = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 3 & 2 & 2 \\ 1 & 0 & 2 & 1 & 1 \\ 3 & 2 & 0 & 1 & 1 \\ 2 & 1 & 1 & 0 & 2 \\ 2 & 1 & 1 & 2 & 0 \end{bmatrix} \end{array} \tag{7}$$

is the solution to the all pairs shortest paths problem.

B. Algorithms

If G represents a network of routers, then we can think of $\mathbf{A}^*$ as resulting from the collective effort of each node i computing row $\mathbf{A}^*(i, \_)$ within some type of distributed computation (assuming the network has stabilized). In protocols such as OSPF and IS-IS each node j floods its *link state* $\{((j, v), A(j, v)) \mid (j, v) \in E\}$ to all other nodes in a network so that each node i can construct a representation of $\mathbf{A}$ and compute the row $\mathbf{A}^*(i, \_)$. Typically this computation is done with some version of Dijkstra's algorithm. Another common approach (BGP, RIP) is to use a version of the distributed Bellman-Ford algorithm where link-states are known only locally while the *computation* is distributed [20].

Needless to say, in network routing we are not just interested in computing the weights of shortest paths as captured in $\mathbf{A}^*$. After all, the whole point of a routing protocol is to populate forwarding tables in such a way that traffic can be forwarded along best paths. There are various ways of augmenting Dijkstra's algorithm and Bellman-Ford to record a shortest path or a set thereof. For example, Cormen et. al. [11] record a predecessor node $\pi(i, j) \in V$ that is a last-hop on some shortest path from node i to node j. Relying on this information is cumbersome in an IP-like next-hop forwarding context, given that node i needs to work its way backward, from the destination j through the successive predecessors to identify the next-hop to forward to. Alternatively, it is easy to modify that version of the algorithm so that $\pi(i, j)$ represents a next-hop or a set thereof, for shortest paths from node i to node j.

We present versions of these algorithms where $\pi(i, j)$ is a set of paths from i to j. Of course this may not be practical since the size of path sets can grow exponentially in the size of the network. However, presenting the more general algorithm allows us to more easily explore which path sets may or may not be implemented efficiently, say by next-hop sets together with ECMP forwarding.

Let $\mathrm{epaths}(E)$ represent the set of elementary (loop free) paths over E. If X and Y are subsets of $\mathrm{epaths}(E)$, then we define

$$X \diamond Y \equiv \{pq \in \mathrm{epaths}(E) \mid p \in X,\ q \in Y\}. \tag{8}$$

That is, $X \diamond Y$ represents the pair-wise concatenation of paths in X and Y that result in elementary paths.

Figure 3 (a) presents a generalized Dijkstra's algorithm, while Figure 3 (b) presents a (model of) distributed Bellman-Ford. The results of these computations are $\mathbf{D} \equiv \mathbf{D}_{n-1}$ and $\pi \equiv \pi_{n-1}$. Both algorithms are generalized for semiring metrics and compute best weights in matrix $\mathbf{D}$ and associated

```
DIJKSTRA_GENERIC_WITH_PATHS(⊕, ⊗, 0̄, 1̄, A, i)
(1)  for each q ∈ V do D_0(i, q) ← 0̄; π_0(i, q) ← {}
(2)  S_0 ← {}; D_0(i, i) ← 1̄; π_0(i, i) ← {ε}
(3)  for each k = 1, 2, ..., |V| do
(4)        find q_k ∈ V − S_{k−1} with D_{k−1}(i, q_k) minimal w.r.t. ≤_⊕
(5)        S_k ← S_{k−1} ∪ {q_k}
(6)        for each j ∈ V − S_k do
(7)            if D_{k−1}(i, q_k) ⊗ A(q_k, j) = D_{k−1}(i, j)
(8)            then π_k(i, j) ← π_{k−1}(i, j) ∪ (π_{k−1}(i, q_k) ⋄ {(q_k, j)})
(9)            else if D_{k−1}(i, q_k) ⊗ A(q_k, j) <_⊕ D_{k−1}(i, j)
(10)           then D_k(i, j) ← D_{k−1}(i, q_k) ⊗ A(q_k, j);
(11)                π_k(i, j) ← π_{k−1}(i, q_k) ⋄ {(q_k, j)}
```

(a) Generalized Dijkstra's Algorithm, with explicit path computation.

```
BELLMANFORD_GENERIC_WITH_PATHS(⊕, ⊗, 0̄, 1̄, A)
(1)  for each i, j ∈ V do D_0(i, j) ← 0̄; π_0(i, j) ← {}
(2)  D_0(i, i) ← 1̄; π_0(i, i) ← {ε}
(3)  for k ← 1 to |V| − 1 do
(4)      for each i ∈ V do
(5)          for each j ∈ V do
(6)              if i = j
(7)              then D_k(i, i) ← 1̄; π_k(i, i) ← {ε}
(8)              else D_k(i, j) ← 0̄; π_k(i, j) ← {}
(9)                   for each q ∈ V do
(10)                      if A(i, q) ⊗ D_{k−1}(q, j) = D_k(i, j)
(11)                      then π_k(i, j) ← π_k(i, j) ∪ ({(i, q)} ⋄ π_{k−1}(q, j))
(12)                      else if A(i, q) ⊗ D_{k−1}(q, j) <_⊕ D_k(i, j)
(13)                      then D_k(i, j) ← A(i, q) ⊗ D_{k−1}(q, j);
(14)                          π_k(i, j) ← {(i, q)} ⋄ π_{k−1}(q, j)
```

(b) Proxy algorithm for distributed Bellman-Ford, with explicit path computation.

Fig. 3: Generic algorithms computing best path weights together with associated paths. Abstraction with respect to $(\min,\ +,\ 0)$. The notation $a <_\oplus b$ means that $a = a \oplus b \neq b$. The results of these computations are $\mathbf{D} \equiv \mathbf{D}_{n-1}$ and $\pi \equiv \pi_{n-1}$. Each run of Dijkstra's algorithm computes only the i-th row of $\mathbf{D}$.

path sets in π. Dijkstra's algorithm takes a node i and computes the i-th row of $\mathbf{D}$, while our version of Bellman-Ford computes the all-pairs solution in $\mathbf{D}$. Our version of Bellman-Ford is intended to model the way distributed Bellman-Ford algorithms work (BGP, RIP) while at the same time ignoring the complexities associated with distributed asynchronous algorithms. Note that the algorithm never directly compares $\mathbf{D}_k(i,\ j)$ with $\mathbf{D}_{k-1}(i,\ j)$ as the Floyd-Warshall algorithm does [11]. Put another way, there is no *memory* at node i recording the best path seen so far to node j. Rather, all paths are recomputed at each step (value of k) from the paths of the previous iteration $(k-1)$.

C. Do the algorithms compute all best paths?

We return to the example illustrated in Figure 2 (a) and represented by $\mathbf{A}$ in Equation 6. If we instantiate the algorithms of Figure 3 with the shortest path semiring $\mathrm{sp} \equiv (\mathbb{N}^\infty,\ \min,\ +,\ \infty,\ 0)$, then both will compute the correct value $\mathbf{D} = \mathbf{A}^*$ of Equation 7 (of course Dijkstra's algorithm must be run n times, once for each row of $\mathbf{A}^*$).

In addition, both algorithms compute the same path sets in the matrix π. For example, it is easy to see that in Figure 2 (a) that there are three paths in both directions between nodes 0 and 2, each with optimal weight 3. Indeed, both algorithms compute

$$\begin{array}{rcl} \pi(0,\ 2) & = & \{(0,1,2),\ (0,1,3,2),\ (0,1,4,2)\} \\ \pi(2,\ 0) & = & \{(2,1,0),\ (2,3,1,0),\ (2,4,1,0)\}. \end{array}$$

(Note that we normally compress the path representation and write $(0,1,4,2)$ instead of $(0,1)(1,4)(4,2)$.)

We now turn to simple QoS metrics that are constructed from the composition of several semirings. We will focus on the *lexicographic product* of semirings,

$$(S, \oplus, \otimes, \overline{0}, \overline{1}) \ \equiv \ (S_1, \oplus_1, \otimes_1, \overline{0}_1, \overline{1}_1) \ \vec{\times} \ (S_2, \oplus_2, \otimes_2, \overline{0}_2, \overline{1}_2),$$

that is defined when the order associated with $\oplus_1$ is total. The carrier set S is $S_1 \times S_2$. The multiplicative operator $\otimes$ is defined point-wise as

$$(a,\ b) \otimes (c,\ d) \quad \equiv \quad (a \otimes_1 c,\ b \otimes_2 d),$$

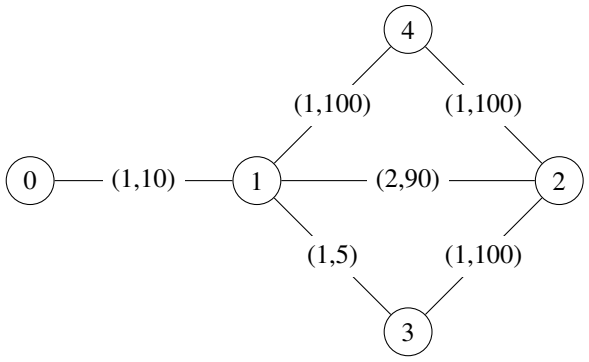


Fig. 4: Widest-shortest paths: weights (d, b) represent distance d and bandwidth b.

and so the multiplicative identity is $\overline{1} \equiv (\overline{1}_1, \overline{1}_2)$. The additive operator $\oplus$ is defined as a lexicographic product

$$(a,b) \oplus (c,d) \quad \equiv \quad \begin{cases} (a,b) & a <_{\oplus_1} c \\ (c,d) & c <_{\oplus_1} a \\ (a, b \oplus_2 d) & \text{otherwise} \end{cases}$$

and it is easy to check that the additive identity is $\overline{0} \equiv (\overline{0}_1, \overline{0}_2)$. This operator introduces the idea of a priority on metrics, where weights from the second component are used to break ties when the first components are equal.

Our first example of a lexicographic product is *most reliable shortest-paths* modeled with the metric

$$\text{sp} \vec{\times} \text{rel} = (\mathbb{N}^\infty,\ \min,\ +,\ \infty,\ 0) \vec{\times} ([0,\ 1],\ \max,\ \times,\ 0,\ 1).$$

An example weighted graph is presented in Figure 2 (b) which augments the link weights of Figure 2 (a) with link reliabilities.

The optimal weight between 0 and 2 is $(3, 0.405)$ and the associated paths are given by:

$$\begin{array}{rcl} \pi(0,2) & = & \{(0,1,2),\ (0,1,3,2)\} \\ \pi(2,0) & = & \{(2,1,0),\ (2,3,1,0)\}. \end{array}$$

These sets exclude the only non-optimal paths (which go through the link between 1 and 3) with a weight $(3, 0.05)$

Now consider the metric for *widest-shortest paths*,

$$\text{sp} \vec{\times} \text{bw} = (\mathbb{N}^\infty,\ \min,\ +,\ \infty,\ 0) \vec{\times} (\mathbb{N}^\infty,\ \max,\ \min,\ 0,\ \infty).$$

Figure 4 augments the link weights of Figure 2 (a) with link capacities. This graph is then represented as the adjacency matrix

$$\mathbf{A} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} \overline{0} & (1,10) & \overline{0} & \overline{0} & \overline{0} \\ (1,10) & \overline{0} & (2,90) & (1,5) & (1,100) \\ \overline{0} & (2,90) & \overline{0} & (1,100) & (1,100) \\ \overline{0} & (1,5) & (1,100) & \overline{0} & \overline{0} \\ \overline{0} & (1,100) & (1,100) & \overline{0} & \overline{0} \end{bmatrix} \end{array}$$

Running either of our algorithms (Dijkstra n times) results in an all-pairs matrix $\mathbf{D} = \mathbf{A}^*$,

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} (0,\infty) & (1,10) & (3,10) & (2,5) & (2,10) \\ (1,10) & (0,\infty) & (2,100) & (1,5) & (1,100) \\ (3,10) & (2,100) & (0,\infty) & (1,100) & (1,100) \\ (2,5) & (1,5) & (1,100) & (0,\infty) & (2,100) \\ (2,10) & (1,100) & (1,100) & (2,100) & (0,\infty) \end{bmatrix} \end{array}$$

Note that the first component of each entry corresponds to the shortest-path weight, while the second corresponds to the largest capacity among the shortest paths. We can see (by inspection) that there are now only two paths in each direction between nodes 0 and 2, each with optimal weight $(3, 10)$:

$$\begin{array}{rcl} \pi_{\text{optimal}}(0,2) & = & \{(0,1,2),\ (0,1,4,2)\} \\ \pi_{\text{optimal}}(2,0) & = & \{(2,1,0),\ (2,4,1,0)\} \end{array}$$

Paths $(0, 1, 3, 2)$ and $(2, 3, 1, 0)$, each of weight $(3, 5)$, are now eliminated due to the bottleneck link between nodes 1 and 3.

However, we can check that neither of our algorithms now computes these paths! It is not too hard to check that Dijkstra's algorithm computes the following paths between nodes 0 and 2.

$$\begin{array}{rcl} \pi_{\text{Dijkstra}}(0,2) & = & \{(0,1,2),\ (0,1,4,2)\} \\ \pi_{\text{Dijkstra}}(2,0) & = & \{(2,4,1,0)\} \end{array}$$

That is, when computing from the source node $i = 2$, Dijkstra's algorithm will find only the path $(2, 4, 1, 0)$ and miss the path $(2, 1, 0)$. One way of understanding this is that when node i runs this algorithm and inserts node 1 into the set S it has found this node along path $(2, 4, 1)$ which has weight $(2, 100)$, which is better than $(2, 90)$, the link weight of arc $(2, 1)$. Note also that the best path from node 1 to node 2 is not next-hop consistent with path $(0, 1, 2)$,

$$\pi_{\text{Dijkstra}}(1,2) \quad = \quad \{(1,4,2)\}$$

In other words, for destination 2, both paths from node 0 collapse to the same next hop entry (go to node 1), and node 1 knows only one path (with next-hop 4). To implement both paths we would need some form of tunneling from node 0 to node 2.

On the other hand, our Bellman-Ford algorithm computes the following paths between nodes 0 and 2.

$$\begin{array}{rcl} \pi_{\text{Bellman}}(0,2) & = & \{(0,1,4,2)\} \\ \pi_{\text{Bellman}}(2,0) & = & \{(2,1,0),\ (2,4,1,0)\} \end{array}$$

Note that in this case only path $(0, 1, 4, 2)$ is computed while path $(0, 1, 2)$ is not. However, by construction all paths computed can be implemented with next-hop and ECMP forwarding. (It is not always the case, as it is here due to the simplicity of the example, that the optimal paths can be obtained from a union of the paths computed by the two algorithms.)

In summary, we see that Bellman-Ford and Dijkstra compute different sets of paths, and that neither algorithm can compute all widest-shortest paths. In addition, some paths computed by Dijkstra's algorithm cannot be implemented with next-hop forwarding. In the next two sections we provide a general algebraic framework that helps clarify such anomalies.

III. Semirings and Beyond

In this section we first provide more details on the semiring-based approach to path finding. Results stated without proof or citation are standard and can be found in [9], [19], [11], [12], [13]. We then review recent results that move beyond semirings by dropping some of the semiring axioms.

A. Semirings axioms

The structure $(S, \oplus, \otimes, \overline{0}, \overline{1})$ is a *semiring* when

- $\oplus$ and $\otimes$ are associative binary operators over S
- $\oplus$ is commutative
- $\overline{0}$ is the identity for $\oplus$
- $\overline{1}$ is the identity for $\otimes$
- $\overline{0}$ is an annihilator for $\otimes$, $a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$

and left and right distributivity hold,

$$\begin{array}{rcccl} \text{LD} & : & a \otimes (b \oplus c) & = & (a \otimes b) \oplus (a \otimes c) \\ \text{RD} & : & (a \oplus b) \otimes c & = & (a \otimes c) \oplus (b \otimes c) \end{array}$$

All of the examples we consider are also *idempotent*, where

$$a = a \oplus a.$$

The order $\leq_\oplus$ is then a partial order. It is a total order when $\oplus$ is *selective*,

$$a \oplus b \in \{a,\ b\}.$$

A semiring is *bounded* when the multiplicative identity is also an additive annihilator,

$$a \oplus \overline{1} = \overline{1} \oplus a = \overline{1}.$$

In this case we have for every a, $\overline{1} \leq_\oplus a \leq_\oplus \overline{0}$.

B. Matrix Iterations, Equations

We can extend the operations of a semiring S to the collection of square matrices of order n over S in a natural way. For any two such matrices $\mathbf{X}$, $\mathbf{Y}$ addition is defined as

$$(\mathbf{X} \oplus \mathbf{Y})(i,\ j) \equiv \mathbf{X}(i,\ j) \oplus \mathbf{Y}(i,\ j)$$

and multiplication as

$$(\mathbf{X} \otimes \mathbf{Y})(i,\ j) \equiv \bigoplus_{q \in V} \mathbf{X}(i,\ q) \otimes \mathbf{Y}(q,\ j).$$

Here we assume for the sake of convenience that the matrix indices range over V. The matrix

$$\mathbf{I}(i,j) = \begin{cases} \overline{1} & (i = j) \\ \overline{0} & (\text{otherwise}) \end{cases}$$

is the multiplicative identity, while the matrix where all its entries are $\overline{0}$ is the additive identity. It is not too hard to show that the collection of all such matrices forms a semiring.

There are many algorithms for computing $\mathbf{A}^*$, when it exists, and here we present only two iterative methods. *Left iteration* is defined as

$$\begin{array}{rcl} \mathbf{A}^{\langle 0 \rangle} & \equiv & \mathbf{I} \\ \mathbf{A}^{\langle k+1 \rangle} & \equiv & (\mathbf{A} \otimes \mathbf{A}^{\langle k \rangle}) \oplus \mathbf{I} \end{array}$$

while *right iteration* is defined as

$$\begin{array}{rcl} {}^{\langle 0 \rangle}\mathbf{A} & \equiv & \mathbf{I} \\ {}^{\langle k+1 \rangle}\mathbf{A} & \equiv & ({}^{\langle k \rangle}\mathbf{A} \otimes \mathbf{A}) \oplus \mathbf{I}. \end{array}$$

For bounded semirings it turns out that we need only iterate $n-1$ times to converge to $\mathbf{A}^*$,

$$\mathbf{A}^* \ = \ \mathbf{A}^{\langle n-1 \rangle} \ = \ {}^{\langle n-1 \rangle}\mathbf{A}.$$

In other words, we have

$$\begin{array}{rcl} \mathbf{A}^{\langle n-1 \rangle} & = & (\mathbf{A} \otimes \mathbf{A}^{\langle n-1 \rangle}) \oplus \mathbf{I} \\ {}^{\langle n-1 \rangle}\mathbf{A} & = & ({}^{\langle n-1 \rangle}\mathbf{A} \otimes \mathbf{A}) \oplus \mathbf{I}, \end{array}$$

and thus we have also solved the matrix equations

$$\mathbf{L} = (\mathbf{A} \otimes \mathbf{L}) \oplus \mathbf{I}, \tag{9}$$

$$\mathbf{R} = (\mathbf{R} \otimes \mathbf{A}) \oplus \mathbf{I}. \tag{10}$$

C. Semiring of elementary paths

Here we introduce a semiring that is associated with only a partial order — the semiring of elementary paths (sep). The carrier set is the power set of all elementary paths $\mathrm{epaths}(E)$, the additive operation is union, and the multiplicative operation is $\diamond$ (defined in Equation 8),

name	S	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
sep	$\mathcal{P}(\mathrm{epaths}(E))$	$\cup$	$\diamond$	$\{\}$	$\{\epsilon\}$

The adjacency matrix $\mathbf{B}$ for graph $G = (V, E)$ is defined as

$$\mathbf{B}(i,\ j) = \begin{cases} \{(i,\ j)\} & (i,\ j) \in E \\ \{\} & (i,\ j) \notin E. \end{cases}$$

It is not hard to check [10] that

$$\mathbf{B}^*(i,\ j) = \text{the set of all elementary paths from } i \text{ to } j.$$

This results holds even though the semiring sep is not closed — the additive annihilator ($\mathrm{epaths}(E)$) is not equal to the multiplicative identity ($\{\epsilon\}$). This semiring will play an important role in the next section.

D. On the loss of distributivity

As mentioned in the introduction, research initially inspired by BGP has forced us to consider algebraic structures that violate some of the semiring axioms [21], [22], [14]. The most common violations seen are related to a loss of distributivity of $\otimes$ over $\oplus$. Examples of such violations are most easily exhibited using the lexicographic product defined in the previous section.

Let us consider only the axiom for left distributivity,

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c).$$

When instantiating this with the result of a lexicographic product of two semirings

$$(S, \oplus, \otimes, \overline{0}, \overline{1}) \ \equiv \ (S_1, \oplus_1, \otimes_1, \overline{0}_1, \overline{1}_1) \ \vec{\times} \ (S_2, \oplus_2, \otimes_2, \overline{0}_2, \overline{1}_2),$$

we have for the left-hand-side of the distributivity axiom

$$\mathrm{lhs} \equiv (a_1,\ a_2) \otimes ((b_1,\ b_2) \oplus (c_1,\ c_2)).$$

By the definitions of $\oplus$ and $\otimes$ (Section II-C) we see that there are three possibilities,

	lhs	when
lhs.1	$(a_1 \otimes_1 b_1,\ a_2 \otimes_2 b_2)$	$b_1 <_{\oplus_1} c_1$
lhs.2	$(a_1 \otimes_1 c_1,\ a_2 \otimes_2 c_2)$	$c_1 <_{\oplus_1} b_1$
lhs.3	$(a_1 \otimes_1 b_1,\ a_2 \otimes_2 (b_2 \oplus_2 c_2))$	otherwise

Now consider the righ-hand-side,

$$\begin{array}{rcl} \text{rhs} & \equiv & ((a_1,\ a_2) \otimes (b_1, b_2)) \oplus ((a_1,\ a_2) \otimes (c_1,\ c_2)) \\ & = & (a_1 \otimes_1 b_1,\ a_2 \otimes_2 b_2) \oplus (a_1 \otimes_1 c_1,\ a_2 \otimes_2 c_2) \end{array}$$

where again there are three possibilities,

	rhs	when
rhs.1	$(a_1 \otimes_1 b_1,\ a_2 \otimes_2 b_2)$	$a_1 \otimes_1 b_1 <_{\oplus_1} a_1 \otimes_1 c_1$
rhs.2	$(a_1 \otimes_1 c_1,\ a_2 \otimes_2 c_2)$	$a_1 \otimes_1 c_1 <_{\oplus_1} a_1 \otimes_1 b_1$
rhs.3	$(a_1 \otimes_1 b_1, a_2 \otimes_2 (b_2 \oplus_2 c_2))$	otherwise

We simplified case rhs.3 using distributivity on $(a_2 \otimes_2 b_2) \oplus_2 (a_2 \otimes_2 c_2)$. Clearly, in order to guarantee that lhs $=$ rhs we would like to have, for all $x,\ y,\ z$,

$$x <_{\oplus_1} y \implies z \otimes_1 x <_{\oplus_1} z \otimes_1 y.$$

Unfolding the definition of $<_{\oplus_1}$, we need

$$x = x \oplus_1 y \neq y \implies z \otimes_1 x = (z \otimes_1 x) \oplus_1 (z \otimes_1 y) \neq z \otimes_1 y,$$

or

$$\begin{array}{rlcl} (1) & x = x \oplus_1 y & \implies & z \otimes_1 x = (z \otimes_1 x) \oplus_1 (z \otimes_1 y) \\ & & = & z \otimes_1 x = z \otimes_1 (x \oplus_1 y) \\ (2) & x \oplus_1 y \neq y & \implies & (z \otimes_1 x) \oplus_1 (z \otimes_1 y) \neq z \otimes_1 y \\ & & = & z \otimes_1 (x \oplus_1 y) \neq z \otimes_1 y \end{array}$$

Since (1) always holds, we are left with (2), which can be simplified to

$$w \neq y \quad \implies \quad z \otimes_1 w \neq z \otimes_1 y.$$

This is simply the contrapositive of what is normally called (left) *cancellativity* for the multiplicative component $\otimes_1$,

$$\forall w, y \in S_1, z \in S_1 \backslash \{\overline{0}_1\}, z \otimes_1 w = z \otimes_1 y \implies w = y$$

Without cancellativity the left-hand side may fall into cases lhs.1 or lhs.2, while the right-hand side falls into case rhs.3 because with $a_1 \otimes_1 b_1 = a_1 \otimes_1 c_1$. Assume we have lhs.1, then a problem will arise if we can select values in S_1 so that

$$\text{lhs} \equiv (a_1 \otimes_1 b_1,\ a_2 \otimes_2 b_2) \neq (a_1 \otimes_1 b_1, a_2 \otimes_2 (b_2 \oplus_2 c_2)) \equiv \text{rhs}.$$

If the semiring $(S_2, \oplus_2, \otimes_2, \overline{0}_2, \overline{1}_2)$ is non-trivial, then we should be able to find such values unless $\otimes_2$ is itself somewhat trivial in that it is (left) *constant*,

$$\forall z, y, z \in S_2,\ z \otimes_2 x = z \otimes_2 y.$$

For right distributivity analogous results hold using corresponding notions of right cancellative and right constant operations. An algebraic structure is distributive, multiplicatively cancellative or constant, if both the left and right versions of the properties hold.

Putting this together we have the following theorem [16].

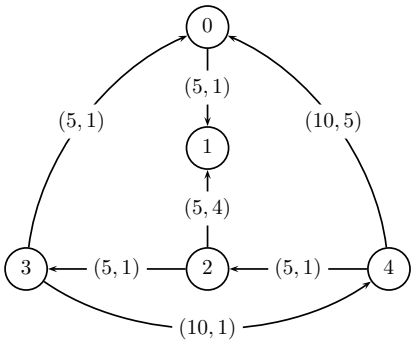


Fig. 5: Shortest widest-paths example from [14] that gives three distinct solutions for $\mathbf{A}^*$, $\mathbf{L}$, and $\mathbf{R}$.

Theorem 1. *The lexicographic product of two semirings is distributive if and only if the two components are distributive and $\otimes_1$ is cancellative or $\otimes_2$ is constant*[2].

Consider the three examples,

	definition	distributive?
most reliable shortest-paths	$\text{mrsp} \equiv \text{sp} \stackrel{\rightarrow}{\times} \text{rel}$	Yes
widest shortest-paths	$\text{wsp} \equiv \text{sp} \stackrel{\rightarrow}{\times} \text{bw}$	Yes
shortest widest-paths	$\text{swp} \equiv \text{bw} \stackrel{\rightarrow}{\times} \text{sp}$	No

The example swp, shortest widest-paths, is not a semiring [7] because the multiplicative operation of min is not cancellative and it appears in the first component of the composite semiring. Thus for semirings, use of *concave* metrics, such as bandwidth, is restricted to all but the last component in a sequence of lexicographic compositions.

Figure 5 presents shortest widest-paths example from [14] that gives three distinct solutions for $\mathbf{A}^*$, $\mathbf{L}$, and $\mathbf{R}$. Here we have

$$\mathbf{A}^* = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{ccccc} (\infty,0) & (5,1) & (0,\infty) & (0,\infty) & (0,\infty) \\ (0,\infty) & (\infty,0) & (0,\infty) & (0,\infty) & (0,\infty) \\ (5,2) & (5,3) & (\infty,\ 0) & (5,1) & (5,2) \\ (10,6) & (5,2) & (5,2) & (\infty,0) & (10,1) \\ (10,5) & (5,4) & (5,1) & (5,2) & (\infty,0) \end{array}\right] \end{array}$$

$$\mathbf{L} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{ccccc} (\infty,0) & (5,1) & (0,\infty) & (0,\infty) & (0,\infty) \\ (0,\infty) & (\infty,0) & (0,\infty) & (0,\infty) & (0,\infty) \\ \mathbf{(5,7)} & (5,3) & (\infty,0) & (5,1) & (5,2) \\ (10,6) & (5,2) & (5,2) & (\infty,0) & (10,1) \\ (10,5) & (5,4) & (5,1) & (5,2) & (\infty,0) \end{array}\right] \end{array}$$

$$\mathbf{R} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{ccccc} (\infty,0) & (5,1) & (0,\infty) & (0,\infty) & (0,\infty) \\ (0,\infty) & (\infty,0) & (0,\infty) & (0,\infty) & (0,\infty) \\ (5,2) & (5,3) & (\infty,\ 0) & (5,\ 1) & (5,2) \\ (10,6) & \mathbf{(5,6)} & (5,2) & (\infty,0) & (10,1) \\ (10,5) & \mathbf{(5,5)} & (5,1) & (5,2) & (\infty,0) \end{array}\right] \end{array}$$

The *bold* entries in $\mathbf{L}$ and $\mathbf{R}$ highlight where they differ from $\mathbf{A}^*$.

[2]The *constant* condition may appear at first glance to be rather silly, but it was put to good use in modeling the eBGP/iBGP distinction with the definition of the *scoped product* [16].

IV. Folding paths into the metric

We now have most of the concepts and results needed to explain the path anomalies described in Section II. Suppose that

$$\text{metric} \equiv (S_1,\ \oplus_1,\ \otimes_1,\ \overline{0}_1,\ \overline{1}_1)$$

is a bounded and totally ordered semiring. We then fold the path computations *into the metric* using the semiring of elementary paths (sep, defined in Section III-C) and the lexicographic product,

$$\text{guarded}(\text{metric}) \equiv \text{metric} \stackrel{\rightarrow}{\times} \text{sep}.$$

We call this a *guarded* algebra of paths over the semiring metric. Using the results of the previous section and the fact that the multiplicative component of sep is not constant, we conclude that guarded(metric) will be a semiring *exactly when* the multiplicative operation of metric is cancellative.

If metric is not cancellative, then guarded(metric) will not be a semiring (due to a lack of distributivity) and we conclude that guarded(metric) can exhibit the same kind of behavior as shortest widest-paths described in the Section III-D. That is, there may be networks with three distinct solutions for $\mathbf{A}^*$, $\mathbf{L}$, and $\mathbf{R}$.

Let us consider three examples.

metric	Is guarded(metric) a semiring?
sp	Yes
mrsp	Yes
wsp	No, since wsp is not cancellative.

The first two semirings, sp and mrsp, are cancellative and so the guarded algebra is a semiring and we would expect $\mathbf{A}^* = \mathbf{L} = \mathbf{R}$. However, wsp is not cancellative, so guarded(wsp) is not a semiring and the algebra can exhibit different solutions depending on the algorithm used.

Only two things remain to be shown. First, that when the Bellman-Ford algorithm of Figure 3 (b) is instantiated with the semiring metric it is really computing a a left solution $\mathbf{L}$ for the algebra guarded(metric). Second, when Dijkstra's algorithm of Figure 3 (a) is instantiated with the semiring metric it is really computing (the i-th row of) a right solution $\mathbf{R}$ for guarded(metric).

A. Application of A2A method to Bellman-Ford

Recall the definition of matrix multiplication from Section III-B, which is presented in an algebraic style,

$$(\mathbf{A} \otimes \mathbf{Y})(i,\ j) \equiv \bigoplus_{q \in V} \mathbf{A}(i,\ q) \otimes \mathbf{Y}(q,\ j).$$

We could rewrite this in a more algorithmic style, putting the results of multiplication in matrix $\mathbf{L}$ as

$$\mathbf{L} \leftarrow \mathbf{A} \otimes \mathbf{Y}.$$

Unfolding this even further we arrive at a standard algorithm for matrix multiplication,

```
(1)  for each i ∈ V do
(2)    for each j ∈ V do
(3)      L(i, j) ← 0̄;
(4)      for each q ∈ V do
(5)        L(i, j) ← L(i, j) ⊕ (A(i, q) ⊗ Y(q, j))
```

Note that line (3) initializes $\mathbf{L}(i,\ j)$ while lines (4) and (5) accumulate the sum as q ranges over V.

Next, consider how we could modify this algorithm to compute

$$\mathbf{L} \leftarrow (\mathbf{A} \otimes \mathbf{Y}) \oplus \mathbf{I}.$$

There are several ways to do this, but we will add a condition for $i = j$,

```
(1)  for each i ∈ V do
(2)    for each j ∈ V do
(3)      if i = j
(4)      then L(i, j) ← I(i, j)
(5)      else L(i, j) ← 0̄
(6)           for each q ∈ V do
(7)             L(i, j) ← L(i, j) ⊕ (A(i, q) ⊗ Y(q, j))
```

We can now modify this algorithm to capture the (left) iteration described in Section III-B,

$$\begin{aligned} \mathbf{A}^{\langle 0 \rangle} &\equiv \mathbf{I} \\ \mathbf{A}^{\langle k+1 \rangle} &\equiv (\mathbf{A} \otimes \mathbf{A}^{\langle k \rangle}) \oplus \mathbf{I} \end{aligned}$$

where at each step k we will store the result $\mathbf{A}^{\langle k \rangle}$ in $\mathbf{L}_k$,

```
     ITERATION_GENERIC(⊕, ⊗, 0̄, 1̄, A)
(0)  L_0 ← I
(1)  for k ← 1 to |V| − 1 do
(2)    for each i ∈ V do
(3)      for each j ∈ V do
(4)      if i = j
(5)      then L_k(i, j) ← I(i, j)
(6)      else L_k(i, j) ← 0̄;
(7)           for each q ∈ V do
(8)             L_k(i, j) ←
                  L_k(i, j) ⊕ (A(i, q) ⊗ L_{k−1}(q, j))
```

This is beginning to resemble the Bellman-Ford algorithm of Figure 3 (b)! However that algorithm is working with *two* matrices, $\mathbf{D}_k$ and π_k, a $\oplus$ defined as the lexicographic product of $\oplus_1$ and $\oplus_2 = \cup$, and a $\otimes$ defined as the direct product of $\otimes_1$ and $\otimes_2 = \diamond$.

Some additional notation will be helpful. Suppose that $\mathbf{X}$ is a matrix with entries in S_1, and $\mathbf{Y}$ is a matrix with entries in S_2. If $\mathbf{X}$ and $\mathbf{Y}$ are the same size, then we can define the matrix $(\!|\mathbf{X},\ \mathbf{Y}|\!)$ over entries in $S_1 \times S_2$ as

$$(\!|\mathbf{X},\ \mathbf{Y}|\!)(i,\ j) \equiv (\mathbf{X}(i,\ j),\ \mathbf{Y}(i,\ j)).$$

Now consider instantiating ITERATION_GENERIC with the algebra guarded(metric). We can express an adjacency matrix over this algebra as $(\!|\mathbf{A},\ \mathbf{B}|\!)$, where $\mathbf{A}$ is the adjacency matrix for the semiring metric and $\mathbf{B}$ is the adjacency matrix for sep, as described in Section III-C. Define

$$\mathbf{L}_k \equiv (\!|\mathbf{D}_k,\ \pi_k|\!).$$

Then lines $(1) - (2)$ of Figure 3 (b) can be seen to correspond to line (0) of ITERATION_GENERIC

$$(\!|\mathbf{D}_0,\ \pi_0|\!) \leftarrow (\!|\mathbf{I}_1,\ \mathbf{I}_2|\!)$$

where $\mathbf{I}_1$ and $\mathbf{I}_2$ are the multiplicative identity matrices for semiring metric and sep, respectively. Similar observations can be made to see that lines $(7) - (8)$ of Figure 3 (b) correspond to lines $(4) - (5)$ of ITERATION_GENERIC.

DIJKSTRA_GENERIC$(\oplus,\ \otimes,\ \overline{0},\ \overline{1},\ \mathbf{A},\ i)$

(1) **for each** $q \in V$ **do** $\mathbf{R}_0(i,\ q) \leftarrow \overline{0}$
(2) $S_0 \leftarrow \{\};\ \mathbf{R}_0(i,\ i) \leftarrow \overline{1}$
(3) **for each** $k = 1, 2, \ldots, \mid V \mid$ **do**
(4) find $q_k \in V - S_{k-1}$ with $\mathbf{R}_{k-1}(i,\ q) \leq_\oplus$-minimal
(5) $S_l \leftarrow S_{k-1} \cup \{q_k\}$
(6) **for each** $j \in V - S_k$ **do**
(7) $\mathbf{R}_k(i,\ j) \leftarrow$
$\mathbf{R}_{k-1}(i,\ j) \oplus (\mathbf{R}_{k-1}(i,\ q_k) \otimes \mathbf{A}(q_k,\ j))$

Fig. 6: Dijkstra's Algorithm Generalized to a totally ordered semiring [10], [12], [13].

DIJKSTRA_GUARDED$(\oplus,\ \otimes,\ \overline{0},\ \overline{1},\ \mathbf{A},\ i)$

(1) **for each** $q \in V$ **do** $\mathbf{R}_0(i,\ q) \leftarrow \overline{0}$
(2) $S_0 \leftarrow \{\};\ \mathbf{R}_0(i,\ i) \leftarrow \overline{1}$
(3) **for each** $k = 1, 2, \ldots, \mid V \mid$ **do**
(4) find $q_k \in V - S_{k-1}$ with $\mathbf{R}_{k-1}(i,\ q_k) \leq_1$-minimal
(5) $S_k \leftarrow S_{k-1} \cup \{q_k\}$
(6) **for each** $j \in V - S_k$ **do**
(7) $\mathbf{R}_k(i,\ j) \leftarrow$
$\mathbf{R}_{k-1}(i,\ j) \oplus (\mathbf{R}_{k-1}(i,\ q_k) \bar{\otimes}\ \mathbf{A}(q_k,\ j))$

Fig. 7: Dijkstra's algorithm for guarded semirings. Order only first component of the guarded semiring. Only assume that algebra is guarded, so we are working over an algebra of the form $(S, \oplus, \otimes, \overline{0}, \overline{1}) = \mathrm{guarded}(\mathrm{metric})$.

The only remaining line to consider is line (8) of ITERATION_GENERIC. By using $(\!|\mathbf{D}_k,\ \pi_k|\!)$ for $\mathbf{L}_k$ and unfolding the definition of $\otimes$ (Section II-C) we can rewrite this line as

$$(\!|\mathbf{D}_k,\ \pi_k|\!)(i,\ j) \leftarrow (\!|\mathbf{D}_k,\ \pi_k|\!)(i,\ j) \oplus (\mathbf{A}(i,\ q) \otimes_1 \mathbf{D}_{k-1}(q, j),\ \{(i,\ q)\} \diamond \pi_{k-1}(q,\ j)).$$

Finally, by unfolding the definition of the lexicographic $\oplus$ (Section II-C), and recalling that $\cup$ is the additive operation of sep, this line can be expressed more algorithmically exactly as lines $(10) - (14)$ of Figure 3 (b),

(10) **if** $\mathbf{A}(i,\ q) \otimes \mathbf{D}_{k-1}(q,\ j) = \mathbf{D}_k(i,\ j)$
(11) **then** $\pi_k(i,\ j) \leftarrow \pi_k(i,\ j) \cup (\{(i,\ q)\} \diamond\ \pi_{k-1}(q,\ j))$
(12) **else if** $\mathbf{A}(i,\ q) \otimes \mathbf{D}_{k-1}(q,\ j) <_\oplus \mathbf{D}_k(i,\ j)$
(13) **then** $\mathbf{D}_k(i,\ j) \leftarrow \mathbf{A}(i,\ q) \otimes \mathbf{D}_{k-1}(q,\ j)$;
(14) $\pi_k(i,\ j) \leftarrow \{(i,\ q)\} \diamond\ \pi_{k-1}(q,\ j)$

B. Application of A2A method to Dijkstra

Figure 6 presents a generic version of Dijkstra's algorithm for totally ordered closed semirings [10], [12], [13]. It is fairly easy to show, by induction on k, that the k-th iteration of this algorithm computes the i-th row of a right-local (partial) solution for elements of S_k.

Theorem 2. *For all k such that $0 \leqslant k \leqslant \mid V \mid$, we have that:*

$$\forall\ j \in V : \mathbf{R}_k(i,j) = \mathbf{I}(i,j) \oplus \bigoplus_{q \in S_k} (\mathbf{R}_k(i,q) \otimes \mathbf{A}(q,j))$$

Letting $\mathbf{R} \equiv \mathbf{R}_n$ and noting that $S_n = V$, we then have

$$\forall\ j \in V : \mathbf{R}(i,j) = \mathbf{I}(i,j) \oplus \bigoplus_{q \in V} (\mathbf{R}(i,q) \otimes \mathbf{A}(q,j)).$$

That is, Figure 6 computes the i-th row of a right solution,

$$\mathbf{R} = (\mathbf{R} \otimes \mathbf{A}) \oplus \mathbf{I}.$$

It is tempting to instantiate the algorithm of Figure 6 with a $\mathrm{guarded}(\mathrm{metric})$ and to use it to solve the associated path problem. This approach would unfortunately fail, since DIJKSTRA_GENERIC requires $\leq_\oplus$ to be total, something which is not true for a $\mathrm{guarded}(\mathrm{metric})$. The semiring of elementary paths, sep, uses set union as its additive law and the associated order is partial. As a consequence, the order associated with the additive law of any $\mathrm{guarded}(\mathrm{metric})$ is also partial. However, as our starting point, DIJKSTRA_GENERIC_WITH_PATHS, does not use at all the order associated with the paths, only the one associated with the metric. This motivates us to alter Line (4) of DIJKSTRA_GENERIC by introducing a special version of the order

$$(a, b) \leq_1 (c, d) \equiv a \leq_{\oplus_1} c$$

The resulting algorithm, DIJKSTRA_GUARDED, can be instantiated to a $\mathrm{guarded}(\mathrm{metric})$.

It is then possible to modify the proof of Theorem 2 and conclude that for the new algorithm of Figure 7 we have for all $j \in V$,

$$\mathbf{R}(i,j) = (\!|\mathbf{I}_1,\ \mathbf{I}_2|\!)(i,j) \oplus \bigoplus_{q \in V} (\mathbf{R}(i,q) \otimes (\!|\mathbf{A},\ \mathbf{B}|\!)(q,j)).$$

That is, we have solved for the i-th row of a right solution to equation

$$\mathbf{R} = (\mathbf{R} \otimes (\!|\mathbf{A},\ \mathbf{B}|\!)) \oplus (\!|\mathbf{I}_1,\ \mathbf{I}_2|\!).$$

Lastly, we need to check that the algorithm of Figure 7 instantiated to $\mathrm{guarded}(\mathrm{metric})$ computes the same result as the the algorithm of Figure 3 (a) when instantiated with the semiring metric. This can be accomplished by defining

$$\mathbf{R}_k \equiv (\!|\mathbf{D}_k,\ \pi_k|\!).$$

and then applying the same type of algorithmic unfolding as was used above for Bellman-Ford.

V. OPEN PROBLEMS

Some researchers have argued that ECMP does not result in a sufficient diversity of paths and our results can be seen as strengthening this argument. Proposals have been made to use not just best the paths, but k-best paths, for some small values of k, while at the same time avoiding forwarding loops [23], [24], [25], [26]. Since the metric for k-best paths can be captured using a semiring [27], [12] it might be interesting to see if the A2A method described in Section I could be applied to an analysis of these proposals to move beyond ECMP.

IP routing protocols normally do not propagate addresses, but rather prefixes representing sets of addresses. Connectivity is implemented with forwarding tables containing prefixes together with *longest match* forwarding. We could model this with the A2A method by using an approach similar to [17] — with the prefix length as a part of the routing metric as

a *cold potato* value preferring longer masks — combined with exact match forwarding. (This is an example where the A2A method would be applied *only* to build a model since implementing forwarding this way could lead to exponentially large forwarding tables.). The real difficulty would then be to model the issues of route aggregation that have been described in [28]. This would most likely lead to metrics that are not semirings.

REFERENCES

[1] S. Chen and K. Nahrsted, "An overview of quality of service routing for next-generation high-speed networks: problems and solutions," *Network, IEEE*, vol. 12, no. 6, pp. 64–79, 1998.

[2] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for qos routing," *Communications Magazine, IEEE*, vol. 40, no. 12, pp. 50–55, 2002.

[3] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *Selected Areas in Communications, IEEE Journal on*, vol. 14, no. 7, pp. 1228–1234, 1996.

[4] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, no. 1, pp. 95–116, 1984.

[5] X. Masip-Bruin, M. Yannuzzi, J. Domingo-Pascual, A. Fonte, M. Curado, E. Monteiro, F. Kuipers, P. V. Mieghem, S. Avallone, G. Ventre, P. Aranda-Gutirrez, M. Hollick, R. Steinmetz, L. Iannone, and K. Salamatian, "Research challenges in qos routing," *Computer Communications*, vol. 29, no. 5, pp. 563 – 581, 2006.

[6] R. A. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *IEEE GLOBECOM*, 1997, see also RFC 2676, 1999.

[7] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 541–550, 2002.

[8] H. De Neve and P. Van Mieghem, "Tamcra: a tunable accuracy multiple constraints routing algorithm," *Comput. Commun.*, vol. 23, no. 7, pp. 667–679, Mar. 2000.

[9] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[10] B. Carré, *Graphs and Networks*. Oxford University Press, 1979.

[11] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 1st ed. McGraw-Hill, 1990.

[12] M. Gondran and M. Minoux, *Graphs, Dioids and Semirings : New Models and Algorithms*. Springer, 2008.

[13] J. Baras and G. Theodorakopoulos, *Path Problems in Networks*. Morgan and Claypool, 2010.

[14] J. a. L. Sobrinho and T. G. Griffin, "Routing in Equilibrium," in *19th International Symposium on Mathematical Theory of Networks and Systems*, 2010.

[15] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Transactions on Networking*, pp. 681–692, Dec. 2001.

[16] A. J. T. Gurney and T. G. Griffin, "Lexicographic products in metarouting," in *ICNP*, 2007.

[17] J. N. Billings and T. G. Griffin, "A Model of Internet Routing Using Semi-modules," *RelMICS*, 2009.

[18] M. A. Alim and T. G. Griffin, "On the interaction of multiple routing algorithms," in *ACM CoNEXT*, 2011.

[19] R. C. Backhouse and B. A. Carré, "Regular Algebra Applied to Path-finding Problems," *Journal of Applied Mathematics*, vol. 15, pp. 161–186, 1975.

[20] R. Perlman, *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, 2nd ed. Addison-Wesley, 2000.

[21] J. L. Sobrinho, "Network routing with path vector protocols: Theory and applications," in *SIGCOMM*, 2003.

[22] ——, "An algebraic theory of dynamic network routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1160–1173, 2005.

[23] M. Rios, V. Marianov, and A. Avagliano, "Multiple path routing algorithm for {IP} networks," *Computer Communications*, vol. 28, no. 7, pp. 829 – 836, 2005.

[24] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," in *SIGCOMM*, 2006.

[25] P. Mrindol, P. Francois, O. Bonaventure, S. Cateloin, and J.-J. Pansiot, "An efficient algorithm to enable path diversity in link state routing networks," *Computer Networks*, vol. 55, no. 5, pp. 1132 – 1149, 2011.

[26] M. Dasgupta and G. Biswas, "Design of multi-path data routing algorithm based on network reliability," *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1433 – 1443, 2012.

[27] A. Wongseelashote, "Semirings and Path Spaces," *Discrete Mathematics*, vol. 26, pp. 55–78, 1979.

[28] F. Le, G. G. Xie, and H. Zhang, "On route aggregation," in *SIGCOMM*, 2011.