Prediction of BGP Routes within an ISP

Ashley Flavel¹, Jeremy McMahon², Aman Shaikh³, Matthew Roughan⁴ and Nigel Bean⁴

¹ AT&T Labs
ashley.flavel@gmail.com

² TRC Mathematical Modelling, University of Adelaide
jeremy.mcmahon@adelaide.edu.au

³ AT&T Labs - Research
ashaikh@research.att.com

⁴ School of Mathematical Sciences, University of Adelaide
{matthew.roughan, nigel.bean}@adelaide.edu.au

Abstract. Internet Service Providers (ISPs) often collect routing data to troubleshoot, analyze and predict the behavior of their network. However, data collected from multiple interacting routing protocols is often incomplete and difficult to manually analyze. In this paper we present a systematic approach to combine the pieces of measured routing data to obtain a more complete picture of a network’s routing state. Our technique is efficient, has no assumptions about router configuration and is accurate. We present a case-study of a large Tier-2 ISP, finding that 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.

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 iBGP topology. We now extend this model to determine the network solution of a general route-reflector 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’ analyses. Compared to the methodology proposed by Feamster and Rexford [8], which provides similar functionality, our methodology is applicable to any route-reflector iBGP configuration, not just configurations satisfying the recommendations of Griffin and Wilfong [9]. Further, our approach is topology independent making it extensible to topologies other than a route-reflector iBGP configuration.

We applied our methodology to the topology of a large Tier-2 AS and using measurements collected from 15% routers (mostly route-reflectors), 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 router decisions. 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.

Our technique consists of two major components. The first component converts a network topology to a reliance graph. Although we primarily focus (in Sections 3 and 4) on the route-reflector topology due to its common use in many ASes, in Section 7 we also consider the conversion of several other iBGP topologies. The second component uses the resulting reliance graph to determine the routing decisions of all routers within the network — independent of the actual topology type. We outline this component in Sections 5 before evaluating our technique on a large Tier-2 ISP in Section 6. For the reader’s benefit we provide a brief background and summary of related work in Section 2.

5 A preliminary analysis was presented in [7]. This article significantly expands the algorithm description, performance evaluation and generalizes the techniques to alternative iBGP topologies.
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. The Border Gateway Protocol (BGP) [10] 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 complete route dissemination (via iBGP), routers inside an AS need to form a full mesh of sessions. However, in large networks, a route-reflector hierarchy is used to mitigate scalability issues of a full mesh. One consequence of route-reflector hierarchy is that the set of routes available at each router is usually a subset of all routes learned across the AS [5].

Prior work has recommended guidelines for designing iBGP configurations [11–13], proposed alterations to iBGP to disseminate more information AS-wide [14,15] and proposed centralized AS-wide route selection [16–18]. 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 [19]). Using this approach, however, a significant number of intermediate router states are evaluated prior to converging to an arbitrary final solution, and also there may be multiple feasible final 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 feasible 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 [8]. 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 route-reflectors to prefer a route learned from a client (a directly connected router in a lower level
of the route-reflector hierarchy) over any other. This constraint is a *sufficient* condition to prevent persistent oscillation and guarantees a unique solution [9]. However as the condition is not *necessary*, it can be overly restrictive and not satisfied in practice [4, 12]. 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, as is the case in our study, can be non-deterministic, resulting in an even greater number of feasible network solutions. Our technique always converges to a feasible solution and in almost all instances, converges to a solution consistent with the observed data.

Let us now consider two example networks, shown in Figure 1, where Griffin and Wilfong’s constraint is not satisfied.

![Figure 1](image)

**Fig. 1.** Stable egress instances violating Griffin and Wilfong’s condition. Black nodes are route-reflectors, and white nodes are client routers. Solid lines represent iBGP sessions, and dashed lines indicate IGP distances to non-client routers. IGP distances are shown next to lines connecting nodes.

For these networks, we are searching for a solution which describes the egress selected at each router. In the case of Figure 1(a), the solution is \{4, 6, 6, 4, 5, 6\}, where the \(i\)th element of the solution vector represents the selected egress of router \(i\). Notice that although router 1 is, with respect to Interior Gateway Protocol (IGP) distance, closer to the egress via router 5 than the egress via router 4, it never learns of this route as 2 does not select this egress. Router 2 selects an egress via 6 which it learns from 3 — not a client router. Hence Griffin and Wilfong’s constraint is violated, but a unique solution still exists.

Although a unique solution guarantees the configuration will not oscillate ad infinitum [9], we saw in [4] that non-uniqueness does not imply a configuration will oscillate. In Figure 1(b) there are two possible solutions depending on whether router 1 makes its decision before — or after — router 2 (the solutions are \{3, 3, 3, 4\}, \{4, 4, 3, 4\}). In the above examples, as Griffin and Wilfong’s constraint is violated, the technique in [8] may not find the solution selected by
routers or may even find an infeasible solution. For example in Figure 1(a), the technique of Feamster and Rexford [8] may return \{5, 6, 6, 4, 5, 6\} which is not valid (as 1 is selecting a route of which it has no knowledge). The pitfall of the technique in [8] is that it relies on the assumption that networks are designed to satisfy Griffin and Wilfong’s constraint. However, we demonstrated in [4] that ensuring this constraint is satisfied is difficult under all failure scenarios and with the current BGP decision process (especially in multi-level hierarchies). In contrast, we make no assumptions on the configuration of the topology making our technique applicable to any network scenario. Further, our technique is highly amenable to the inclusion of measurement data to influence which of multiple network solutions is actually chosen by the network. We present results in Section 6 of a case-study where we always predict a feasible solution. We found this solution was consistent with observed data in 99.9999% of cases.

3 Two-level Route-Reflector Reliance Graph

We first introduced the concept of reliance between router decisions, in [4], 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 \( v \). Reliances are represented by a directed edge in the direction of information flow in the reliance graph and we denote this reliance as \( u \leftrightarrow v \). Routing information can only flow over iBGP sessions between routers, and consequently the reliance graph is a sub-graph of the 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.

Co-reliance groups are strongly connected components of the reliance graph. Co-reliance groups form an acyclic structure representing the reliances between co-reliance groups. We visit each co-reliance group in a topological order, evaluating the decision of all routers in a co-reliance group before moving to the next. When only one router exists in a co-reliance group, the BGP decision process is evaluated exactly once. However, in a non-singleton co-reliance group, the decisions of some routers may be dependent on the decision of other router’s within a co-reliance group. Consequently, we may need to re-evaluate the decisions of routers within a non-singleton co-reliance group to ensure updated information does not alter a router’s decision. Importantly, a co-reliance group is never re-visited. We explicitly describe the ordering of co-reliance groups and the evaluation of router decisions within a co-reliance group in Section 5.

The reliance graphs, based on the rules outlined in [4], and their corresponding co-reliance groups for the examples in Figure 1 are shown in Figure 2.

In Figure 2(a), router 1 is reliant on 2 because if it learns of egress 5 it will select it. However, router 3 is not reliant on 1 as its closest egress is 6 — which it always learns of and hence will never select any other route. In Figure 2(b), router 1 will egress via 4 if it learns of it. Thus it is reliant on 2. Further, if router
2 learns of the egress via 3, it will select it. Hence 2 is reliant on 1 and so routers 1 and 2 form a co-reliance group. Evaluating router decisions in any topological ordering (the numerical ordering \( D_1, D_2, \ldots \) in both Figure 2(a) and Figure 2(b) are examples of possible topological orderings) will result in a feasible solution. In Figure 2(b) the order in which we evaluate the decisions of router 1 and 2 in co-reliance group \( D_3 \) determines which of two solutions is realized. For example, if we evaluate the decision of router 1 first, it will only have knowledge of the egress via router 3. Hence, it will select to egress via 3 and propagate this choice to router 2. Router 2 will also learn of the egress via router 4 and consequently will choose to egress via 3 (as it has a shorter IGP distance). Router 1 will not alter its decision as it will not learn of the egress via router 4. Hence, a network solution is \( \{3, 3, 3, 4\} \). Conversely, if we evaluate the decision of router 2 prior to router 1, an alternative network solution is \( \{4, 4, 3, 4\} \). In Section 5 we use network measurements to determine which of these alternatives is actually selected by the network.

The reliance rules in a multi-level hierarchy are somewhat more complicated than in the two-level case examined in [4]. We first present a brief recap of the notation used in [4] before outlining the new reliance rules and examining an example of their application. Recall, all reliance rules are based on where a router can learn of its best route after convergence. Similar techniques can be applied to any iBGP topology. We examine several other topologies in Section 7.

### 4 General Route-Reflector Reliance Graph

The reliance rules of [4] were only applicable to the two-level route-reflector 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 route-reflectors can hide information propagated to
their parents [20]. Therefore, to assist the reader in following concepts as we describe such rules, we use an example topology shown in Figure 3(a).

In Figure 3(a), routers 1, 2, 3 and 4 form the central core mesh of route-reflectors, while c, d, g, h, i, l and n form the middle level route-reflectors. 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, where the ranking function $\lambda$ is defined in Section 4.1. 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 Route Reflection Notation Recap

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

An iBGP configuration $C$ is a pair $C = (G_p, G_s)$ where $G_p$ is the physical graph on which the IGP is run to determine the shortest path between two routers. The iBGP signaling graph $G_s = (V, A_s)$ is overlaid on top of the physical graph with routers $V$ connected by directed edges in $A_s$.

Three types of edges exists in $A_s$. An edge $(u, v) \in$ down represents an edge from a route-reflector $u$ to one of its clients $v$. An edge $(u, v) \in$ up if and only if $(v, u) \in$ down. Edges in up are acyclic — consistent with a hierarchy rather than an arbitrary network design. An edge $(u, v) \in$ over represents an iBGP session from router $u$ to $v$. If $(u, v) \in$ over then $(v, u) \in$ over.

For a signaling path $S$ to be valid, it must be able to be split into sub paths $S = PQR$ where $P$ contains zero or more edges $p_i \in$ up, $R$ contains zero or more edges $r_i \in$ down and $Q$ is either empty or consists of a single arc $q \in$ over. Note that up to two of $P$, $Q$ or $R$ may be empty.

An egress instance [9], $I = (C, X)$, 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 [8]. In our example of Figure 3(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, $E = \{b, d, e, f, g, h, i, m, l, 1, 2, 3\}$. 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 $\lambda_u$ for a router $u$ such that if an egress via $a$ is preferred over an egress via $b$ at router $u$, then $\lambda_u(a) > \lambda_u(b)$. If two egresses via $a$ and $b$ (routes) are equivalent up-to the tie-break option and the actual route chosen is dependent on message timing, then $\lambda_u(a) = \lambda_u(b)$. For convenience, we denote the preference of the null route $\phi$ as $\lambda_u(\phi) = -\infty$. 
Fig. 3. An example 3-level route-reflector topology. Black nodes represent route-reflectors at the top level. Arrowed lines represent a reliance. We explicitly define the following preferences: $\lambda_3(b) > \lambda_3(e) > \lambda_3(f)$, $\lambda_2(b) > \lambda_2(e) > \lambda_2(f)$, $\lambda_3(i) > \lambda_3(m)$, $\lambda_4(m) > \lambda_4(i) > \lambda_4(b) > \lambda_4(e) > \lambda_4(f)$. 
4.2 Reliance Rules for Route Reflection

In this section we generalize the reliance rules defined for the two-level route-reflector hierarchy in [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, where larger co-reliance groups than necessary may be created. 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. This is the policy of the AS we examine.

**Downstream Egress Set:** Let us generalize the best downstream egress function defined in [4] to return a set of downstream egresses $A(u)$ for a router $u$. If $u$ has no downstream egresses, $A(u) = \emptyset$. Unlike the two-level hierarchy, in an arbitrary hierarchy, it is no longer guaranteed that a router will learn of all downstream egresses since the set of available routes is restricted by the selection of intermediate routers.

We first define $A_1(u)$ 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 $u \not\in X$,

$$A_1(u) = \{ v \in X : (u,v) \in \text{down} \text{ and } \lambda_u(v) = \max_{w \in X : (u,w) \in \text{down}} \lambda_u(w) \}$$

We show, in Table 1, the sets $A_1(u)$ for all routers of our example in Figure 3. Notice that as $\lambda_g(e) > \lambda_g(f)$, $A_1(g) = e$.

<table>
<thead>
<tr>
<th>Router(u)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>g</th>
<th>h</th>
<th>l</th>
<th>all other routers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1(u)$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$b$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>$A_n(u)$</td>
<td>$b$</td>
<td>$\emptyset$</td>
<td>$c$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

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 $A_n(u)$ and define it for $u \not\in X$ as,

$$A_n(u) = \bigcup_{w \in E \setminus X : (u,w) \in \text{down}} \{ v \in A(w) \setminus A_1(u) : \lambda_u(v) \geq \max_{r \in A_1(u)} \lambda_u(r) \}.$$
Note that egresses not preferred over an “always available egress” are not in \( A_n(u) \). These sets are also shown in Table 1 for our example of Figure 3. As router 2 can learn of both \( e \) and \( f \) from its children \( (g \text{ and } h) \), both egresses are in \( A_n(2) \). Also notice that as \( i \) is an always available downstream egress of 3 and \( i \) is preferred over \( m \), \( m \) is not in \( A_n(3) \). Finally, we define \( A(u) = A_1(u) \cup A_n(u) \). Note, \( A(u) \) is well defined as we define \( A(u) \) 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. Edges in the reliance graph are a subset of the edges in the signaling graph \( A_S \). There are three types of edges in \( A_S \) which may be part of the reliance graph. Consider the edge \((u,v) \in A_S:\)

1. \((u,v) \in \text{down}: a \text{ route-reflector } u \text{ is reliant on its child } v \text{ iff } u \not\in X \text{ and } v \in E.\)
2. \((u,v) \in \text{up}: a \text{ client } u \text{ is reliant on its parent } v \text{ iff } u \not\in X.\)
3. \((u,v) \in \text{over}: a \text{ router } u \not\in X \text{ is reliant}
   \begin{align*}
   \text{(a) on another router } v \in E \setminus X \text{ iff } \\
   \min_{r \in A(u)} \lambda_u(r) \leq \max_{s \in A(v) \setminus A_1(u)} \lambda_u(s)
   \end{align*}
   \begin{align*}
   \text{(b) on another router } v \in X \text{ iff } \\
   \min_{r \in A(u)} \lambda_u(r) \leq \lambda_u(v)
   \end{align*}

We demonstrate the above rules for our example in Figure 3(b). A summary of all the reliances for this topology is included in the first two columns of Table 2.

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) \( u \leftrightarrow v \text{ if } (u,v) \in \text{down and } v \in X \setminus A_1(u). \)

That is, a route-reflector 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 Figure 3(c).
Table 2. Reliances for example topology of Figure 3 and the rule which identifies them. Also included is whether they are prunable and non-trivial reasons why or why not prunable.

<table>
<thead>
<tr>
<th>Reliance</th>
<th>Rule</th>
<th>Prunable?</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \sim d$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$2 \sim 1$</td>
<td>over(a)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$2 \sim g$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$2 \sim h$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$3 \sim i$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$3 \sim l$</td>
<td>down</td>
<td>Yes (2)</td>
<td>$(3, l) \in \text{down}$ and $l \in E \setminus X$ and $i \in A_1(3)$, $A(l) = {m}$ and $\lambda_3(m) &lt; \lambda_3(i)$</td>
</tr>
<tr>
<td>$4 \sim 1$</td>
<td>over(a)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$4 \sim 2$</td>
<td>over(a)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$4 \sim 3$</td>
<td>over(a)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$a \sim c$</td>
<td>up</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$a \sim b$</td>
<td>over(b)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$c \sim d$</td>
<td>over(a)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$c \sim 1$</td>
<td>up</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$d \sim b$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$d \sim 1$</td>
<td>up</td>
<td>Yes (3)</td>
<td>$\lambda_d(b) &gt; \lambda_d(*)$.</td>
</tr>
<tr>
<td>$g \sim c$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$g \sim f$</td>
<td>down</td>
<td>Yes (1)</td>
<td>$(g, f) \in \text{down}$ and $f \in X$ and $f \notin A_1(g)$</td>
</tr>
<tr>
<td>$g \sim 2$</td>
<td>up</td>
<td>No</td>
<td>$\lambda_2(e) &lt; \lambda_3(b)$ and $\lambda_2(e) &lt; \lambda_2(b)$</td>
</tr>
<tr>
<td>$h \sim f$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$h \sim 2$</td>
<td>up</td>
<td>Yes (3)</td>
<td>$\lambda_h(f) &gt; \lambda_h(*)$.</td>
</tr>
<tr>
<td>$j \sim i$</td>
<td>up</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$k \sim i$</td>
<td>up</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$l \sim m$</td>
<td>down</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$l \sim 3$</td>
<td>up</td>
<td>Yes (3)</td>
<td>$\lambda_l(m) &gt; \lambda_l(*)$</td>
</tr>
<tr>
<td>$n \sim 4$</td>
<td>up</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$o \sim n$</td>
<td>up</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>$p \sim n$</td>
<td>up</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
2) \( u \leftrightarrow v \) if \((u, v) \in \textbf{down}, v \in E \setminus X \) and
\[
\min_{r \in A_1(u)} \lambda_u(r) > \max_{s \in A_1(v)} \lambda_u(s)
\]
That is, if a route-reflector \( u \) 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 Figure 3(c).

Our next rule is for \( \textbf{up} \) edges. Before specifying the rule for \((u, v) \in \textbf{up}, \) 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, route-reflector hierarchies do not tend to be larger than three-levels, and for three levels, we can formally define
\[
L(u,v) = \bigcup_{w \in E : v \rightarrow w} A(w) \setminus A_1(u).
\]
3) \( u \leftrightarrow v \) if \((u, v) \in \textbf{up} \) and
\[
\min_{r \in A_1(u)} \lambda_u(r) > \max_{s \in L(u,v)} \lambda_u(s)
\]
That is, a route-reflector 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 egresses at the route-reflector. In our example, \( m \) is an always available and downstream egress from \( l \). Also, \( l \) does not have a preference for any other upstream egresses and so \( l \) will prefer \( m \) over any egress it can learn from \( 3 \). Hence the reliance of \( l \) on \( 3 \) is pruned. Other reliances which are pruned using this technique are \( h \leftrightarrow 2 \) and \( d \leftrightarrow 1 \). Notice \( g \leftrightarrow 2 \) is not prunable as \( g \) may select an egress learned from \( 2 \).

We include in the third column of Table 2 whether reliances are able to be pruned and in column four, we provide non-trivial reasons why a reliance can (or cannot) be pruned.

5 Finding Solutions

5.1 Finding a Feasible 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 Figure 3(d). One topological ordering (any topological order will result in the same network solution) is the numerical ordering of \( D_1 - D_{19} \) in Figure 3(d). If multiple routers are present in a co-reliance group (such as
the decisions of the routers may be dependent on message timing and we evaluate their decisions until a feasible solution is found. The ordering in which we evaluate the routers within a co-reliance group determines which of possibly multiple feasible 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.

5.2 Finding a Consistent Solution

A walk of the reliance graph can have multiple feasible 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 a solution which is consistent with the actual decisions made by routers. If the decisions of some routers in the network are known, through measurement, we can use them as constraints while determining the solution, noting that there may still be multiple feasible solutions which match these known route selections. 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, although this approach can result in situations where we reach a co-reliance group and there are no possible solutions satisfying the constraints. Such situations can arise where a lack of constraints at a non-singleton co-reliance group may cause a random ordering of router evaluation that subsequently results in a route being unavailable for selection at a later co-reliance group. Also, a random tie-break decision (when the oldest-route tie-break is used) may also cause a route to be unavailable for selection at a later co-reliance group in our topological ordering. To address this issue, we could backtrack along the reliance graph to resolve such discrepancies. In our examined network, however, there were only seven occasions, out of over 12.7 million measured decisions (that is, measurements taken from a subset of routers for all available prefixes), where we could not form a consistent solution at a particular router. Therefore, at this stage we have not implemented a backtracking algorithm.

We saw in the example of Figure 2(b), evaluating the decision of routers within a non-singleton co-reliance group determines to which of multiple feasible solutions we converge. We now outline our heuristic to converge to a consistent solution, satisfying all measurement constraints.

5.3 Ordering of Routers Within a Co-reliance Group

We currently have an ordering of co-reliance groups to visit. However, a co-reliance group can contain multiple routers. The order in which we evaluate
the BGP decision process on routers within a co-reliance group determines upon
which of possibly multiple network solutions we converge. Network measurement
infrastructure in the form of route-monitors (or any other available source) can
provide us with additional information as to the current network state. For co-
reliance groups with multiple routers, we can order the routers to increase the
probability of converging to a solution consistent with route-monitors such that
we do not need to re-visit a co-reliance group.

We use the comparison subroutine, compare_routers shown in Figures 4 and
5, along with sorting algorithm to order routers within a co-reliance group (we
happen to have used the inbuilt sort function of Perl [21]).

\[
\begin{array}{|l|l|}
\hline
T_r & \text{Available routes learned from neighboring routers according to topology rules} \\
\alpha(T_r) & \text{Returns preferred route from } T_r \text{ at router } r \\
b_r & \text{Selected route at router } r \\
\beta_{D_k} & \text{Routers with decisions evaluated in co-reliance group } D_k \\
\beta_D & \text{Co-reliance groups already visited} \\
\mathcal{M}(r) & \begin{cases} 
1 & \text{if monitor available for router } r \text{ and a downstream egress is preferred,} \\
0 & \text{if no monitor is available on router } r, \\
-1 & \text{if a monitor is available and a non-downstream egress is preferred} 
\end{cases} \\
\mathcal{R}(r) & \text{Set of routers reliant on } r \\
\mathcal{H}(r) & \text{Number of down iBGP edges a router } r \text{ is from a downstream egress} \\
\text{min} & \text{First element in a sorted set} \\
\Gamma & \text{Routers which modify their decision on backtracking} \\
\hline
\end{array}
\]

Fig. 4. Function and variable definitions used in the compare_routers and the network
solver algorithm.

If compare_routers(a,b) returns 1, we evaluate the BGP decision process
at router a before router b. If compare_routers(a,b) returns −1, we evaluate
the BGP decision process at router a after router b. If compare_routers(a,b)
returns 0, we have no information indicating whether the BGP decision process
at a or b should be evaluated first. This algorithm is easily combined with Perl’s
inbuilt sort. We describe the reasoning for the router comparison below.

Firstly, to minimize visits to routers within a non-singleton co-reliance group,
downstream routers (routers with fewest iBGP hops to the egress) are evaluated
first to ensure maximum information is available as early as possible.

For a monitored router to choose the route it is known to select, the route
must be in its set of available routes. We can increase the likelihood of this by
ensuring if a monitored router prefers its downstream egress, we evaluate this
router prior to other routers. In addition, if a monitored router prefers a non-
downstream egress, then we would like this route to be in the set of available
routes when we visit the monitored router. Hence, we evaluate the monitored
router after other routers.
sub compare_routers(a,b)
    // Evaluate routers closest to egress first
    if H(a) < H(b)
        return 1
    elsif H(a) > H(b)
        return -1
    end-if
    // Evaluate Monitored routers preferring downstream first
    if M(a) > M(b)
        return 1
    elsif M(a) < M(b)
        return -1
    end-if
    // Evaluate routers whose reliant routers need downstream route
    parent_reliance_a=0
    if M(a) == 1
        foreach a_i ∈ R(a)
            if M(a_i) ∈ Λ(a)
                parent_reliance_a=1
            end-if
        end
    end-if
    parent_reliance_b=0
    if M(b) == 1
        foreach b_i ∈ R(b)
            if M(b_i) ∈ Λ(b)
                parent_reliance_b=1
            end-if
        end
    end-if
    if parent_reliance_a < parent_reliance_b
        return 1
    elsif parent_reliance_a > parent_reliance_b
        return -1
    end-if
    // No information to indicate ordering of routers
    return 0
end-sub

Fig. 5. Router comparison subroutine for a non-singleton co-reliance group. By ordering routers within a co-reliance group, we can increase the likelihood of converging to a solution consistent with constraints learned from the measurement infrastructure. If the compare_routers(a,b) returns 1, we evaluate the BGP decision process of router a prior to router b. If the compare_routers(a,b) returns -1, we evaluate the BGP decision process of router a after router b. This subroutine can be used in conjunction with a generic sorting algorithm (we use the inbuilt Perl sort) to determine a sorted list of routers.
There are still issues to ensure consistency with the BGP monitor. These may occur when we have, for example, one monitored router which prefers another route-reflector’s egress for which we do not have a BGP monitor. We attempt to ensure this egress is available when we evaluate this router’s decision by evaluating the decisions of parents of such egresses prior to other routers.

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 as shown in Figure 6. 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 network solver 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.

\[
\begin{align*}
\beta_D & \leftarrow \phi \\
\text{while } \exists D_k \notin \beta_D & \quad D_k = \min\{D_j | D_j \notin \beta_D\} \\
\beta_{D_k} & \leftarrow \phi \\
& \quad // \text{Reset visited routers} \\
& \quad // \text{Initializing router decisions} \\
\text{while } \exists r \notin \beta_{D_k} & \quad r = \min\{r_i | r_i \notin \beta_{D_k}\} \\
b_r & \leftarrow \alpha_r(T_r) \\
\beta_{D_k} & \leftarrow \beta_{D_k} \cup r \\
\end{align*}
\]

\[
\beta_D \leftarrow \beta_D \cup D_k
\]

Fig. 6. Network solver algorithm. The covering \textbf{while} loop enumerates all co-reliance groups. The first inner loop enumerates all routers within a co-reliance group. We have separated, for clarity, the process of checking if all router’s decisions are stable in a non-singleton co-reliance group. However, in practice, this loop can be integrated into the first inner loop.
We saw in [4] that it is possible that a co-reliance group never converges to a solution. 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 then investigate possible corrective actions.

5.4 Breaking Ties

When the oldest-route tie-breaker is used and there could be multiple routes available at a router with equal IGP distances to an egress, then any route with this equal-best IGP distance may be chosen by a router. This can increase the number of feasible network solutions. Similar to Section 5.3 we would like to converge to a solution satisfying our measurements. Obviously, if we have a route-monitor at a router and the route selected by the monitor is in the set of equal-best IGP distance routes, then we select this route. However, if no route-monitor is available for a router, we select the route that satisfies the greatest number of monitors connected to reliant routers. Once a monitor on a reliant router has been satisfied, it is marked so other tie-breaking decisions will satisfy other monitors. If the tie is still not broken, we select a route at random.

5.5 Dynamic IGP

IGP distances are dynamic and can change when the underlying network changes. Consequently, our approach must be amenable to IGP changes as well as BGP changes. An OSPF monitor [22] is able to record the dynamics of the IGP, and this information can be fed into our model. The design of the reliance graph (which abstracts the actual IGP distances), allows us to re-evaluate router decisions only when the changes in distance affect router decisions. This process is shown in Figure 7. The ranking of egresses at each router can be thought of as the index in a sorted list (based on IGP distance). If the lowest-router-id is used as the tie-break option, then the list can be completely sorted. If the updated distance between two routers does not affect the ranking of the egress, then the decision of all routers will not be altered. However, if the ranking of the egress is altered, then the reliance graph may be changed and consequently the decisions of some routers may also be changed.

We only need to examine the egress instances in which the reliance graph can be altered. That is, those egress instances in which both the egress affected by the new distance and another egress which alters its ranking are part of the egress instance. Even if the best egress is not affected by a change in IGP distance, we re-calculate the reliance graph to ensure all reliance graphs are synchronized with the current state of the network.

6 Evaluation

We have implemented our technique using a shared Sun-Solaris server using Perl v5.8.8 [21] and evaluated them using data collected from a large Tier-2
Fig. 7. Subroutine igp_change for determining the reliance graphs requiring recalculation when an IGP distance changes.

AS. This data included router-configuration files, IGP distances from an OSPF monitor [22] and BGP routes from a route-monitor connected to approximately 15% of routers, a majority of which were route-reflectors. 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. Where no IGP distance is available (which may be due to OSPF monitor limitations in certain areas of the network), we assume the egress is unreachable. However, if we have a route-monitor indicating a route is better than others but we have no IGP distance, we assume the IGP distance is less than all other routes. The AS has a three-level RR hierarchy and uses the oldest-route tie-break option. The MED attribute is reset by the AS.

Our technique 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 following the examined point in time. Our evaluation is based on data collected on 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 confirmed occurrences
of this configuration error with operators, and they were subsequently corrected. We have excluded the prefixes affected by these configuration errors from our analysis.

We group all prefixes with the same egress instance into *molecules*. A single reliance graph exists for each molecule. We were able to cluster the 224,870 stable prefixes into 827 reliance graphs — a significant reduction in required computation. However, as there are multiple feasible solutions for a single reliance graph (due to the ordering of routers within a co-reliance group and the oldest-route tie-break), we split molecules into *atoms*. Atoms are clusters of prefixes with the same egress instance and also all route-monitors indicate the same egress is selected for each router. Each atom requires a ‘walk’ of the reliance graph. For the 224,870 stable prefixes, we discovered the egress router selected by all routers (including the 85% of routers without route-monitors) with 1,154 walks of the reliance graph.

As our technique is based on the rules of route propagation, it will *always* find a feasible 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, our technique always found feasible solutions, and as mentioned in Section 5.2, only very rarely did it produce a solution that was inconsistent with the supplied constraints. Even in these rare cases, the predicted egresses were in the same PoPs as the actual selected egress, and backtracking to alter the random aspects of decision making 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 Generalized Topologies

Our technique for determining the network solution has two components. The first component is the creation of the reliance graph from specific topology rules. The second component is the calculation of router decisions based on the abstract reliance graph. We have so far focused on the creation of a reliance graph for the route-reflector topology with the MED attribute filtered. However, the solution of other, more complicated topologies can also be calculated by only altering the first component of our technique. In this section, we now consider several other examples of more complex topologies and how rules governing their route propagation can be used to create reliance graphs.
7.1 Route-Reflection with MED

The Multi-Exit-Discriminator, or MED value, can be associated with announced routes to indicate which, of multiple inter-connecting links is preferred by the announcing AS. Many ASes, such as the AS we examined in Section 6 filter this attribute to prevent neighboring ASes influencing internal traffic flow. However, some ASes do not filter this attribute to allow neighboring ASes to indicate their link preference. Consequently, it is important to consider the impact to the network solution when the MED attribute is not filtered. Hence, we now consider a route-reflector topology where an AS respects MEDs (which are set by neighboring ASes). That is, MED values are only compared if they are learned from the same AS. Consequently, it is no longer valid to assume all routers select routes that are equally attractive through the MED step of the BGP decision process. However, it is valid to assume that all routers select a route equally attractive up to the MED step \[8\]. It is also no longer valid to assume routers prefer their direct egress. However, all direct egress routers with the best AS-wide MED value (for each neighboring AS) will still always select their direct egress. A direct egress router with a non-optimal MED value will select a better route if it learns of it. Thus, all valid paths in the signaling graph such that the non-optimal egress can learn of this route must be in the reliance graph. The increase in the number of edges in the reliance graph can increase the size of the co-reliance groups. However, the maximum co-reliance group size is still bounded by the size of the egress ancestor set (routers with either a direct egress or can learn of an egress from a client) which is commonly an order of magnitude smaller than the total number of routers. Feamster and Rexford \[8\] recommend a simulator as the most efficient method of solving this case, but our approach involves significantly reduced complexity.

Consider the example in Figure 8(a). In this example, we assume all route-reflectors are closer to their clients than any other router. Two ASs (black and white) send routes with MED attributes depicted by a number inside the large arrows. Recall that reliances only indicate where a router may learn of its best route. There are more possibilities in this case as border routers with a direct egress can choose to exit the network via an indirect route (if they learn of a route from the same AS with a lower MED value).

Firstly, as shown in Figure 8(a) we add reliances as before based on IGP distances. As we have assumed for this example Constraint A is satisfied, no router with a client egress can be reliant on another router in the same level of the hierarchy.

Secondly, we add reliances resulting from the MED announcements from white. Router 2 has a direct egress and hence it may select this route. However, 1 has an egress with a better MED value than the direct egress via 2. If 2 ever learns of the better route via 1, it will select it. Thus, we must add reliances on every feasible signaling path from 1 to 2. In this two-level hierarchy, there is only one feasible signaling path \((1, 3, 4, 2)\). Hence, we add the reliances between router 3 and 4 and 4 and 2. Router 7 also learns of a route from white. If 7 ever learns of the egress via 1 or 2, it will select it. Hence, we add all feasible paths...
(a) IGP distance reliances. In this example, all client egress routers are closer than all non-client egress routers.

(b) Addition of reliances from white. Router 1 is not reliant on any other router, as no white MED value is better than its own direct egress MED value. However, if router 2 learns of the route from 1, then it will select it over its own direct egress. Hence, on all possible signaling paths from 1 to 2, we create reliances (in this example, there is only one valid signaling path (1, 3, 4, 2)).

(c) Black reliances and co-reliance groups. Router 9 may alter its decision if it learns the black route from router 7. We create reliances on the signaling path between 7 and 9. Co-reliance groups are the strongly connected components of the reliance graph.

Fig. 8. An example topology where the MED attribute is respected. The shading of the large arrows represent the neighboring AS originating the route, and the MED value is depicted within the arrow.
from 1 to 7 and 2 to 7. Reliances between routers 3 and 5, 4 and 5, and 5 and 7 are inserted into the reliance graph. All the above reliances are shown in Figure 8(b).

Thirdly, reliances can be created from the announcements of black. Router 9 will modify its selected route if it learns of the route via 7. Thus, as the only feasible signaling path from 7 to 9 is (7, 5, 4, 9), we add the reliances between 5 and 4 and 4 and 9.

Now that we have the reliance graph for this topology, we find the strongly connected components which form the co-reliance groups shown in Figure 8(c), and apply our general algorithm for determining router decisions, as described in Figure 6.

Feamster and Rexford [8] stated simulation is the best way to solve the network solution when MEDs are compared per AS. However, we have shown that this is not the case. In the best case, our techniques are linear in the number of routers, and in the worst case router decisions only need to be evaluated multiple times in the egress ancestor set — which is much smaller than the total number of routers.

### 7.2 Full Mesh

A full mesh is the simplest of iBGP topologies but it can still be difficult to analyze in the presence of MEDs. When the MED attribute is either ignored or compared AS-wide, all routers with a direct egress (AS-wide best-route up-to the IGP distance step) will select it as best and hence are not reliant on any other router. In a full mesh, all other routers have an iBGP session with all routers and thus the only reliance rule required for the full-mesh is for those routers with direct egress routes.

Consider the example in Figure 9(a). Here we have two routes equally attractive up to the IGP distance step arriving at routers 2 and 8. Given the simple reliance graph, the analysis is almost trivial.

When MED is introduced, we are still able to determine reliances in a similar way to the route reflection example in Section 7.1. We simply place reliances on all routers with larger MED values (per neighboring AS). In Figure 9(b) we see router a has a direct egress, however it is reliant on b as it will select the route learned at b from black (as it has a lower MED value). However, it will not select the route learned at h as the white and black MED values are not comparable. Therefore, the reliance graph is simple, and the analysis is trivial once again.

An interesting example, shown in Figure 9(c), is derived from Griffin and Wilfong’s ‘Mashed Potato’ configuration [23]. Here black and white each announce routes at h and b with black preferring the egress at h and white the opposite. This results in a non-singleton co-reliance group $D_1$. In this situation, there are two feasible solutions for this co-reliance group which are $(b, h) = (\text{black}_{20}, \text{white}_{20})$ or $(\text{white}_{30}, \text{black}_{30})$. Either is a possibility due to tie-breaking decisions at whichever of $b$ or $h$ happens to be evaluated first (as MED values are not compared for egresses to different ASes). It is worth noting
Fig. 9. Reliance graphs for a full-mesh topology with the MED attribute respected.
that the solution \((b, h) = (\text{white}_{30}, \text{black}_{30})\) results in both backup routes being selected, which breaks the semantics of the MED attribute [23].

7.3 Confederations

Confederations of sub-ASs are used as an alternative to route-reflection in large networks where a full iBGP mesh is infeasible. The large AS is split into a confederation of sub-ASs. Within a sub-AS, the reliances can be calculated as with a full-mesh topology. Additional reliances are required between routers in separate confederations with iBGP sessions as they are able to propagate any route learned. Complicated topologies such as a combination of route reflection and confederations (although we are unaware of this topology being used in any network) can be solved if reliance rules can be found. In the worst case every router could be part of a single co-reliance group, and the algorithm is effectively a network simulation.

Consider the example in Figure 10(a). The AS is split into four sub-ASs. Each sub-AS has a full-mesh topology. Links between sub-ASs exist between routers \((d, j), (d, g), (h, n), (m, j)\). Equally good routes enter the AS at router \(a\) and \(f\). As shown in Figure 10(b), within each sub-AS, reliances are found as with the full mesh topology. If we assume the intra-sub-AS distances are closer than inter-sub-AS distances then all co-reliance groups in sub-ASs with direct egresses are singular. However, \(D_{10}\) is non-singular as \(j\) is closer to \(f\) than \(a\) and \(m\) is closer to \(a\) than \(f\). The order in which messages are passed determines which solution is found.

\[ \text{Fig. 10. An example confederation of sub-ASes and the corresponding reliance graph.} \]
When multiple routers are present in a co-reliance group, we use a similar ordering of routers within the co-reliance groups as to the route-reflection topology. We evaluate routers with monitors using an egress with the fewest sub-AS hops first. For example, in Figure 10(b), if \( j \) chooses to egress via \( a \) we evaluate \( a \)'s decision prior to \( m \)'s (as \( m \) would choose \( a \) as well).

A possible extension of the description of confederations with reliances is the description of inter-AS relationships and the prediction of Internet-wide routes. A topology inferred by a technique such as Mühlbauer et al. [24] 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. In addition, the ordering of router decisions outlined in this paper could also be used to improve the convergence times of existing BGP simulators such as C-BGP [19].

8 Conclusions

In this paper, we presented a technique using a reliance graph to determine the routing decisions for all routers within an AS. This technique is generally applicable on any topology describable as a reliance graph. The conversion of a physical topology to a reliance graph makes our technique applicable to practical networks. This conversion requires knowledge of the rules of the actual topology analyzed. Throughout this paper we presented an in-depth examination of the topology of a large Tier-2 AS and in Section 7 we considered several other possible iBGP topologies.

One significant benefit of using a reliance graph technique 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 [25].

References