Guaranteeing BGP Stability With a Few Extra Paths

Rachit Agarwal, Virajith Jalaparti, Matthew Caesar, P. Brighten Godfrey
University of Illinois at Urbana-Champaign, IL, USA
{agarwa16, jalapar1, caesar, pbg}@illinois.edu

***Abstract*—Policy autonomy exercised by Autonomous Systems (ASes) on the Internet can result in persistent oscillations in Border Gateway Protocol, the Internet's inter-domain routing protocol. Current solutions either rely on globally consistent policy assignments, or require significant deviations from locally assigned policies, resulting in significant loss of autonomy of ASes.**

In this paper, we take a different approach that guarantees stability with less restrictive policies. Namely, we propose multipath routing to find a better trade-off between AS policy autonomy and system stability. We design an algorithm, Stable Path(s) Assignment (SPA), that provably detects persistent oscillations and eliminates these oscillations by assigning multiple paths to some ASes in the network. Such an assignment allows each AS to use its most-preferred available path, while requiring very few ASes to carry transit traffic along additional paths in order to break oscillations.

We design a distributed protocol for SPA and present tight bounds on the number of paths assigned to the ASes in the network. Using simulations on the AS graph, we show that in presence of oscillations, SPA assigns at most two paths to any AS in the network (in 99.9% of the instances), with an extremely small fraction of ASes assigned the extra path.

***Keywords*-Internetworking, Routing, Algorithms, Border Gateway Protocol**

I. Introduction

The Internet connects thousands of independently operating networks, known as *Autonomous Systems* (ASes), corresponding to Internet service providers, companies, universities, etc. Individual ASes must cooperate to construct routes, yet at the same time, they often compete for business and have very different operational goals. To allow this highly diverse set of networks to cooperate, the *Border Gateway Protocol* (BGP) was developed. BGP is a routing protocol that computes paths across ASes, and comes with a highly flexible set of policy knobs and configuration parameters to allow individual ASes to express their routing preferences. However, it has been shown that this freedom of route selection exercised by the ASes can cause instability in inter-domain routing manifesting in the form of persistent route oscillations [1], [2]. Several studies have demonstrated that such routing oscillations can significantly degrade the end-to-end performance of the Internet [3], [4].

Currently, there are two main approaches to avoid persistent route oscillations in BGP. First, convergence is guaranteed if ASes use a restricted set of policies known as valley-free routes [5]. However, though valley-free routes are common, it is unrealistic to expect *all* ASes to conform to these constraints. A second approach requires ASes to dynamically detect persistent oscillations, and deviate from local preferences when oscillations do occur [6]. This approach has the advantage that a limitation on the permitted policies is imposed only in the presence of persistent oscillations. However, in such cases, it requires a significant deviation from locally assigned policies: ASes involved in an oscillation must switch to a (potentially much) lower preferred path, both for the traffic originating at the AS and the transit traffic.

In this paper, we take a different approach which guarantees stability with less restrictive policies. Our key observation is that oscillations are fundamentally caused by the interdependence of routes across ASes, and this interdependence can be broken through *multipath routing*. Specifically, we show that a selection of routes is possible such that every AS uses its most-preferred available path, and some ASes may be required to carry transit traffic along additional paths in order to break oscillations. Thus, instead of requiring ASes to change their preferences over the set of paths as in [5], [6], we restrict policies only by requiring a few ASes to carry transit traffic along certain paths in addition to their most preferred path. Any scheme which guarantees stability must restrict ASes' policies in some way; we believe our approach may offer a good tradeoff in order to guarantee stability, as long as it is rare that an AS is forced to use one of these less-preferred transit paths. Therefore, a key question for our approach is how to guarantee convergence while assigning only a small number of additional paths. We additionally note that a multipath-based approach may be desirable as it has several benefits in its own right, including improved security, availability, and flexibility [7], [8], [9].

To formalize the idea of stability using multipath routing, we present the Multiple Stable Paths Problem (MSPP) (Section III), where an AS is allowed to be assigned a set of paths, and the set is stable if it includes the AS's most preferred path. We present a *centralized* algorithm Stable Path(s) Assignment (SPA) (Section IV), that given an instance of MSPP, assigns paths to each AS in the network in a way that guarantees: (1) each AS is assigned its highest preferred path among the set of available paths, and (2) the set of assigned paths is small.

If the set of assigned paths are all of size one, then the solution corresponds to the route assignment using BGP. The two key challenges addressed by SPA are: one, in absence of routing oscillations, the solution of SPA must correspond to that of BGP and two, a way of keeping the size of the set of assigned paths small when routing oscillations do occur, which corresponds to taking away only a little autonomy from the ASes.

In order to understand the behavior of SPA in distributed settings, we design a distributed protocol for SPA (Section V) and evaluate the performance of SPA on the AS graph (Section VI). Our evaluation results suggest two main conclusions; first, misconfigurations in the network can have significant impact on the routing stability in the network. In particular, we show that even with 1% of the ASes misconfigured, the network may have dispute wheels with overwhelming probability. Second, assigning at most one extra path to very few ASes in the network may be sufficient to guarantee routing stability. In particular, in 99.9% of the networks that had a dispute wheel, SPA assigned at most one extra path to any AS in the network. Furthermore, in 99.9% of the instances when SPA assigned more than one path to any AS, less than 0.04% of the ASes were assigned the extra path.

Our evaluation results also suggest that SPA can effectively detect (with high probability) networks that have a stable state but can potentially face persistent oscillations. In particular, assuming that all the networks that had a dispute wheel also had a stable state, SPA detected (and assigned) the stable state for more than 91% of these networks. This means that for more than 91% of the networks that had a dispute wheel, SPA assigned a single path to each AS in the network.

II. Related Work

The possibility of persistent oscillations in BGP was first noticed in [1]. Griffin et al. [2] introduced Stable Paths Problem as a formal model for policy routing with path vector protocols. They proved that a sufficient condition for guaranteed convergence of BGP is the absence of *dispute wheels*, a structure that captures the conflicting routing policies of the ASes that are involved in the oscillations. The essence of [2], and a series of theoretical results that followed [10], [11], [12], is that the problem of routing oscillations is a fundamentally inherent property of any path-vector routing protocol that allows each AS to select its highest preferred available path. Hence, by definition, any scheme that guarantees stability would require to limit the autonomy of the ASes in some form. However, the degree to which autonomy must be limited to achieve stability is an open problem.

A natural approach to avoid persistent oscillations is to restrict the set of paths an AS can select its path from. Gao and Rexford [5] presented a set of such restrictions that guarantee convergence of BGP. However, restricting each AS in the Internet to select paths based on these restrictions is unrealistically restrictive: an AS can not choose a higher preferred path that is restricted, even when there are no policy conflicts in the network. Moreover, doing this consistently across ASes (which do not share a global view of the topology) is a challenging problem. Finally, even if one can achieve such a path assignment, violations to these policies due to complex business agreements and misconfigurations can still induce persistent oscillations.

Another approach is to dynamically detect persistent routing oscillations and limit autonomy only when persistent oscillations are detected. Such schemes would induce restrictions on local policies at some ASes when a dispute wheel does occur [6], [13], [14]. While this restricts limiting AS autonomy only when necessary, the proposed schemes require ASes to select a lower preferred path for their own traffic as well as the transit traffic. Our solution, on the other hand, allows each AS in the network to route its traffic through its highest preferred available path, while requiring (very few) ASes to route some of the transit traffic through a lower ranked path. Moreover, we show through simulations that our scheme can guarantee stability while requiring use of lower-ranked paths on only an extremely small fraction of ASes; this extent of autonomy loss was not evaluated in [6], [13], [14].

Exploiting multipath routing as a tool to design stable routing protocols has been explored in at least two other solutions. Haxell and Wilfong [15] introduced a fractional model of BGP, where an AS is assigned multiple paths to the same destination. They proved that every instance of a network under Fractional-BGP model is solvable. However, their proof is non-constructive and in fact the path assignment for Fractional-BGP is "hard" (a PPAD-complete problem) [16]. Furthermore, in fractional-BGP, some ASes in the network may be required to be assigned exponentially many paths in order to guarantee convergence, even when each AS is involved in a single minimal dispute wheel. In contrast, our scheme guarantees stability by assigning no more than a single extra path per minimal dispute wheel an AS is involved in (for a comparison of our scheme to Fractional-BGP, see [17]).

Wang *et al.* introduced NS-BGP [18], a neighbor-specific routing model that allows ASes to select multiple paths, each of which is used by the AS to transit at least one of its neighbor's traffic. It was shown that such a path-assignment protocol is inherently more stable when compared to BGP. However, despite multiple path assignments, NS-BGP does not guarantee stability without any restriction on the set of paths an AS can select its path from, leading to similar issues as with [5]. Our algorithm SPA, on the other hand, guarantees stability and as discussed earlier, evaluation results suggests that SPA assigns just a single extra path to very few ASes in the network.

III. Multiple Stable Paths Problem

We start by giving an example as an intuition to why multipath routing may help in stabilizing the BGP route selection process. Using this example, we discuss some of the issues with designing an efficient multipath routing solution to the BGP instability problem. We then give a formal definition of the Multiple Stable Paths Problem (MSPP), and summarize the results from the following sections.

Assume, for the following example, that all an AS cares about is to route its own traffic through its most preferred available path.

Example 1: Consider an example network shown in Fig. 1. This network has no stable state if each node insists on routing all the traffic (both, the originating and the transit traffic) through its highest preferred available path [2]. However, if we allow some nodes in the network to be assigned multiple paths, it is not very difficult to realize that persistent oscillations in the network can be avoided. For instance, one of the path assignment using Stable Path(s) Assignment for MSPP is $\{\{130, 10\}, 210, 30, 430\}$, *i.e.*, node 1 is assigned paths $\{130, 10\}$, node 2 is assigned path 210 and so on. To see this, note that each of the nodes in the network is assigned its highest preferred *available* path to the destination (paths 3420 and 420 are not available).

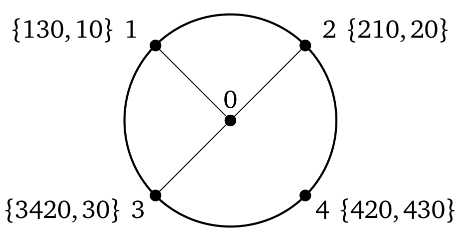

Figure 1. Bad Gadget. The set next to the node denotes the set of permissible paths at that node in decreasing order of ranking.

There are several important issues that need to be addressed in light of the above example. Our approach requires some ASes (node 1 in Example 1) in the network to select and advertise multiple routes. The first problem is to design a scheme that allows such path selection so as to guarantee stability in the network. Furthermore, in practice, the assumption made before the example could be unrealistic: an AS cares not only about the path taken by the traffic originating at the AS, but also about the path taken by the transit traffic. Hence, requiring an AS to select and advertise multiple paths can be regarded as limiting autonomy of that AS in the following sense: an AS advertising multiple routes to its neighbors might require to route its neighbor's traffic through a path which is not the highest preferred path of that AS. For instance, node 1 in Example 1 routes 2's traffic through path 10, which is not 1's highest preferred path. Note that, in the absence of a monetary payment, node 1 has no incentive to do that, other than stability. We remark, however, that in all the earlier schemes node 1 will route its traffic and 2's traffic through path 10. In this sense, our scheme for the network of Example 1, limits less autonomy than all the previous schemes. In general, the scheme should assign as few paths as possible to an AS in the network so as to limit as less an autonomy as possible.

We revisit these issues in following sections, where we show that our algorithm, with extremely high probability, assigns a single extra path to ASes in the network while guaranteeing stability. We start the discussion on MSPP by describing the notation used in this paper.

A. Preliminaries

The network is represented as an AS graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is the set of ASes and each edge $e = (u\ v) \in E$ represents a BGP session between ASes u and v. We assume that AS 0 is the destination AS. A path in G is either the empty path, denoted by ϵ, or a sequence of ASes, $(v_k, v_{k-1}, \ldots, v_0)$. If P and Q are paths with u and v as their last and first ASes respectively, and if $(u\ v) \in E$, we denote by $P(u\ v)Q$ the path formed by the concatenation of these paths.

For each $v \in V$, let $\mathscr{P}^v$ denote the set of *permitted paths*[1] from v to the destination (AS 0). For the destination AS, we assume that $\mathscr{P}^0 = \{(0)\}$. Let $\mathscr{P} = \cup_v \mathscr{P}^v$ be the union of all sets of permitted paths. For each $v \in V$, there is a non-negative integer valued *ranking function* λ^v, defined over $\mathscr{P}^v$, which represents how AS v ranks its permitted paths. We assume strictness of the ranking functions, namely, for any AS v and two paths $P_1, P_2 \in \mathscr{P}^v$, $\lambda^v(P_1) = \lambda^v(P_2)$ if and only if $P_1 = P_2$. Furthermore, for any two paths $P_1, P_2 \in \mathscr{P}^v$ and $\lambda^v(P_1) < \lambda^v(P_2)$, then P_2 is said to be preferred over P_1. We assume that for each AS $v \in V - \{0\}$, the empty path is contained in the set of permitted paths and for each path $P \in \mathscr{P}^v - \epsilon$, $\lambda^v(P) > \lambda^v(\epsilon)$. Let $\Lambda = \{\lambda^v | v \in V - \{0\}\}$.

An instance of the MSPP, $\mathscr{I} = (G, \mathscr{P}, \Lambda)$, is an undirected graph G together with the set of permitted paths at each AS $\mathscr{P}$ and the ranking functions for each AS Λ.

B. Stable Path Assignment

We describe the path assignment process at each AS in the network and the collective outcome of the processes at each AS: a stable path assignment.

Definition 1 (Path Assignment): Given an instance $\mathscr{I}$ of MSPP, a *path assignment* $\pi : V \to \mathscr{P}$ is a function π that maps each AS $u \in V$ to a set of paths $\pi(u) \subseteq \mathscr{P}^u$. By definition, $\pi(0) = \{(0)\}$.

[1]An AS may apply its local policies to decide a subset of available paths that are allowed to be used at that AS. Such paths are referred to as permitted paths at the ASes.

We will generally write a path assignment as a vector of sets, $(\pi(0), \pi(1), \ldots, \pi(n))$, where $\pi(u) \subseteq \mathscr{P}^u$ (recall that $\pi(0) = \{(0)\}$). The set of choices that an AS $u \in V$ gets to choose a path from are given by:

$$\text{choices}(\pi, u) = \begin{cases} \{(u\ v)\pi(v) | (u\ v) \in E\} \cap \mathscr{P}^u & \text{if } u \neq 0 \\ (0) & \text{otherwise} \end{cases}$$

Note that $\pi(v)$ could possibly be a set of paths and hence AS u could potentially have several choices of paths that have as next hop the same neighbor v. For any AS $u \in V$, we define the **best available path** (denoted as best(choices(π, u))) as the highest preferred path in choices(π, u). Note that the best "available" path depends on the path assignment π. When the context is clear, we will denote the best available path for an AS u as $P^*(u)$.

Definition 2 (Stable Path Assignment): Given an instance $\mathscr{I}$ of MSPP, a path assignment π is said to be *stable at AS u* if $\pi(u)$ contains best(choices(π, u)). The path assignment π *is stable* if it is stable at each AS in the network.

Note that the autonomy issues discussed earlier are captured within the definition of MSPP stable path assignment. Indeed, the only restriction we put for stability of an instance of MSPP is that an AS has its highest preferred available path in the set of paths assigned to that AS.

C. Summary of Results

We first summarize the results in the following sections:

- We present an algorithm Stable Path(s) Assignment (SPA), that produces a path assignment π that is stable, as defined in Definition 2.
- SPA imposes no restriction on the set of permitted paths at any AS in the network, and assigns each AS its highest preferred available path.
- If the network has no dispute wheel, SPA assigns paths exactly as BGP would.
- If the network does have a dispute wheel, then SPA requires that some ASes support routing along paths other than their highest preferred path. We give a tight upper bound on the number of paths assigned by SPA to any AS in the worst-case. However, using simulations on the AS graph, we show that in 99.9% of the cases, SPA assigns no more that two paths (*i.e.*, one extra path) to any AS in the network.
- We design a distributed protocol for SPA.

IV. Stable Path(s) Assignment

In this section, we present Stable Path(s) Assignment (SPA), a *centralized* algorithm which when given an instance $\mathscr{I} = (G, \mathscr{P}, \Lambda)$ of MSPP, assigns paths to each AS in the network such that the final path assignment corresponds to a stable path assignment of MSPP. We give a distributed protocol that implements SPA in the next section.

We also introduce a graph theoretic structure *dispute graph* and show that a directed cycle in a dispute graph corresponds to the possibility of persistent routing oscillations in the network. We then exploit this observation to provide proofs of correctness of SPA. We close the section with some properties of SPA including a tight bound on the maximum number of paths assigned by SPA to any AS in the network. To start with, we give some definitions which will allow us to succinctly describe the algorithm.

Definition 3 (Partial Path Assignment): A *partial path assignment* $\pi = \{\{(0)\}, \pi(1), \pi(2), \ldots, \pi(n)\}$ for $V' \subseteq V$ is a path assignment such that for every $u \in V'$, every AS in $\pi(u)$ is in V'.

Definition 4 (Consistent Path): For an AS v_0, a path $P = (v_0, v_1, \ldots, v_k, 0) \in \mathscr{P}^{v_0}$ is said to be *consistent with path assignment* π if for each $v_i \in P$, the subpath $P_i = (v_i, \ldots, v_k, 0)$ is either in $\pi(v_i)$ or is ranked higher than each of the paths in $\pi(v_i)$. If $P \in \pi(v_1)$, then P is said to be a **direct path**, else we call it an **indirect path**.

Example 2 (Example 1 continued): Consider again the network shown in Fig. 1 and a run of BGP. Consider a snapshot during the path assignment process with $\pi = \{0, 130, 20, 30, 430\}$. First, note that for $V' = \{0, 1, 2, 3\}$, π can be called a partial path assignment but for $V' = \{0, 1, 2\}$, π is not a partial path assignment. Next, consider node 1 ($\pi(1) = \{130\}$). We claim that path 10 is inconsistent with the path assignment π. This is due to the fact that $\lambda^1(10) < \lambda^1(\pi(1))$. Next consider nodes 4 and 3. The claim is that path 420 is a direct path for node 4 that is consistent with π, while the path 3420 is an indirect path for node 3 that is consistent with π. We leave it as an exercise to confirm the claims (see [17]).

It is easy to see and particularly important to note that for two partial path assignments, π_i and π_j, if for each AS u, $\pi_i(u) \subseteq \pi_j(u)$ and if a path P is *not* consistent with π_i, then P can not be consistent with π_j.

Definition 5 (Stable and Unstable Sets): The *stable set* $\mathscr{S}$ is defined to be the set of ASes $u \in V$, for which the highest ranked path consistent with π is assigned to u. The *unstable set* is the set of ASes that are not in the stable set.

Definition 6 (Dispute Wheel [2]): A *dispute wheel* $\mathscr{W} = \{\{u_1, u_2, \ldots, u_k\}, \{Q_1, Q_2, \ldots, Q_k\}, \{R_1, R_2, \ldots, R_k\}\}$ is a sequence of nodes $u_1, u_2, \ldots, u_k$, a sequence of non-empty paths $Q_1, Q_2, \ldots, Q_k$ and a sequence of paths $R_1, R_2, \ldots, R_k$ such that for each $1 \leq i \leq k$, we have (1) R_i is a path from u_i to u_{i+1}, (2) $Q_i \in \mathscr{P}^{u_i}$, (3) $R_i Q_{i+1} \in \mathscr{P}^{u_i}$, and $\lambda^{u_i}(Q_i) < \lambda^{u_i}(R_i Q_{i+1})$ (all subscripts taken modulo k).

Definition 7 (Minimal Dispute Wheel [2]): A *minimal dispute wheel* is a dispute wheel in which for each $1 \leq i \leq k$, either $R_i R_{i+1} Q_{i+2}$ is not permitted at u_i or $\lambda^{u_i}(R_i R_{i+1} Q_{i+2}) < \lambda^{u_i}(R_i Q_{i+1})$.

Examples illustrating the idea of dispute wheels and minimal dispute wheels can be found in [2], [17].

Figure 2. Pseudo-Code for Initial Path Assignment

```
Initial Path Assignment:
While $V_i \neq V$ Do
  If $\mathscr{S}_i \neq \emptyset$
      Assign each node in $\mathscr{S}_i$ its highest ranked path consistent with $\pi_{i-1}$
      $V_{i+1} = V_i \cup \mathscr{S}_i$
  Else If $\mathscr{D}_i \neq \emptyset$
      Let $v$ be an arbitrarily chosen node from $\mathscr{D}_i$
      Assign $v$ its highest ranked direct path consistent with $\pi_{i-1}$
      $V_{i+1} = V_i \cup \{v\}$
  Else
      Assign each node in $V - V_i$ the empty path
      $V_{i+1} = V$
  $i = i + 1$
Let $\pi = \pi_i$
Let $\mathscr{S} = \cup_i \mathscr{S}_i$
```

Figure 3. Pseudo-Code for Stable Path(s) Assignment – SMDW

```
Stable Path(s) Assignment – SMDW:
Run Initial Path Assignment to compute $\pi$ and $\mathscr{S}$
Let $\mathscr{U} = V - \mathscr{S}$
Initialize $j$
While $|\mathscr{U}| \neq 0$
  Let $\mathscr{U}_j \subseteq \mathscr{U}$ be the set of nodes whose highest ranked path consistent with $\pi$ is a direct path
  Assign each node in $\mathscr{U}_j$ its highest ranked path consistent with $\pi$
  $\mathscr{U} = \mathscr{U} - \mathscr{U}_j$
  $j = j + 1$
```

A. Stable Path(s) Assignment Algorithm

We present Stable Path(s) Assignment (SPA) algorithm. To give an intuition for the underlying functioning of the algorithm, we first consider a simpler case, when the instance $\mathscr{I}$ of MSPP contains a single minimal dispute wheel. We then give a simple generalization to the case of networks having multiple minimal dispute wheels.

The algorithm (for the case of single minimal dispute wheel) runs in two phases: in first phase Initial Path Assignment, each AS is assigned a single path to the destination (some ASes may be assigned empty paths); the second phase Stable Path(s) Assignment – SMDW (SMDW for single minimal dispute wheel) assigns each AS in the unstable set its highest ranked path consistent with the current path assignment. Both phases assign paths in a greedy fashion, first assigning paths to the ASes whose highest ranked path consistent with current path assignment is a direct path. If no such AS is available, the algorithm will arbitrarily choose an AS that has direct paths to the destination and assign the highest ranked path among these (direct) paths to the AS. The algorithms Initial Path Assignment and Stable Path(s) Assignment – SMDW are given in Fig. 2 and Fig. 3 respectively.

More formally, the first phase Initial Path Assignment, constructs a sequence of subsets of V, $\{0\} = V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_k = V$, together with a sequence of partial path assignments $\pi_0, \pi_1, \ldots, \pi_k$, where each π_i is a partial path assignment for V_i. Let $\mathscr{D}_i$ be the set of ASes $u \in V - V_i$ that have a direct path consistent with π_i. Let $\mathscr{S}_i \subseteq \mathscr{D}_i$ be the set of ASes whose highest ranked path consistent with π_i (*i.e.*, best(choices(π_i, u))), is a direct path. The second phase takes the ASes in the unstable set and assigns, in each iteration, at least one AS its highest ranked path consistent with the current path assignment. Note that for correctness of the algorithm, it suffices to prove that in each iteration of Stable Path(s) Assignment – SMDW, there is at least one AS whose highest ranked path consistent with current path assignment is a direct path, *i.e.*, for all j, either $|\mathscr{U}_j| > 0$ or $|\mathscr{U}| = 0$.

Example 3: Consider again the network shown in Fig. 1. Table I gives the step-wise path assignment when the algorithm Stable Path(s) Assignment – SMDW is run on the Bad Gadget. It is easy to see that the final path assignment using the algorithm, $\pi = \{0, \{10, 130\}, 210, 30, 430\}$, is indeed a stable path assignment for MSPP.

Next, we prove the correctness of the algorithm.

Table I
ITERATIONS OF STABLE PATH(S) ASSIGNMENT – SMDW FOR NETWORK OF FIG. 1

iteration i	$\mathscr{V}_i$	π_i	$\mathscr{D}_i$	$\mathscr{S}_i$	$\mathscr{U}$	$\mathscr{U}_i$
0	$\{0\}$	$\{0,-,-,-,-\}$	$\{1,2,3\}$	$\{\}$	$\{1\}$	-
1	$\{0,1\}$	$\{0,10,-,-,-\}$	$\{2,3\}$	$\{2\}$	$\{1\}$	-
2	$\{0,1,2\}$	$\{0,10,210,-,-\}$	$\{3\}$	$\{3\}$	$\{1\}$	-
3	$\{0,1,2,3\}$	$\{0,10,210,30,-\}$	$\{4\}$	$\{4\}$	$\{1\}$	-
4	$\{0,1,2,3,4\}$	$\{0,10,210,30,430\}$	$\{\}$	$\{\}$	$\{1\}$	-
0	$\{0,1,2,3,4\}$	$\{0,10,210,30,430\}$	-	-	$\{1\}$	$\{1\}$
1	$\{0,1,2,3,4\}$	$\{0,\{10,130\},210,30,430\}$	-	-	$\{\}$	$\{\}$

B. Correctness of SPA

We start by defining *dispute graph*, a directed graph that captures the set of conflicting rankings between ASes. We then show that edge-disjoint cycles in a dispute graph correspond to distinct minimal dispute wheels and exploit this to give a simple proof for the correctness of the algorithm.

Definition 8 (Dispute Graph): Given an instance $\mathscr{I} = (G,\mathscr{P},\Lambda)$ and a path assignment π, a dispute graph corresponding to π is a directed graph $\mathscr{G}(\mathscr{I},\pi)$ constructed as follows: For each node $u \in V\backslash\{0\}$, we create a node in the dispute graph. Let $P^*(u) = (u, v_1, v_2, \ldots, v_k, 0)$ denote the highest ranked path of u consistent with π. For each *intermediate* node v_i, we create an edge $(u\ v_i)$ in the dispute graph.

Definition 9: A cycle $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_k \rightarrow u_1$ is said to be minimal if there do not exist any $i, j \neq i+1$ such that $u_1 \rightarrow \cdots \rightarrow u_i \rightarrow u_j \rightarrow \cdots \rightarrow u_1$ is also a cycle.

Lemma 1: Let $\mathscr{I} = (G,\mathscr{P},\Lambda)$ be an instance of MSPP and let $\mathscr{G}(\mathscr{I},\pi)$ be the dispute graph for $\mathscr{I}$ corresponding to a path assignment π. Then if $\mathscr{G}(\mathscr{I},\pi)$ has k edge-disjoint cycles, $\mathscr{I}$ has at least k minimal dispute wheels.

Proof: We show that each edge-disjoint cycle corresponds to a minimal dispute wheel in the network. Consider a minimal cycle $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_k \rightarrow u_1$ in $\mathscr{G}(\mathscr{I},\pi)$. Then, by definition of a dispute graph, for any node u_i in the cycle, u_{i+1} must be in the highest ranked path of u_i that is consistent with π. This implies that one can construct a minimal dispute wheel as follows: Let $P^*(u_i)$ denote the highest rated path of u_i and let R_i be the subpath of $P^*(u_i)$ from u_i to u_{i+1} and let Q_{i+1} be the remaining subpath of $P^*(u_i)$. This will give a minimal dispute wheel $W = \{\{u_1,u_2,\ldots,u_k\},\{Q_1,Q_2,\ldots,Q_k\},\{R_1,R_2,\ldots,R_k\}\}$. ■

Lemma 2: Let $\mathscr{U}$ denote the unstable set after INITIAL PATH ASSIGNMENT and let W be a dispute wheel in $\mathscr{U}$. For each $u \in W$, let $P^*(u)$ denote the highest ranked path of u consistent with π. Then, $P^*(u)$ must contain at least 4 nodes from V.

Proof: Consider any node $u \in W$ and let $P^*(u)$ be its highest ranked path consistent with π. Note that since u is in a dispute wheel $P^*(u)$ must not be a direct path for u. Clearly, $P^*(u)$ must contain at least two nodes u and 0. If $\pi(u) = \epsilon$, then $(u\ 0) \notin \mathscr{P}^u$ and if $\pi(u) \neq \epsilon$, then $\lambda^u(\pi(u)) \geq (u\ 0)$, hence $P^*(u)$ must contain at least three nodes. We need to show that if $u \in W$, $P^*(u)$ must contain at least four nodes. For sake of contradiction, assume that it contains only three nodes. Then, $P^*(u) = uu'0$ for some $u' \in V$. Clearly, $(u'\ 0) \in \pi(u')$ since $P^*(u)$ is consistent with π. But then, $P^*(u)$ is a direct path for u leading to the contradiction that $u \in W$. Hence, $P^*(u)$ must have at least 4 nodes. ■

Lemma 3 ([19]): A k-regular directed graph with no parallel edges contains at least $5k/2 - 2$ edge-disjoint cycles.

Theorem 1: The algorithm STABLE PATH(S) ASSIGNMENT – SMDW solves all instances $\mathscr{I} = (G,\mathscr{P},\Lambda)$ of *MSPP* if $\mathscr{I}$ does not contain multiple minimal dispute wheels.

Proof: Consider the first iteration of STABLE PATH(S) ASSIGNMENT. If there is no dispute wheel in $\mathscr{U}$, then clearly each node is assigned a stable path. However, if there is a dispute wheel in $\mathscr{U}$, then we claim that there must be multiple minimal dispute wheels in the network.

To prove this, let $\mathscr{I} = (G,\mathscr{P},\Lambda)$ be an instance of MSPP and let $\mathscr{G}(\mathscr{I},\pi)$ be the dispute graph for $\mathscr{I}$ corresponding to current path assignment. Let W be a dispute wheel in the unstable set $\mathscr{U}$ after INITIAL PATH ASSIGNMENT. By Lemma 2, the highest ranked path of each node in W must have at least four nodes, out of which two nodes must be different from u and the destination node. Hence, by construction, each node in $\mathscr{G}(\mathscr{I},\pi)$ must have an out-degree of at least 2. Then, by Lemma 3, there must be at least three edge-disjoint cycles in $\mathscr{G}(\mathscr{I},\pi)$. This in turn, by Lemma 1, means that there must be multiple minimal dispute wheels in $\mathscr{I}$. ■

The above results naturally generalize to the case of multiple minimal dispute wheels. We give the algorithm for the case of multiple minimal dispute wheels in the following subsection. We close this section with discussion on the properties of the SPA solutions. Due to lack of space, proofs for the following theorems and claims have not been included and can be found in [17].

Figure 4. Pseudo-code for Stable Path(s) Assignment

```
Stable Path(s) Assignment:
While $\mathscr{S} \neq V$
  Run Initial Path Assignment with $V = \mathscr{U}$ to compute $\pi$ and $\mathscr{S}$.
  Let $\mathscr{U} = V - \mathscr{S}$.
  Initialize $j$ to 0.
  Let $\mathscr{U}_0 \subseteq \mathscr{U}$ be the set of nodes whose highest ranked path consistent with $\pi$ is a direct path.
  While $|\mathscr{U}_j| \neq 0$
      Assign each node in $\mathscr{U}_j$ its highest ranked path consistent with $\pi$
      $\mathscr{U} = \mathscr{U} - \mathscr{U}_j$
      Compute $\mathscr{U}_{j+1}$: the set of nodes whose highest ranked path consistent with $\pi$ is a direct path.
      $j = j + 1$
  Let $\mathscr{S} = V - \mathscr{U}$
```

C. *Case of Multiple Minimal Dispute Wheels*

When there are multiple minimal dispute wheels in $\mathscr{I}$, one may need to assign more than two paths to a single node. This can be achieved by running algorithm Initial Path Assignment every time a dispute wheel is found in the unstable set $\mathscr{U}$. Each iteration of Initial Path Assignment will result in breaking at least one minimal dispute wheel. However, we believe (and show using evaluation results) that for most of the networks, multiple minimal dispute wheel case is also resolved in a single run of Initial Path Assignment. The pseudo-code for Stable Path(s) Assignment algorithm is shown in Fig. 4.

Theorem 2: The algorithm Stable Path(s) Assignment-MMDW solves all instances $(G, \mathscr{P}, \Lambda)$ of *MSPP*.

D. *Properties of SPA*

We discuss some of the properties of SPA solutions. The proofs for the claims can be found in [17].

Claim 1: In absence of dispute wheels, Stable Path(s) Assignment assigns a single path to each node. The paths assigned correspond to the paths assigned by BGP.

Claim 2: No node in (a single minimal dispute wheel of) a network is assigned more than two paths by Stable Path(s) Assignment – SMDW.

Claim 3: Given an instance $\mathscr{I} = (G, \mathscr{P}, \Lambda)$ of MSPP, let $m(u)$ denote the number of minimal dispute wheels that an AS u is in. Then Stable Path(s) Assignment assigns no more than $m(u)+1$ paths to u. This is tight.

We make two observations. First, the bound of Claim 3 is valid for any multipath assignment algorithm that guarantees stability and that operates in a distributed fashion, in the sense that it does not restrict the order in which nodes in dispute wheel are assigned paths. Second, the number of paths assigned in the worst case could be as many as the number of nodes in the dispute wheel. Whether or not it is large is an interesting question; however, we show in Section VI that in almost all the cases, just one extra path suffices to guarantee stability.

V. Distributed Stable Path(s) Assignment

Previous sections have described our overall approach. However, thus far we have assumed that each AS in the network can observe the entire graph. In this section, we describe some simple extensions to our technique to allow it to operate as a dynamic routing protocol in distributed environments. For simplicity, we concentrate only on sketching the main ideas of our approach, relying on previous work in detecting dispute wheels [13], [6], multipath routing [8], [20], and BGP implementations [21] to handle details not discussed here.

The distributed protocol (corresponding to a single destination prefix) is shown in Fig. 5. The algorithm consists of two key steps.

A. *Dispute detection*

First, each AS continuously checks to determine if it is part of a dispute wheel. This can be performed by adapting any of the known techniques [6], [13]. We do this by extending the SPVP formulation [13], which allows ASes to detect their being in dispute by maintaining a *route history* and detecting cycles in the history. Our extension [17] allows an AS that receives an advertisement containing *multiple paths* to detect if it is still in dispute; the originial SPVP formulation did not need to handle this case. Using this technique, each AS can determine whether it is currently in dispute with other ASes.

B. *Handling disputes*

Once an AS has detected its involvement in a dispute, the protocol places a *restriction* on the path selection process of the AS. The restriction disallows withdrawals that are caused by availability of a higher preferred path (such withdrawals are implicit in BGP). When an AS processes a new (and higher preferred) path, it selects this path and forwards it as an additional path to break the dispute. In particular, when the restriction is enabled, the AS *marks* the current path that is advertised. The AS

Figure 5. Distributed Stable Path(s) Assignment Algorithm

```
DISTRIBUTED STABLE PATH(S) ASSIGNMENT:
For each node $v$
On receiving an advertisement for path $P$:
  Check whether or not in dispute
  If $P$ preferred over all paths in $\pi(v)$
    If in dispute wheel
      Select the path $P$ (as an additional path)
      Let $\pi(v) = \pi(v) \cup \{P\}$
    Else
      Select the path $P$
      Let $\pi(v) = \{P\}$
  Advertise $\pi(v)$ to neighbors

On receiving a withdrawal for path $P$:
  $\pi(v) = \pi(v) \backslash \{P\}$
  Withdraw path $P$ from neighbors
```

then additionally marks any new more-preferred paths that are received after the restriction is enabled. The AS then advertises the *set* of these paths to its neighbor. Note that this procedure does not affect how physical failures and other withdrawals are handled – only path changes due to receiving a more-preferred path are affected. On a failure (or a path withdrawn by the neighbor), the AS would receive a withdrawal message, causing the path to be unmarked and deleted from its own routing table (and subsequently deleted from the set advertised to the AS's neighbors).

Note that the description above handles the network dynamics. Indeed, during the convergence process, an AS u that has detected its presence in a dispute wheel may either receive a path advertisement or a path withdrawal. A path advertisement does not affect u's adherence to the restrictions imposed in order to handle the disputes; the path advertisement received by an AS under restriction will not cause path withdrawals. An explicit path withdrawal, on the other hand, may be due to a change in the topology of the underlying network or due to one of u's neighbors, say AS v, withdrawing the path being advertised by u to its neighbors. Consider AS v. Either v is in the dispute wheel or not. If not, then the path withdrawal by AS v is allowed by the protocol (recall, the handling disputes above puts restriction only when an AS is in dispute wheel). If v is indeed in dispute wheel, at some point of time v will detect its presence in the dispute wheel and at that point of time, both u and v will stop withdrawing the paths. Following this process, at some point, all the ASes in the dispute wheel will stop withdrawing the paths and all the advertisements will eventually be consumed by the ASes in the network, resulting in the network reaching a stable state.

VI. Evaluation Results

In this section, we report on our implementation of Stable Path(s) Assignment (SPA) and its evaluation on an AS-level network topology. We discuss below the methodology used for evaluating the performance of the algorithm (Sec. VI-A). We then describe our results on the effect of misconfigurations on routing instability for traditional single-path routing (Sec. VI-B), and the number of paths assigned by SPA to guarantee convergence (Sec. VI-C).

A. Methodology

SPA requires several inputs: (1) a network topology, (2) a set of possible paths for each AS, and (3) a ranking of these paths to define policies.

Topology: We use an AS-level network topology from CAIDA [22], which used route dumps from January 2009 to construct an AS-level network which consists of 24,113 ASes.

Possible paths: Distributed protocols such as ours and BGP require only a function to compare the preference of two paths. However, SPA requires an explicit list of possible paths sorted in order of preference. We begin by generating a list of possible paths. Unfortunately, there could be exponentially many paths from each AS to each destination. Hence, we resorted to a heuristic preprocessing approach: we implemented a "policy-free" path-vector protocol in which each AS simply advertises every path it receives, up to at most 100 paths to each neighbor and at most 10 paths of each length, for each destination. After running this protocol, we collect the set of advertised paths at each AS, and use these as the possible paths that are the input to SPA. We believe this set of paths—which includes paths through all neighbors and of various lengths—is a representative set given that the policies we will test are based on the path's next-hop and length.

Policies: To rank the set of possible paths, we implemented a simple shortest-path based preference ranking function for most ASes. If an AS had multiple paths of the same length through the same neighbor, ties were broken arbitrarily. If all ASes used these policies, convergence would be guaranteed. However, in order to simulate misconfigurations or unusual policies, we select a small uniform-random set of ASes whose policies are defined by a random ordering of the possible paths at that AS.

Trials: In each trial of our algorithm we select a single random destination, a random set of misconfigured ASes, and random policies for those ASes. We varied the fraction of ASes that are misconfigured; for each fraction, we ran 10,000 trials.

We note that the performance of SPA may vary if different assumptions are made about the topology, set of possible paths, or policies. We leave a detailed study of a broad range of environments to future work.

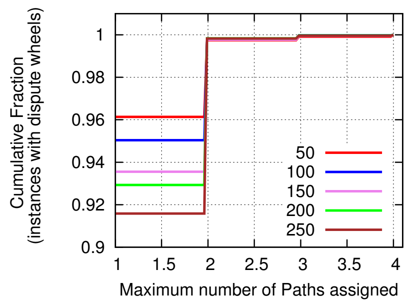

Figure 7. Cumulative fraction of instances (among all instances that had a dispute wheel) versus the maximum number of paths assigned to any AS for various number of misconfigured ASes

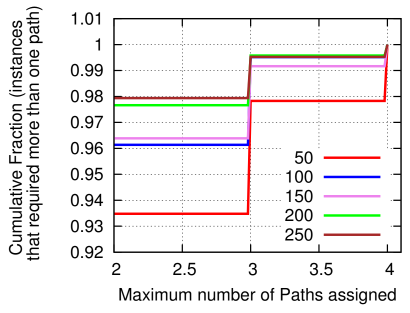

Figure 8. Cumulative fraction of instances (among all instances in which some AS was assigned more than one path) versus the maximum number of paths assigned to any AS for various number of misconfigured ASes

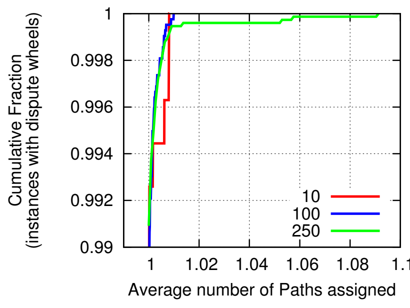

Figure 9. Cumulative fraction of instances (among all instances that had a dispute wheel) versus the average number of paths assigned (over all ASes) for various number of misconfigured ASes

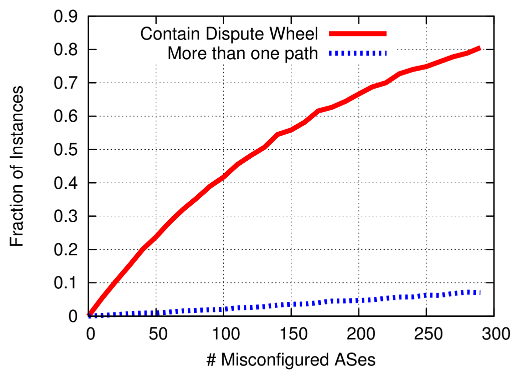

Figure 6. Fraction of instances with dispute wheel and fraction of instances in which SPA assigned more than one path in order to stabilize the network (out of 10,000 instances) versus number of misconfigured ASes.

B. *Misconfigurations and Instability*

Our first metric of interest is the intensity of the routing instability problem for traditional single-path routing. In particular, the effect of misconfigurations on the routing instability has not been evaluated in earlier works.

Of course, deciding convergence properties of an instance is hard: it is NP-complete to decide whether a stable state exists [2], and PSPACE-complete to decide whether BGP can oscillate [23]. We therefore study two upper bounds on the fraction of instances that are unstable. First, we compute the *fraction that have dispute wheels;* the absence of a dispute wheel is the classic sufficient condition for a unique stable state to exist [2]. Second, we compute the *fraction on which SPA converges with each AS in the network assigned a single path*.

Our evaluation results (shown in Fig. 6) suggest that even a few ASes being misconfigured can cause dispute wheels in the network. In particular, even with 0.05% of the ASes misconfigured (10 out of 24,113 ASes), more than 5% of the instances (out of 10,000 instances) had dispute wheels. The lower line in the figure shows that despite the presence of dispute wheels, SPA was usually able to find a stable state while giving each AS a single path to the destination. This indicates either that (1) the "no dispute wheel" condition is a very pessimistic bound on the fraction of instances that definitely converge, or (2) in a large fraction of cases divergence is possible, but requires an unlikely ordering of events. An interesting question for future work is to resolve which of these cases is most common.

However, there are still cases in which, when running SPA, the network did not converge with one path per AS. For example, just 0.1% of the ASes misconfigured (20 out of 24,113 ASes) led to non-single-path convergence in 0.45% percent of trials. Given the importance of convergence, this fraction is non-negligible. Moreover, each trial uses a single destination, so the chance of non-convergence when routing to all 24,112 destinations is presumably much higher.

C. *Number of Paths*

Finally, we study our main performance metric for SPA: the number of paths assigned to ASes in the network. Note that Claim 3 implies that the number of paths assigned to an AS by SPA could be as large as the number of ASes in the dispute wheel. However, in (almost) all the instances, SPA stabilized the network with at most one extra path assigned to (very) few ASes in the network. In particular, in 99.9% of the instances that had a dispute wheel, even with a large number of misconfigured ASes, SPA assigned at most one extra path to any AS in the network (see Fig. 7). Furthermore, the number of ASes that were assigned this extra path were an extremely small fraction of the total number of ASes: in more than 99% of the networks in which any AS was assigned an extra path, at most 0.04% of the ASes were assigned an extra path.

For networks in which the algorithm assigned more than two paths to any AS in the network, almost 97.5% of the instances were assigned at most three paths (see Fig. 8). The remaining instances were some of the unlucky instances, in which the algorithm ended up assigning four paths to some ASes in the network. In all the 300,000 iterations (over 30 values of number of misconfigured ASes), not even a single AS (in any instance) was assigned more than four paths.

This resulted in an extremely small number of paths assigned on an average (averaged over all the ASes in the network), as shown in Fig. 9.

VII. Conclusion

In this paper, we have presented a solution towards resolving the conflict between policy autonomy of Autonomous Systems in the Internet and stability of the inter-domain routing protocol BGP. The key tool we leverage is the use of multipath routing by ASes in conflict. We have presented an algorithm Stable Path(s) Assignment (SPA) that detects persistent oscillations and eliminates these oscillations by assigning multiple paths to some ASes in the network. SPA achieves a better trade-off between AS policy autonomy and system stability by allowing each AS to use its most preferred available path, while requiring very few ASes in the network to carry transit traffic along additional paths in order to break oscillations. We have shown that in absence of dispute wheels, SPA assigns path exactly as BGP. SPA is complemented with a simple distributed protocol that assigns multiple stable paths in a distributed fashion.

Through simulations, we have shown that with very high probability, SPA assigns at most one extra path to ASes in conflict while guaranteeing stability of the network.

References

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

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

[3] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," in *Proc. ACM SIGCOMM*, 1997, pp. 115–126.

[4] C. Labowitz, G. R. Malan, and F. Jahanian, "Origins of Internet routing instability," in *Proc. IEEE INFOCOM*, 1999, pp. 218–226.

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

[6] C. T. Ee, B. Chun, V. Ramchandran, K. Lakshminarayanan, and S. Shenker, "Resolving inter-domain policy disputes," in *Proc. ACM SIGCOMM*, 2007, pp. 157–168.

[7] D. Wendlandt, I. Avramopoulos, D. G. Andersen, and J. Rexford, "Don't secure routing protocols, secure data delivery," in *Proc. ACM HotNets*, 2006.

[8] W. Xu and J. Rexford, "Miro: Multi-path interdomain routing," in *Proc. ACM SIGCOMM*, 2006, pp. 171–182.

[9] J. He and J. Rexford, "Towards internet-wide multipath routing," *IEEE Network*, vol. 22, no. 2, pp. 16–21, 2008.

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

[11] T. G. Griffin, A. D. Jaggard, and V. Ramchandran, "Design principles of policy languages for path vector protocols," in *Proc. ACM SIGCOMM*, 2003, pp. 61–72.

[12] N. Feamster, R. Johari, and H. Balakrishnan, "Implications of autonomy for the expressiveness of policy routing," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1266–1279, 2007.

[13] T. G. Griffin and G. Wilfong, "A safe path vector protocol," in *Proc. IEEE INFOCOM*, 2000, pp. 490–499.

[14] J. A. Cobb, M. G. Gouda, and R. Musunuri, "A stabilizing solution to the stable paths problem," in *Proc. ACM Symposium on Self-Stabilizing Systems, 2704 LNCS*, Springer-Verlag, 2003, pp. 169–183.

[15] P. E. Haxell and G. T. Wilfong, "A fractional model of the border gateway protocol (BGP)," in *Proc. ACM-SIAM SODA*, 2008, pp. 193–199.

[16] S. Kintali, L. J. Poplawski, R. Rajaraman, R. Sundaram, and S. Teng, "Reducibility among fractional stability problems," in *Proc. IEEE FOCS*, 2009, pp. 283–292.

[17] R. Agarwal, M. Caesar, and P. B. Godfrey, "Stable path(s) assignment for inter-domain routing," University of Illinois at Urbana-Champaign, IL, USA, Tech. Rep., June 2009. [Online]. Available: www.ifp.illinois.edu/∼agarwa16

[18] Y. Wang, M. Schapira, and J. Rexford, "Neighbor-specific BGP: more flexible routing policies while improving global stability," in *Proc. ACM SIGMETRICS*, 2009, pp. 217–228.

[19] N. Alon, C. McDiarmin, and M. Molloy, "Edge-disjoint cycles in regular directed graphs," *Journal of Graph Theory*, vol. 22, no. 3, pp. 231–237, 1996.

[20] E. Rosen, A. Viswanathan, and R. Callon. (2001) Multiprotocol label switching architecture, RFC 3031.

[21] Y. Rekhter and T. Li. (1995) A border gateway protocol - 4, RFC 1771.

[22] The cooperative association for internet data analysis, URL: http://www.caida.org/home/.

[23] A. Fabrikant and C. Papadimitriou, "The complexity of game dynamics: BGP oscillations, sink equilibria, and beyond," in *Proc. ACM-SIAM SODA*, 2008, pp. 844–853.