

Design Principles of Policy Languages for Path Vector Protocols

Tim Griffin

AT&T Research

`griffin@research.att.com`

Aaron Jaggard

University of Pennsylvania

`adj@ee.upenn.edu`

Vijay Ramachandran

Yale University

`vijayr@cs.yale.edu`

Overview

- BGP is configured using low-level languages describing local policies.
- Vendor languages are expressive enough that local policies can interact to produce global routing anomalies.
- Lack of design of high-level policy languages

Research Goal

- Provide a theoretical framework for the interdomain routing problem in order to design languages for routing policies that are:
 - “Locally enforceable,” so that they produce “global sanity”
 - Not overly restrictive

Our Achievements

- A characterization of interdomain routing involving local policies
- Local conditions that guarantee stable routes
- [In progress] Development of policy languages that can enforce these local conditions

Outline

- **Definition of Path Vector Systems**
- Relationship to SPP
- Examples of Path Vector Systems
- Conditions that give unique solutions (and related results)
- Applying the above results to policy-language design

Path Vector Systems

$T = (G, R, F, \leq, r_o)$ is a **path vector system**.

- Simple, undirected graph $G(V, E)$
- A set R of **reachability objects**
 - R is a totally ordered set such that all $r \leq \infty$
- An object $r_o \in R$ associated with origin $o \in V$
- A pair of functions for each edge (i, j) :
 - **Import function** at i , $I_{(i,j)}(r) = s$, where $r, s \in R$
 - **Export function** at j , $E_{(i,j)}(r) = s$, where $r, s \in R$(Let F be the set of all of these functions.)

Path Vector Solutions

- $f_{(v,u)}(\bullet) = I_{(v,u)}(E_{(v,u)}(\bullet))$.
- **Object assignment:** $\rho: V \rightarrow R$ with $\rho(o) = r_o$.
- **Candidates:** $C(\rho, o) = \{r_o\}$,
 $C(\rho, v) = \{r \in R \mid (v,u) \in E \text{ and } r = f_{(v,u)}(\rho(u))\}$.
- ρ is a **solution** for a path vector system if $\rho(v) \in \min(C(\rho, v))$.
- Let $G_\rho = (V, A)$, where $(v, u) \in A$ iff $\rho(v) = f_{(v,u)}(\rho(u)) \neq \infty$. ρ is **acyclic** if G_ρ is.

Path Vector Systems, Intuitively

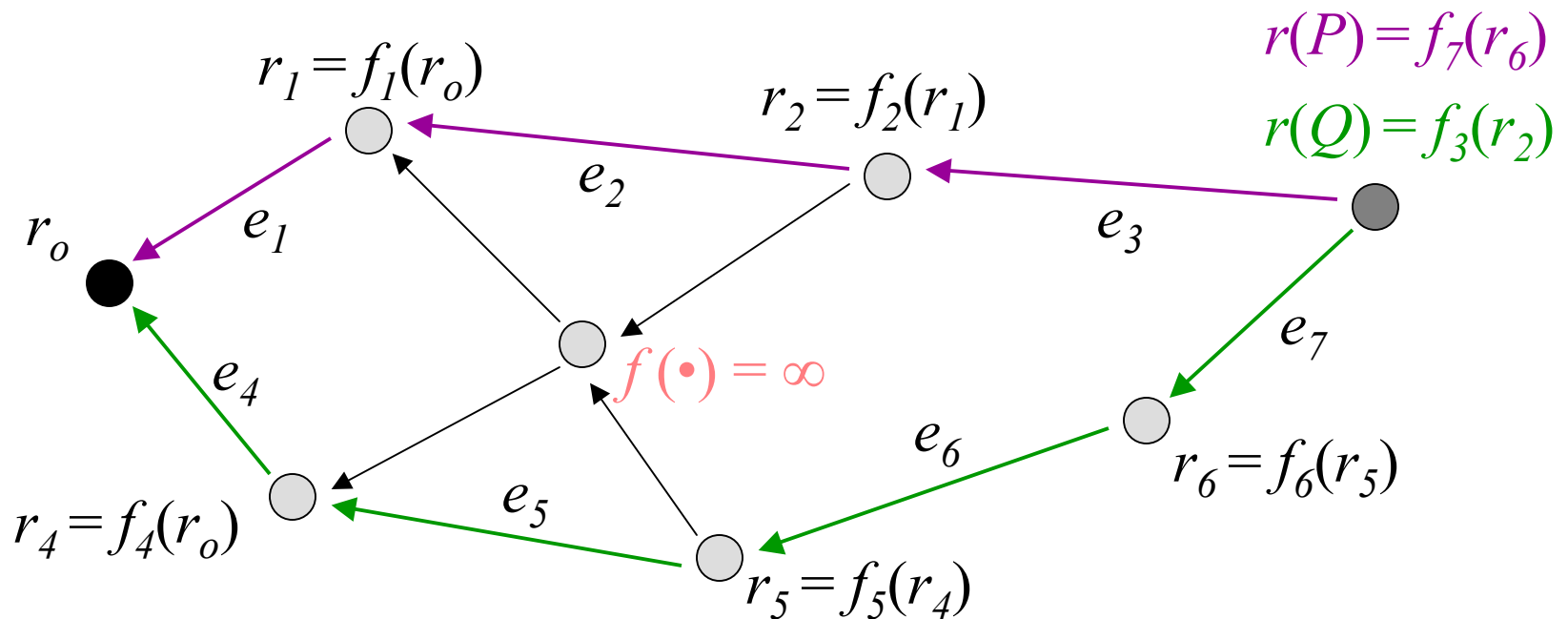
- Reachability objects “correspond” to nodes and paths, and describe the relative preference in using them to reach the origin.
- Local preferences are described using the import and export functions.
- Functions are specified at nodes for each adjacent edge, so knowledge of the entire graph is unnecessary at each node.

Outline

- Definition of Path Vector Systems
- **Relationship to SPP**
- Examples of Path Vector Systems
- Conditions that give unique solutions (and related results)
- Applying the above results to policy-language design

Path Vector System to SPP

- Set SPP rankings to match the order of **reachability objects of paths.**



SPP to Path Vector System

- Given an SPP, let R be the set of all permitted paths P .
- Choose an ordering consistent with the SPP rankings.
- Set $f_{(v,u)}(p) = \begin{cases} (v,u)p, & \text{if } p \text{ is permitted at } u \text{ and} \\ & (v,u)p \text{ is permitted at } v; \\ \infty, & \text{otherwise.} \end{cases}$

Outline

- Definition of Path Vector Systems
- Relationship to SPP
- **Examples of Path Vector Systems**
- Conditions that give unique solutions (and related results)
- Applying the above results to policy-language design

Example: LCP with Filtering

- $R = \mathbf{N} \cup \{\infty\}$ with $r_o = 0$.
- Local policies are defined by:
 - $E_{(v,u)}(r) = r + n$, for $n \geq 0$.
 - $I_{(v,u)}(r) = r + m$, for $m > -n$.
- To filter out a route, set $f_{(v,u)}(r) = \infty$.
- This corresponds to lowest-cost-path routing with edge cost $c(v,u) = n + m$ if $f_{(v,u)}(r) = r + m + n$.

Example: Simple BGP

- Let P be the set of simple paths in G terminating at the origin.
- Let R be $(\mathbf{N} \times P \times V) \cup \{\infty\}$ with lexical ordering $(n, p, w) \leq (m, q, u)$, $m \leq n$, $|p| \leq |q|$, $w \leq u$ (by IP address).
- Let $f_{(v,u)}(l, p, w) = (l', up, u)$.

Outline

- Definition of Path Vector Systems
- Relationship to SPP
- Examples of Path Vector Systems
- **Conditions that give unique solutions (and related results)**
- Applying the above results to policy-language design

Caveat

- Ignore the possibility that paths are equally ranked at a node.
- That is:
 - No cyclic path vector solutions
 - No ties in SPP rankings
- BGP does not allow ties, so this is fine as a model for BGP, but...
- ... ties will be addressed in the future.

Partially-Ordered SPPs

- Let $(p_1, p_2) \in R_1$ if p_1 is a subpath of p_2 .
- Let $(p_1, p_2) \in R_2$ if p_1 and p_2 are permitted at some node v and p_1 is ranked higher than p_2 .
- Let R be the transitive closure of $R_1 \cup R_2$.
- An SPP is **partially ordered** iff R is partially ordered.

Dispute Wheels and Partially-Ordered SPPs

- R is the ordering to choose (if possible) on reachability objects when converting SPPs to path vector systems.
- **Theorem 1:** An instance of SPP is dispute-wheel free *iff* it is partially ordered.
- **Proof:** Omitted, but similar to proof of Theorem 2 (next slide).

Increasing Path Vector Systems

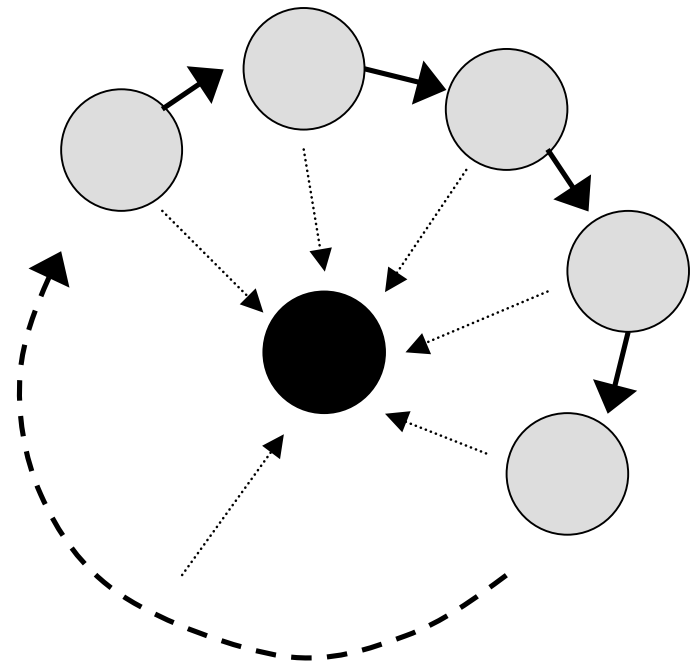
- A path vector system is **increasing** if, for each path $(i, j)_p$,

$$r((i, j)_p) > r(p), \text{ if } r(p) \neq \infty.$$

- **Theorem 2:** If T is an increasing path-vector system, then the corresponding SPP is dispute-wheel free (hence, partially ordered).

Proof of Theorem 2

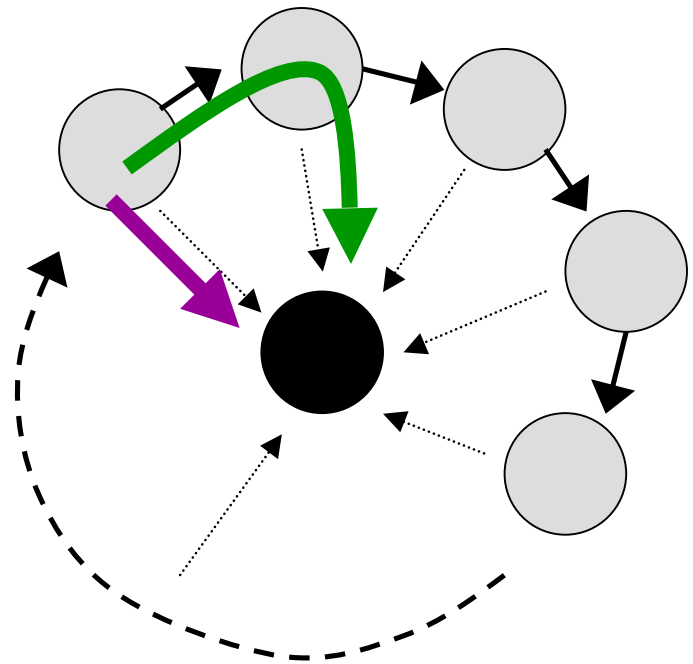
Proof: Suppose not: the corresponding SPP to the IPVS has a dispute wheel.



Proof (continued)

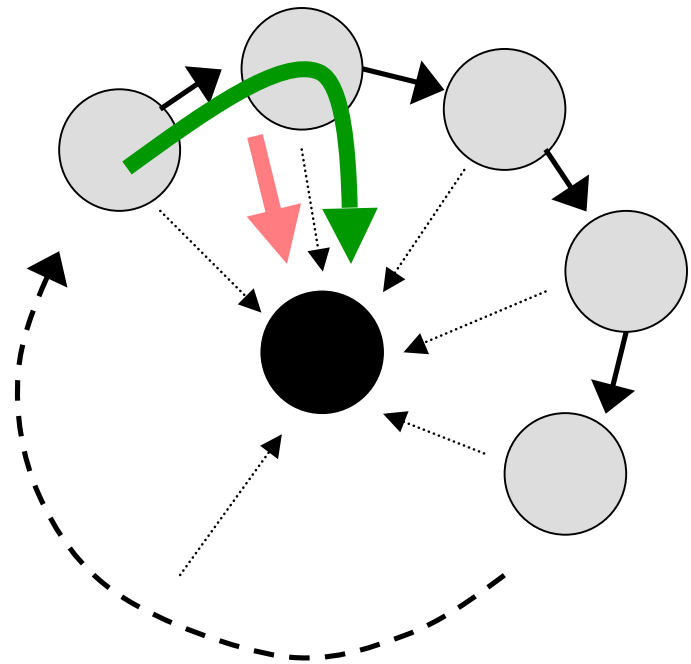
The path through the neighbor, $(i, j)P$,
is preferred over the path to the origin, Q .

So, $r((i, j)P) < r(Q)$.



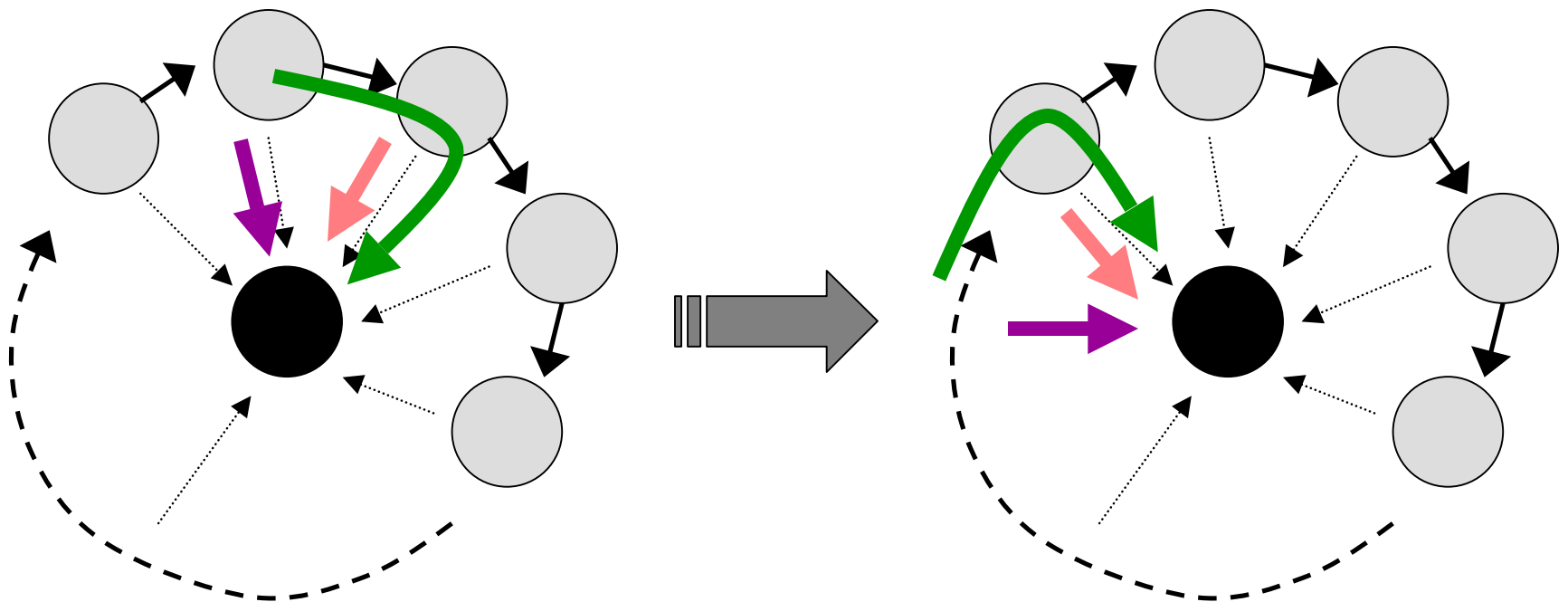
Proof (continued)

Because we have an IRS, $r(P) < r((i, j)P)$.



Proof (continued)

This is true all the way around the wheel;
so $r(Q) < r(Q)$, which is impossible.



Constructing an IPVS from an SPP

- Given a partially-ordered SPP, we can construct an IPVS corresponding to it:
 - over a set of reachability objects where, given an object, we can always find one larger (until the algorithm converges)
 - taking the reachability object set to be $\mathbf{N} \cup \{\infty\}$
- Given a path vector system T where $SPP(T)$ is partially-ordered, there exists an increasing T' with $SPP(T) = SPP(T')$. (Notion of equivalence?)
- The algorithm to do this requires **global knowledge** of the graph.

Algorithm (informally)

- At each node, list permitted paths in decreasing order.
- Assign each successive path the smallest available reachability object so that $f_e(\bullet)$, where e is the adjacent edge on the path, is increasing.
- Repeat, adjusting based on the reachability objects assigned by neighbors.

Algorithm Properties

- If the SPP is partially ordered (*i.e.*, dispute-wheel free), the algorithm will converge so that every path is assigned a reachability object and the functions are increasing.
 - How fast?
- If the SPP is not partially ordered, the algorithm will diverge.
 - Can we detect this is happening?

Outline

- Definition of Path Vector Systems
- Relationship to SPP
- Examples of Path Vector Systems
- Conditions that give unique solutions (and related results)
- **Applying the above results to policy-language design**

Local Condition for Sanity

- As long as

$$I_{(v,u)}(E_{(v,u)}(r)) > r, \quad r \neq \infty$$

for each edge, the corresponding SPP has a unique solution and is robust against failures.

- **Can we design languages that are expressive, yet enforce this condition?**

Enforcing Sanity

- We could:
 - Fix a reachability object order relation
 - Allow operators to choose import and export policy functions
 - Filter out route objects that violate the increasing rule
- But, it may be hard to make sense out of policies described in this way!

Languages Equivalent to IPVSs

Hierarchical BGP with Backup Routes

Let $R = (\mathbb{N} \times D \times \mathbb{N} \times P \times V) \cup \{\infty\}$.

Backup Level \rightarrow \mathbb{N}
 Hierarchically ordered tuples; component order: \rightarrow D
 Lexically ordered tuples; component order: \rightarrow \mathbb{N}
 Paths to the origin \rightarrow P
 Local preference \rightarrow V

$(b, d, n, p, w) \leq (b', d', m, q, u)$ then:
 $b' \leq b, d' \leq d, m \leq n, |p| \leq |q|, w \leq u$.

Let $r_o = (0, \text{self}, 0, \varepsilon, o)$.

Hierarchical Policies

- Each node specifies, for each (path, next hop) pair:
 - its local preference for it
 - its backup level
 - its hierarchy description
(*customer, provider, peer*)
- Customer, provider, peer relationship rules determine the import and export functions given the above.

Converting to an IPVS

- Because a node can specify an arbitrary integer for local preference, the path vector system that results is not automatically increasing.
- **But...** we can use our algorithm to convert it to an increasing one, basically “adjusting” preference values while keeping the implied ranking.

An Automatic IPVS

Hierarchy-Only Language

- At each node, describe each neighbor as:
 - A customer
 - A provider
 - A peer
- Use the relationship rules to generate import and export functions.
- This gives an IPVS! (Proof omitted.)

Summary of Progress

- Simple, local characterization of routing policies that result in global sanity, using the model of path vector systems.
 - Relationship with earlier models
 - Simple languages with desirable properties
- To do:
 - Define the concept “locally enforcible”
 - Can we exactly capture the class of IPVSs using a language that is locally enforcible?