Humpty Dumpty: Putting iBGP Back Together Again

Ashley Flavel[1], Jeremy McMahon[2], Aman Shaikh[3], Matthew Roughan[1], and Nigel Bean[1]

[1] School of Mathematical Sciences, University of Adelaide
{ashley.flavel,matthew.roughan,nigel.bean}@adelaide.edu.au
[2] TRC Mathematical Modelling, University of Adelaide
jeremy.mcmahon@adelaide.edu.au
[3] AT&T Labs - Research
ashaikh@research.att.com

Abstract. Humpty Dumpty is the anthropomorphic nursery-rhyme egg broken into many pieces. Similarly, we have many pieces of measurement data to represent the current iBGP state. However, unlike the nursery-rhyme where the King's men couldn't put Humpty together again, we present a systematic approach to putting all the pieces of measured iBGP data together to obtain a more complete picture of a network's routing state.

Our technique determines the decisions made by all routers in a network. It is efficient, has no assumptions about router configuration and is accurate. We present a case-study of a large Tier-2 ISP, finding for those routers with adequate measurement infrastructure, we consistently find the egress location for 99.9999% of $(router, prefix)$ pairs. Further, for the 85% of routers without measurement infrastructure we predict their decisions. This technique has been successfully applied in a 'what-if' scenario and has future applications in the real-time analysis of routing decisions.

Keywords: iBGP, Route prediction.

1 Introduction

Measurement plays a crucial role in management of IP networks since it allows operators to determine how the network is currently operating. The measurement data can be used for tasks such as deriving traffic demands in operational networks [1], finding traffic matrices [2] and their dynamics [3], and oscillation detection [4]. A majority of such tasks require some knowledge of the path traffic takes through a network — hence the need for routing measurements. However, due to high storage requirements, operational setup costs and dependency between routes selected across routers, BGP monitors collecting BGP routing information are often only connected to a subset of routers. In this paper we provide a methodology to make use of the high dependency between router decisions to systematically "fill in the gaps" left by partial measurements.

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 52–65, 2009.
© IFIP International Federation for Information Processing 2009

An Autonomous System's (AS) BGP routing decisions are not atomic. When multiple routes are available to a destination, individual routers within the AS can make different decisions as to their selected route based on their own perspective of the 'best' route. The *network solution*, that is, the decision of all routers in the network for a particular destination, is dependent on the *subset* of AS-wide routes which are learned at each individual router. Hence, it is not valid to assume *all* AS-wide routes are learned at each router for selection [5]. The iBGP configuration employed, such as full mesh or route-reflection [6], determines whether all routes or a subset of routes are available at every router. In this paper we focus on the route reflector iBGP configuration since it is used widely in large enterprise and service provider networks.

In [4], we introduced a model to analyze the oscillatory properties of a two-tier route-reflector (RR) iBGP topology. We now extend this model to determine the network solution of a general RR iBGP topology. The model captures the reliance of a router on other routers for choosing its best route. This model underpins a methodology for determining routes selected by *all* routers based on the knowledge of routes selected by a subset of routers and the iBGP configuration. Another benefit of our methodology is that it can also be used for 'what-if' analysis. Compared to the methodology proposed by Feamster and Rexford [7], which provides similar functionality, our methodology is applicable to *any* RR iBGP configuration, not just configurations satisfying recommendations of Griffin [8]. Further, our approach is topology independent making it extensible to topologies other than a RR iBGP configuration.

We applied our methodology to the topology of a large Tier-2 AS and using measurements collected from 15% routers (mostly RRs), we could determine routes for all the routers in the network. Of over 12.7 million routing decisions, we predicted a decision consistent with the observed data for all but seven routers. In the process, we also detected several configuration and data collection issues on routers when routes predicted by our methodology were inconsistent with the measurement data — highlighting an additional benefit of our analysis.

2 Background and Related Work

The Internet is comprised of a collection of Autonomous Systems (ASes) which co-operate to ensure connectivity between any two hosts. BGP is the protocol used to disseminate reachability information between ASes. A given AS has several routers at its border that connect with other ASes. These routers use BGP to learn routes to external prefixes from other ASes. These routes are then propagated to other routers inside the AS using iBGP (internal BGP). Every router selects a 'best' route from all the routes learned for every prefix. The selected route is then sent to other routers. For route dissemination (via iBGP), routers inside an AS need to form a full mesh of sessions. However, in large networks, a RR hierarchy is used to mitigate scalability issues of a full mesh. One consequence of RR hierarchy is that the set of routes available at each router is usually a subset of all routes learned across AS [5].

Prior work has recommended guidelines for designing iBGP configurations [9, 10, 11], proposed alterations to iBGP to disseminate more information AS-wide [12,13] and proposed centralized AS-wide route selection [14]. Our approach is orthogonal to these. Our aim is very pragmatic — to understand the current operation of a network — irrespective of whether it satisfies certain guidelines or not.

One approach to discovering the network solution is to simulate BGP by propagating information in an arbitrary order between routers until no router alters its decision (for example, C-BGP [15]). However using this approach, a significant number of intermediate router states are evaluated prior to converging to an arbitrary final solution (there may be multiple feasible solutions [4]). Consequently, determining if a network is in the convergence process or if it is persistently oscillating is difficult. In contrast, we avoid many intermediate states to quickly find a valid solution and more importantly, converge to a solution consistent with observed data. This enables our approach to predict the route traffic actually takes in the network. In addition, if the configuration has an oscillatory state, we can quickly identify it and pinpoint the responsible routers.

The most closely related work to ours is by Feamster and Rexford [7]. Their motivation was to predict the network solution *as designed*. That is, they assume recommended guidelines for network configuration are satisfied resulting in a unique network solution. We make no such assumptions allowing the network to be analyzed as it is currently operating — whether it satisfies guidelines or not. In addition, they assume complete visibility of input routes. In contrast, our technique works with even limited knowledge of input routes from the network. Further, our technique is designed to use observed data to influence which of multiple network solutions is actually chosen by the network. Finally, it can efficiently analyze the impact of small changes to the network without a significant re-analysis.

The primary assumption of Feamster and Rexford requires all RRs to prefer a client (a directly connected router in a lower level of the RR hierarchy) learned route over any other. This constraint is a *sufficient* condition to prevent persistent oscillation and guarantees a unique solution [8]. However as the condition is not *necessary*, it can be overly restrictive and not satisfied in practice [10, 4]. Removing this assumption removes the benefit of always converging to a unique solution — the timing of BGP updates can determine which of multiple solutions is settled upon. In addition, the tie-breaking option employed in operational networks, such as the one we examined, can be non-deterministic, resulting in an even greater number of feasible network solutions. Our technique always converges to a valid solution and in almost all instances, converges to a solution consistent with the observed data.

3 Two-Level Route-Reflector Reliance Graph

We first introduced, in [4], the concept of *reliance* between router decisions to determine if a network configuration was oscillatory. In this paper, we use reliances to efficiently and accurately determine the *actual* routes selected by

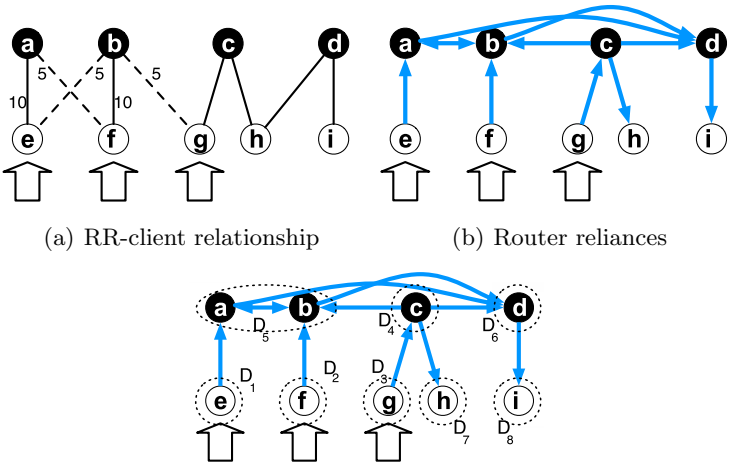
any router. We say a router u is reliant on another router v if it can learn of its best route for a particular prefix (after convergence) from v. We denote this reliance as $u \leftsquigarrow v$. Reliances are represented by a directed edge in the direction of information flow in the reliance graph. Routing information can only flow over iBGP sessions between routers. Consequently, the reliance graph is a sub-graph of the iBGP signaling graph. The rules governing route-propagation in an iBGP topology determine which links are pruned from the signaling graph to form the reliance graph. Note that a reliance graph is a directed graph and we term strongly connected components *co-reliance groups*[1].

By the construction of the reliance graph, the only location a router can learn of its best route is from an inbound edge from a neighboring router in the reliance graph. Consequently, if all neighbor's decisions are static, then the router's decision is also static (as the set of available routes remains constant). Hence, if there are only singleton co-reliance groups in the reliance graph, then a topological sort of all routers will result in an ordering where all routers' decisions must only be evaluated once. However, if there are non-singleton co-reliance groups, then we cannot topologically sort the routers (as there is at least one loop in the reliance graph). We can still topologically sort the co-reliance groups. By evaluating each co-reliance group in a topological ordering, we can ensure we do not need to re-visit a co-reliance group, but we may need to visit routers *within* a non-singleton co-reliance group multiple times.

Consider Fig. 1(a) as an example. In this example we have not shown the full-mesh iBGP sessions between RRs. When a RR is closer to a non-client egress router[2] the IGP distances are shown. Routers e, f and g learn routes from outside the AS. We create reliances when a router can learn of its best route from another (see Fig. 1(b)). For example, a is reliant on e as e propagates its externally learned route to a. However a is also reliant on b as b can inform a of the route which it learns from f. Similarly, b is reliant on a. As c is closer to its own client (g) than any non-client, it will never select any route learned via another RR. Router d will select the best route from routes learned from other RRs. Clients h and i will only ever select a route which they learn from their parent RRs. In Fig. 1(c) we highlight all strongly connected components in the reliance graph. Evaluating router decisions in any topological ordering of co-reliance groups (for example the numerical ordering $D_1, D_2, ...$) will result in a valid solution. Notice there are two valid solutions to this example based on the order of evaluation of routers in D_5. If we evaluate a's decision first, both routers in D_5 will select the egress e. Conversely if we evaluate b's decision first, both routers will select egress f.

The reliance rules in the three-level hierarchy are somewhat more complicated than the two-level case examined in [4]. We first present a brief recap of the notation used in [4] before outlining generalized reliance rules and showing an example of how they apply. Similar techniques can be applied to any iBGP topology if the topology can be abstracted to a reliance graph.

[1] For every u and v in a strongly connected component, there is a path from u to v.
[2] An egress router is a router which learns a route directly from a neighboring AS.



(a) RR-client relationship (b) Router reliances

(c) Co-reliance groups

Fig. 1. Example two level RR topology. Solid nodes are RRs and transparent nodes are clients. IGP distances shown when non-client router is closer than client router, thus for example a prefers f over e when possible. Large arrows indicate a route is learned from a neighboring AS.

4 General Route-Reflector Reliance Graph

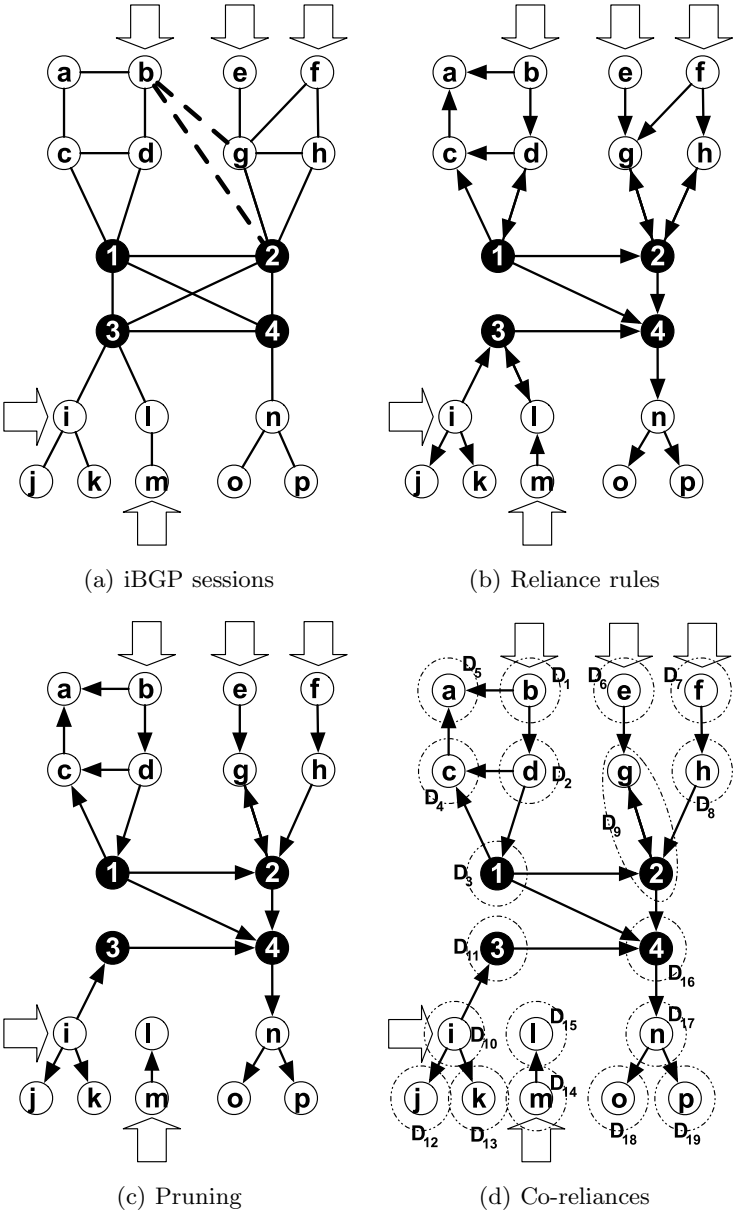
The rules we examined in [4] were only applicable to the two-level RR hierarchy. We now generalize these rules to a multi-level hierarchy. The rules governing reliance in the multi-level hierarchy are significantly more complex than the two-level case as RRs can hide information propagated to their parents (see [16] for details). Consequently, we use an example topology shown in Fig. 2(a) as we describe the rules to assist the reader in following the concepts. In this figure, routers $1, 2, 3$ and 4 form the central core mesh of RRs, while c, d, g, h, i, l and n form the middle level RRs. The solid lines represent iBGP sessions and the dashed lines represent a router's preference for a non-downstream egress. Where no dashed line exists, a downstream egress is preferred. We explicitly define several vital preferences in the caption of the figure [3]. Routes learned directly from neighbor ASes are denoted by large arrows, i.e., routers b, e, f, i and m are the egress routers.

4.1 Notation Recap

We now present a brief recap of the important notation used for the remainder of this chapter. For a thorough description, we refer the reader to [4].

An iBGP configuration C is a pair $C = (\mathcal{G}_\mathcal{P}, \mathcal{G}_\mathcal{S})$ where $\mathcal{G}_\mathcal{P}$ is the physical graph on which the IGP is run to determine the shortest path between two

[3] The ranking function λ is defined later in Section 4.1.



(a) iBGP sessions (b) Reliance rules

(c) Pruning (d) Co-reliances

Fig. 2. An example 3-level RR topology. Black nodes represent RRs at the top level. Arrowed lines represent a reliance. We explicitly define the following preferences: $\lambda_g(b) > \lambda_g(e) > \lambda_g(f), \lambda_2(b) > \lambda_2(e) > \lambda_2(f), \lambda_3(i) > \lambda_3(m), \lambda_4(m) > \lambda_4(i) > \lambda_4(b) > \lambda_4(e) > \lambda_4(f)$.

routers. The iBGP signaling graph $\mathcal{G}_S = (V, A_S)$ is overlaid on top of the physical graph with routers V connected by directed arcs in A_S.

Three types of arcs exists in A_S. An arc $(u, v) \in$ **down** represents an arc from a RR u to one of its clients v. An arc $(u, v) \in$ **up** if and only if $(v, u) \in$ **down**. Arcs in **up** are acyclic — consistent with a hierarchy rather than an arbitrary network design. An arc $(u, v) \in$ **over** represents a vanilla iBGP session from router u to v. If $(u, v) \in$ **over** then $(v, u) \in$ **over**.

A valid signaling path S satisfies the following property. The path S can be split into sub paths $S = PQR$ where $P = p_1 p_2 ... p_a$ for some $a \geq 0$ such that each $p_i \in$ **up**, $R = r_1 r_2 ... r_b$ for some $b \geq 0$ such that each $r_i \in$ **down** and Q consists of at most a single arc $q \in$ **over**. Note that P, Q or R may be empty.

An *egress instance* [8] $I = (C, X)$ corresponds to a pair of a configuration C and a set of egress routers X. The set X consists of all egress routers that learn an external BGP route to a particular prefix which are not eliminated by the BGP decision process (up-to the IGP distance step) when compared with all AS-wide routes [7]. In our example of Fig. 2(a), b, e, f, i and m form X. An *egress ancestor set* E can be recursively defined as the set of egress routers X and all parents of routers in E. In our example, $E = \{b, d, e, f, g, h, i, m, k, l, 1, 2, 3\}$. Note that although an egress router may learn multiple routes (to a prefix) it will only advertise its best route to neighbors. Hence, there is a one-to-one mapping from egress routers to available routes. Therefore, we will refer to an egress router and its available route interchangeably.

The BGP decision process is denoted by a ranking function λ_u for a router u such that if a route a_k is preferred over a route a_j at router u, then $\lambda_u(a_k) > \lambda_u(a_j)$. If two routes a_k and a_j are equivalent up-to the tie-break option and the actual route chosen is dependent on message timing, then $\lambda_u(a_k) = \lambda_u(a_j)$. For convenience, we denote the preference of the null route ϕ as $\lambda_u(\phi) = -\infty$.

4.2 Reliance Rules for Route Reflection

In this section we generalize the reliance rules we previously defined for the two-level RR hierarchy [4] to an arbitrary hierarchy. Although there is a strict set of reliances which are a subset of arcs (of type **up, down** and **over**) in the signaling graph A_S, defining where a router can learn of its best route in an n-level hierarchy is more difficult than in the two-level case. An important consideration is that failing to define reliances can result in *incorrect* decisions, while defining additional reliances simply increases the computational complexity of predicting selected routes (as it may create larger co-reliance groups than really exist). Consequently, we start with a relatively conservative definition of reliances before pruning many of those which cannot exist. We assume the MED attribute is filtered or compared AS-wide in this section.

Downstream Egress Set. Let us generalize the best downstream egress function defined in [4] to return a *set* of *downstream* egresses $\Lambda(u)$ for a router u. If u has no downstream egresses, $\Lambda(u) = \phi$. Unlike the two-level hierarchy, in an arbitrary hierarchy, it is no longer guaranteed that a router will learn of all

Table 1. Downstream egress sets for routers in example topology of Fig. 2

Router(u)	1	2	3	d	g	h	l	all other routers
$\Lambda_1(u)$	ϕ	ϕ	i	b	e	f	m	ϕ
$\Lambda_n(u)$	b	e, f	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

downstream egresses since the set of available routes is restricted by the selection of intermediate routers.

We first define $\Lambda_1(u)$ as the set of best downstream egresses which are one downstream iBGP hop away from u. Router u is guaranteed to learn of these routes due to the direct iBGP session and so these routes will always be available. Formally, for $u \notin X$,

$$\Lambda_1(u) = \left\{ v \in X : (u,v) \in \textbf{down} \text{ and } \lambda_u(v) = \max_{w \in X : (u,w) \in \textbf{down}} \lambda_u(w) \right\}$$

We show for our example in Fig. 2 the sets $\Lambda_1(u)$ for all routers in Table 1. Notice that as $\lambda_g(e) > \lambda_g(f)$, $\Lambda_1(g) = e$.

Now, let us consider other egresses u could learn (and select as best) from clients which are not direct egresses, i.e., those more than one hop away. We denote this set by $\Lambda_n(u)$ and define it for $u \notin X$ as,

$$\Lambda_n(u) = \bigcup_{w \in E \setminus X : (u,w) \in \textbf{down}} \left\{ v \in \Lambda(w) \setminus \Lambda_1(u) : \lambda_u(v) \geq \max_{r \in \Lambda_1(u)} \lambda_u(r) \right\}.$$

Note that egresses not preferred over an "always available egress" are not in $\Lambda_n(u)$. We also show in Table 1 for our example in Fig. 2 the sets of $\Lambda_n(u)$ for all routers. As router 2 can learn of both e and f from its children (g and h), both egresses are in $\Lambda_n(2)$. Also notice that as i is an always available downstream egress of 3 and i is preferred over m, m is *not* in $\Lambda_n(3)$. Finally, we define $\Lambda(u) = \Lambda_1(u) \cup \Lambda_n(u)$. Note, $\Lambda(u)$ is well defined as we define $\Lambda(u)$ recursively up the hierarchy.

Rules for Reliance. Reliance rules are adapted from the route propagation rules [6] and indicate where a router can learn of its best route. Arcs in the reliance graph are a subset of the arcs in the signaling graph A_S. There are three types of arcs in A_S which may be part of the reliance graph. Consider the arc $(u,v) \in A_S$:

1. $(u,v) \in$ **down:** a RR u is reliant on its child v iff $u \notin X$ and $v \in E$.
2. $(u,v) \in$ **up:** a client u is reliant on its parent v iff $u \notin X$.
3. $(u,v) \in$ **over:** a router $u \notin X$ is reliant
   (a) on another router $v \in E \setminus X$ iff

$$\min_{r \in \Lambda(u)} \lambda_u(r) \leq \max_{s \in \Lambda(v) \setminus \Lambda_1(u)} \lambda_u(s)$$

(b) on another router $v \in X$ iff

$$\min_{r \in \Lambda(u)} \lambda_u(r) \leq \lambda_u(v)$$

We demonstrate the above rules for our example in Fig. 2(b). Applying rule 1, we see any router in E which does not have a direct egress is reliant on its children, for example, 2 is reliant on h. Applying rule 2, all client routers are reliant on their parents (unless they are a direct egress), for example l is reliant on 3. Rule 3 applies when a router can learn of a better route via an **over** edge than any client-learned route. For instance, rule 3(a) applies to the reliance of 2 on 1, as 2 will select the route from b if it ever learns of it, whereas rule 3(b) applies for the reliance of a on b, as a can learn the egress directly from b, via an **over** edge.

Pruning Reliances. Our technique for determining router decisions would work on the reliance graph defined by the rules above. However, ideally we would like to have the smallest possible co-reliance groups in the reliance graph to minimize computation. The reliance rules are essentially a pruning of the signaling graph. We can continue in this vein by pruning even more reliances:

1) $u \nleftrightsquigarrow v$ if $(u, v) \in$ **down** and $v \in X \backslash \Lambda_1(u)$.

That is, a RR is reliant only on its best client with a direct egress. In our example, as g prefers e over f, and as e is always available, f will never be chosen and is pruned in Fig. 2(c).

2) $u \nleftrightsquigarrow v$ if $(u, v) \in$ **down**, $v \in E \backslash X$ and

$$\min_{r \in \Lambda_1(u)} \lambda_u(r) > \max_{s \in \Lambda(v)} \lambda_u(s)$$

That is, if a RR u has a client r with a direct egress and no possible egress which it can learn from another client v is better than r, then u cannot be reliant on v. In our example, as 3 prefers i over m, 3 is not reliant on l and this edge is also pruned in Fig. 2(c).

Our next rule is for **up** edges. Before specifying the rule for a $(u, v) \in$ **up**, we define $L(u, v)$ as the egresses that can be learned by a router u from the parent router v which are not available from a direct client. For a general hierarchy, the exact form of $L(u, v)$ can be quite complicated. In practice, RR hierarchies do not tend to be larger than three-levels, and for three levels, we can formally define

$$L(u, v) = \bigcup_{w \in E : v \rightsquigarrow w} \Lambda(w) \backslash \Lambda_1(u).$$

3) $u \nleftrightsquigarrow v$ if $(u, v) \in$ **up** and

$$\min_{r \in \Lambda_1(u)} \lambda_u(r) > \max_{s \in L(u,v)} \lambda_u(s)$$

That is, a RR in the second level of the hierarchy is only reliant on its parent if the parent can learn a better route than any of the always available egress

at the RR. In our example, l prefers m over any egress it can learn from 3 (as $\lambda_l(m) > \lambda_l(i)$). Hence the reliance of l on 3 is pruned. Other reliances which are pruned using this technique are $h \leftsquigarrow 2$ and $d \leftsquigarrow 1$. Notice $g \leftsquigarrow 2$ is not prunable as g may select an egress learned from 2.

Finding a Valid Solution. As the reliance graph precisely identifies which routers' decisions a particular router is dependent upon, if there are no cycles in the reliance graph structure, we can topologically sort the routers and evaluate them in-order (visiting them exactly once). However, as shown in our example, it is likely there are cycles in the reliance graph. Hence, we partition the reliance graph into co-reliance groups before undertaking a topological sort on co-reliance groups. We show these co-reliance groups in Fig. 2(d). One topological ordering (any topological order will result in the same network solution) is the numerical ordering of $D_1 - D_{19}$ in Fig. 2(d). If multiple routers are present in a co-reliance group (such as D_9) the decisions of the routers may be dependent on message timing and we evaluate their decisions until a *valid* solution is found. The ordering in which we evaluate the routers within a co-reliance group determines which of possibly multiple valid solutions we converge upon [4]. Our desire is to converge to the *actual* solution selected by the network. We describe how we achieve this in the next section.

5 Finding the Actual Solution

A walk of the reliance graph can have multiple valid solutions when either

1. co-reliance groups have multiple routers; or
2. the non-deterministic oldest-route tie-breaker is used.

When such conditions exist, we want to find the *actual* solution chosen by routers. If decisions of some routers in the network are known (through measurement), we can use them as constraints while determining the solution[4]. Two feasible approaches to using these constraints are: (i) find all possible solutions and select one which satisfies the constraints; or (ii) gravitate towards a solution satisfying all constraints by ensuring that when we visit each co-reliance group, we select a solution consistent with the constraints. We take the latter approach as it reduces unnecessary computation of infeasible solutions. However, it can result in discrepancies if we reach a co-reliance group and there are no solutions satisfying the constraints. We could backtrack along the reliance graph to resolve discrepancies, however in our examined network, we found only seven in over 12.7 million decisions had such discrepancies and so we did not implement a backtracking algorithm.

The ordering of router evaluation within a co-reliance group determines which of multiple valid solutions we converge to. Due to space constraints, we refer the reader to [16] for full details of ordering of routers within a co-reliance group.

[4] There may be multiple feasible solutions which match the known route selections.

Once we have an ordering for co-reliance groups and an ordering for routers within a co-reliance group we can calculate the decisions of all routers by simply walking the reliance graph. We visit each co-reliance group in-order, and visit each router within the co-reliance group in-order. If the co-reliance group is non-singleton, we continue evaluating the router decisions until no routers alter their decision. Our algorithm does not rely on the underlying topology, only its description in terms of a reliance graph. Consequently, any topology describable by a reliance graph can be analyzed using this algorithm.

It is possible that a co-reliance group never converges to a solution [4]. However, if this is the case, then the actual network also has oscillatory properties. The non-convergent co-reliance group isolates the routers responsible for oscillatory modes so administrators can take corrective action.

When the oldest-route tie-breaker is used, there may be multiple routes available at a router with equal IGP distances to an egress. Any route with this equally good IGP distance may be chosen by the router. We again use any available constraints to assist our decision (see [16] for details).

6 Evaluation

We have implemented our techniques and tested them on a large Tier-2 AS. The AS has a three-level RR hierarchy and uses the oldest-route tie-break option. The MED attribute is reset by the AS. A BGP monitoring infrastructure collects BGP updates from 15% of routers, majority of which are RRs. We use the known routes from these routers as the set of input routes to the network. Each such route contains a "next-hop" attribute which corresponds to the egress router for the route. IGP distances are determined based on data collected by an OSPF monitor [17].

Our algorithm discovers the decisions made by routers once the network has converged to a solution. Consequently, we only examine *stable* prefixes – those prefixes with no updates witnessed from any router under observation in the 6 hours prior and 6 hours past the examined time. Our evaluation is based on data collected on the 26th May 2008, although we found similar results for several other examined intervals. During the analysis process, our model discovered several minor configuration errors. In this case, our model predicted the "correct" outcome, although the network selected an "incorrect" outcome due to a configuration error on several egress routers. We exclude the prefixes affected by these configuration errors from our analysis.

To speed up our analysis, we group all prefixes with the same set of egress routers and the same egress selection by all routers that have session to the BGP monitor. This allowed us to discover the egress router selected by all routers (including the remaining 85% of routers without BGP monitors) for 224,870 stable prefixes with only 827 reliance graphs and 1154 "walks" of the reliance graph.

As our technique is based on the rules of route propagation, it will *always* find a valid solution given any configuration. With the addition of monitor information (or any other constraints available), we can converge to a solution

satisfying such constraints. In practice, we found our technique always found valid solutions and only seven inconsistencies with BGP monitor data in over 12.7 million known $(prefix, router)$ pairs. This discrepancy was resulting from a tie-break decision at a router without a BGP monitor session. The predicted egresses were in the same PoPs as the actual selected egress. Backtracking to alter the random tie-break decision would correct this.

We found 99.99% of co-reliance groups were singleton and the maximum size of a co-reliance group was five routers which occurred only four times ensuring our technique very rarely required the re-evaluation of router decisions.

The execution time of this case study lasted several hours. However, the evaluation time was dominated by the conversion of the enormous amounts of compressed binary BGP data on disk to an ASCII readable format in memory. The construction and walk of the reliance graph did not significantly contribute to the execution time. Consequently, the incorporation of incremental changes to BGP and IGP may allow near real-time analysis of router decisions. We leave the incorporation of such routing dynamics to future work.

7 General Router Reliance Graph

In previous sections, we focused on the reliance rules for an iBGP route-reflector topology. However, our technique is applicable to *any topology* describable by a series of reliances. That is, if a set of rules defining where a router can possibly learn of its best route can be defined, our technique will find a solution. This is in contrast to [7] where different iBGP topologies required separate algorithms. Further, for the iBGP topology examined in this paper the conditions required by Feamster-Rexford technique are not satisfied. Our technique, in the best case can precisely order the execution of router decisions (when all co-reliance groups are singular) and in the worst case (all routers in a general topology form a single co-reliance group) reverts to an arbitrary ordering of router decisions (such as in [15]). The addition of reliance rules increases the efficiency of our technique.

An example of a different iBGP topology is an RR topology with the MED attribute respected. The authors of [7] recommended a simulator as the best technique to evaluate the network solution to this topology. However, in [16] we detail reliance rules for such a topology, finding that although the number of edges in the reliance graph can increase (in comparison to when the MED attribute is not respected), routers outside the egress ancestor set do not form non-singleton co-reliance groups. Hence, the maximum co-reliance group size is bounded by the size of the egress ancestor set which is commonly an order of magnitude smaller than the total number of routers — making our technique significantly more efficient than a pure simulator. Consequently, the ordering of router decisions outlined in this paper may also be used to improve the convergence times of existing BGP simulators such as C-BGP [15].

The separation of the topology and the rules for route propagation from the algorithm used to evaluate the network solution has possible applications not only in iBGP described in this paper, but also in the eBGP context. A topology

inferred by a technique such as [18] could form a starting point to predict the (Internet-wide) solution for a particular prefix and may help to answer Internet-wide 'what-if' questions.

8 Conclusions

In this paper, we presented a reliance graph model to capture the dependence amongst routers for route selection. The input to the model is the iBGP topology and IGP distances. The model allows one to efficiently calculate the network solution (set of routes selected by all routers) with no assumptions on the iBGP configuration. The model also works when only partial information about routes is available. We demonstrated the efficacy of the model by applying it to a Tier-2 containing over $220,000$ prefixes. Our methodology was able to find a valid solution and a solution consistent with observed routes for all but seven $(prefix, router)$ pairs even when routes from only about 15% routers were known.

One significant benefit of using a reliance graph model is that dynamics of iBGP topology or IGP distance that do not affect the reliance graph does not have any effect on the actual routing choices. Furthermore, BGP route dynamics only require the re-evaluation of routers in the portion of the network so affected. We believe that these two features should allow our methodology to work in real-time for filling gaps to BGP monitors as well as for 'what-if' analyses. In fact, we have applied the methodology successfully to determine the current router decisions and predict changes under modified route availability [16].

BGP monitors are used to determine the route selected by routers in the network *i.e.*, the selected network solution. Our reliance graph analysis identifies where routes can be learned from. Consequently, a new direction of research could be to identify the optimal placement of BGP monitors to minimize the number of random tie-break decisions while maximizing the information about the available egresses.

References

1. Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., True, F.: Deriving Traffic Demands for Operational IP Networks: Methodology and Experience. IEEE/ACM Transactions on Networking (2001)
2. Zhang, Y., Roughan, M., Lund, C., Donoho, D.: Estimating Point-to-Point and Point-to-Multipoint Traffic Matrices: An Information-Theoretic Approach. IEEE/ACM Transactions on Networking 13(5), 947–960 (2005)
3. Teixeira, R., Shaikh, A., Griffin, T.G., Voelker, G.M.: Network Sensitivity to Hot-Potato Disruptions. In: ACM SIGCOMM (2004)
4. Flavel, A., Roughan, M., Bean, N., Shaikh, A.: Where's Waldo? Practical Searches for Stability in iBGP. In: IEEE International Conference on Network Protocols (2008)
5. Buob, M., Meulle, M., Uhlig, S.: Checking for Optimal Egress Points in iBGP Routing. In: International Workshop on the Design of Reliable Communications Networks (2007)

6. Bates, T., Chandra, R., Chen, E.: BGP Route Reflection - An Alternative to Full Mesh IBGP, RFC 2796 (2000)
7. Feamster, N., Rexford, J.: Network-Wide Prediction of BGP Routes. IEEE/ACM Transactions on Networking 15(2), 253–266 (2007)
8. Griffin, T., Wilfong, G.: On the Correctness of IBGP Configuration. In: ACM SIGCOMM (2002)
9. Feamster, N., Balakrishnan, H.: Correctness Properties for Internet Routing. In: Forty-third Allerton Conference on Communication, Control, and Computing (2005)
10. Vutukuru, M., Valiant, P., Kopparty, S., Balakrishnan, H.: How to Construct a Correct and Scalable iBGP Configuration. In: IEEE INFOCOM, Barcelona, Spain (April 2006)
11. Buob, M., Uhlig, S., Meulle, M.: Designing Optimal iBGP Route-Reflection Topologies. In: IFIP Networking (2008)
12. Bonaventure, O., Uhlig, S., Quoitin, B.: The Case for More Versatile BGP Route Reflectors, Work in progress, draft-bonaventure-bgp-route-reflectors-00.txt (2004)
13. Poduri, K., Alaettinoglu, C., Jacobson, V.: BST - BGP Scalable Transport. In: NANOG 27 (2003)
14. Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A., van der Merwe, J.: Design and Implementation of a Routing Control Platform. In: Symposium on Networked Systems Design and Implementation (2005)
15. Quoitin, B., Uhlig, S.: Modeling the Routing of an Autonomous System with CBGP. IEEE Network Magazine, Special Issue on Interdomain Routing (2005)
16. Flavel, A.: BGP, Not As Easy As 1-2-3. Ph.D thesis, University of Adelaide (2009)
17. Shaikh, A., Greenberg, A.: OSPF Monitoring: Architecture, Design and Deployment Experience. In: Symposium on Networked Systems Design and Implementation (2004)
18. Mühlbauer, W., Maennel, O., Uhlig, S., Feldmann, A., Roughan, M.: Building an AS-Topology Model that Captures Route Diversity. In: ACM SIGCOMM (2006)