Rapid Convergence versus Policy Expressiveness in Interdomain Routing

Alexander J. T. Gurney
Comcast
alexander_gurney@cable.comcast.com

Sanjeev Khanna
University of Pennsylvania
sanjeev@cis.upenn.edu

Yang Li
University of Pennsylvania
yangli2@cis.upenn.edu

***Abstract*—In interdomain routing, competing network operators encode policies about possible routes in routing protocol configuration. The operation of the protocol should lead to satisfactory routes for all operators, but this process may not terminate or take a long time, exploring exponentially many alternative paths before stabilizing. In this paper, we study convergence for the partial policy specification model where preferences are set for only some paths and the ranking for the remaining paths is indifferent to the network operator. We consider policy restrictions that ensure a network to stabilize quickly. Specifically, we show that even when each operator only specifies preferences for two paths and each path has at most three hops, a network may still encounter exponentially many steps before convergence. However, restricting the policy any further ensures poly-time convergence. From another direction, it is well known that preferences based only on the 'next-hop' node always converge within linear-time. We show that even relaxing the preference to be based on the 'next-two-hop' leads to exponential-time convergence. Finally, we further study policy completion that leads to a stable state that minimizes the hop-length of the longest path, and establish a hardness result along with an approximation algorithm.**

I. Introduction

The Internet is a network of networks, connected via a common routing protocol which allows data paths to be found that meet the local policy of each network. This protocol is 'BGP', the Border Gateway Protocol [19]. Its most important difference from other route-finding methods is the way in which local policy controls the selection and propagation of routes: rather than there being a single global definition of 'best' route (as in shortest-path protocols), every constituent network has its own interpretation. This is because of the asymmetry in the economic relationships among these network participants; accordingly, BGP is best perceived as a protocol that tries to compute a Nash equilibrium in a game where all parties are trying to obtain good routes (by their local definition) but are constrained by the choices of others (one cannot pick a route through a neighbor without agreement) [17], [2].

BGP policy configuration can be extremely complex and nuanced. Preferences among possible network paths may be set arbitrarily. In practice, typical preferences are rather more structured. A standard configuration practice is for the first preference decision to be based on the identity of the neighbor from which the route was received; and then, among all routes coming from the most-favored neighbors, other characteristics such as path length are used to break ties [4], [1], [25]. Note that even in this restricted case, the preference classes over the neighbors may not be consistent across the entire network.

In general, network policies can conflict to the extent that BGP will be unable to find a stable solution: in this case, the protocol will *oscillate* indefinitely [13], [12], [16]. Furthermore, a set of policies may support the existence of multiple possible outcomes, which is typically also felt to be an error (because policy was not expressed well enough to yield the truly intended single outcome) [6]. There has been a great deal of work on identifying sufficient criteria for BGP to converge to a unique stable state [5], [7], [9], [23], [4], [22]. Unlike as previously suspected, the observed issues with convergence may not arise from router bugs or network anomalies, nor even from mistakes in the protocol definition, but may be *inherent* to the nature of the routing problem being solved. This was shown by the use of abstract models (in particular, *stable paths problems [8]*) displaying the same effects, without any of the complicating apparatus of BGP or its environment.

Even if BGP *does* converge, experience shows that it may take some time to do so [12], [3], and even in the absence of policy, when link failure occurs, the path-vector protocol used by BGP may still need exponential time to converge [11]. The slow convergence causes practical difficulties for network operators, and degradation or loss of service for their customers. The technology of 'route flap damping', which modifies the timing behavior of BGP in order to avoid propagating temporary oscillations, has been developed [24], criticized [15], and readjusted [18], but in a purely empirical fashion that does not address the root cause of delayed convergence.

The main focus in this paper is on the theoretical aspect of the convergence *time* of BGP. In particular, we aim to understand *when convergence is guaranteed*, how much time it requires for the network to actually converge (or to stabilize). We study convergence time when different restrictions are applied on the structure of the preferences, and establish both necessary and sufficient conditions (i.e., dichotomy theorems) for convergence in *polynomial time*. To the best of our knowledge, prior to our work, the only related studies concerning polynomial time convergence are given by [4], [20], which only established polynomial time convergence for the Gao-

Rexford criteria[1].

Our results are backed by a general model of routing preference, based on the idea of partial policy specification [10], [25]. While BGP requires all paths to be ranked in a linear order (as otherwise it cannot choose a single best path in all circumstances), operators do not actually think of policy in this way. It is more natural to imagine the linear preference order as being determined by a combination of a general operator-determined policy, and subsequent tie-breaking actions that do not reflect 'genuine' preferences. The general policy, as implemented in standard BGP systems via match-action rules, amounts to partitioning the set of possible paths into disjoint classes: within a class, no preference is given, and between the classes, preferences might exist. For example, 'all routes from neighbor 17' could be a class, which is preferred to the class 'all routes from neighbor 4'.

We show that convergence is achieved when all possible routes can be put into a global linear order consistent with the given preferences (i.e., a linearization), which also matches known conditions for the existence of a unique stable solution [10]. We further establish that the construction of a linearization (when one exists), though the number of paths being linearized may be exponentially many, can always be done in time polynomial in the number of explicitly given preferences (Section III).

For the setting where, instead of linearizing, the routers only follow the specified partial preferences and act indifferently between unordered paths (hence never voluntarily switch to a path that is *not strictly better* than the current path), we present a detailed study of convergence time of networks (which are guaranteed to converge) with restricted preference systems. In particular, if each node only specifies preference over at most two paths, where each path has at most three hops, there still exist instances of networks that may take exponentially many (in the total number of nodes) steps before convergence. On the other hand, restricting the preference any further ensures poly-time convergence (Section IV). If a path's degree of preference is determined only by the identity of the next-hop neighbor, poly-time convergence is guaranteed [21]. However, using only the next two hops to determine preference already leads to instances that take exponentially many steps before convergence (Section V).

Finally, we observe that a given partial policy may admit many possible linearizations which may result in distinct stable states for the network. A natural question is if one can efficiently compute a linearization that results in a stable state with some desirable properties. We consider the problem of computing a linearization that minimizes the hop-length of the longest path in the resulting stable state. We establish a strong hardness result by showing that the problem is NP-hard to approximate to within a factor of $\Omega(n)$. To complement the hardness result, we further provide a poly-time algorithm that finds a path linearization where the length of the longest path only incurs an additive error of at most l, where l is the length of the longest path in the preference.

[1][20] also contains a general result regarding convergence time in terms of the number of *phases* (see Section III-A for the definition and more details).

Organization: The paper is structured as follows. After introducing some fundamental concepts and notation (Section II), we explore the linearization concept and its complexity (Section III). We then show how possible restrictions of preference expressivity affects convergence time (Section IV, V). Finally, we investigate the problem of completing policies that minimizes the length of the longest path (Section VI). We conclude (Section VII) by relating our results to prior work on the algorithmic complexity of BGP.

II. Preliminaries

In BGP, each network router (node) is capable of expressing independent preferences about its paths to each possible destination. It is well established that it suffices to focus on a single destination.

Definition II-.1 (Network)**.** *A* network N *is defined as a tuple* $\langle G(V,E), L, t\rangle$*, where* $G(V,E)$ *is a directed graph,* L *is the preference (or policies, see definition below), and* t *is the designated destination node.*

Throughout this paper, we denote by n the number of nodes in V. We assume that any $v \in V$ is connected to t in $G(V,E)$. Let $\mathcal{P}$ denote the set of all simple paths in $G(V,E)$ that terminate at t, and let $\mathcal{P}_v$ for each v in V, be the set of simple paths from v to t. The preference L contains path preferences expressed as partial orders on $\mathcal{P}_v$ for each node v in V.

Definition II-.2 (Preference)**.** *A preference* L_v *for a node* v *is a partial ordering* $\succ_v$ *on* all *simple paths from* v *to* t*,* P_v*. For any* $p, q \in P_v$*,* p *is 'better' than* q *iff* $p \succ_v q$*. Otherwise,* p *and* q *are unordered, and* v *is 'indifferent' between them.*

Generally, we write $p \succ_L q$ (or $p \succ q$ if L is clear from the context) if p is better than q for some v. In addition, we denote an empty path as ϵ, which is worse than any path in $\mathcal{P}$. We say a path p is *specified* in a preference L_v, denote by $p \in L_v$, if there exists a path q with $p \succ_v q$ or $q \succ_v p$.

In the protocol execution, each node tries to find a 'good' path to t, according to its own order, but subject to the requirement that it cannot choose a path unless the relevant neighbor has chosen the suffix of that path. Since forwarding is destination-based and hop-by-hop, a node may select a 'path', but data need not be constrained to follow that path if the other nodes along it have chosen differently. Therefore, it is more appropriate to say that a node chooses an outgoing edge to send traffic; it might continue to send traffic along that edge even after further network events have diverted the path. We now define the possible routing states in the protocol execution more precisely.

Definition II-.3 (State)**.** *A* state *of the protocol execution is a function* S *from* V *to* $E \cup \{\perp\}$*, such that for each node* v*, other than* t*, we either have* $S(v) = (v,u)$ *when* v *has selected the edge* (v,u) *in* E*, or* $S(v) = \perp$ *if no neighbor is assigned to* v*.*

A node v is *connected* to t in a state S iff v is connected to t in the graph induced by the edges chosen in S. We will write $P_S(v)$ for the path induced by S from v to t. If v is not connected, then $P_S(v) = \epsilon$[2]. If v is connected, $P_S(v) = S(v)P_S(u)$ when $S(v) = (v, u)$ (this will always be a simple path); indeed, any intermediate node appearing on $P_S(v)$ is assigned the corresponding suffix of $P_S(v)$. When the network converges, these paths collectively form a tree directed towards t. During the execution, a node is constrained to only choose among the *available paths*.

Definition II-.4 (Available paths)**.** *If a node v has k out-neighbors $u_1, u_2, \ldots, u_k$ in $G(V, E)$, then the set of* available paths *for v in state S is* $\mathrm{AP}(S, v) = \{(v, u_i)P_S(u_i) \mid 1 \leq i \leq k, (v, u_i)P_S(u_i) \in \mathcal{P}_v\} \cup \{\epsilon\}$.

If the preference specifies a total order over all available paths, a node can always choose the best available path according to such an order. However, if only given a partial preference, for the unordered paths, a node still needs to make a decision among them. We consider the case where for each node v and paths starting from v, *those specified in L_v are better than the remaining*, and a node can change its state (and hence the state of the network) only if it has an *improving move*.

Definition II-.5 (Improvement)**.** *A node v has an* improving *move in a state S iff there exists a path $p^* = (v, u^*)P_S(u^*)$ in $\mathrm{AP}(S, v)$ such that $S(v) \neq (v, u^*)$ and p^* is better than $P_S(v)$. The corresponding* improved state *S' will have $S'(v) = (v, u^*)$. A state is* stable *iff no node has an improving move.*

Consequently, whenever v is choosing among a set of paths without a total order, v can switch to an available path that is strictly better than his current path if any. If there is no such path, v will always stick with the current path (and hence has no improving move).

Remark 1 (The stable paths problem)**.** *Our partial preference system generalizes the case of the well known* stable paths problem *(SPP) [8] in the following sense. In SPP, each path is assigned with a* rank *(which reflects the priority) and for each node, paths with the same rank must have the same next-hop (so called the* strictness*). Since for each node, different paths with the same next-hop never appear simultaneously (the next-hop node can only have one path to t), order them arbitrarily will not affect the behavior of nodes in SPP. In other words, the preference of SPP for each node is (implicity) a total order over all paths. Partial preference system clearly captures total orders on paths, and it is more general since paths with different next-hop are allowed to be unordered.*

A node can only attempt to make an improvement when it is *activated* according to a *schedule*.

Definition II-.6 (Activation)**.** *An* activation *on $v \in V$ at a state S is a state transformation from S to a state S' such that only v can change its state to a improved state (if v has an improving move). We use ρ to denote the transformation function, i.e., $S' = \rho(S, v)$.*

Definition II-.7 (Schedule)**.** *A* schedule *is a sequence of activations defined by a function α from $\mathbb{N}$ to V, where $\alpha(\tau) = v$ means that the node v is activated at the time step τ. A schedule is* starvation-free *if for each node v and each time τ, there exists some $\Delta > 0$ such that $v = \alpha(\tau + \Delta)$. We will only consider starvation-free schedules.*

For an initial state S, we denote by $\mathsf{S}_\alpha^{(\tau)}$ the state reached after activating the list of nodes $\alpha(1), \alpha(2), \ldots, \alpha(\tau)$ in order (and $\mathsf{S}_\alpha^{(0)}$ is taken to be S). In other words, $\mathsf{S}_\alpha^{(\tau+1)} = \rho(\mathsf{S}_\alpha^{(\tau)}, \alpha(\tau + 1))$, for any positive τ. The following claim is an immediate consequence of the protocol execution model.

Claim II.1. *If $P_S(v) \neq \epsilon$ for some node v in some state S, for any schedule α and any $\tau > 0$, $P_{\mathsf{S}_\alpha^{(\tau)}}(v) \neq \epsilon$.*

Definition II-.8 (Convergence)**.** *A network has* converged *given a schedule α at time τ from a starting state S, if for any $\Delta > 0$, we have $\mathsf{S}_\alpha^{(\tau)} = \mathsf{S}_\alpha^{(\tau+\Delta)}$.*

Claim II.2. *If a network converges, it always converges to a stable state.*

Definition II-.9 (Convergence time)**.** *Given a starting state and a schedule, if τ is a time step at which the network N has converged ($\tau = +\infty$ if N never converges), we define the* convergence time *to be the number of improvements made up to τ. The* maximum convergence time*, denoted by $CT_{\max}(N)$, of N is the longest convergence time over all possible initial states and schedules.*

Note that since only the improving moves are counted, the convergence time is not necessarily equal to τ (though it is at most τ), and the definition of maximum convergence time is consistent for any τ at which N has converged.

III. Path Linearization

In this section, we show that whenever the specified partial preferences are free of dispute wheels (a particular cyclic arrangement of preferences), there is always a total order over all possible paths that is consistent with the preference such that a unique stable state will be reached after some bounded amount of time. We refer to such a network as a *linearizable network*, and to the process of finding such a total order as *path linearization*. Our approach is based on exploiting the structure of the "path digraph" associated with the input instance, taking advantage of a well-known connection between path digraphs and dispute-wheels. Moreover, we provide a path linearization algorithm that, though computing a total order on possibly exponentially many paths, runs in time polynomial in the size of the input graph and the preference.

A. Path Digraph and Dispute Wheels

A *path digraph* is a graph representation for 'BGP-like' path problems, where individual nodes' preferences govern route selection in a path-vector algorithm. For a network

[2]Note that ϵ is an empty path while $\perp$ denotes an 'empty' edge.

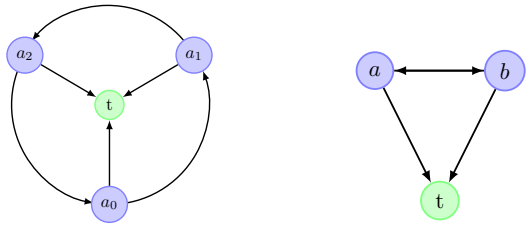
$\langle G(V,E), L, t\rangle$, the nodes of the path digraph are all the simple paths in G. The nodes for paths p and q are connected by a directed edge (p,q) if either p is a suffix of q or some node in G prefers p to q. Thus in the path digraph any suffix p of a path q (denoted by $p \to q$) is always considered better than the path q itself.

Definition III-A.1 (Linearization)**.** *For a given network $\langle G(V,E), L, t\rangle$, a* linearization $\succ_{\mathcal{P}}$ *of all paths $\mathcal{P}$ is a total order compatible with both L and the suffix relations. Specifically, for any $p, q \in \mathcal{P}$, (a) if $p \succ_L q$, then $p \succ_{\mathcal{P}} q$ (preference compatibility), and (b) if $p \to q$ then $p \succ_{\mathcal{P}} q$ (suffix compatibility).*

Define a *phase* of a schedule to be an interval of time during which all nodes are activated at least once. Then, linearizable networks have the following property.

Theorem 1. *Any linearizable network will converge to a unique stable state. Moreover, n phases suffice for any linearizable network on n nodes to converge under any initial state and any schedule, where n is the number of nodes.*

Proof. By [8], [20], absence of dispute wheels implies that the network always converges to a unique stable state, and convergence will happen in at most n phases. Absence of dispute wheels is equivalent to acyclicity of the path digraph [10], which is equivalent to linearizablility of a network. □



(a) Topology of an unstable network. (b) Topology of an network with two stable states.

Fig. 1: Some network examples.

On the other hand, without such a global linearization, convergence is not guaranteed. Figure 1a shows a typical example where the network will never converge. Assume each node a_i has preference $a_i a_{i+1} t \succ a_i t \succ a_i a_{i+1} a_{i+2} t$, where subscripts are interpreted modulo 3. In fact, this network has no stable state, and hence it can never converge. To see this, consider the number of nodes choosing the direct edge to t in a stable state if there exists one. If none of the nodes takes the path $a_i t$, any a_i will take $a_i t$ when activated (hence unstable). If only a_0 chooses the direct edge $a_0 t$, a_2 will choose $a_2 a_0 t$ and leave a_1 no choice but also choosing the direct edge to t. If a_0, a_1 both choose the direct edges to t, then a_0 will switch to $a_0 a_1 t$. If all nodes choose the direct edge to t, then a_0 will switch to $a_0 a_1 t$.

B. A Polynomial Time Algorithm for Linearization

We assume in the following that the input network is linearizable, and show that an efficiently computable linearization always exists. As our goal is to create a linearization for possibly exponentially many paths in polynomial time, the output can not be an explicit representation of the linear order. Hence we design a poly-time computable function that takes as input an ordered pair of simple paths (p, p') and outputs *yes* whenever p is ranked higher than p' in the linearization, and *no* otherwise. We establish the following.

Theorem 2. *Any linearizable network has a linearization that can be computed in polynomial time.*

To linearize a network $\langle G(V,E), L, t\rangle$, we first consider the path digraph with respect to only the *specified paths* $\mathcal{P}_L$. Formally, each path in $\mathcal{P}_L$ is (i) the trivial path that has only one node t, (ii) a path in the preference system L, or (iii) a subpath (i.e., a suffix) of some path in L. We call the following linear order of $\mathcal{P}_L$ a *spine*.

Definition III-B.1 (Spine)**.** *A spine on $\langle G(V,E), L, t\rangle$ is a linear order $\succ_{\mathcal{P}_L}$ on $\mathcal{P}_L$ such that for any $p, q \in \mathcal{P}_L$, (a) if $p \succ_L q$ then $p \succ_{\mathcal{P}_L} q$, and (b) if $p \to q$ then $p \succ_{\mathcal{P}_L} q$. Note that path t must be the largest element in any spine.*

A spine $\succ_{\mathcal{P}_L}$ can be created efficiently since the total number of paths in $\mathcal{P}_L$ is at most $|L| + n|L| + 1 = O(n|L|)$ (which respectively corresponds to the paths in L, the suffixes of paths in L, and the trivial path t) and a topologically sorted order of $\mathcal{P}_L$ can found in $O(n^2|L|^2)$ time.

Now, we show how to use a spine $\succ_{\mathcal{P}_L}$ to find a linearization of the network. For each path p in $\mathcal{P} \setminus \mathcal{P}_L$, among all suffixes of p that are in $\mathcal{P}_L$, map p to the smallest suffix according to the order $\succ_{\mathcal{P}_L}$; the order between two paths p, q that are mapped to two different paths p', q' in $\mathcal{P}_L$ is compatible with the order between p' and q' in $\succ_{\mathcal{P}_L}$; all paths assigned to the same path in $\mathcal{P}_L$ are ordered lexicographically.

It is not difficult to verify that the above process gives a linearization of the network in polynomial time. For the interest of space, we defer to Appendix A for a formal proof of the correctness, and the running time.

IV. Convergence Time for Restricted Preferences: A Dichotomy Theorem

Although path linearization guarantees convergence after n phases of execution, this does not mean that the convergence time is polynomially bounded since there could be arbitrarily long sequences of improving moves during a phase. In this and the next sections, we study the convergence time of linearizable, but not linearized, networks (i.e., executing the protocol under partial preferences) over restricted families of preference systems.[3] We start with the following families of preference systems.

Definition IV-.1 ($\langle s, l\rangle$-Preference Systems)**.** *A preference L is a $\langle s, l\rangle$-preference system iff for the preference L_v of each*

[3]The execution model for partial preferences implicitly assumes that paths specified in the preference is better than the rest. By simply examining the resulting path digraph, it can be verified that having these additional preferences does not affect the linearizability of a network.

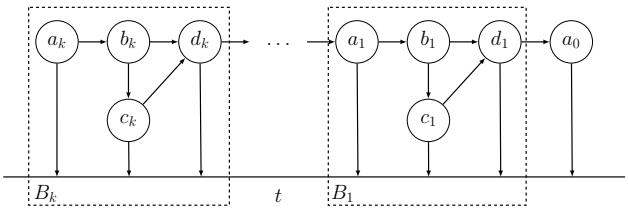


Fig. 2: A network with constant preference size and constant-length paths, that takes an exponentially long time to converge.

node v, (i) there are at most s paths in $|L_v|$, i.e., $|L_v| \leq s$, and (ii) for each path p in L_v, the length of p is at most l, i.e., $\max_{p \in L_v} \{|p|\} \leq l$.

We establish here the following dichotomy: even a linearizable network with only a $\langle 2,3 \rangle$-preference system could encounter exponentially many improving moves before convergence, while any linearizable network with a $\langle 2,2 \rangle$-preference system (resp. a $\langle 1,3 \rangle$-preference system) always converges after at most n^2 (resp. $2n$) improvements.

A. Exponential Convergence Time for $\langle 2,3 \rangle$-Preferences

Our first result shows that there exists a family of (even acyclic) networks such that the preference of any node contains at most two paths where each path has length at most three, and yet convergence may take time exponential in the size of the network.

Theorem 3. *For any $k \geq 1$, there exists a network $N\langle G(V,E), L, t\rangle$, where G is a DAG on $n = 4k+2$ nodes, and L is a $\langle 2,3 \rangle$-preference system, s.t. the maximum convergence time of N is $2^{\Omega(n)}$.*

Note that G is a DAG immediately implies that the network is free of dispute wheel, and hence linearizable. A network N satisfying the conditions of the above theorem is depicted in Figure 2, with the destination node t shown as the 'ground'. The network consists of a special node a_0, followed by $k = \lfloor n/4 \rfloor$ blocks B_i of four nodes each, named a_i, b_i, c_i and d_i. The preferences for each node in a block B_i are $L_{a_i} = \{a_i b_i d_i t \succ a_i t\}$, $L_{b_i} = \{b_i c_i t \succ b_i d_i t\}$, $L_{c_i} = \{c_i d_i t \succ c_i t\}$ and $L_{d_i} = \{d_i a_{i-1} t \succ d_i t\}$. The number of paths in the preference of for each node is 2 and every path has at most 3 hops.

The central idea is a pattern of activations for the nodes in each block B_i, whereby a 'flip' for node a_{i-1} (that is, when the node changes its state and then returns to the previous state) triggers two flips for node a_i. Accordingly, the sequence of blocks can be made to amplify a single flip at a_1 into exponentially many flips for the subsequent a_i.

For the interest of space, in the following, we will present the activation sequence of each block and explain how it leads to amplification, while a detailed proof of correctness is supplied in Appendix B1. The order of each block is defined as follows.

State	Activate	Old edge	New edge	Improvement reason
1	d_i	(d_i, a_{i-1})	(d_i, t)	$d_i t \succ_{d_i} d_i a_{i-1} b_{i-1} \cdots$
2	a_i	(a_i, t)	(a_i, b_i)	$a_i b_i d_i t \succ_{a_i} a_i t$
3	b_i	(b_i, d_i)	(b_i, c_i)	$b_i c_i t \succ_{b_i} b_i d_i t$
4	a_i	(a_i, b_i)	(a_i, t)	$a_i t \succ_{a_i} a_i b_i c_i t$
5	c_i	(c_i, t)	(c_i, d_i)	$c_i d_i t \succ_{c_i} c_i t$
6	b_i	(b_i, c_i)	(b_i, d_i)	$b_i d_i t \succ_{b_i} b_i c_i d_i t$
7	a_i	(a_i, t)	(a_i, b_i)	$a_i b_i d_i t \succ_{a_i} a_i t$
8	d_i	(d_i, t)	(d_i, a_{i-1})	$d_i a_{i-1} t \succ_{d_i} d_i t$
9	c_i	(c_i, d_i)	(c_i, t)	$c_i t \succ_{c_i} c_i d_i a_{i-1} \cdots$
10	a_i	(a_i, b_i)	(a_i, t)	$a_t \succ_{a_i} a_i b_i d_i a_{i-1} \cdots$

TABLE I: Sequence of improving moves for Lemma IV-A.1.

Definition IV-A.1 (Block Order)**.** *For $1 \leq i \leq k$, let σ_i be the following ordering of nodes in B_i:*

$$d_i, a_i, b_i, a_i, c_i, b_i, a_i, d_i, c_i, a_i.$$

We will refer to this ordering as the block order *of B_i.*

We consider the block order of activation starting with the following state.

Definition IV-A.2 (State S_1)**.** *For any $1 \leq i \leq k$, the S_1 state of the block B_i is defined as $S_1(a_i) = (a_i, t)$, $S_1(b_i) = (b_i, d_i)$, $S_1(c_i) = (c_i, t)$, and $S_1(d_i) = (d_i, a_{i-1})$.*

The activation of the block order on B_i starting from the state S_1 is illustrated in Figure 3, where the red (or thick) edges represent the current state. The following lemma provides the essential 'amplification' step for the exponential time result. Recall that we say that a node 'flips' when it changes its assigned edge and then changes back. The lemma shows that a flip for node a_{i-1} can cause a double flip for node a_i by activating the nodes in B_i according to the block order (with the activation of a_{i-1} interposed). Therefore the node a_{i+1} can be made to flip four times, and so on.

Lemma IV-A.1. *For any $1 < i \leq k$, suppose that the block B_i is in state S_1, the state of a_{i-1} is (a_{i-1}, b_{i-1}), and the node a_{i-1} will shortly change to (a_{i-1}, t). Then there exists a schedule such that each node makes an improving move when activated, and the final state of B_i is still S_1. Moreover, during activating under this schedule, the state of d_i flips once (from (d_i, a_{i-1}) to (d_i, t) and then back), and a_i flips twice (from (a_i, t) to (a_i, b_i) and back, twice).*

Proof. The schedule we use will activate the nodes of B_i in the block order σ_i with the activation of a_{i-1} to adopt (a_{i-1}, t) occurring partway through.

The progression of states leading to required effect is shown in Figure 3, and explained in detail in Table I. Note that a_{i-1} will activate and adopt (a_{i-1}, t) after state 2 but before state 9.

Consequently, the transition to state 2, where node d_i switches from an unranked path beginning with (d_i, a_{i-1}) to its second-best path $d_i t$ is an improving move, and so is the transition to state 9, where a_{i-1} has switched to (a_{i-1}, t) and so d_i can improve to its best path, $d_i a_{i-1} t$.

The claims of the lemma statement can readily be verified from the presented sequence. □

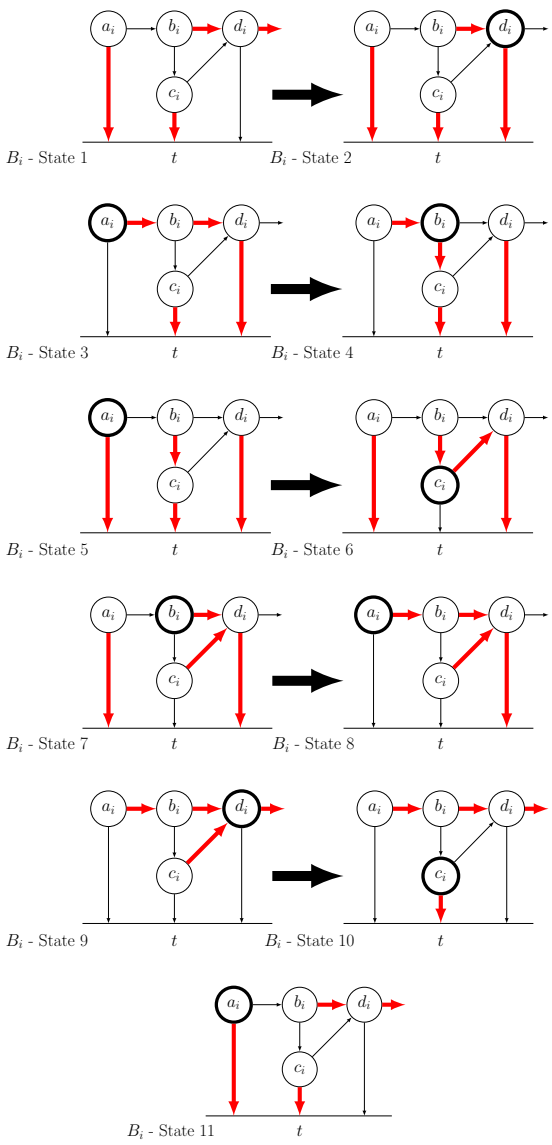


Fig. 3: States resulting from applying activation sequence σ_i to block B_i.

B. An Upper Bound on Convergence Time

We now investigate the maximum convergence time of linearizable networks. Firstly, note that any $\langle 1, l\rangle$-preference system for any positive integer l, will converge after at most $2n$ improvements since a node v can only switch from being not connected to t to choosing some neighbor u where u has a path to t, or switch from choosing some neighbor u to choosing the neighbor of the only path in v's preference (when this path becomes available). Hence, we only need to establish maximum convergence time for $\langle 2, 2\rangle$-preference systems, and combining with our construction for showing that $\langle 2, 3\rangle$-preference systems lead to exponentially many improvements (Theorem 3), we have a dichotomy theorem. In fact, we will establish the maximum convergence time for networks with $\langle 2, l\rangle$-preference systems for any positive integer l, which, as a special case, implies poly-time convergence for $\langle 2, 2\rangle$-preference systems.

We define the function $\mathcal{L}(l, n)$ for any positive integers l and n, to be the sum of a list of integers $\mathcal{L}(l, n) = \sum_{i=1}^{n} a_i$, where $a_1 = 0$ and $a_i = (l-1)a_{i-1} + 2$ for all i, and establish the following theorem.

Theorem 4. *For any positive integer l, for any linearizable network $N\langle G(V, E), L, t\rangle$ where L is a $\langle 2, l\rangle$-preference system, the maximum convergence time of N is at most $\mathcal{L}(l, n)$, where n is the number of nodes in G.*

We first establish that the total number of improvements made by any node v is upper bounded by the total number of improvements made by the nodes on the best path of v (Lemma IV-B.1). To utilize this fact, we introduce a process for finding a sequence of the nodes such that every node v appears after any other node on the best path of v (Lemma IV-B.2). Examining the nodes under this sequence, we show that for any i, the i-th node in the sequence can make at most a_i improvements, hence proving that $\mathcal{L}(l, n)$ is an upper bound of the total number of improvements.

We denote by $Imp(v)$ the maximum number of improvements that the node v can make. (The proofs of Lemma IV-B.1, IV-B.2 are deferred to Appendix B2.)

Lemma IV-B.1. *For any node v in a $\langle 2, l\rangle$-preference system network whose best path is denoted by $vu_1u_2 \dots u_j t$, $Imp(v) \leq \sum_{i=1}^{j} Imp(u_i) + 2$.*

We use the notion of "good" node to define/find a sequence of the nodes when examining their number of improvements.

Lemma IV-B.2. *For any linearizable network $\langle G(V, E), L, t\rangle$, for any $T \subsetneq V$ with $t \in T$, there exists at least one node $v \in V \setminus T$ s.t. either $L_v = \emptyset$, or for the best path of v $vu_1 \dots u_j t$, $u_i \in T$ for all i. We call such a node v a* good node for T.

Proof of Theorem 4. We will create a list of subsets of V, $\{A_i\}_{i=1}^{n}$ by starting from $A_1 = \{t\}$, adding one node every step, and ending with $A_n = V$. Let $a_i = \max\{Imp(v) \mid v \in A_i\}$. We will prove in the following that $a_i \leq (l-1)a_{i-1} + 2$. Since the node added at the i-th step can make at most a_i improvements, by summing up the number of improvements of all nodes, we achieve the upper bound of $\mathcal{L}(l, n)$ total improvements.

Starting from $A_1 = \{t\}$, we now show the transition from A_{i-1} to A_i, i.e., how to pick a node v that forms $A_i = A_{i-1} \cup \{v\}$ with $a_i \leq (l-1)a_{i-1} + 2$. By Lemma IV-B.2 with $T = A_{i-1}$, there exists a node v with either empty preference, or for v's best path, $vu_1u_2 \dots u_j t$, all nodes except v belong to A_{i-1}. We pick any such node v (i.e., any good node for A_{i-1}). For the first case, v has an empty preference implies that the only improvement v can make is from $\perp$ to one of its neighbor. For the second case, by Lemma IV-B.1, $Imp(v) \leq \sum_{i=1}^{j} Imp(u_i) + 2$. Since $j \leq (l-1)$ and all u_i belongs to A_{i-1}, we have $Imp(v) \leq \sum_{i=1}^{j} a_{i-1} + 2 \leq (l-1)a_{i-1} + 2$. Since $a_i = \max\{a_{i-1}, Imp(v)\}$, $a_i \leq (l-1)a_{i-1} + 2$. □

A simple calculation shows that $\mathcal{L}(2,n) \leq n^2$, and for any $l \geq 3$, $\mathcal{L}(l,n) \leq 2(l-1)^n$. Therefore, in particular, for any network with a $\langle 2,2\rangle$-preference system, the maximum convergence time is at most n^2. For a network with $\langle 2,3\rangle$-preference system, the maximum convergence time is at most 2^{n+1}. Combined with Theorem 3, which shows that there exist networks with $\langle 2,3\rangle$-preferences that take $2^{\Omega(n)}$ time to converge, we establish that the *convergence time complexity* of networks with $\langle 2,3\rangle$-preference systems is $2^{\Theta(n)}$.

V. Hop-Based Preference Systems

In this section we examine the maximum convergence time of hop-based preference systems.

Definition V-.1 (k-Hop Preference Systems)**.** *In a* k-hop preference system*, every node chooses paths based only on the next* k *hops of the paths.*

We use $vu_1u_2\dots u_k* \succ_v vw_1w_2\dots w_k*$ to denote that the prefix $vu_1u_2\dots u_k$ is better than $vw_1w_2\dots w_k$ for the node v. The well-known preference scheme of Gao and Rexford [4] is an example of a 2-hop preference system. All adjacencies are classified as customer-provider or peer-peer. Nodes are required to prefer customer routes over all others, which is a 1-hop preference rule. In addition, so-called 'valley' paths are worst of all: a path that go from a provider, 'down' to a customer, and then back 'up' to another provider. The extended transit provider guidelines of Liao et al. [14] can be treated in the same way. However, it is clear that the definition of k-hop preference is more general than these, even for the case when k is 2. The simplest case, when k is 1, covers the pure local preference scheme where initial preference decisions are based on the identity of the next-hop neighbor.

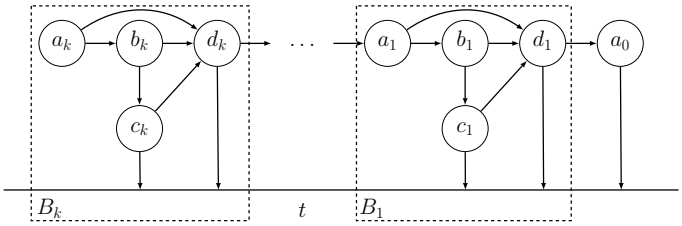
We establish here the following dichotomy result: any network with a 1-hop preference system converges in linear time while there exists a family of linearizable networks with 2-hop preferences systems such that the maximum convergence time is exponential.

A. Exponential Convergence Time for 2-Hop Preferences

We start by establishing the exponential convergence time result for linearizable networks with 2-hop preference systems. Note that the example shown in Figure 1a can be directly transformed into a 2-hop preference system which suggests that there are networks with 2-hop preference systems that never converge, which of course have unbounded maximum convergence time. The focus of this section is to show exponential convergence time for networks guaranteed to converge (in fact, even for DAGs).

Theorem 5. *For any* $k \geq 1$*, there exists a network* $N\langle G(V,E),L,t\rangle$ *with* $n = 4k+2$ *nodes forming a DAG and a* 2*-hop preference system such that the maximum convergence time of* N *is* $2^{\Omega(n)}$.

A slight modification of the construction in Theorem 3 will demonstrate the same result for 2-hop preference systems. Consider the network whose topology is shown in Figure 4a.



(a) Topology.

Node	1	2	3	4
a_i	$d_ia_{i-1}*$	b_id_i*	d_it	b_ic_i*
b_i	c_it	d_it	c_id_i*	$d_ia_{i-1}*$
c_i	d_it	t	$d_ia_{i-1}*$	
d_i	$a_{i-1}d_{i-1}*$	t	$a_{i-1}b_{i-1}*$	

(b) Preferences from best (1) to worst (4).

Fig. 4: A network with 2-hop preferences and exponential convergence time.

The only difference from the network used in Theorem 3 is that here a_i has an edge pointing to d_i, instead of t. The 2-hop preferences are shown in Table 4b. In Appendix C1, we supply a detailed account of the exponential-length activation sequence, which is based on the same idea as the construction of the $\langle 2,3\rangle$-preference systems.

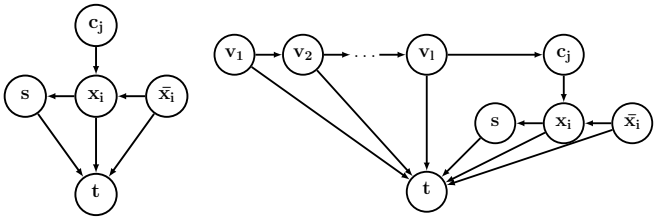
B. Linear-Time Convergence for 1-Hop Preferences

It is well known that whenever BGP preferences constitute a 1-hop preference system, the network always converges to a stable state after a linear number of improvements.

Theorem 6. *For any network* $N\langle G(V,E),L,t\rangle$*, where* L *is a* 1*-hop preference system, the maximum convergence time of* N *is at most* $n+m$*.*

For completeness, we give a simple proof of this theorem in Appendix C2. Note that Theorem 6 does not rely on absence of dispute wheels. This indicates that for 1-hop preference, there always exists at least one stable state. The network in Figure 1b is a typical example where there is a dispute wheel (which leads to multiple stable states) but the network always converges. Here the nodes a and b prefer each other to the direct edge to t. There are two stable states: $\{(a,b),(b,t)\}$ and $\{(b,a),(a,t)\}$. Which one is reached depends on which node is activated first.

Note that if allowing activation for multiple nodes simultaneously for networks with dispute wheels, even if there are stable states, convergence is not guaranteed. Again, consider the network in Figure 1b. If we start from the state $S(a) = (a,t)$ and $S(b) = (b,t)$, and activate both a and b. The path abt is available to a, and the path bat is available to b. Hence the new state will be $S(a) = (a,b)$ and $S(b) = (b,a)$. If we activate both a and b again, both of them will realize that they are not connected to t, and hence will switch back to $S(a) = (a,t)$ and $S(b) = (b,t)$. Therefore, the network will keep flipping between those two states and never converge.



(a) Basic gadget for the clause $c_j = x_i$.

(b) Final gadget for the clause $c_j = x_i$.

Fig. 5: The construction for showing the hardness of PLM.

VI. Linearization that Minimizes Path-Length

In this section, we study the problem of finding a path linearization that minimizes the length of the longest path in the stable state, which we refer to as the path-length minimization problem (PLM). We define the "length" of a path linearization to be the length of the longest path in the stable state, and hence PLM is to find a path linearization with the minimum length. We will only linearize the paths to the extent where the stable state is unique, and the remaining paths can either be linearized arbitrarily, or left unordered.

In the absence of preferences, PLM can be solved by simply performing a breath first search from the sink. Surprisingly, the presence of preference makes the problem almost impossible to tackle: it is NP-hard to approximate PLM to within a factor of $\Omega(n)$, (n is the number of nodes in the network). We further complement the hardness result by presenting an algorithm that finds a path linearization with length at most $(OPT + l)$, where OPT is the minimum length achievable by a path linearization and l is the length of the longest path in the preference.

A. Hardness of the Path-Length Minimization Problem

Theorem 7. *For any $2 \leq l \leq \Theta(n)$, it is NP-hard to distinguish whether the minimum length of a path linearization for a given network on n nodes is 2 or at least l.*

Proof. We use a reduction from 3SAT. Given a 3SAT instance $C = c_1 \wedge c_2 \wedge \ldots c_y$ with variables $x_1, x_2, \ldots, x_z$, we create the sink t and a special node s with edge (s, t). For each variable x_i, create two nodes that respectively represent x_i and $\bar{x}_i$, with edges (x_i, t), (x_i, s), $(\bar{x}_i, t)$, and $(\bar{x}_i, x_i)$. For each clause c_j, create a node that represents c_j with edges (c_j, x_i) if x_i is in c_j, and $(c_j, \bar{x}_i)$ if $\bar{x}_i$ is in c_j (see Figure 5a for the resulting topology of a simple clause $c_j = x_i$). We further create a list of nodes $v_1, v_2, \ldots v_l$ with edges (v_1, t), (v_2, t) $\ldots (v_l, t)$, and (v_1, v_2), (v_2, v_3) $\ldots (v_{l-1}, v_l)$. In addition, for each c_j node, create an edge (v_l, c_j). The final topology of a simple clause $c_j = x_i$ is shown in Figure 5b. The x_i nodes have no preference, each $\bar{x}_i$ node has preferences $\bar{x}_i x_i t \succ \bar{x}_i t \succ \bar{x}_i x_i s t$, each c_j node prefers any path of length 2 to those longer than 2. For any $1 \leq i \leq l$, v_i prefers the path $v_i v_{i+1} \ldots v_l p$ to the path $v_i t$, for any path p starting at a c_j node to t of length 3. It is straightforward to verify that the reduction is poly-time.

If C is satisfiable by an assignment A, consider the preference completion where a x_i node prefers the direct path $x_i t$ to the path $x_i s t$ iff x_i is true in A. The preference of $\bar{x}_i$ ensures that among x_i and $\bar{x}_i$, only one can take a path of length 1 and the other will take a path of length 2. Consequently, each c_j node has a path of length 2 (through the literal that satisfies c_j in A), and hence each v_i node will take the path $v_i t$. The length of the path linearization is 2. On the other hand, if C is not satisfiable, for any path linearization, define an assignment where x_i is true iff the node x_i prefers the path $x_i t$ to $x_i s t$. At least one c_j node is not satisfied by this assignment, and hence every out-going neighbor of c_j has a path of length 2, and c_j must end up taking a path of length 3. Then, the node v_l can take the path through this c_j, and each v_i will take the path through v_{i-1}. Consequently, v_1 will end up with a path of length more than l. Since l can be made $\Omega(n)$ in this construction, distinguishing whether the minimum length of a path linearization is 2 or $\Omega(n)$ implies satisfiability of C. □

B. Approximate the Path-Length Minimization Problem

Theorem 7 establishes that PLM is hard to approximate to within a factor of $\Omega(n)$. However, note that this hardness result relies on the length of the longest path in the preference being large (that is, $\Omega(n)$). We complement the hardness result by the positive result below.

Theorem 8. *There is a poly-time algorithm that for any linearizable network $N\langle G(V, E), L, t\rangle$, outputs a path linearization with length at most $(OPT + l)$, where OPT is the minimum length of a path linearization of N, and l is the length of the longest path in L. Moreover, in the path linearization the algorithm outputs, every node only has preference over at most $(|L| + 1)$ paths.*

We say a path p is *compatible* with a state S, if for any edge (u, v) on p, the state of u in S, $S(u)$, is either $\perp$ (i.e., no edge is selected) or (u, v). A path p is said to be *fully compatible* with S if for any edge (u, v) on p, $S(u) = (u, v)$. A fully compatible path in S is always available. In addition, we said a node v has *stabilized* at a state S if v will not change its state under any schedule.

Proof. To find a path linearization with the properties stated in the theorem, we first create a spine (see Definition III-B.1) for the given network $\langle G(V, E), L, t\rangle$, and argue that the spine guarantees a unique stable state S for a subset T of nodes where (i) for each node v in T, the path of v in the state S belongs to the spine, and (ii) for any node u in $V \setminus T$, no path specified in the preference of u, i.e., L_u, is compatible with S. Since all paths in the graph induced by S belong to the spine, the longest path among them has length at most l.

For the remaining nodes, since for any node v in $V \setminus T$, no path in L_v is compatible with S, any path p (starting at v) compatible with S can essentially be made the best path of v in S by letting p be less preferred than any path (starting at v) in the spine, but better than the rest. To find a path compatible with S for each node in $V \setminus T$ such that the length of the

longest path is bounded, we treat all nodes in T as a 'super sink', and perform a BFS from the super sink to the remaining nodes. Consequently, every path in the resulting graph would be a combination of a path induced by S and a path in the BFS tree. Since the depth of the BFS tree is a lower bound of the minimum length of a path linearization by adding the path of each node v in the resulting graph to the preference of v, we achieve a stable state where path-length are at most $OPT + l$. For every node v, since every path in L can add at most one new path from v to t to the spine, v will have preferences on at most $(|L|+1)$ paths. It remains to show how to find such a state S and a set of nodes T. We establish the following lemma where the proof is deferred to Appendix D.

Lemma VI-B.1. *Given a network with a spine, where the original preference is replaced by the spine, in any state S, if T is a subset of nodes that have stabilized, then either (i) there exists a node $v \in V \setminus T$, where the best compatible (with S) path in L_v (if any) is fully compatible, or (ii) for any $v \in V \setminus T$, no path in L_v is compatible with S.*

Then, starting from $T = \{t\}$ and an empty state S, we can repeatedly apply Lemma VI-B.1, find nodes whose best fully compatible path is available (which will be selected eventually), add them to T, and update S accordingly, until for any remaining node, no path in the preferences is compatible with S. □

VII. Conclusions

Partially specified preferences match the way that router configuration is envisaged: definite policy is established at coarse granularity, and further tie-breaking decisions are essentially arbitrary. In this paper, we studied convergence time for partial routing preference systems and introduced the notion of linearizability to connect partial preference systems to earlier work on routing correctness and convergence.

We formally established that the routing convergence time is exacerbated by the complexity of the routing policy. This demonstrates that in order to improve the convergence speed of BGP, we must think not only about the router implementations, the network environment, and the protocol design, but also about the nature of the specified policies. In particular, our result in Section IV shows that the real source of slow convergence is not the number of alternative paths, nor the length of paths, nor the presence of cycles, but the ability to express 'fine-grained' route preferences. Even if preferences are only allowed to be based on the two-hop neighborhood, exponential convergence time is possible.

On the positive side, restricting the preference system *does* help, though consequently, nodes might be only allowed to specify preference over either the next hop or a few short paths. Our work on 1-hop preference systems generalizes Schapira et al. [21]; they additionally require that path preferences follow the guidelines of Gao and Rexford [4]. The next-hop restriction is in fact a reasonable one, since it matches a typical first-cut BGP policy—setting local preference based on the neighbor's identity.

In other prior works, Fabrikant et al. [3] consider a braid-like network that may encounter exponentially many improvements before convergence, using essentially a $\langle 4, 3 \rangle$-preference system. Our work completes theirs by clarifying how much restriction one needs to place on the preference system in order to guarantee poly-time convergence.

Acknowledgement. Supported in part by National Science Foundation grants CCF-1116961, CCF-1552909, and IIS-1447470.

References

[1] Caesar, M., and Rexford, J. BGP routing policies in ISP networks. *IEEE Network 19*, 6 (Nov/Dec 2005), 5–11.

[2] Fabrikant, A., and Papadimitriou, C. H. The complexity of game dynamics: BGP oscillations, sink equilibria, and beyond. In *Proc. ACM SODA* (2008), pp. 844–853.

[3] Fabrikant, A., Syed, U., and Rexford, J. There's something about MRAI: Timing diversity exponentially worsens BGP convergence. In *Proc. IEEE INFOCOM* (2011).

[4] Gao, L., and Rexford, J. Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking* (2001), 681–692.

[5] Griffin, T., and Wilfong, G. T. An analysis of BGP convergence properties. In *SIGCOMM* (1999), pp. 277–288.

[6] Griffin, T. G., and Huston, G. RFC 4264: BGP wedgies, 2005.

[7] Griffin, T. G., Shepherd, F. B., and Wilfong, G. Policy disputes in path vector protocols. In *IEEE ICNP* (1999).

[8] Griffin, T. G., Shepherd, F. B., and Wilfong, G. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking 10*, 2 (2002), 232–243.

[9] Griffin, T. G., and Wilfong, G. A safe path vector protocol. In *Proc. IEEE INFOCOM* (2000).

[10] Gurney, A. J. T., Jia, L., Wang, A., and Loo, B. T. Partial specification of routing configurations. In *Workshop on Rigorous Protocol Engineering (WRiPE)* (2011).

[11] Karloff, H. J. On the convergence time of a path-vector protocol. In *SODA* (2004), J. I. Munro, Ed., SIAM, pp. 605–614.

[12] Labowitz, C., Ahuja, A., Abose, A., and Jahanian, F. An experimental study of BGP convergence. In *ACM SIGCOMM* (2000).

[13] Labowitz, C., Malan, G. R., and Jahanian, F. Internet routing instability. *IEEE/ACM Transactions on Networking* (1998).

[14] Liao, Y., Gao, L., Guerin, R. A., and Zhang, Z.-L. Safe inter-domain routing under diverse commercial agreements. *IEEE/ACM Transactions on Networking 18*, 6 (2010), 1829–1840.

[15] Mao, Z. M., Govindan, R., Varghese, G., and Katz, R. H. Route flap damping exacerbates Internet routing convergence. In *Proc. ACM SIGCOMM* (2002).

[16] McPherson, D., Gill, V., Walton, D., and Retana, A. RFC 3345: Border Gateway Protocol (BGP) persistent route oscillation condition, 2002.

[17] Papadimitriou, C. H. Algorithms, games, and the Internet. In *Proc. ACM STOC* (2001).

[18] Pelsser, C., Maennel, O., Mohapatra, P., Bush, R., and Patel, K. Route flap damping made usable. In *PAM* (2011).

[19] Rekhter, Y., Li, T., and Hares, S. RFC 4271: A Border Gateway Protocol 4 (BGP-4), 2006.

[20] Sami, R., Schapira, M., and Zohar, A. Searching for stability in interdomain routing. In *Proc. IEEE INFOCOM* (2009), pp. 549–557.

[21] Schapira, M., Zhu, Y., and Rexford, J. Putting BGP on the right path: the case for next-hop routing. In *ACM HotNets* (2010).

[22] Sobrinho, J. Network routing with path vector protocols: Theory and applications. In *Proc. ACM SIGCOMM* (2003), pp. 49–60.

[23] Varadhan, K., Govindan, R., and Estrin, D. Persistent route oscillations in interdomain routing. *Computer Networks* (2000).

[24] Villamizar, C., Chandra, R., and Govindan, R. RFC 2439: BGP route flap damping, 1998.

[25] Zhao, M., Zhou, W., Gurney, A. J. T., Haeberlen, A., Sherr, M., and Loo, B. T. Private and verifiable interdomain routing decisions. In *Proc. ACM SIGCOMM* (2012).

APPENDIX

A. Omitted Details from Section III

We present here the omitted details from Section III-B, completing the proof of Theorem 2. We start by reviewing the high-level overview of the steps used in our linearization algorithm, and then proceed to a formal description and analysis.

(a) Create a linear order, called a *spine*, on $\mathcal{P}_L$, i.e., the set of paths that either appear in L or are a suffix of some path in L.

(b) Every other path $p \in \mathcal{P}$, is mapped to a unique path p' on the spine, ordering p between p' and the successor of p' on the spine (if there exists one).

(c) All paths assigned to the same path p' on the spine are ordered lexicographically.

We write $p \geq_{\text{lex}} q$ for the relation on paths that holds whenever p is shorter than q, or they are the same length and p lexicographically precedes q (considering them as words over the alphabet V, with some arbitrary fixed order on V).

Claim A.1. *If a network $\langle G(V,E), L, t\rangle$ is linearizable, there must exist a spine on $\mathcal{P}_L$.*

In fact, the restriction of any linearzation to $\mathcal{P}_L$ is a spine, by Definitions III-A.1 and III-B.1.

Definition A.1 (Vertebra)**.** *For any path p in $\mathcal{P} \setminus \mathcal{P}_L$, the* vertebra *of p is $v(p) = \min_{\succ_{\mathcal{P}_L}} \{q \in \mathcal{P}_L \mid q \to p\}$, i.e., the minimal spine path among all of its suffixes. Note that because t is in $\mathcal{P}_L$, $v(p)$ is always well defined.*

We now prove Theorem 2 by demonstrating the following order.

Definition A.2 (Spinal order)**.** *Given a spine $\succ_{\mathcal{P}_L}$ on network $\langle G(V,E), L, t\rangle$, define the* spinal order $\succ_{\mathcal{P}}$ *on $\mathcal{P}$ by $p \succ_{\mathcal{P}} q$ if and only if $v(p) \succ_{\mathcal{P}_L} v(q)$, or $v(p) = v(q)$ and $p \geq_{\text{lex}} q$.*

We introduce the following three lemmas to prove Theorem 2.

Lemma A.1. *The spinal order is a linearization of $\mathcal{P}$.*

Proof. We must show that the order $\succ_{\mathcal{P}}$ is a total order that is consistent with preference and suffix compatibility rules.

First of all, the order $\succ_{\mathcal{P}}$ is indeed a total order.

- **Reflexivity.** Obvious from the definition.
- **Transitivity.** Suppose $p \succ_{\mathcal{P}} q \succ_{\mathcal{P}} r$; consider $v(p)$, $v(q)$ and $v(r)$. If $v(p) = v(q) = v(r)$, we must have $p \geq_{\text{lex}} q \geq_{\text{lex}} r$ and so $p \geq_{\text{lex}} r$. Hence $p \succ_{\mathcal{P}} r$. If $v(p) \succ_{\mathcal{P}_L} v(q)$, then either $v(q) \succ_{\mathcal{P}_L} v(r)$ or $v(q) = v(r)$. For the first case, since the spine is transitive, we have $v(p) \succ_{\mathcal{P}_L} v(r)$; in the second case, $v(q) \succ_{\mathcal{P}_L} v(r)$ trivially. Thus $v(p) \succ_{\mathcal{P}} v(r)$. The argument for $v(q) \succ_{\mathcal{P}_L} v(r)$ is exactly parallel.
- **Antisymmetry.** Suppose $p \succ_{\mathcal{P}} q$ and $q \succ_{\mathcal{P}} p$. Clearly we must have $v(p) = v(q)$, from which it follows that $p \geq_{\text{lex}} q$ and $q \geq_{\text{lex}} p$. Hence $p = q$ as required.
- **Totality.** Recall that v is defined for all paths. Suppose that $\neg(p \succ_{\mathcal{P}} q)$, for some distinct p and q. Then $\neg v(p) \succ_{\mathcal{P}_L} v(q)$, and since the spine is a total order, we have either $v(q) \succ_{\mathcal{P}_L} v(p)$ or $v(p) = v(q)$. The first case directly implies $q \succ_{\mathcal{P}} p$. In the second, it must be that $\neg(p \geq_{\text{lex}} q)$ according to the definition of this order. Since $\geq_{\text{lex}}$ is a total order, we must have $q \geq_{\text{lex}} p$, from which it follows that $q \succ_{\mathcal{P}} p$, as required.

Secondly, the order is compatible with the preferences in L. Suppose that $p \succ_L q$. Then $p = v(p)$, $q = v(q)$ and $v(p) \succ_{\mathcal{P}_L} v(q)$, so $p \succ_{\mathcal{P}} q$ as required.

Finally, the order is compatible with the suffix relation. Suppose that $p \to q$. By definition of vertebra, it must be that $v(p)$ is at least as good as $v(q)$ on the spine, because any suffix of p is also a suffix of q. If $v(p) \succ_{\mathcal{P}_L} v(q)$ then $p \succ_{\mathcal{P}} q$ and we are done. Otherwise, $v(p) = v(q)$; but in this case, p is a shorter path than q and so lexicographically precedes it. We have $p \geq_{\text{lex}} q$ and $p \succ_{\mathcal{P}} q$ as required. □

Lemma A.2. *If a spine exists, then it can be constructed in $O(|\mathcal{P}_L|^2)$ time.*

Proof. Recall that a spine is a linear order on $\mathcal{P}_L$, which extends both the preferences in L and the suffix relations. Let Π be a binary relation on $\mathcal{P}_L$ which contains exactly those pairs of paths (p, q) for which $p \succ_L q$ or $p \to q$. A topological sort of Π will either fail, indicating the presence of a cycle and hence nonexistence of a spine, or succeed, and produce a spine.

Extending L to Π can be done in $O(|\mathcal{P}_L|^2)$ time. A topological sort of all paths in Π can be done in $O(|\mathcal{P}_L| + |\mathcal{P}_L|^2)$ time, where $|\mathcal{P}_L|^2$ is an upper bound on the number of relations in Π. Therefore, the entire running time is bounded by $O(|\mathcal{P}_L|^2)$. □

Lemma A.3. *Given any two paths p and q from $\mathcal{P}$, the determination of whether $p \succ_{\mathcal{P}} q$ or $q \succ_{\mathcal{P}} p$ holds can be made in polynomial time.*

Proof. We may assume a spine has already been constructed, since this takes polynomial time in any case. To identify the ordering of p and q, we must evaluate and compare $v(p)$ and $v(q)$, and if they are equal, then compare p and q lexicographically.

It takes $O(|V|)$ time to evaluate $v(p)$ for a given p. The length of p is at most $|V|$, since $\mathcal{P}$ contains only simple paths. We can find $v(p)$ by enumerating all suffixes of p, beginning at p itself, and returning the first one that is in $\mathcal{P}_L$.

Comparison of $v(p)$ and $v(q)$ is done according to the spine. If they are equal, lexicographic comparison of p and q takes $O(|V|)$ time.

We conclude that preference queries, for two given paths, take $O(|V|)$ time overall, once a spine has been constructed. □

Proof of Theorem 2. Lemma A.1 guarantees that there exists at least one spine. By Lemma A.1, the corresponding spine order is a linearization of $\mathcal{P}$. In addition, by Lemma A.2

and A.3, we can construct as well as determine preferences between two paths in polynomial time. □

B. Omitted Details From Section IV

We now present a detailed analysis of the restricted family of instances described in Section IV that take exponentially long to converge in the worst-case.

1) Exponentially Many Improvements for $\langle 2, 3\rangle$-Preference Systems: Recall from Section IV-A the example depicted in Figure 2.

To prove Theorem 3, we will construct an activation sequence for this network of exponential length: specifically, each node a_i will make an improving move 2^i times.

Lemma B.1. *Any improving move by a node in a block B_i does not affect the state of a node in a block B_ℓ with $\ell < i$.*

Proof. As the graph G is a DAG, each node in B_ℓ appears before any node in B_i in the topologically sorted order. Thus an improving move by a node in B_i cannot affect the choice of nodes in B_ℓ. □

The pieces are now in place for the presentation of the main result. The proof below provides an activation sequence with exponentially many improving moves. Specifically, the a_i nodes will each activate and improve 2^i times.

Proof of Theorem 3. Consider the network with graph shown in Figure 2 and preference described above.

We start with the following initial state of the network. For all $i > 1$, B_i is in state S_1 and B_1 is in State 2 of Figure 3, i.e., a_1 takes (a_1, t), b_1 takes (b_1, d_1), c_1 takes (c_1, t), and d_1 takes (d_1, t).

The reason why we treat B_1 differently is that the block B_i can be used to trigger improving moves in B_{i+1} when the node a_i flips. But for B_1, a_0 only has one outgoing edge and will never be able to improve. But since the ultimate goal is just to flip a_1 twice, and the improvement occurs at State 3, 5, 8 and 11, starting from State 2 is enough.

For any sequence σ, let σ^R denote the sequence σ in reverse. Also, let σ_1' denote the block order σ_1 with the first occurrence of d_1 removed.

We now create the activation sequence Σ using a stack of nodes O, and a counter χ_i to record the number of times that each node a_i has been popped. We start by pushing the sequence ${\sigma_1'}^R$ on to the stack; so the sequence will get popped out as σ_1'. We now start popping nodes from the stack, and each popped node is appended to the sequence Σ. If it is one of the a_i nodes, then we increment and examine χ_i.

- If χ_i is odd (that is, if a_i has been activated an odd number of times), and if $i \neq k$, then we push the sequence $(d_{i+1}, a_{i+1}, b_{i+1}, a_{i+1}, c_{i+1}, b_{i+1}, a_{i+1})^R$ into the stack O. This corresponds to σ_{i+1} restricted to the transitions from state 1 to state 8.
- If χ_i is even, and if $i \neq k$, then we push the sequence $(d_{i+1}, c_{i+1}, a_{i+1})^R$ into the stack O. This corresponds to the remaining steps of σ_{i+1}, going from state 8 to state 11.

The process continues until the stack is exhausted.

We now establish that (a) every appearance of the node a_i in Σ leads to an improving move, and (b) χ_i will end up as 2^i. Since $n = 4k + 2$, it follows that the node a_k makes $2^{\Omega(n)}$ improving moves in the sequence Σ and hence we get exponentially long convergence time.

We start by establishing the assertion (a) above by induction on i where i ranges from 1 to k. The state transformation in Lemma IV-A.1 implies that for any block B_i, starting from the state S_1, every appearance of a node in the block order leads to an improving move, including a_i.

For the base case, we consider the node a_1. The nodes in B_1 start in state S_2 instead of S_1. However, the construction in Lemma IV-A.1 shows that skipping the transition from state 1 to state 2 does not affect the subsequent transitions from state 2 through to state 11, and so the appearance of each a_1 will still make an improving move when applying the block order σ_1'. The process defined above will push additional nodes on to the stack when pushing ${\sigma_1'}^R$. However, for any $1 \leq i < k$, when seeing a node a_i, the process only pushes nodes in B_ℓ where $\ell > i$. Now by Lemma B.1, any improving moves by nodes in B_ℓ where $\ell > i$ do not affect the state of nodes in B_1. Thus a_1 will still make an improving move whenever it is given the chance. Second, no more occurrences of the node a_1 will be pushed onto the stack, besides the initial push of the sequence ${\sigma_1'}^R$.

As a result, every appearance of the node a_1 in Σ leads to an improving move. Observe furthermore that the node a_1 has only two out-neighbors and the initial state ensures that it starts with the edge (a_1, t). Each improving move causes a_1 to switch between its neighbors. Therefore, between any successive odd appearance and even appearance of a_1, it changes from (a_1, t) to (a_1, b_1) and back.

For the inductive step, assume that every appearance of nodes a_1 to a_i in the sequence Σ leads to an improving move. We furthermore assume that for $1 \leq j \leq i$, between any successive odd appearance and even appearance of the a_j, it changes from (a_j, t) to (a_j, b_j) and then back to (a_j, t). We establish that the same property holds for the node a_{i+1}.

Node a_{i+1} can get pushed into the stack only when node a_i is popped. Consider a pair of successive odd and even appearances of node a_i. In the construction of Σ, upon an occurrence of a_i with χ_i odd, we push the sequence $\{d_{i+1}, a_{i+1}, b_{i+1}, a_{i+1}, c_{i+1}, b_{i+1}, a_{i+1}\}^R$ onto the stack. Each pop of an a_{i+1} corresponds to an improving move, by Lemma IV-A.1. During popping of these nodes from the stack, additional nodes from blocks B_{i+2} to B_k might be pushed into the stack. But as before, any improving moves by nodes in the blocks B_{i+2} to B_k cannot affect the state of B_{i+1}, and moreover, no further occurrences of a_{i+1} will get pushed into the stack.

The next occurrence of a_i (which will be even) happens after each of the above seven nodes has been popped, and B_{i+1} is in state 8. This activation of a_i results in its adoption of (a_i, t), which is what enables d_{i+1} to make an improving move, and reach state 9 of B_{i+1} (as in Lemma IV-A.1).

Accordingly, the sequence $(d_{i+1}, c_{i+1}, a_{i+1})^R$ is pushed onto the stack, so that d_{i+1} will be the next node popped, improving from $d_{i+1}t$ to $d_{i+1}a_it$, after which B_{i+1} will be in state 9, allowing c_{i+1} and then a_{i+1} to make their improving moves. So a_{i+1} has improved again.

As in the case above for odd χ_i, the additional nodes pushed do not affect the state of B_i, and no more occurrences of a_{i+1} are pushed into the stack. Therefore every occurrence of the node a_{i+1} corresponds to an improving move. and furthermore, a_{i+1} changes from (a_{i+1}, t) to (a_{i+1}, b_{i+1}) and then back to (a_{i+1}, t).

We next establish assertion (b), that is, the node a_i appears 2^i times in the sequence Σ. Whenever we pop an odd occurrence of node a_{i-1}, three occurrences of a_i will be pushed into the stack. Whenever we pop an even occurrence of a_{i-1}, one occurrence of a_i is pushed into the stack. Therefore, any successive odd and even occurrence of a_{i-1} results in four occurrences of a_i getting pushed into the stack. Considering the final values of the χ_i counters, if χ_i is even then $\chi_{i+1} = 2\chi_i$. Since $\chi_1 = 2$, which is even, it follows that $\chi_i = 2^i$ for all i.

Therefore, the total number of improvements is at least $\sum \chi_i > 2^k$, and hence the maximum convergence time $CT_{\max}(N) = 2^{\Omega(n)}$. □

2) The Maximum Convergence Time for Restricted Preference Systems:

Proof of Lemma IV-B.1. For any node v, the first improvement could be switching from $\perp$ to some neighbor u with a valid path. To make any more improvements, since v will always have a valid path through u (Claim II.1), at least one path from L_v with next-hop different than u must be available. Therefore, if $L_v = \emptyset$, v can make no more improvement, and if $|L_v| = 1$, v can make at most two improvements.

For $|L_v| = 2$, assume the best path is p_1 and the second best path is p_2, and the next hop of p_1 is w_1 and the next hop of p_2 is w_2. After making at most two improvements, v must choose either (v, w_1) or (v, w_2) and the corresponding path is available when the previous improvement is made.

To make any more improvement, we argue that it can only happen after some node in p_1 make some improvement first. If the current state is (v, w_1), the only neighbor v could switch to is w_2 (when p_1 is no longer available and p_2 becomes available). Since at the time v chooses w_1, p_1 must be available, v can make such improvement only after some node in p_1 made an improvement and switched to another path.

If the current state is (v, w_2), by Claim II.1, the path through w_2 to t is always valid. Hence the only neighbor v can switch to is w_1 (when p_1 becomes available). Since at the time v chooses w_2, p_1 must not be available, v can switch only after every node in p_1 (except v) is choosing its next-hop in p_1.

Either case, v can make an improvement only after at least one node in p_1 (except v itself) made at least one improvement. Hence, if $p_1 = vu_1u_2 \dots u_jt$, the total improvements v can make is $Imp(v) \leq \sum_{i=1}^{j} Imp(u_i) + 2$. □

Proof of Lemma IV-B.2. We can assume that for every node $v \in V \setminus U$, vt is not the best path, since otherwise, v is a good node (hence, if $l = 1$, where the only possible path in L is of form vt, all nodes in $V \setminus U$ are good). Now we assume $l \geq 2$ and prove by contradiction. Suppose none of the nodes in $V \setminus U$ is good. Denote $p_{[v]}$ the sub-path of p starting from node v, if $v \in p$.

Pick any node $v_1 \in V \setminus U$. There exists a node v_2 in v_1's best path p_1 s.t. $v_2 \in V \setminus U$. By suffix relationship, $p_{1[v_2]} \to p_1$. Since $v_2 \in V \setminus U$, there exists a node v_3 in v_2's best path p_2 s.t. $v_3 \in V \setminus U$. By suffix relationship and preference relationship, respectively, $p_{2[v_3]} \to p_2$ and $p_2 \succ p_{1[v_2]}$. Eventually, this process will encounter the same node twice. If we denote the nodes we encountered by $v_1, v_2, \dots$, then there exists some k s.t., the best path of v_k, p_k contains the node v_i for some $i < k$. Hence,

$$p_i \succ p_{k[v_i]} \to p_k \succ \dots \to p_{i+1} \succ p_{i[v_{i+1}]} \to p_i$$

This is a dispute wheel by definition, which contradicts the fact that the network is linearizable. □

C. Omitted Details From Section V

1) 2-Hop Preference Systems: As before, we break the network into blocks and denote them as B_i and define a new S_1 state similar to the previous case.

Definition C.1 (S_1 state, 2-hop case)**.** *The state S_1 of B_i is the state $S_1(a_i) = (a_i, d_i)$, $S_1(b_i) = (b_i, d_i)$, $S_1(c_i) = (c_i, t)$, and $S_1(d_i) = (d_i, a_{i-1})$.*

Lemma C.1. *For any B_i, starting from state S_1, there is an activation sequence of improving moves, terminating in S_1 again, during which node d_i changes from (d_i, a_{i-1}) to (d_i, t) and back, and node a_i changes from (a_i, d_i) to (a_i, b_i) and back, twice.*

State	a_i	b_i	c_i	d_i
1	(a_i, d_i)	(b_i, d_i)	(c_i, t)	(d_i, a_{i-1})
2	(a_i, d_i)	(b_i, d_i)	(c_i, t)	$\star(d_i, t)$
3	$\star(a_i, b_i)$	(b_i, d_i)	(c_i, t)	(d_i, t)
4	(a_i, b_i)	$\star(b_i, c_i)$	(c_i, t)	(d_i, t)
5	$\star(a_i, d_i)$	(b_i, c_i)	(c_i, t)	(d_i, t)
6	(a_i, d_i)	(b_i, c_i)	$\star(c_i, d_i)$	(d_i, t)
7	(a_i, d_i)	$\star(b_i, d_i)$	(c_i, d_i)	(d_i, t)
8	$\star(a_i, b_i)$	(b_i, d_i)	(c_i, d_i)	(d_i, t)
9	(a_i, b_i)	(b_i, d_i)	(c_i, d_i)	$\star(d_i, a_{i-1})$
10	(a_i, b_i)	(b_i, d_i)	$\star(c_i, t)$	(d_i, a_{i-1})
11	$\star(a_i, d_i)$	(b_i, d_i)	(c_i, t)	(d_i, a_{i-1})

TABLE II: State sequence of block B_i.

Proof. We use the same block order as before (Definition IV-A.1). Table II shows the resulting states; the annotation '$\star$' shows which node made the improving move. □

When a_i changes to (a_i, b_i), d_{i+1} will lose its best path and switch to the second best (d_i, t), which is always available. When a_i flips back to (a_i, d_i), the best path of d_{i+1} appears again, and d_{i+1} will change to it. As a result, each flip of d_i will cause a_i to flip twice and each flip of a_i will cause d_{i+1} to flip twice. The construction proceeds as in Theorem 3.

2) 1-Hop Preference Systems – Proof of Theorem 6:

Proof. We prove this through a simple potential function argument. For any state S of the network, we define an n-dimensional vector W that records the 'quality' of the current solution for each node. Specifically, for each node $v \in V$, we set entry $W(v) = i$ if $S(v)$ is the i-th best next hop for v, and $W(v) = |k_v| + 1$ otherwise (i.e. if $S(v) = \perp$); here k_v denotes the number of out-neighbors of v.

By Claim II.1, after making an improvement, v will be connected to t and can always stick with its current path, and hence the value $W(v)$ is non-increasing over time.

On the other hand, whenever v makes an improvement, it must switch to a better neighbor, and the value of $W(v)$ will strictly decrease. It follows that the total number of improving moves is bounded by

$$\sum_{v \in V} W(v) \leq \sum_{v \in V} (k_v + 1) \leq m + n$$

Hence the system always converges to a stable state in $n+m$ steps. □

D. Omitted Details From Section VI

Proof of Lemma VI-B.1. We prove by contradiction. For simplicity, a (fully) compatible path always refers to a path (fully) compatible to the state S. Suppose that for any node $v \in V \setminus T$, the best compatible path in L_v (if any) is not fully compatible, and there exists a node $v_0 \in V \setminus T$ who has compatible paths in L_{v_0}, and the best compatible path is p_0. By our assumption, p_0 is not fully compatible. Hence, there exists a sub-path of p_0 denoted by $v_1 q_0$ where q_0 is a path to t induced by S and $v_1 \notin T$ (i.e., the shortest sub-path to t not induced by S). Since the path $v_1 q_0$ is on the spine, it is a path in the preference of v_1, L_{v_1} that is fully compatible. By our assumption, $v_1 q_0$ is not the best compatible path in L_{v_1}. Consider the best compatible path p_1 of L_{v_1}, and using the same process, we can find a sub-path of p_1, $v q_1$, where the best compatible path of v_2 is p_2. Eventually, the process must reach the same node twice and the paths $\{p_i, v_{i+1} q_i\}$ forms a dispute wheel, which contradicts the fact that the network is linearizable. □