On the Computational Complexity of Policy Routing

Márton Zubor[1,3], Attila Kőrösi[1,2,4], and András Gulyás[1,2,4] and Gábor Rétvári[1,2,5]

[1] Budapest University of Technology and Economics, Hungary
[2] Department of Telecommunication and Mediainformatics
[3] Department of Algebra
[4] Hungarian Academy of Science (MTA) Information system research group
[5] MTA-BME Future Internet Research Group
{zubor, korosi, gulyas, retvari}@tmit.bme.hu

Abstract. With the advent of new network architectures, like Software Defined Networks, the rules governing the way traffic is routed through the network are becoming increasingly complex. In this paper, we revisit the theoretic underpinnings of policy routing in the light of the new requirements. We show that certain simple but plausible algebraic properties already induce intractable path selection instances, and we extend the algebraic description of policies for which the related path selection problem is guaranteed to be tractable with a new class, called polynomial finite algebras, which captures many real-life application domains.

Keywords: policy routing, path selection, routing algebras, computational complexity

1 Introduction

Policy routing is the art and science of determining optimal forwarding paths under complex operational constraints. Originally in shortest path routing the rule was simply to pick the least cost path with respect to some additive link weights, but network operators have increasingly turned towards more sophisticated policies like path reliability and resilience [17], bandwidth and perceived congestion [8, 16], business relations and service level agreements [2], etc. These routing policies, and the computational complexity of the path selection problem thereof, are well-understood today, thanks to the theory of routing algebras [4, 5, 14, 15].

Recently, however, new routing architectures have surfaced for which today's algebraic routing theory does not provide adequate complexity characterization. Data centers, for instance, operate over starkly optimized topologies and services, defined by rules significantly more customized than allowed by classical routing policies. With the emergence of Software Defined Networks (SDN), furthermore, operators now enjoy complete freedom to shape their routing preferences [9]. A perfect example of how the seemingly simple problem of calculating the preferred path between two nodes can become surprisingly complicated is the upcoming

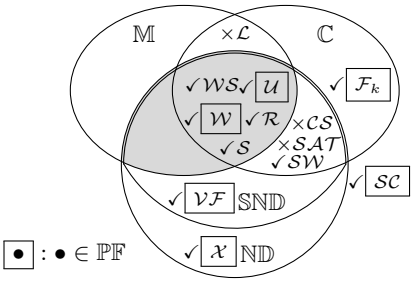

Fig. 1: A classification of routing algebras based on the algebraic properties of the corresponding path selection problem. $\mathbb{ND}$ denotes non-decreasing, $\mathbb{SND}$ strongly non-decreasing, $\mathbb{M}$ monotone, $\mathbb{PF}$ polinomial fininte and $\mathbb{C}$ commutative algebras. Algebras in the gray area can be solved by the Dijkstra algorithm, and algebras $\mathcal{S}$, $\mathcal{W}$, $\mathcal{WS}$, $\mathcal{SW}$, $\mathcal{R}$, $\mathcal{U}$, $\mathcal{VF}$, $\mathcal{X}$, $\mathcal{SC}$, $\mathcal{SAT}$, $\mathcal{CS}$, and $\mathcal{F}_k$, marked with $\checkmark$ (or $\times$), give examples for tractable (intractable, resp.) routing policies (see Section 6).

Service Chaining paradigm [10]. Here, service functions, like intrusion detection, network address translation, or video transcoding, are realized as standalone or virtualized middleboxes scattered throughout the network, and packets must pass through these functions in specific order. In more recent deployments the particular service realized by a middlebox has become dynamically configurable, and the policy routing task involves jointly optimizing both the placement of the services *and* the forwarding paths themselves [9].

In these routing paradigms packet loops are a fairly natural consequence of the routing policy itself, something that falls completely outside the model of classical algebraic policy routing. Even our very capability to compute the preferred path is under scrutiny now, as it is no longer obvious whether what we are looking for is in fact a path or a walk. Or, in a similarly troubling scenario, a network operator might accidentally pose a service chaining rule that, even though extremely desirable from an operational perspective, happens to induce an intractable path selection problem, making that policy essentially unrealizable in practice[6]. Regrettably, conventional routing theory does not provide sufficient guidelines to identify such pathologic cases.

The goal of this paper is to take the first steps to extend the theory of policy routing into this brave new era of networking. Our main goal is to separate routing policies over which computing the preferred path/walk is "easy" from those that admit intractable path/walk selection instances. However, instead of giving piecemeal complexity characterizations for each routing policy we rather provide a *general* treatment. In particular,

(i) we describe the algebraic properties that are sufficient and necessary in order for the optimal walk to coincide with the optimal path;
(ii) we extend the "safe algebraic regime" under which path selection is guaranteed to be polynomially tractable; and
(iii) we identify the ensembles of simple algebraic properties that can induce NP-complete path selection instances.

As a main contribution of the paper, we provide the first ever comprehensive classification of routing algebras based on the tractability of the related path selection problem (see Fig. 1 and the discussion in Section 6).

The rest of the paper is organized as follows. First, we introduce the algebraic model used throughout the paper in Section 2. Then, in Section 3 we reveal the relation between the preferred walk and preferred path selection problems, in

[6] Later in Section 6 we shall show real-life routing policies that exhibit this problem.

Section 4 we define a new algebraic property, polynomial finite algebras, that warrants a fast path selection algorithm, in Section 5 we discuss the reverse case, that is, algebras inducing intractable path selection instances, in Section 6 we highlight important practical consequences of our findings, and finally in Section 7 we conclude the paper.

2 An algebraic model for policy routing

The network is modeled by a finite, connected graph $G(V, E)$, $|V| = n$ and $|E| = m$. An $s - t$ *walk* is a sequence of nodes $q = (s = v_1, v_2, \ldots, v_k = t)$, where k is the length of the walk and $(v_i, v_{i+1}) \in E : \forall i = 1, \ldots, k-1$. A *cycle* is a walk with $s = t$, a *path* p is a walk that visits a node at most once, and a *preferred walk* q^* is the one that is favored from the set of available $s - t$ walks $\mathcal{Q}_{st}$ according to some predefined policy routing rules.

A concise model for this setting is that of *routing algebras* [4, 5, 14, 15]. In this paper, a routing algebra $\mathcal{A}$ is defined as a totally ordered monoid with a compatible infinity element. Formally, $\mathcal{A} = (W, \oplus, \preceq)$, where W is the set of (abstract) weights that can be assigned to edges with a special infinity weight $\phi \in W$ meaning that an edge/walk is not traversable, $\oplus$ is a composition operator for weights, and $\preceq$ is weight comparison.

Formally, the following properties are presumed.

1. $(W, \oplus)$ is a monoid (semigroup with identity) with zero element:
   - Closure: $w_1 \oplus w_2 \in W$ for all $w_1, w_2 \in W$.
   - Associativity: $(w_1 \oplus w_2) \oplus w_3 = w_1 \oplus (w_2 \oplus w_3)$ for all $w_1, w_2, w_3 \in W$.
   - Identity element: $\exists 0 \in W$ such $0 \oplus w = w \oplus 0 = w$ for every $w \in W$.
   - Zero element: $\exists \phi \in W$ such $\phi \oplus w = w \oplus \phi = \phi$ for every $w \in W$.
2. $\preceq$ is a total order on W with a maximal element:
   - Reflexivity: $w \preceq w$ for any $w \in W$.
   - Anti-symmetry: if $w_1 \preceq w_2$ and $w_2 \preceq w_1$, then $w_2 = w_1$ for any $w_1, w_2 \in W$.
   - Transitivity: if $w_1 \preceq w_2$ and $w_2 \preceq w_3$, then $w_1 \preceq w_3$ for any $w_1, w_2, w_3 \in W$.
   - Totality: for all $w_1, w_2 \in W$ either $w_1 \preceq w_2$ or $w_2 \preceq w_1$.
   - Maximal element: $\exists \infty \in W$ such $w \preceq \infty$ for all $w \in W$.
3. $\preceq$ and $\oplus$ are consistent:
   - The zero element for $\oplus$ is the maximal element for $\preceq$: $\phi = \infty$.

We sometimes use the shorthand notation ab instead of $a \oplus b$.

Given a walk $q = (v_1, v_2, \ldots, v_k)$, we obtain the weight $w(q)$ of q by combining the weights of its constituent edges:

$$w(q) = \bigoplus_{i=1}^{k-1} w(v_i, v_{i+1}) \ .$$

With this notation at hand, the preferred walk q^* in $\mathcal{A}$ between nodes s and t is simply the one with the smallest weight from the set of all $s - t$ walks $\mathcal{Q}_{st}$ over the relation $\preceq$. The path selection problem over $\mathcal{A}$ can take two alternative forms, based on whether we allow the output to be a walk or we require it to be strictly a path.

Definition 1 *$Q_{\mathcal{A}}(G,s,t)$ (Preferred walk computation problem): given a graph $G(V,E)$ with weight function $w : E \to W$ and a node pair $s,t \in V$, return an $s-t$ walk q^* so that*

$$w(q^*) \preceq w(q) \text{ for all } q \in \mathcal{Q}_{st} \ . \tag{1}$$

We further define the *preferred-path computation problem* $P_{\mathcal{A}}(G,s,t)$ similarly as above, whereas we also require the output to be a path. In addition, with a slight abuse of notation we define the "decision-form" of the problems $Q_{\mathcal{A}}(G,s,t,b)$ ($P_{\mathcal{A}}(G,s,t,b)$), where the task is to decide whether a walk q^* (path p^*) exists with weight $w(q^*) \preceq b$ (resp. $w(p^*) \preceq b$) for some given bound $b \in W$. Trivialy, $P_{\mathcal{A}}(G,s,t,b)$ is in NP but $Q_{\mathcal{A}}(G,s,t,b)$ is not necessarily, as the size of a preferred path is always polynomial as the function of the input but the size of the preferred walk is not.

The lexicographic product operator is a useful tool to compose complex routing algebras from simple ones [5]. Given two routing algebras $\mathcal{A} = (W_{\mathcal{A}}, \oplus_{\mathcal{A}}, \preceq_{\mathcal{A}})$ and $\mathcal{B} = (W_{\mathcal{B}}, \oplus_{\mathcal{B}}, \preceq_{\mathcal{B}})$, the *lexicographic product* of $\mathcal{A}$ and $\mathcal{B}$ is a routing algebra $\mathcal{A} \times \mathcal{B} = (W, \oplus, \preceq)$, where the weight composition operator is defined as $(w_1,v_1) \oplus (w_2,v_2) = (w_1 \oplus_{\mathcal{A}} w_2, v_1 \oplus_{\mathcal{B}} v_2)$ for all $w_1, w_2 \in W_{\mathcal{A}}$ and $v_1, v_2 \in W_{\mathcal{B}}$, and

$$(w_1,v_1) \preceq (w_2,v_2) = \begin{cases} v_1 \preceq_{\mathcal{B}} v_2 & \text{if } w_1 =_{\mathcal{A}} w_2 \\ w_1 \preceq_{\mathcal{A}} w_2 & \text{otherwise} \end{cases}$$

One easily checks that shortest path routing corresponds to the algebra $\mathcal{S} = (\mathbb{N}, \infty, +, \leq)$, while widest path routing, where preferred paths are those with the largest bottleneck capacity, is $\mathcal{W} = (\mathbb{N}, 0, \min, \geq)$ [14]. The lexicographic products of $\mathcal{S}$ and $\mathcal{W}$ also correspond to practically relevant routing policies: widest-shortest path routing $\mathcal{WS} = \mathcal{S} \times \mathcal{W}$ prefers from the set of shortest paths the one with the highest free capacity [1], and shortest-widest path $\mathcal{SW} = \mathcal{W} \times \mathcal{S}$, just contrarily, prefers the shortest from the set of widest paths [8, 16].

3 When preferred paths and walks coincide

Our aim in this paper is to study the computational complexity of the preferred-walk computation problem $Q_{\mathcal{A}}$ and the preferred-path computation problem $P_{\mathcal{A}}$, in order to extend the traditional algebraic characterization to new routing architectures and policies. For this, first we have to make clear under which conditions we can at least hope for a solution. This is the goal in this section.

There are the following three cases: *(i)* there is no solution for $Q_{\mathcal{A}}$ but for $P_{\mathcal{A}}$ there is (e.g., if $\mathcal{A}$ is the shortest path algebra with negative cycles); *(ii)* both $Q_{\mathcal{A}}$ and $P_{\mathcal{A}}$ are solvable but the solutions differ (see Fig. 2); and *(iii)* both problems are solvable with equal result. Clearly, from a practical standpoint case *(iii)* is favorable, as in this case a simple preferred walk computation algorithm will readily deliver the preferred path as well (after removing cycles). Thus, in this section we give a sufficient and necessary condition for *(iii)* to hold.

Consider the following definitions:

– Non-Decreasing ($\mathbb{ND}$): $w_1 \preceq w_1 \oplus w_2$ and $w_1 \preceq w_2 \oplus w_1$ for every $w_1, w_2 \in W$.

- Strongly Non-Decreasing ($\mathbb{SND}$): non-decreasing with $w_1 \oplus w_3 \preceq w_1 \oplus w_2 \oplus w_3$ for every $w_1, w_2, w_3 \in W$.
- Monotone ($\mathbb{M}$): $w_1 \preceq w_2$ implies $w_1 \oplus w_3 \preceq w_2 \oplus w_3$ and $w_3 \oplus w_1 \preceq w_3 \oplus w_2$ for every $w_1, w_2, w_3 \in W$.
- Commutative ($\mathbb{C}$): $w_1 \oplus w_2 = w_2 \oplus w_1$ for every $w_1, w_2 \in W$.
- Polynomial Finite ($\mathbb{PF}$): $f_{\mathcal{A}}(n) = O(p(n))$ where $p()$ is a polynomial and

$$f_{\mathcal{A}}(n) \;=\; \sup\{|\mathcal{B}| \;:\; \mathcal{B} \text{ is a subalgebra of } \mathcal{A} \text{ generated by } n \text{ elements}\} \;.$$

The properties $\mathbb{C}$, $\mathbb{ND}$, and $\mathbb{M}$ are familiar from the literature [4, 14]. In addition, we introduce two new properties.

First, $\mathbb{PF}$ formalizes the idea that the subalgebras generated by any small set of elements of an algebra are not "too large". In particular, let $\mathcal{A} = (W, \oplus, \preceq)$ and let $W_G \subset W$ denote the set from which the links of a graph G take their weights from (easily, $|W_G| = O(n)$). Then, the number of weights that can be assigned to any walk of length n is at most $f_{\mathcal{A}}(n)$, and the $\mathbb{PF}$ property simply requires that this quantity is polynomial in the size of the network (i.e., n). For example, in the case of the widest path routing algebra $\mathcal{W}$ the weight of a walk will be one of its constituting link weights, and if there are n different link weights then the size of their generated algebra is n, hence $f_{\mathcal{W}}(n) = n$. Later, we shall see that this property is particularly useful to define fast path selection algorithms.

Second, our $\mathbb{SND}$ property extends $\mathbb{ND}$ to non-commutative algebras. The $\mathbb{ND}$ property in fact guarantees that, at least for commutative algebras, *(i)* every preferred path is a preferred walk and *(ii)* a preferred path can be obtained from a preferred walk by dropping all cycles[7]. For non-commutative algebras, however, we need $\mathbb{SND}$ for *(i)* and *(ii)* to hold, as shown below. (Note that $\mathbb{SND} \Leftrightarrow \mathbb{ND}$ over commutative algebras.)

Lemma 1. *For any graph $G(V, E)$ with weight function w, if $\mathcal{A}$ is $\mathbb{SND}$ then $Q_{\mathcal{A}}(G, s, t)$ is solvable for any $s, t \in V$.*

Proof. First, we introduce some notation. Let Σ be a finite alphabet, let S_Σ denote the set of finite words on Σ, and let $(S_\Sigma, \lhd)$ be a partially ordered set (poset) where $x \lhd y$ for some $x, y \in S_\Sigma$ if x can be obtained from y by deleting symbols. In addition, we define an antichain of $(S_\Sigma, \lhd)$ as a set $R \subseteq S_\Sigma$ so that neither $x \lhd y$ nor $y \lhd x$ holds for any $x, y \in R, x \neq y$. We use the following claim from [7, page 106-107]:

Proposition 1 *All antichains of $(S_\Sigma, \lhd)$ are finite.*

Now, suppose indirectly that there exists a $\mathbb{SND}$ algebra $\mathcal{A}$, a graph $G(V, E)$ with weight function w, and $s, t \in V$, such that there is no preferred walk from s to t. Hence, there is an infinite sequence of walks $s_i : i \in \mathbb{N}$ in G with $w(s_i) \succ w(s_{i+1})$ and obviously this has an infinite subsequence such that the length of s_i is strictly less than that of s_{i+1} for any $i \in \mathbb{N}$. By choosing $\Sigma = \{w(e) : e \in E\}$

[7] Note that the known path selection algorithms (e.g. Dijkstra, Bellmann-Ford [12]) in fact search for a walk, but this walk is guaranteed to also be a path whenever the underlying routing algebra is $\mathbb{SND}$, like e.g., the shortest-path algebra $\mathcal{S}$.

and applying Proposition 1 to the poset $(s_n : n \in \mathbb{N}, \lhd)$, we find at least one pair of walks s_i and s_j with $i < j$ and $s_i \lhd s_j$. Observe that, by $i < j$ and transitivity of $\preceq$, we have $s_i \succ s_j$. On the other hand, by applying $\mathbb{SND}$ repeatedly we have $s_i \preceq s_j$, which contradicts $s_i \succ s_j$.

Theorem 1 *For any graph $G(V, E)$ with weight function w and any $s, t \in V$,* (i) *solution q^* to $Q_{\mathcal{A}}(G, s, t)$ and solution p^* to $P_{\mathcal{A}}(G, s, t)$ both exist and* (ii) *$w(q^*) = w(p^*)$, if and only if $\mathcal{A} \in \mathbb{SND}$.*

Proof. According to Lemma 1 if $\mathcal{A}$ is $\mathbb{SND}$ then a solution q^* to $Q_{\mathcal{A}}$ always exists. ($P_{\mathcal{A}}$ always has a solution, as the set of $s-t$ paths is finite and $\preceq$ is total.) From this the statement $w(q^*) = w(p^*)$ will follow, since $\mathbb{SND}$ guarantees that deleting a cycle from q^* to obtain p^* cannot increase the weight (see Fig. 2).

To see the reverse direction, observe that if $\mathcal{A}$ is not $\mathbb{SND}$ then there are $a, b, c \in \mathcal{A}$ with $abc \prec ac$ and then one easily constructs a graph on which $q^* \neq p^*$ (see again Fig. 2).

Corollary 1. *If $\mathcal{A}$ algebra is $\mathbb{PF}$, then $Q_{\mathcal{A}}(G, s, t, b)$ is in NP.*

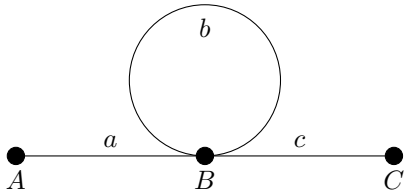

Fig. 2: Illustration of the $\mathbb{SND}$ property: $\mathbb{SND}$ guarantees that if we remove the cycle(s) from a preferred $A - C$ walk of weight abc then what we obtain is a preferred path since $ac \preceq abc$. On the other hand, if $\mathcal{A}$ is not $\mathbb{SND}$ then $abc \prec ac$ for some $a, b, c \in \mathcal{A}$ and on the above graph the preferred $A - C$ walk abc does not coincide with the preferred $A - C$ path ac.

We close this discussion by noting that $\mathbb{C} \wedge \mathbb{ND} \Rightarrow \mathbb{SND}$ and $\mathbb{ND} \wedge \mathbb{M} \Rightarrow \mathbb{SND}$. The proofs are trivial and omitted here.

4 Polynomial time algorithms

Now, we are in the position to draw the algebraic silhoutette of routing policies that admit a polynomial time path selection algorithm. This work was spearheaded by Sobrinho with the following claim [14]:

Proposition 2 *If $\mathcal{A} \in \mathbb{ND} \cap \mathbb{M}$, then $Q_{\mathcal{A}}$ and $P_{\mathcal{A}}$ can be solved in polynomial time by the generalized Dijksta algorithm.*

Note that Sobrinho's algebraic framework differs somewhat from ours, in that we do not require $\mathbb{M}$ neither minimality of 0. However, it is easy to see that $\mathbb{ND}$ implies the minimality of 0, and $\mathbb{M}$ and the minimality of 0 imply $\mathbb{ND}$. Also note that the result holds only if the primitive operations $\oplus$ and $\preceq$ are $O(1)$. We shall use this assumption in the rest of the paper.

Sobrinho's result is important in that it guarantees a polynomial algorithm for many practically important policies (see Section 6). Unfortunately, it does not cover some crucial non-monotone cases (like the valley-free routing algebra of BGP [4]) and many exotic routing policies in SDN and service chaining (again, see Section 6). Our next result extends the family of routing policies for which polynomial time algorithm is warranted to the set of polynomial finite algebras.

Theorem 2 *If $\mathcal{A} \in \mathbb{PF}$ then for any $G(V, E)$ with weight function w and $s, t \in V$, $Q_{\mathcal{A}}(G, s, t)$ can be solved in time polynomial in $|V|$.*

Proof. Let $\mathcal{B}$ denote the sub-algebra of $\mathcal{A}$ generated by $\{w(e) : e \in E\}$. Construct a new unweighted digraph $G_{\mathcal{B}}(V_{\mathcal{B}}, E_{\mathcal{B}})$ with $V_{\mathcal{B}} = V \times \mathcal{A}$ and

$$E_{\mathcal{B}} = \{((v, a), (u, b)) : a \oplus w(v, u) = b\} \ .$$

Clearly, there is a walk in $G_{\mathcal{B}}$ from $(s, 0)$ to (t, r) if and only if there is a walk in G from s to t with weight r. Algorithm 1 below uses this observation to solve $Q_{\mathcal{A}}(G, s, t)$.

Algorithm 1 Preferred walk computation for $\mathbb{PF}$ algebras

1: Determine algebra $\mathcal{B}$.
2: Build graph $G_{\mathcal{B}}$.
3: Run a BFS traversal on $G_{\mathcal{B}}$ from $(s, 0)$ to every node in $\{(t, a) : a \in \mathcal{B}\}$.
4: Use binary search over $\preceq$ to find the least element of
$\{a \in \mathcal{B} :$ there is a walk from $(s, 0)$ to (t, a) in $G_{\mathcal{B}}\}$.

The complexity of the algorithm is as follows. Let $n = |V|$ and let $m = |E|$. Since $\mathcal{A}$ is $\mathbb{PF}$ and $\mathcal{B}$ is generated by m elements, there is a polynomial $p(m)$ so that $|\mathcal{B}| = O(p(m))$. Easily, line 1 takes $|V_{\mathcal{B}}| = O(p(m)n)$ time, line 2 and line 3 run in $O(E_{\mathcal{B}}) = O((p(m)n)^2)$ time, and finally line 4 terminates in $O(\log(|\mathcal{B}|))$ time, so the total running time is a polynomial in n as $m = O(n^2)$.

Observe that the $\mathbb{PF}$ property guarantees only that we can compute the preferred *walk* in polynomial time using Algorithm 1. To actually obtain the preferred *path*, we also need the $\mathbb{SND}$ property in addition to $\mathbb{PF}$, which, by Theorem 1, states that the latter can be obtained from the former in polynomial time by deleting cycles.

The question remains how to actually check whether a particular routing algebra is $\mathbb{PF}$. Easily, the naive approach will not work as the number of subalgebras of a routing algebra can be very large. As we shall see, the below algebraic properties are particularly useful to detect whether or not an algebra is $\mathbb{PF}$:

- Selective: $w_1 \oplus w_2 \in \{w_1, w_2\}$ for each $w_1, w_2 \in W$.
- Left-condensed: $w_1 \oplus w_2 = w_1 \oplus w_3$ for each $w_1, w_2, w_3 \in W$.

Lemma 2. *If $\mathcal{A}$ is a selective then $\mathcal{A}$ is $\mathbb{PF}$.*

Proof. If $\mathcal{A}$ is selective then every subset $\mathcal{B} \subseteq \mathcal{A}$ is closed under $\oplus$ and so $\mathcal{B}$ is a subalgebra. Thus, if $|\mathcal{B}| = n$ then the subalgebra generated by $\mathcal{B}$ is exactly $\mathcal{B}$, so it also holds n elements and hence $f_{\mathcal{A}}(n) = n$.

Lemma 3. *If $\mathcal{A}$ is a left-condensed then $\mathcal{A}$ is $\mathbb{PF}$.*

Proof. Suppose that $\mathcal{A}$ is left-condensed and let $\mathcal{B} \subseteq \mathcal{A}$ such that $|\mathcal{B}| = n$. Then, the subalgebra generated by $\mathcal{B}$ is $\{0, \infty\} \cup \mathcal{B} \cup \mathcal{B}^2$, it has at most $2n+2$ elements, and thus $f_{\mathcal{A}}(n) = 2n + 2$ is again polynomial.

From Lemma 2 it immediately follows that widest path algebra $\mathcal{W}$ is $\mathbb{PF}$, due to it being selective [11].

Table 1: Algebraic properties and computational complexity (of the preferred path selection problem) for some routing policies.

Algebra	Notation/Definition	Properties	$f_{\mathcal{A}}(n)$	Complexity
Shortest path	$\mathcal{S} = (\mathbb{N}, \infty, +, \leq)$	$\mathbb{ND}$, $\mathbb{M}$, $\mathbb{C}$, $\mathbb{SND}$	–	Polynomial
Widest path	$\mathcal{W} = (\mathbb{N}, 0, \min, \geq)$	$\mathbb{ND}$, $\mathbb{M}$, $\mathbb{C}$, $\mathbb{PF}$, $\mathbb{SND}$	n	Polynomial
Most reliable path	$\mathcal{R} = ((0,1], 0, *, \geq)$	$\mathbb{ND}$, $\mathbb{M}$, $\mathbb{C}$, $\mathbb{SND}$	–	Polynomial
Usable path	$\mathcal{U} = (\{0,1\}, 0, *, \geq)$	$\mathbb{ND}$, $\mathbb{M}$, $\mathbb{C}$, $\mathbb{PF}$, $\mathbb{SND}$	2	Polynomial
Widest-shortest path	$\mathcal{WS} = \mathcal{S} \times \mathcal{W}$	$\mathbb{ND}$, $\mathbb{M}$, $\mathbb{C}$, $\mathbb{SND}$	–	Polynomial
Shortest-widest path	$\mathcal{SW} = \mathcal{W} \times \mathcal{S}$	$\mathbb{ND}$, $\mathbb{C}$, $\mathbb{SND}$	–	Polynomial
SAT	$\mathcal{SAT}$	$\mathbb{ND}$,$\mathbb{C}$, $\mathbb{SND}$	$2^{3n}+1$	NP-complete
Constrained shortest path	$\mathcal{CS}$	$\mathbb{C}$	–	NP-complete
Longest path	$\mathcal{L} = (\mathbb{N}, \infty, +, \geq)$	$\mathbb{M}$, $\mathbb{C}$	–	NP-complete
Valley-free routing	$\mathcal{VF}$	$\mathbb{ND}$, $\mathbb{PF}$, $\mathbb{SND}$	6	Polynomial
Proxy	$\mathcal{X}$	$\mathbb{ND}$,$\mathbb{PF}$	–	Polynomial
Static Service-chaining	$\mathcal{SC}$	$\mathbb{PF}$	$n(n-1)/2$	Polynomial
Fixed Service-chaining	$\mathcal{F}_k$	$\mathbb{PF}$, $\mathbb{C}$	$k+2$	Polynomial

5 NP-hardness results

It seems that the family of routing policies for which a polynomial time path selection algorithm is available is reassuringly broad. Currently, it is an open question whether the algebraic characterization provided by Proposition 2 and Algorithm 1 can be broadened further (see Fig. 1). Below, we argue that this is non-trivial, because even as simple as commutative and non-decreasing algebras *can* already induce intractable path selection instances. Again, our characterization is algebraic, which allows us to sidestep piecemeal treatment for each new policy arising in practice. Instead, we identify entire "unsafe" algebraic regions where protocol designers may accidentally hit an NP-hard instance.

Theorem 3 *There is an algebra $\mathcal{A} \in \mathbb{M} \cap \mathbb{C}$, so that $P_{\mathcal{A}}(G, s, t, b)$ is NP-complete.*

Proof. The well-known NP hard longest path problem [12] can be trivialy formalized in this language. $\mathcal{L} = (\mathbb{N}, \infty, +, \geq)$.

Theorem 4 *There is an algebra $\mathcal{A} \in \mathbb{SND} \cap \mathbb{C}$, so that $Q_{\mathcal{A}}(G, s, t, b)$ and $P_{\mathcal{A}}(G, s, t, b)$ are NP-complete.*

Proof. We show an algebra, $\mathcal{SAT}$, and corresponding Karp-reductions 3-SAT $\propto Q_{\mathcal{SAT}}$ and 3-SAT $\propto P_{\mathcal{SAT}}$. Let a φ be a 3-CNF formula on some set of boolean variables X. Note that if $x \in X$ then $\neg x \in X$ and $\forall x \in X : \neg(\neg x) = x$. Let φ be a conjunction on n clauses $\varphi_1, ..., \varphi_n$ and $\varphi_i = \varphi_{i,a} \vee \varphi_{i,b} \vee \varphi_{i,c}$, where $\varphi_{i,a}, \varphi_{i,b}, \varphi_{i,c} \in X$ for every $i = 1, ..., n$.

We define $\mathcal{SAT}$ as follows. Let S be the power set of X endowed with an infinity element: $S = 2^X \cup \{\infty\}$, and let $\oplus$ be binary operation on S so that

$$\forall a, b \in S \setminus \{\infty\} : a \oplus b = \begin{cases} \infty & \text{if } \exists v \in a : \neg(v) \in b \\ a \cup b & \text{otherwise} \end{cases}$$

$$\forall a \in S : \infty \oplus a = a \oplus \infty = \infty \ .$$

Since $\mathcal{SAT}$ is $\mathbb{SND}$, it is enough to proove that $P_{\mathcal{A}}(G, s, t, b)$ is NP-hard.

We define a multi-graph $G(V, E)$ so that solving $Q_{\mathcal{SAT}}$ on G solves 3-SAT. Let $V = \{0, ..., n\}$, let $E = \{e_{1,a}, e_{1,b}, ..., e_{n,b}, e_{n,c}\}$, where $e_{i,j}$ is a directed edge from v_{i-1} to v_i and $j \in \{a, b, c\}$, and let $w : E \to \mathcal{SAT}$ be a weight function so that $w(e_{i,j}) = \varphi_{i,j}$ for all $i \in \{1, ..., n\}$ and $j \in \{a, b, c\}$. Clearly, G can be built in time polynomial in n. Now, one easily proves the following statement: the weight of the preferred $0 \to n$ path in G is less then ∞ if and only if φ is satisfiable. This completes the proof.

For convenience, we used a multigraph with parallel arcs in the proof, but it is trivial to rewrite it in terms of simple graphs by adding an artificial identity element to $\mathcal{SAT}$ and substituting every parallel arc with a two-hop path with one of the arcs having the identity element as weight. Note also that the NP-hard constrained shortest path problem ($\mathcal{CS}$), which is also in $\mathbb{C} \cap \mathbb{SND}$, [6] can also be posed as a path selection problem over a special algebra.

Our results indicate that each of the algebraic regimes $\mathbb{ND}$ (and $\mathbb{SND}$), $\mathbb{C}$, and $\mathbb{M}$ contain algebras with intractable path selection instances. Even the restrictions $\mathbb{SND} \cap \mathbb{C}$ and $\mathbb{M} \cap \mathbb{C}$ are unsafe from a practical standpoint, in that the related path selection problem might easily end up being prohibitive to solve in practice. This means that designing future routing policies requires extreme care [13].

As a corollary, we state the following.

Corollary 2. *If for a routing policy the preferred path computation problem over cannot be solved by the Dijkstra algorithm, then the existence of a polynomial time algorithm is not guaranteed.*

6 Discussion

The implications of our results are summarized in Table 1, and Fig. 1 provides a comprehensive computational complexity classification. Here, $\mathcal{SAT}$ is defined as above, and $\mathcal{S}$, $\mathcal{WS}$, $\mathcal{R}$, $\mathcal{U}$, and $\mathcal{W}$ are from the literature [14]. As the latter five are $\mathbb{ND}$ and $\mathbb{M}$, the corresponding path-selection problems are polynomially tractable by Proposition 2.

An important example for a policy for which the generalized Dijkstra algorithm is not correct is valley-free routing, used extensively in the Internet inter-domain routing system [4]. Here, links are labeled according to the business relationship they represent as either *provider* (p), *customer* (c), or *peer* (r), and a preferred "valley-free" path is one that does not contain "economically unreasonable" portions (i.e., cp, cr, rp, or rr sub-paths). This is described by the algebra $\mathcal{VF} = (W, \oplus, \preceq)$ (adopted from [4]), where $W = \{0, c, r, p, prc, \infty\}$ (prc is introduced to maintain associativity), $\oplus$ is defined by the Cayley table below, and the precedence is $0 \preceq c \preceq r \preceq p \preceq prc \preceq \infty$. Note that $\mathcal{VF}$ is not $\mathbb{M}$ nor $\mathbb{C}$, but it is $\mathbb{PF}$ so Algorithm 1 warrants the availability of a tractable path selection algorithm.

We wish to point out that the difference between $\mathbb{ND}$ and $\mathbb{SND}$ is, although subtle, not arbitrary. It is often the case in inter-domain routing, for instance, that some entity c rejects any packet received directly from a unless it has

also been proxied through a third-party, say, b. In practice, such policy considerations are realized in the control plane, encoded by sophisticated BGP route filtering rules and BGP communities. The corresponding proxy algebra is $\mathcal{X} = (\{a, b, c\}, \oplus, \preceq)$, where $\oplus$ is the usual concatenation operator with the restriction that $a \oplus c = \infty$ and $\preceq$ is essentially arbitrary[8]. Then, $\mathcal{X}$ is $\mathbb{ND}$ but not $\mathbb{SND}$, as $a \oplus b \oplus c \prec \infty = a \oplus c$, and hence solving $P_{\mathcal{X}}$ is far from trivial. However, it easy to convert $\mathcal{X}$ to $\mathbb{PF}$, so again Algorithm 1 warrants polynomial tractability.

To demonstrate how new architectures bring up exotic new routing algebras, consider a simple service chaining example [10]. Here there are three functions, network address translation (n), deep packet inspection (d) and an virus scanning (v), and each flow must pass through each function exactly once and exactly in the order n, d, v. The corresponding algebra is $\mathcal{SC} = (\{0, n, d, v, nd, dv, ndv, \infty\}, \oplus, \preceq)$, where $\oplus$ is as

$\oplus$	0	c	r	p	prc	∞
0	0	c	r	p	prc	∞
c	c	c	∞	∞	∞	∞
r	r	prc	∞	∞	∞	∞
p	p	prc	prc	p	prc	∞
prc	prc	prc	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞

$\oplus$	0	n	d	v	nd	dv	ndv	∞
0	0	n	d	v	nd	dv	ndv	∞
n	n	∞	nd	∞	∞	ndv	∞	∞
d	d	∞	∞	dv	∞	∞	∞	∞
v	v	∞	∞	∞	∞	∞	∞	∞
nd	nd	∞	∞	ndv	∞	∞	∞	∞
dv	dv	∞	∞	∞	∞	∞	∞	∞
ndv	ndv	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞

$\mathcal{SC}$ is not $\mathbb{ND}$ as $n \oplus d \oplus v \prec n \oplus d$, but it is $\mathbb{PF}$ so Algorithm 1 again gives a polynomial time complexity characterization.

In many upcoming service chaining deployments, the exact function realized by a particular middlebox is not fixed but instead freely configurable by the network operator or an SDN controller [9]. For instance, a service point may realize network address translation or deep packet inspection (but not both) at different points in time, and switching between the two can be done by downloading the adequate rules to the middlebox. In this dynamic service chaining architecture, the preferred-path computation problem $P_{\mathcal{A}}(G, s, t)$ involves both deciding which particular functionality to realize at a particular service point in G *and* to actually compute the preferred path itself between s and t. Thinking of the previous service chaining example, middleboxes are now wild cards, capable to realize network address translation (n), deep packet inspection (d), or virus scanning (v), and our task is to find route, from s to t, *through exactly* 3 *middleboxes*. Once we have the route, we can switch the traversed middleboxes to the required functionality.

This scenario perhaps best demonstrates how careful one must be when defining new routing policies. First, assume that the number of service points k a packet must traverse is not restricted. In this case, in an n node network we can easily require the packet to pass through exactly $n-2$ service points, and hence our path selection problem boils down to a Hamiltonian path problem, a famous

[8] Note that in our model link weights are assigned to links and not to nodes as would be needed in this and the subsequent scenarios, but it is easy to convert between the two cases.

NP-hard problem. Or, if we require the packet to visit *at least* k middleboxes we arrive to the longest path problem $\mathcal{L}$. If, in addition, it is also a requirement to minimize the cost (or maximize the bandwidth) of the path, then the resultant algebra is equivalent to $\mathcal{L} \times \mathcal{S}$ ($\mathcal{L} \times \mathcal{W}$, resp.), problems that seem even more difficult.

Based on these considerations, it is plausible to impose a strict upper bound on the number of services a packet is required to visit. Let this bound be k. The difference from the above setting is that now k is a fixed constant and it is not allowed to vary with the input, and hence the above simple reductions to NP-hard problems, where k is part of the input, do not apply. Suppose now that the task is to route a packet through at most k middleboxes[9]. Then, we arrive to the algebra $\mathcal{F}_k = (W, \oplus, \preceq)$, where $W = [0, k] \cup \infty$ and

$$a \oplus b = \begin{cases} a + b & \text{if } a + b \leq k \\ \infty & \text{otherwise} \end{cases}$$
$$\infty \oplus a = a \oplus \infty = \infty \ .$$

The ordering is $k \preceq k - 1 \preceq \ldots 1 \preceq 0$. Easily, $\mathcal{F}_k$ is $\mathbb{C}$, and since it is also $\mathbb{PF}$ the path selection problem in this case can be solved with Algorithm 1. Moreover, the related lexicographic products $\mathcal{F}_k \times \mathcal{S}$ and $\mathcal{F}_k \times \mathcal{W}$ can also be solved in polynomial time.

Finally, we note that there are some practically important routing policies which fall outside our characterization. For instance, $\mathcal{SW}$ is not $\mathbb{M}$ therefore the Dijkstra algorithm does not work, neither it is $\mathbb{PF}$ so Algorithm 1 does apply either. For this particular case a special algorithm guarantees polynomial time path selection [16], but further extending the algebraic classification presented in this paper to the general case (if at all possible) is currently an open problem.

7 Conclusion

Routing theory is often counted as a "cold" research area [3], suggesting that we can sit back and relax knowing that the major questions that can be raised in connection with routing are more or less well answered. However, it turns our that the latest developments concerning the core philisophy of networking (data centers, SDN, service chaining, etc.) pose considerable challenge for today's routing theory. We have shown that new routing policies are emerging at the near horizon, which may for instance embrace routing loops to facilitate meeting strict policy considerations, whereas in today's routing theory loops count as heresy.

The main message of this paper is to point out that there is still much to do out there and it is time to rehash routing theory to cope with the upcoming challenges. We have taken the first steps towards realizing this ambitious goal. We have extended the algebraic policy routing theory with a sufficient

[9] The settings when we require the packet to meat *at least* k middleboxes or *exactly* k middleboxes are handled similarly.

and necessary characterization for the preferred-walk and preferred-path selection problems to be both solvable with the same output and we have provided a comprehensive classification of routing policies based on the computational complexity of the corresponding path selection problem. Our findings indicate that defining routing policies in these upcoming routing architectures requires extreme forethought [13], as seemingly simple routing policies, even as simple as commutative, non-decreasing, and monotone ones, can easily give rise to intractable routing problems.

Acknowledgements

Gábor Rétvári was supported by the OTKA/PD-104939 grant. This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013) and the High Speed Networks Laboratory.

References

1. Apostolopoulos, G., Guerin, R., Kamat, S., Tripathi, S.K.: Quality of service based routing: A performance perspective. In: SIGCOMM. pp. 17–28 (1998)
2. Caesar, M., Rexford, J.: BGP routing policies in ISP networks. Tech. Rep. UCB/CSD-05-1377, EECS Department, University of California, Berkeley (2005)
3. Crowcroft, J.: Cold topics in networking. SIGCOMM Comput. Commun. Rev. 38(1), 45–47 (Jan 2008)
4. Griffin, T., Sobrinho, J.: Metarouting. In: SIGCOMM '05. pp. 1–12 (2005)
5. Gurney, A., Griffin, T.: Lexicographic products in metarouting. In: Network Protocols, IEEE International Conference on. pp. 113–122 (2007)
6. Handler, G.Y., Zang, I.: A dual algorithm for the constrained shortest path problem. Networks 10(4), 293–309 (1980)
7. Lothaire, M.: "Combinatorics on Words". Cambridge Mathematical Library (1997)
8. Ma, Q., Steenkiste, P.: On path selection for traffic with bandwidth guarantees. In: Proceedings of the 1997 International Conference on Network Protocols (ICNP '97). p. 191 (1997)
9. Qazi, Z.A., Tu, C.C., Chiang, L., Miao, R., Sekar, V., Yu, M.: SIMPLE-fying middlebox policy enforcement using SDN. In: Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. pp. 27–38. SIGCOMM '13 (2013)
10. Quinn, P.: Network service chaining problem statement. Internet draft (2013)
11. Rétvári, G., Gulyás, A., Heszberger, Z., Csernai, M., Bíró, J.J.: Compact policy routing. In: Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing. pp. 149–158. PODC '11 (2011)
12. Sedgewick, R., Wayne, K.: Algorithms. Pearson Education (2011), http://books.google.hu/books?id=idUdqdDXqnAC
13. Seehra, A., Naous, J., Walfish, M., Mazieres, D., Nicolosi, A., Shenker, S.: A policy framework for the future Internet. HotNets-VIII (2009)
14. Sobrinho, J.: Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet. IEEE/ACM Trans. Netw. 10, 541–550 (August 2002)
15. Sobrinho, J.: Network routing with path vector protocols: theory and applications. In: SIGCOMM '03. pp. 49–60 (2003)
16. Wang, Z., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications. IEEE J.Sel. A. Commun. 14(7), 1228–1234 (Sep 2006)
17. Younis, O., Fahmy, S.: Constraint-based routing in the Internet: Basic principles and recent research. Communications Surveys Tutorials, IEEE 5(1), 2–13 (2003)