Modeling BGP Route Selection within an AS

Nick Feamster
MIT Computer Science & AI Lab
feamster@csail.mit.edu

Jennifer Rexford
AT&T Labs–Research
jrex@research.att.com

Abstract

This paper presents a provably correct model that computes the outcome of the BGP decision process for each router in a *single* AS, without simulating the complex details of BGP message passing. The model requires only static inputs that can be easily obtained from the routers: the set of candidate routes to a destination (and the routers in the AS at which they were learned), import policies and other session-level parameters, and internal topology. We propose algorithms for computing route selection under four different network configurations: with the MED attribute compared across all routes, and compared only across routes from the same neighboring AS; and with a "full mesh" internal BGP (iBGP) topology versus an iBGP topology that uses a scalability technique called "route reflection". For each scenario, we derive general properties of the routes that routers ultimately select, present an efficient algorithm for computing the outcome of BGP route selection, and prove the algorithm's correctness. Studying the general properties and computational overhead of modeling the route selection process in each of these cases provides insights into the unnecessary complexity introduced by the MED attribute and route reflection; we use these insights to propose improvements to BGP that achieve the same goals as MED and route reflection without introducing the negative side effects of these features. We have implemented some of the algorithms from this paper in a prototype and have shown them to be efficient and accurate enough for many traffic engineering tasks.

1. Introduction

To control the flow of traffic through their networks, operators need to know how configuration changes affect the decisions made by the routers. In large backbone networks, the selection of paths depends on the routes advertised by neighboring domains and the internal topology, as well as the interdomain routing policies and intradomain link weights. To avoid costly debugging time and catastrophic mistakes, operators must be able to make predictions quickly based on an accurate, network-wide model of the routing protocols. In this paper, we present efficient algorithms for computing the routing decision at each router in an Autonomous System (AS). In solving the problem, we grapple with two features of the Border Gateway Protocol (BGP) that leave routers with no consistent ranking of the candidate routes and limit visibility into the routing options.

1.1 Backbone Network Engineering

To model the selection of routes inside a backbone network, we must capture the interaction between three routing protocols, as shown in Figure 1:

- **External BGP (eBGP):** The AS uses external BGP (eBGP) to exchange route advertisements with neighboring domains. For example, the routers W, X, and Y each have eBGP sessions with neighboring ASes. The routers may apply an *import* policy to modify the attributes of the routes learned from the neighbor, with the goal of influencing the decision process in Table 1 that each router applies to select a single best BGP route for each destination prefix.

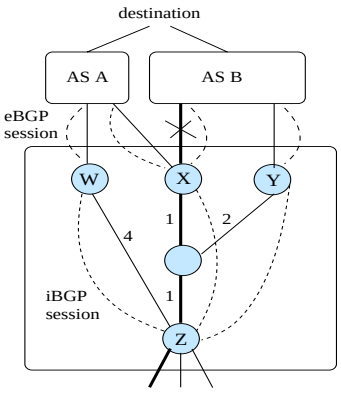

Figure 1: Network with three egress routers connecting to two neighboring ASes: Solid lines correspond to physical links (annotated with IGP link weights) and dashed lines correspond to BGP sessions. Thick lines illustrate the shortest path from one router to its closest egress point for reaching the destination.

- **Internal BGP (iBGP):** The routers use internal BGP (iBGP) to disseminate the routes to the rest of the network. In the simplest case, each pair of routers has an iBGP session, forming an "full mesh." If two routes are equally good through the first four steps in Table 1, the router favors an eBGP-learned route over an iBGP-learned one. In Figure 1, router Z receives three iBGP routes, from routers X, Y, and Z, forcing the decision to proceed to the next step in Table 1.

- **Interior Gateway Protocol (IGP):** The routers inside the AS run an Interior Gateway Protocol (IGP) to learn how to reach each other. The two most common IGPs are OSPF and IS-IS, which compute shortest paths based on configurable link weights; the routers also use the IGP path costs in the sixth step in Table 1. In Figure 1, router Z selects the route with the smallest IGP path cost of 2, learned from router X[1].

After completing the path-selection process, the router combines the BGP and IGP information to construct a forwarding table that maps the destination prefix to the outgoing link along the shortest path. In Figure 1, the forwarding path consists of the thick lines from the ingress link at router Z to the egress link at router X.

[1] If two routes have the same IGP path cost, the router performs an arbitrary tiebreak in the seventh step in Table 1.

1. Highest local preference
2. Lowest AS path length
3. Lowest origin type
4. Lowest MED (with *same* next-hop AS)
5. eBGP-learned over iBGP-learned
6. Lowest IGP path cost to egress router
7. Lowest router ID of BGP speaker

Table 1: Steps in the BGP decision process

If the link between X and AS B becomes persistently congested, the network operator may need to adjust the configuration of the routing protocols to direct some of the traffic to other egress routers. For example, the operator could modify the import policy at router X to make the BGP routes for some destinations look less attractive than the routes learned at other routers [1]. Changing the import policy causes X to advertise a BGP route with a different attribute value (e.g., a smaller *local preference*) to influence the routing decisions throughout the network. Ultimately, the change in the import policy at X has the *indirect* effect of directing some of the traffic entering at router Z to egress router Y (the next-closest egress point, in terms of the IGP path costs), thereby alleviating the congestion on the link connecting X to AS B. Network operators make similar kinds of configuration changes for a variety other reasons, such as exploiting new link capacity, preparing for maintenance on part of the network, or reacting to equipment failures.

Operators need to predict the effects of changes to the routing protocol configuration before modifying the live network. Human intuition, while powerful, is not sufficient for understanding the complex interactions between three routing protocols running on a large, dynamic network. Experimenting on the live network runs the risk of making disruptive configuration changes that degrade performance. Instead, we believe that operators should have an accurate and efficient tool that computes the effects of configuration changes on the flow of traffic through the network. The efficiency of the model is of paramount importance, since a network operator (or an automated optimization algorithm) may need to explore numerous candidate configuration changes before identifying a choice that satisfies the engineering goals. Rather than simulating the complex message-passing details of the protocols, the model needs only to compute the *outcome*—the routing decision for each router once the protocols have converged.

1.2 Challenges and Contributions

In this paper, we present a model that accurately determines how the network configuration and the routes learned via eBGP affect the flow of traffic through an AS. While some existing tools simulate BGP's behavior [2], our work is the first to develop a model that determines the *outcome* of the BGP path selection process at each router in an AS without simulating the dynamics of the protocol. This problem would be easy to solve if (i) the decision process in Table 1 allowed each router to form a consistent ranking of the candidate routes and (ii) the distribution of routes in iBGP allowed each router to learn every route. However, two features of BGP conspire to make the problem much more challenging:

- **The Multiple Exit Discriminator (MED) attribute breaks the ranking of routes:** An eBGP neighbor can set the MED attribute of a route advertisement to influence the behavior of the receiving AS. For example, in Figure 1 AS B might send a route with a MED of 10 to router Y and a MED of 20 to router X; as a result, Z would select the route from Y with the smaller MED, even though the IGP path to X is shorter. In practice, the MED comparison in step 4 of the decision process applies only to routes learned from the *same* next-hop AS. When MEDs are used, the decision process does *not* impose a total ordering on the routes at each router; in fact, the choice of one route over another may depend on the presence or absence of a third route [3].
- **Route reflectors limit the visibility their clients have of the iBGP routes:** The quadratic scaling of a full-mesh iBGP configuration forces large networks to distribute routes in a hierarchical fashion. A router configured as a route reflector selects a single best route and forwards the route to its clients. Using route reflectors reduces the number of iBGP sessions, as well as the number of routes the clients need to receive and store. However, since a route reflector forwards only its *best* route to its iBGP neighbors, the choices available at one router depend on decisions at other routers. In particular, the route reflector may make a different choice in the sixth step of the decision process than its clients would have.

These two features of BGP cannot be ignored because they are commonly used to provide flexibility and scalability, respectively.

In this paper, we present a model of BGP path selection inside a single AS, as well as algorithms to compute the routing decision at each router. Our main contributions are:

- **Network-wide model of BGP path selection:** Rather than analyzing BGP *dynamics*, we model the *outcome* of the distributed path-selection process. When a routing system converges, the outcome does not depend on the order and timing of the messages, allowing our algorithms to model a message ordering that computes the outcome efficiently.
- **Capturing the influence of MED and route reflectors:** After presenting a simple algorithm for networks that disable MEDs and have a full-mesh iBGP configuration, we present algorithms that handle MED and route reflectors in isolation. Then, we present another algorithm that captures the complex interaction between the two features.
- **Proposed improvements to BGP:** In addition to complicating the modeling problem, these two features create difficulties for the operation of BGP itself. We discuss ways to improve the design and operation of BGP to avoid the harmful effects without sacrificing the policy semantics of MEDs and the scalability provided by route reflectors.

In Section 2, we present practical constraints that allow us to compute the outcome of the routing protocols and decompose the modeling problem. Section 3 presents a model for BGP path selection for the simplified case of a full-mesh iBGP configuration and no MED attribute[2]. Section 4 focuses on modeling the MED attribute, assuming a full-mesh iBGP configuration. In Section 5, we consider more complex iBGP session configurations. Section 6 presents improvements to BGP. Section 7 provides an overview of related work, and Section 8 concludes. Appendix A presents a brief summary of the design, evaluation, and validation of a tool that implements our model [4].

2. Modeling Constraints and Overview

Network operators frequently adjust the configuration of BGP policies to control the flow of traffic through an AS. In this section, we define three constraints that a routing system must obey

[2]Throughout the paper, we often describe BGP "without MED". Network configurations "without MED" could also be viewed as a configuration that compares the MED attributes across all routes (e.g., in Cisco IOS, this behavior can be enabled with `always-compare-med` setting).

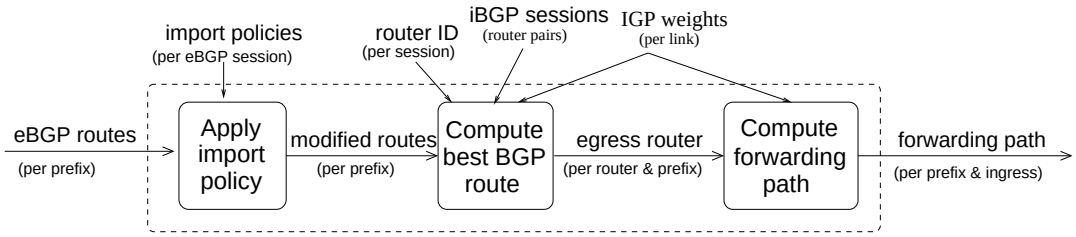

Figure 2: Decomposing *network-wide* BGP route selection into three independent stages

for a model to provide accurate predictions. Next we describe how these constraints enable us to decompose the modeling problem into three stages—applying the import policy to eBGP-learned routes, selecting the best BGP path at each router, and computing the forwarding path. Since the first and third stages are relatively simple, the rest of the paper focuses on the second stage of computing the BGP routing decision at each router for each destination prefix.

2.1 Constraints on Modeling Path Selection

Modeling the effects of a configuration or topology change is possible when three important conditions hold. First, the *inputs* to the model must be relatively stable. In particular,

CONSTRAINT 1. *The eBGP-learned routes change slowly with respect to the timescale of network engineering decisions.*

If the eBGP-learned routes change frequently, the internal routing system does not have time to propagate the effects of one eBGP advertisement before the next one arrives. In practice, most BGP routes are stable for days or weeks at a time [5], and the vast majority of traffic is associated with these stable routes [6]. This allows the routing model to operate on a static snapshot of the eBGP routes. Any eBGP routing change can be treated as a separate problem instance.

Second, the routers in the network must ultimately reach a stable outcome. In particular,

CONSTRAINT 2. *Given stable eBGP-learned routes and fixed iBGP and IGP topologies, each router inside the AS converges to a unique routing decision.*

If the routers continually change their routing decisions, no model could accurately predict how the data traffic would flow. Fortunately, previous work [7] has identified sufficient conditions for an internal routing configuration to satisfy constraint 2. We describe these conditions in more detail in Section 5 when we address the challenges introduced by route reflectors.

Third, the routing decisions at each router should not depend on time or message ordering:

CONSTRAINT 3. *The routing decision at each router must depend only on the routes received from its neighbors, and not the order or timing of the messages.*

For example, some router vendors have an additional step in the BGP decision process that favors the "oldest" route before the final tie-breaking step of comparing the router IDs. The "age-based tie-breaking" favors more stable routes, making the outcome of the BGP decision process dependent on the *order* the router receives the advertisements. Disabling age-based tie-breaking forces a deterministic selection based on the smallest router ID, as in Figure 1; other BGP features, such as route flap damping [8], can help reduce the likelihood of selecting unstable routes.

2.2 Decomposition of the Modeling Problem

The routing decision at each router depends on the eBGP-learned routes, the BGP routing policies, the iBGP session configuration, and the IGP topology. When constraints 1, 2, and 3 hold, a model can compute the effects of changes to these inputs:

OBSERVATION 1. *If a routing system is guaranteed to converge to a unique solution, the solution is independent of the order that routers exchange routes and apply the decision process.*

This observation means that we can consider the evolution of the routing system under *any particular ordering* of the steps in the convergence process, without the risk of arriving at the wrong answer. Following the approach in [9], we express an ordering in terms of an *activation sequence* that "activates" one or more routers in each phase. When activated, a router applies the decision process in Table 1 and propagates the best route to its iBGP neighbors. We construct an activation sequence that allows us to decompose the problem into three stages as shown in Figure 2:

1. Receiving the eBGP routes and applying import policy: The first stage activates all of the edge routers at the same time. This stage assumes that each router receives all of its eBGP-learned routes and applies the import policies, *before exchanging any iBGP update messages*. Each eBGP-learned route has attributes (such as the destination prefix and the AS path) and is associated with an eBGP session. The import policy may filter the route or set certain attributes such as local preference, origin type, and multiple-exit discriminator (MED), according to attributes in the advertised route (e.g., based on ASes in the AS path). Because applying the import policy is a local operation for each eBGP-learned route at each router, the first stage emulates exactly the operations a real router would perform upon receiving each of the eBGP routes. These routes, with modified attributes, form the input to the second stage.

2. Computing the best BGP route at each router: Many routes from the first stage would never be selected by any router as the best route. For example, an eBGP-learned route with a local preference of 90 would *never* be selected over another route with a local preference of 100. In addition, different routers in the AS may select different best BGP routes because they have different IGP path costs to the egress router. Also, a router can only consider the routes advertised by its iBGP and eBGP neighbors, which may influence the final decision at that router. The output of this stage is a single best egress router for each ingress router and destination prefix. Constructing an efficient activation sequence for this stage is challenging, and is the focus of the next three sections of the paper.

3. Computing the forwarding path through the AS: In the third stage, we model the effects of the IGP link weights on the construction of the forwarding path through the AS from an ingress

router toward a destination prefix. Given the chosen BGP route, the ingress router forwards packets along the outgoing link (or links) along shortest paths to the egress router, and the process repeats at the next router. Ideally, the traffic flows along the shortest path (or paths) all the way from the ingress router to the chosen egress router. However, in practice, routers along the shortest path may have chosen a *different* egress router. Such *deflections* can occur if the iBGP session configuration limits the BGP routing options at the routers [7]. By considering one router at a time, the third stage can compute an accurate view of the forwarding path(s) even when deflections occur.

Although the diagram in Figure 2 shows only three stages, we envision that network operators could incorporate other phases for additional functionality. For example, another phase could combine the predicted forwarding paths with traffic data to predict the load on each link in the network. Using the model for traffic engineering assumes that traffic volumes are relatively stable, and that they remain stable in response to configuration changes. In previous work, we found that prefixes responsible for large amounts of traffic have relatively stable traffic volumes over long timescales [1]. Operators could use the routing model to test configuration changes on reasonably slow timescales that affect prefixes with stable traffic volumes. A network operator could also combine measurements or estimates of the traffic arriving at each ingress router for each destination prefix [10] with the link-level paths to predict the load on each link in the network. Another phase might evaluate the optimality of the these link-level paths in terms of propagation delay or link utilization and could search for good configuration changes before applying them on a live network.

3. BGP with Ordering and Full Visibility

In this section, we present some basic properties of BGP route selection when a network employs a full mesh iBGP topology and the MED attribute is compared across all routes. Based on these properties, we describe an algorithm that models BGP path selection for this simple case.

3.1 Notation and Basic Properties

To model BGP path selection more precisely, we must first introduce some notation. Table 2 summarizes the notation we use for the remainder of this paper and summarizes where this notation is introduced. We assume that the AS has a set of N eBGP-learned routes, E, for a given destination prefix, which it learns at R routers. E contains the eBGP-learned routes after import policies have been applied. For convenience, we define $E_r \subseteq E$ as the set of eBGP-learned routes at router $r \in R$. At any given time, a router also has zero or more iBGP-learned routes $I_r \subseteq E$. We also define two functions that we will use throughout the remainder of the paper:

- λ_r, which takes a set of best routes and outputs the best route according to the BGP decision process in Table 1.

  The subscript r on λ_r provides necessary context because different routers can apply the BGP decision process to the same set of routes and obtain different results. For example, in Figure 1, router X would treat the route learned from AS B as an eBGP-learned route with the router ID of the eBGP session with B. On the other hand, Z sees an iBGP-learned route with IGP path cost of 2 and the router ID associated with the iBGP session to X.

Symbol	*Description*	*Section*
	FUNCTIONS ON ROUTES	
λ_r	Takes a set of routes and outputs the best route, according to the BGP decision process applied at router r	3.1
γ	Takes a set of routes and extracts the subset whose attributes are equally good up through the first four steps of the decision process	3.1
σ	Takes a set of routes and extracts the subset whose attributes are equally good up through the first *three* steps of the decision process	4.2
	SETS OF ROUTES OR ROUTERS (INITIAL INPUTS)	
R	routers in the AS	3.1
A	routers that have been activated	5.2
E	eBGP-learned routes	3.1
E_r	eBGP-learned routes at router r	3.1
N	number of eBGP-learned routes (i.e., $\|E\|$)	3.1
	SETS OF ROUTES (INTERMEDIATE AND FINAL OUTPUTS)	
I_r	iBGP-learned routes at router r	3.1
P_r	All routes learned at router r	3.1
b_r	The best route that router r selects.	3.1
C	The set of candidate routes at some intermediate activation. A subset of E.	3.2
B	The set of best routes computed by the algorithm. A subset of C.	3.2
L	The set of routes eliminated at some activation step.	4.2
	IBGP TOPOLOGY	
S	iBGP sessions.	5.2
G	iBGP session graph. $G = (R, S)$.	5.2

Table 2: Description of the notation used in this paper, and the sections where each piece of notation is introduced.

- γ, which takes a set of BGP routes C, and outputs $C' \subseteq C$, such that routes in C' are the best routes based on the first four steps in Table 1.

  Unlike λ_r, the function γ is a function whose context is *global*; that is, its context is not router-specific. The function γ, will be applied to a set of routes to eliminate all routes that could *never* be the best route at any router.

Using Observation 1, we devise an *activation sequence*, which "activates" one or more routers at any given phase. When activated, a router r applies the BGP decision process to compute a best route $\lambda_r(E_r \cup I_r)$, which may then be propagated via iBGP. In reality, routers may be activated in any order and may change their best route many times before the network converges. In the following sections, we devise activation sequences that allow us to *efficiently* compute the final routing decision.

In this section and in Section 4, we will consider a *full mesh* iBGP topology (as described in Section 1). A full mesh iBGP topology provides full visibility of BGP routes at each router; that is, every router learns the complete set of eBGP-learned routes. Furthermore, when the MED attribute is compared across all routes (as opposed to just those from the same neighboring AS) a router's preferences over the set of routes it learns form a total ordering (i.e., $\forall P'_r \subseteq P_r$, if $a \in P'_r \subseteq P_r$ and $\lambda_r(P_r) = a$, then $\lambda_r(P'_r) = a$). These properties allow us to make two important observations about BGP's *steady state* path assignment.

First, when MED is compared across all routes, any router that selects a route from the set of eBGP-learned routes will select its locally-best route. Formally, call the best route that router r ultimately selects b_r. Then, $b_r \in E_r \Rightarrow b_r = \lambda_r(E_r)$.

LEMMA 1. *If the MED attribute is compared across* all *routes, then each router ultimately either selects its own best eBGP-learned*

§	*MED*	*RR*	*Running Time*	Lem. 1	Lem. 2
3.2	No	No	$O(N + \|R\|^2)$	•	•
4.2	Yes	No	$O(N \log N + N\|R\|)$		•
5.2	No	Yes	$O(N + \|S\|)$	•	•
5.3	Yes	Yes	$O(N \log N + N\|R\| + N\|S\|)$		

Table 3: Properties of the BGP path selection models in each of the four cases (with and without MED, and with and without route reflection).

route or some iBGP-learned route. Formally, $b_r \in E_r \Rightarrow b_r = \lambda_r(E_r)$.

Proof. By definition, each router r applies the decision process to the union of the routes it learns via eBGP and iBGP: $b_r = \lambda_r(E_r \cup I_r)$. Therefore, either $b_r \in E_r$ or $b_r \in I_r$. Furthermore, since the MED attribute is comparable across all routes, the router r's preferences over routes in $E_r \cup I_r$ form a total ordering, so either $b_r = \lambda_r(E_r)$ or $b_r = \lambda_r(I_r)$. But, if $b_r \neq \lambda_r(E_r)$, then $b_r = \lambda_r(I_r)$, so $b_r \in I_r$ and $b_r \notin E_r$. □

If the iBGP topology forms a full mesh, each BGP-speaking router ultimately selects a route in $\gamma(E)$; that is, every router ultimately selects a route that has the maximum local preference, minimum AS path length, lowest origin type, and lowest MED (assuming MEDs are compared across all routes).

LEMMA 2. *If the iBGP topology forms a full mesh, every router* r *will ultimately select a route,* b_r *that is in* $\gamma(E)$*, where* E *is the set of all eBGP-learned routes. Formally,* $b_r \in \gamma(E)$.

Proof. Assume that some router r selects $b_r \notin \gamma(E)$, and define $P_r \subseteq E$, the set of routes that router r learns. By definition, $b_r = \lambda_r(P_r)$, so $\lambda_r(P_r) \notin \gamma(E)$. This property implies that $P_r \cap \gamma(E) = \phi$; otherwise, b_r would be better than all routes in $\gamma(E)$, which contradicts the definition of γ. But, if $P_r \cap \gamma(E) = \phi$, then the iBGP topology does not form a full mesh, since at least one router $s \in R$ selects a route from $\gamma(E)$, and, in a full mesh topology, router r would have learned that route from s. □

This lemma also holds when the iBGP topology does *not* form a full mesh as long as the MED attribute is compared across all routes, as discussed in more detail later in Section 5. Table 3 summarizes the roles of Lemmas 1 and 2 in the four scenarios we model; the table also indicates the computational complexity for each algorithm. In the following subsection, we present an algorithm that models the outcome of the BGP decision process in the simple case of a full mesh iBGP topology where the MED attribute is compared across all routes, independent of the next-hop AS.

3.2 Algorithm: Full Mesh, No MED

Lemma 1 makes it possible to propagate the effects of route selection at each router only once, since each router will select its locally best eBGP-learned route or some other router's best route. Lemma 2 makes it possible to compute the route that each router r selects by simply applying λ_r to the set of all locally best routes, B (i.e., $b_r = \lambda_r(B)$). When Lemma 1 holds, selecting the best route at each eBGP-speaking router is straightforward, because it is possible to produce a total ordering of routes at each router. In this case, the algorithm for computing the best route at every eBGP-speaking router is simple, as shown in Figure 3. The algorithm takes as input the set of all eBGP-learned routes (E) and the set of all eBGP-speaking routers (R), and produces the set of best eBGP routes (B). E_r refers to all eBGP-learned routes learned by router

Algorithm: Full Mesh, No MED

SELECTBEST_EBGP(E, R)
Build the set of locally best routes at each router.
This set is the set of candidate best eBGP routes.
$C \leftarrow \cup_r \lambda_r(E_r)$
Eliminate all routes from C which
do not have highest local preference, etc.
$B \leftarrow \gamma(C)$

Figure 3: Algorithm for computing the best route at eBGP routers, assuming that MED is compared across all routes (i.e., that there exists a total ordering of routes at each router).

r, and C represents the set of candidate routes after each router selects the best route from the set of its eBGP-learned routes. The output of this algorithm is $B = \gamma(C)$, the set of all best routes to this destination, such that $b_r = \lambda_r(B)$.

To prove that this algorithm is correct, we must show that this algorithm accurately emulates *one* activation sequence; Observation 1 guarantees that as long as the algorithm correctly emulates a single activation, it will correctly emulate BGP route selection.

THEOREM 1. *When each router can produce a total ordering over all possible candidate routes, the algorithm in Figure 3 correctly computes the outcome of the decision process for all routers that select an eBGP-learned route as their best route.*

Proof. We prove this theorem constructively, by showing that the algorithm correctly emulates an activation sequence and message ordering that could result in BGP. Consider the following ordering:

1. All routers receive routes to the destination via eBGP. Then, every router is activated simultaneously.
2. Every router advertises its locally-best route via iBGP. After all iBGP messages have been exchanged, every router is activated simultaneously.

In the first phase, each router r computes $\lambda_r(E_r)$, resulting in a set of candidate routes $C = \cup_r \lambda_r(E_r)$, as in the first line of the algorithm in Figure 3. Then, each router learns these routes, resulting in $P_r = C$ for all $r \in R$. Note that $B \subseteq C$ by definition, which means that each router that learns a route to the destination via eBGP has either zero or one route in B. We consider both cases. If a router r has a route in C but not in B, then r's eBGP-learned route $b_r = \lambda_r(E_r)$ must have been worse according to the first four steps of the decision process than some other route, $b_s = \lambda_s(E_s)$ in C (otherwise, $\gamma(C)$ would not have eliminated it). But in a full mesh iBGP topology, r would learn a route via iBGP that is at least as good as b_s, so b_r would also be eliminated in phase 2 of the activation. Of course, if a router has a route in C, then that must be the route that it would select after phase 2 of activation: it is equally good as all routes in $\gamma(C)$ through the first 4 steps of the decision process (by construction), and it prefers its own best route over any iBGP-learned route (by step 5 of the decision process). □

Computational Complexity. When each router can form a total ordering over all possible candidate routes, the computational complexity for route prediction is proportional to the total number of routes in the system. The first step of the algorithm scans all N eBGP-learned routes and selects the best eBGP-learned route at each router, if any; at most $|R|$ routes remain after this step. The second step selects, for each router $r \in R$, the best route from R. Thus, the running time will be $O(N + |R|^2)$, where N is the number of eBGP-learned routes, and $|R|$ is the number of routers in

the system (a full mesh iBGP configuration will have $|R|(|R|-1)$ iBGP sessions. When $|R| > N$, the N term is dominated, so the running time is $O(|R|^2)$. When $N > |R|$, however, a simpler approach to the algorithm would simply be to apply $\lambda_r(E)$ at each router, which has $O(N|R|)$ running time.

The algorithm we have presented in this section works as long as routers compare the MED attribute across all candidate routes and when the network does not use route reflection. In practice, some ISPs configure their routers to compare the MED attribute across all candidate routes (often to avoid problems with oscillation), and most small networks do not use route reflection.

4. Modeling Path Selection with MED

In this section, we present how to model path selection when the MED attribute is compared only across routes learned from the same AS, rather than across all routes for a destination prefix. MED prevents each router from having a total ordering over all possible candidate routes, so it is actually possible to have $b_r \in E_r$ without $b_r = \lambda_r(E_r)$. In Section 4.1, we describe this problem in more detail and describe why the simple approach presented in Section 3.2 fails; then, we present an algorithm that accurately computes the outcome of BGP path selection when MED is compared only across routes from the same AS.

4.1 Problems Introduced by MED

The algorithm from Section 3.2 assumes that each router's ranking between two routes is independent of whether other routes are present (i.e., $\lambda_r(\{a,b\}) = a \Rightarrow \lambda_r(\{a,b,c\}) \neq b, \ \forall a,b,c$). When MED is only compared across routes from the same AS, the interaction between MED and router ID prevents the algorithm from simply selecting the locally best route at each router, because $b_r \in E_r \not\Rightarrow b_r = \lambda_r(E_r)$. This point has serious implications, because we can no longer assume that if a router selects an eBGP-learned route to a destination, that eBGP-learned route will be that router's locally best route; rather, the route that the router ultimately selects may be worse than the "best" route at that router when compared only against routes learned via eBGP at that router. Thus, the approach from Section 3.2, which computes b_r by taking the locally best route at each router from $\gamma(E)$, may not compute the correct result. Using the example in Figure 4, we explain why two seemingly-natural approaches to computing the routes do not work:

- *Local route elimination is not correct.* The algorithm in Figure 3 would first apply $\lambda_r(E_r)$ at each router. Given the choice between the two eBGP-learned routes a and c, router X prefers c, since c has a smaller router ID. However, between routes a, c, and d, router X prefers route a, because route d eliminates route c due to its lower MED value. Thus, router X's preference between routes a and b depends on which route Y selects. The algorithm in Figure 3 would compute $\lambda_X(\{a,c\}) = c$ and $\lambda_Y(\{b,d\}) = d$ (resulting in $C = \{c,d\}$), and ultimately compute $B = \{d\}$ because d has a smaller MED value than c. In reality, though, router X would select route a over d, because a is an eBGP-learned route from a different neighboring AS.

- *Global route elimination is not correct.* It might also seem reasonable to apply γ globally, followed by applying λ_r locally at each router. In a global comparison of the routes (i.e., when applying $\gamma(\{a,b,c,d\})$), a and c are first eliminated based on MED, and then router X picks route d (since d is preferred to b based on the router ID comparison applied at router Y). This conclusion is incorrect, because X would *always* prefer route a over route d, since a is learned via eBGP (step 5) and a and d are equally good up through step 4 (recall that a router does not compare the MEDs of routes with different next-hop ASes).

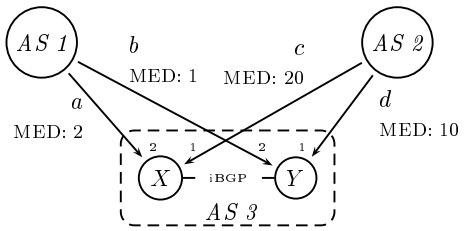

Figure 4: Interaction between MED and router ID in the BGP decision process. Small numbers are router IDs.

Algorithm: Full Mesh, MED

SELECTBEST_EBGP_MED(E, R)
  Build the set of initial locally best routes.
  $C \leftarrow \sigma(E)$
  $B_0 \leftarrow \phi$
  do
    $B_{i+1} \leftarrow \cup_r \lambda_r(C_r \cup B_i)$
  while $B_{i+1} \neq B_i$

Figure 5: Algorithm for computing the best route at eBGP routers, assuming that MED is only compared across routes *from the same neighboring AS*.

The crux of the problem is that the MED attribute makes it impossible to produce an ordering of the routes at X; the relative ranking of two routes depends on the presence or absence of a third route.

4.2 Algorithm: Full Mesh, MED

To correctly handle the interaction between the MED and router ID attributes, the algorithm emulates a message ordering that propagates the effects of MED on each router's best route. Figure 5 summarizes this algorithm. For this algorithm, we define a new function, σ, which takes a set of routes and returns all routes equally good up through the first three steps of the BGP decision process (i.e., local preference, AS path length, and origin type). When applied to the network in Figure 4, the algorithm starts with all routes in $\sigma(E)$ and proceeds as follows:

1. B_1 gets the locally best routes from X and Y: c and d, respectively. That is, $B_1 = \{c,d\}$.
2. On the second iteration, X compares the routes from C that it learns via eBGP, a and c, along with route d from B_1, so $\lambda_X(\{a,c,d\}) = a$. Similarly, $\lambda_Y(\{b,c,d\}) = d$. Thus, $B_2 = \{a,d\}$.
3. On the third iteration, the process repeats, and $B_3 = \{a,d\}$, at which point the algorithm terminates.

This algorithm computes the correct routing decision for each router: a at router X and d at router Y. At router Y, d is better than a (step 5), b (step 7) and c (step 4). At router X, a is better than d (step 5); a is not better than b, but this does not matter because router Y does not select b, and a is not better than c, but this does not matter since c is always worse than d (step 4).

THEOREM 2. *When MED is compared only across routes from the same neighboring AS, the algorithm from Figure 5 accurately*

emulates the results of one activation sequence and message ordering for all routers that select an eBGP-learned route as their best route.

Proof. Computing $\sigma(E)$ removes from C all routes in E that could *never* be the best route at any router (i.e., because they have a lower local preference, higher AS path, or lower origin type). Because the iBGP topology forms a full mesh, as long as there is a route in E at *any* router that is better in the first three steps of the decision process, no router will select a route that is not in $\sigma(E)$. The remainder of the algorithm evaluates a routing system with the routes in $E \setminus \sigma(E)$ removed.

The remainder of the algorithm follows an activation sequence where each phase (or iteration of the loop) activates all of the routers simultaneously. The proof proceeds by induction. After the first iteration of the loop, $B_0 = \phi$ and $b_r = \lambda_r(C_r)$, where C_r is all of the routes learned at router r via eBGP with the highest local preference, shortest AS path length, and lowest origin type. By definition, $\lambda_r(C_r)$ returns each router's locally best route according to the BGP decision process, which is the same as that which the BGP decision process would select for each router after phase 1 of the activation sequence. In a network with a full mesh iBGP configuration, each router r then sends its locally best route, b_r, to every other router.

Suppose the algorithm correctly computes the outcome of the BGP decision process for the first i iterations of the activation sequence. Assume that there is some router r for which the algorithm computes $b'_{r,i+1} \neq b_{r,i+1}$. Then, it must be the case that $b_{r,i+1} \notin C_r \cup B_i$, otherwise λ_r would also have selected $b_{r,i+1}$. Either $b_{r,i+1}$ is an eBGP-learned route or it is an iBGP-learned route. If it is eBGP-learned, then it must be in C_r, as we previously established. If it is iBGP-learned, then it must be in B_i, since every iBGP-learned route is the best route of some other router in the AS. But if either $b_{r,i+1} \in C_r$ or $b_{r,i+1} \in B_i$, then $b_{r,i+1} \in C_r \cup B_i$, which is a contradiction. □

The algorithm in Figure 5 is correct, but it is not efficient: each iteration of the loop repeatedly considers routes that have been "eliminated" by other routes. A more computationally efficient algorithm would eliminate routes from consideration at each iteration if we know that they could never be the best route at any router—such is the spirit of applying $\sigma(E)$ across the initial set of routes. Unfortunately, as we know from Section 4.1, because the MED attribute is not comparable across all routes and comes before the router ID step in the decision process, it is possible for a route that is not in the set B_i to emerge in the set B_j for some $j > i$. We now formally define a condition by which routes may be eliminated.

LEMMA 3. *Suppose there exist two routes $s \in C_r$ at router r and $t \in C_{r'}$ at router $r' \neq r$. If $t \in B_i$ and $\lambda_r(s,t) = t$, then $s \notin B_j \; \forall j > i$.*

Proof. First, note that as long as $t \in B_j$, then $s \notin B_j$ because route t is preferable to s Also note that because all routes in C are equally good up the MED comparison and eBGP-learned routes are preferred over iBGP-learned routes, we know that $\lambda_r(s,t) = t$ because MED(t) < MED(s). Now, suppose there exists some $j > i$ for which $t \notin B_j$. Call the best route at router r' at step i, $v = \lambda_{r'}(C_{r'}) \neq t$; again, we know that MED(v) < MED(t). But this means that MED(v) < MED(s), $\lambda_r(s,v) = v$, and, thus, $s \notin B_j$. □

We can use this result to devise a more efficient route prediction algorithm that eliminates, at every iteration, a router's locally best route if it has a higher MED value (and same next-hop AS) than any other router's locally best route. This algorithm is described in Figure 6 and shown conceptually in Figure 7; it can also be thought of in terms of an activation sequence: (1) each router learns routes via eBGP, selects a locally best route, and readvertises via iBGP; (2) each router compares its locally best route with all other routes learned via iBGP, and *eliminates* its own locally-best route from the system if it is worse than some other locally-best route at another router; (3) the system is restarted (from phase 1) with the eliminated routes removed. This algorithm is computationally more efficient than the one in Figure 5; we now analyze its running time complexity.

Algorithm: Full Mesh, MED (Alternate Algorithm)

SELECTBEST_EBGP_MED(E, R)

Eliminate all routes from E_r (locally at r) which do not have highest local preference, etc.
$C \leftarrow \sigma(E)$

Keep track of the best routes at each router.
do
$B \leftarrow \cup_r \lambda_r(C_r)$
$L \leftarrow B \setminus \gamma(B)$
$C \leftarrow C \setminus L$
while $L \neq \phi$

Figure 6: Computationally efficient algorithm for computing the best route at eBGP routers, assuming that MED is only compared across routes *from the same AS* (i.e., that there is no total ordering of routes).

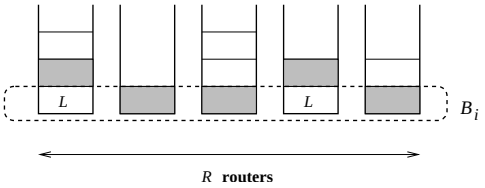

Figure 7: Implementation of the route computation algorithm from Figure 6. Each stack represented one of $|R|$ total routers, and each stack element represents one of L routes. The bottom elements of the $|R|$ stacks represent B_i, the elements marked L represent routes that are worse than the routes at the bottom of the remaining stacks according to the first four steps of the decision process, and the shaded routes represent B_{i+1}.

Computational Complexity. Understanding the running time of the algorithm from Figure 6 is easiest when we consider the implementation of the algorithm shown in Figure 7. In this figure, the eBGP-learned routes at each router are represented as a stack and are sorted *locally* (i.e., compared only to other routes learned at the same router). The bottom of the stack represents the best route learned at that router; the route that is second from the bottom is the second best route, and so forth. Then, the algorithm from Figure 6 can be interpreted as follows:

- $B \leftarrow \cup_r \lambda_r(C_r)$ is the union of all of the elements at the bottom of the stack and does not need to be computed explicitly, assuming each stack is sorted. The complexity of sorting N total routes distributed across $|R|$ stacks is $O(N \log N)$.
- $L \leftarrow B \setminus \gamma(B)$ marks a route at the bottom of a stack if that route is worse than any route at the bottom of another stack, according to the first four steps of the BGP decision

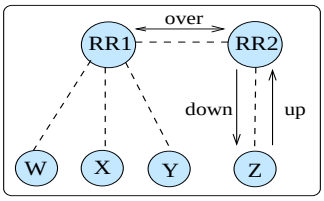

Figure 8: Example iBGP signaling graph

process. This process takes at most two scans of the routes at the bottom of the $|R|$ stacks, so the running time is $O(|R|)$.

- $C \leftarrow C \setminus L$ "pops" the marked routes from the bottom of the stacks, where appropriate. This process requires a single scan through $|R|$ stacks and at most $|R|$ pop operations, so the running time is $O(|R|)$.
- In the worst case, the above three steps repeat until $N-1$ routes are popped from the stacks, and each iteration only pops a single route. Thus, in the worst case, the running time for the algorithm is $O(N \log N + N|R|)$.

5. Route Computation with Route Reflection

A full mesh iBGP topology does not scale to large networks because a network of $|R|$ routers requires $O(|R|^2)$ iBGP sessions. Network operators use a technique called *route reflection*, which improves scalability by introducing hierarchy but makes modeling BGP path selection more complicated. First, we define an iBGP signaling topology, expound on problems introduced by route reflection, and describe constraints on iBGP configuration that must hold for modeling to be possible. Next, we propose an algorithm that efficiently computes the outcome of BGP path selection in a network with route reflection, and then we present a minor modification to the algorithm that is necessary if MED is only compared across routes from the same neighboring AS.

5.1 Problems Introduced by Route Reflection

A router does not normally forward iBGP-learned routes over other iBGP sessions, but it can be configured as a *route reflector* (RR), which forwards routes learned from one of its route-reflector clients to its other clients. The routers in an AS form a directed graph, $G = (R, S)$, of iBGP sessions called a *signaling graph*. Each edge $a = (u, v) \in S$ where $u, v \in R$ corresponds to an iBGP session between a pair of routers. We then define three classes of edges: (1) $a \in$ **down** if v is a route reflector client of u; (2) $a \in$ **up** if u is a route reflector client of v; and (3) $a \in$ **over** if u and v have a regular iBGP session between them. Figure 8 shows an example signaling graph. In a full mesh configuration, every pair of routers has an edge in **over**, and both the **up** and **down** sets are empty.

Previous work has shown that BGP converges to a stable solution as long as the structure of the signaling graph satisfies certain sufficient conditions [7]. Accordingly, we refine Constraint 2 in terms of these sufficient conditions to guarantee that an iBGP topology with route reflection converges:

CONSTRAINT 4. *(a)* $\forall\, u, v, w \in R$, $((u, v) \in$ **down** *and* $(u, w) \notin$ **down**$) \Rightarrow \lambda_u(\{\rho_v, \rho_w\}) = \rho_v$, *where* ρ_v *represents any route learned from* v *and* ρ_w *is any route from* w*; and (b) the edges in* **up** *are acyclic.*

Part (a) is satisfied when routers do not change the attributes of iBGP-learned routes and each router has a lower IGP path cost to its clients than to other routers. The common practices of applying import policies only on eBGP sessions and placing RRs and their clients in the same point-of-presence (i.e., "PoP") ensure that these conditions hold. Part (b) states that if a is an RR for b, and b is an RR for c, then c is not an RR for a, consistent with the notion of a route-reflector hierarchy (rather than an arbitrary signaling graph).

Even a route reflector configuration that converges can wreak havoc on the algorithms from Sections 3.2 and 4.2. Route reflectors hide information by advertising only a *single best route to its iBGP neighbors*. For example, in Figure 8, if W and Z have eBGP-learned routes, router Y learns a single route from its route reflector RR1. Suppose that RR1 selects the eBGP route advertised by Z. Then, Y would pick Z's route as well, even if Y would have preferred W's route over Z's route. Note that Y makes a different routing decision than it would if it could select its best route from all the eBGP routes (i.e., from both W and Z). In large networks, route reflection reduces the number of routing messages and iBGP sessions, which helps scalability, but it makes modeling BGP route selection more complicated in the following ways:

1. A router will not typically learn every route that is equally good up through the first four steps of the decision process. That is, it is possible (and likely) that some routers will not learn every route in $\gamma(B)$. In Section 5.2, we describe an algorithm that handles this case.
2. If a network uses route reflectors, *and* MED is only compared across routes from the same AS, the routes that some routers ultimately select may be *worse* than some eBGP-learned routes, according to the first four steps of the decision process. That is, it may be the case that $b_r \notin \gamma(E)$ for some router r. In Section 5.3, we make a slight modification to the algorithm in Section 5.2 to handle this case.

5.2 Algorithm: Route Reflection, No MED

Route reflection obviates the need for routers in an AS to form a full mesh topology, but it also means that some routers may not learn all routes in $\gamma(B)$. This artifact has two implications. First, the algorithm cannot simply assign non-eBGP-speaking routers the route from the "closest" eBGP-speaking router, because a router may not learn the route. In other words, applying $b_r \leftarrow \lambda_r(B)$ may not always be correct. For example, consider the network shown in Figure 9. W, X, and Y are clients of route reflector RR, and Z is a regular iBGP peer of Y. X and Y have a short IGP path between them, but they are *not* directly connected by an iBGP session. Routers W, X, and Z have eBGP routes that are equally good through the first four steps of the decision process, and have thus selected their own eBGP-learned routes. In this network, Y's closest egress point is X, but Y selects W since RR's closest egress router is W. Second, often there is *no consistent ranking of possible egress routers* from some non-eBGP-speaking router. For example, in Figure 9, RR prefers egress router W because its IGP path cost to W is the shortest. Router Y's preferences over possible egress routes depends on the presence or absence of other routes. If the AS learns routes for some destination via eBGP sessions at routers X and Z, then router Y prefers using X as an egress router. On the other hand, if the AS learned routes at W, X, and Z, then Y prefers using Z, which implies that Y prefers egress Z over X and is inconsistent with Y's choice when only X and Z are available egress routers.

To account for the fact that all routes are not visible at all routers, we design an algorithm that emulates a certain activation sequence, making route assignments at each router where possible and propagating the effects of these decisions to other routers, without ever having to revisit any assignment. This algorithm is shown in Figure 10. The algorithm first activates the routers from the bottom of

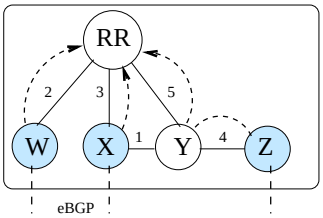

Figure 9: When an AS's iBGP topology uses route reflectors, a router may not always discover the route corresponding to its closest egress router.

Algorithm: Route Reflection, No MED

SELECTBEST_EBGP_RR(E, R)

Proceed up the hierarchy, assigning best routes.
Find a router for which all children are activated.
$A \leftarrow \phi$
while $\exists r \in R$ s.t. $r \notin A$ and $c \in A\ \forall c \in \text{DOWN}(r)$
  $I_r \leftarrow \cup_{c \in \text{DOWN}(r)} b_c$
  $b_r \leftarrow \lambda_r(I_r \cup E_r)$
  $A \leftarrow A \cup r$

Proceed down the hierarchy.
Find a router for which all parents are activated.
$A \leftarrow \phi$
while $\exists r \in R$ s.t. $r \notin A$ and $c \in A\ \forall c \in \text{UP}(r)$
  $I_r \leftarrow \cup_{c \in \text{UP}(r) \cup \text{OVER}(r)} b_c$
  $b_r \leftarrow \lambda_r(I_r \cup b_r)$
  $A \leftarrow A \cup r$

Figure 10: Efficiently computing the best route at each router in a network with route reflection but no MED.

the route reflector hierarchy upwards, which guarantees that each router selects a *down* route where possible, as required by Constraint 4(a). Because the algorithm moves upwards from the bottom of the hierarchy, it performs computations for each router as all of the routes from its clients become known; computations for these routers never need to be revisited, since, by Constraint 4, a router always prefers routes from its children over routes from its peers or parents. Visiting the routers in the *down* direction ensures that the algorithm performs computations for the remaining routers using all available routes from the **up** and **over** sets. Considering the routers in this particular ordering guarantees that no router makes a decision that should change later, after some other router makes a decision. The algorithm defines two partial orderings of the routers based on the elements of the **up** and **down** sets. We can define these two partial orderings because Constraint 4(b) requires that the signaling graph does not have any cycles of these edges, so each partial ordering must have a top and bottom element.

Applying this algorithm to the example in Figure 9, the shaded routers select best routes in the first step, since each of those routers is at the bottom of the hierarchy and, thus, all of their neighbors in **down** have been activated (since they have none). Y is activated, but it does not select a route at this point because it has no neighbors in **down**. Since these four routers are at the same level in the hierarchy, they can be activated in any order. Then RR is activated, since all of its children are activated; it applies $\lambda_{RR}(\{r_W, r_X\})$ and selects r_W because it has the smallest IGP path cost. The routers are all activated again in the downward direction; Y receives r_W from RR and compares it with r_Z, which is its best route to the destination. X and Z also receive r_W but continue to select their own route, since λ_r prefers eBGP routes over iBGP routes.

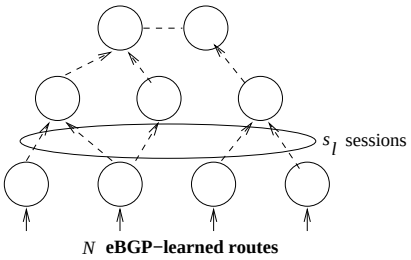

Figure 11: Running time analysis of an iBGP graph walk.

THEOREM 3. *If each router can form a total ordering over the set of all candidate routes, then the algorithm in Figure 10 correctly computes the outcome of the BGP decision process, b_r, for all routers $r \in R$.*

Proof. Assume some router r that selects a route, b_r, that is different than the route assigned by the algorithm in Figure 10, b'_r. Then, the mismatch can occur in one of two cases: (1) when b_r is learned from a session in **down**, or (2) when b_r is learned from a session not in **down** (i.e., in either **up** or **over**).

Consider the case where b_r is learned from a session in **down**. Call b'_r the first case of an incorrect computation (i.e., the algorithm has correctly computed the best route for all routers below r in the hierarchy); since we examine the first such mismatch, I_r is correct. If b'_r is also in **down**, then $b'_r = \lambda_r(I_r \cup E_r)$ when the algorithm traverses up the hierarchy, which implies that b'_r is better than b_r according to the BGP decision process, and r would have actually selected b'_r. If b'_r is in **up** or **over**, then it must have been the case that it was better, according to the BGP decision process, than the displaced route b_r in **down**. But then, by definition of λ_r, router r would have also selected b'_r in BGP. Thus, the algorithm correctly computes b_r for all routers r that select a best route from **down**.

Assume b_r is learned from a session in **up** or **over**. From the first half of the proof, we know that the algorithm correctly computes b_r for all routers that select a route from **down**, so call b'_r the first instance of a mismatch for some router that selects a best route from **up** or **over** (i.e., the algorithm correctly assigns b_r for all routers higher in the hierarchy than r). Again, because we consider the first such mismatch, we know that I_r is correct. If the route that the algorithm selects, b'_r, is in **down**, then, by Constraint 4(a), BGP could not have selected b_r, so we have a contradiction. If both b_r and b'_r are learned from sessions in **up** and **over**, then both are in I_r, and, according to the $\lambda_r(I_r \cup b_r)$ step in the algorithm and by definition of λ_r, both the algorithm and the BGP decision process would select the same route. □

This theorem relates to one from earlier work [11] on sufficient conditions for stable BGP routing *at the AS level*; this work provides a constructive proof showing that the sufficient conditions guarantee stability. In subsequent work, Griffin *et al.* discovered that the sufficient conditions for stable eBGP routing were analogous to those for stable iBGP routing with route reflection [7]. The algorithm from this section applies the iBGP analog of the constructive proof from the work on stable interdomain routing to develop an algorithm for *computing* that stable path assignment.

Computational Complexity. This route computation involves traversing the route reflector hierarchy exactly twice. The running time of this algorithm is $O(N + |S|)$, where N is the number of

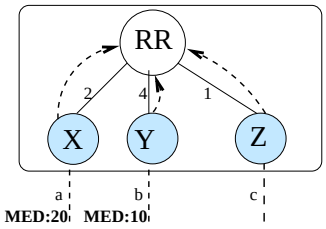

Figure 12: When an AS's iBGP topology uses route reflectors and MED, a router may not always select a route in $\gamma(E)$.

eBGP-learned routes, and $|S|$ is the number of iBGP sessions. To see why this is the case, consider the l-level route reflector hierarchy pictured in Figure 11. Starting from the bottom of the hierarchy, the algorithm must perform comparisons over N routes to determine the routes that the M routers at the bottom of the hierarchy select (the number of routers at the bottom of the hierarchy is inconsequential: these comparisons can be performed by constructing a subset of M routes from the original N routes, which can be performed in a single scan of the N routes). The algorithm them propagates the selection of these M routes to the next level of the hierarchy, where s_l comparisons must be performed across the routers at the next highest level, where s_l is the number of iBGP sessions at level l. Repeating this process up the hierarchy yields a total running time of $O(N + |S|)$.

Recall from Section 3.2 that the running time for route computation in the case of full-mesh iBGP, was $O(N + |R|^2)$, or $O(N + |S|)$. Note that the algorithm for the case with route reflection has the *same* running time complexity as before; the running time for computing the outcome of BGP route selection is no more complex, even though the process for computing the outcome is more involved. In an iBGP topology with route reflection, the number of sessions, $|S|$, will actually be less than $|R|^2$; the running time of the algorithm in this section benefits from the fact that route reflectors reduce the number of sessions in the iBGP topology.

5.3 Algorithm: Route Reflection, MED

When a network uses both route reflection and MED, the graph walk algorithm in Figure 10 no longer works, because it relies on the fact that the routers that all routers will ultimately select a route in $\gamma(E)$. In a network with route reflection *and* MED, this is not always true, because when a router selects a locally best route, a route with a lower MED value might not be visible to that router. As a result, some router in the AS might select an eBGP-learned route that is *worse*, according to the first four steps of the BGP decision process, than eBGP-learned routes selected by other routers!

Consider the example shown in Figure 12. The network learns routes to some destination at routers X, Y, and Z that are equally good up to MED comparison. All three routers are clients of the route reflector RR. The routes at X and Y are learned from the same next-hop AS, and r_Y has a lower MED value. One might think that router X would never select route a, since, after all, it has a higher MED value than route b, but that is not the case in this figure: RR learns routes a, b, and c, and selects route c as its best route, because c has the shortest IGP path cost. Therefore, X never learns route b!

Note that applying the algorithm from Figure 10 does not correctly compute the outcome of the BGP decision process. Proceeding up the hierarchy: (1) routers X, Y, and Z would select routes a, b, and c, respectively; (2) RR selects route c because it has a shortest IGP path; (3) each router selects its own eBGP-learned route, since eBGP routes are preferred over iBGP routes. While the graph walk correctly computes the outcome in this case, a different IGP graph would produce a different result: if RR's IGP path cost to Y were less than that to Z, then X would learn routes a and b, rather than a and c, and it would ultimately select b, because of its lower MED value. On the other hand, if X had learned a second eBGP-learned route, d, that was *better* than b, but worse than a, then X would ultimately select d, which is an eBGP-learned route but *not* its locally best eBGP route.

To account for this, we apply the result from Lemma 3 to derive the algorithm in Figure 13. This algorithm also immediately eliminates all candidate routes that have a lower local preference, longer AS path, or higher origin type than other eBGP-learned routes, as in the algorithms from Figures 5 and 6. The algorithm repeatedly applies the graph walk from Figure 13 and *eliminates* eBGP-learned routes from the set of candidate routes, C, when they are worse, according to MED, than some other route learned as a result of the graph walk. Similarly to the algorithm in Figure 6, the algorithm in Figure 13 can be equated with iteratively restarting the activation sequence after eliminating routes from the system.

THEOREM 4. *If a network uses route reflection,* and *routers cannot form a total ordering over all candidate routes, then the algorithm in Figure 13 correctly computes the outcome of the BGP decision process, b_r, for all routers $r \in R$.*

Proof. Consider the activation sequence where (1) all routers select their locally best eBGP learned route and readvertise these routes via iBGP; (2) every router compares their eBGP-learned routes with routes learned via iBGP.

From Theorem 3, we know that one iteration of the loop in Figure 13 correctly computes the result of one iteration of this activation sequence. From Lemma 3, it is possible to eliminate any locally best eBGP-learned routes that are eliminated by other routes in B_i, since those routes will never appear in a B_j for any $j > i$, and restart the activation sequence with the smaller set of candidate routes. The termination condition, $L = \phi$, corresponds to the condition that no router's best route changes as a result of applying the algorithm from Figure 10, which will also result in no new iBGP messages being sent in the activation sequence. □

Computational Complexity. Sorting a total of N routes locally at each router (to allow for the application of λ_r) has $O(N \log N)$ running time. Each iteration has the following complexity:

1. Once B has been computed, executing the algorithm from Figure 10 has complexity $O(|S|)$.
2. Computing the set L and eliminating those routes from the candidate set requires a comparison at each router, which has complexity $O(|R|)$.

This loop executes a maximum of $N - 1$ times, since, in the worst case, one route is eliminated at each iteration and the ultimate set B has only a single route. Therefore, the total running time is $O(N \log N + N|R| + N|S|))$.

6. Discussion: Proposed Improvements to BGP

Thus far, this paper has focused on modeling BGP route selection inside a single AS. Notably, two artifacts, the MED attribute and route reflection, complicate this modeling. Not only do these attributes make modeling difficult, *they also create problems with the operation of BGP itself*. The use of MED, both with and without route reflection has been shown to cause oscillation [12]; route reflection can also prevent convergence and cause forwarding loops [7]. The MED attribute is intended to allow a neighboring AS

Algorithm: Route Reflection, MED

```
SELECTBEST_EBGP_MED_RR(E, R)
    C ← σ(E)
    do
        B ← ∪_r λ_r(C_r)
        SELECTBEST_EBGP_RR(B, R)
        L ← ∪_{r|b_r≠λ_r(E_r)} λ_r(E_r)
        C ← C \ L
    while L ≠ φ
```

Figure 13: Efficiently computing the best route at each router in a network with route reflection, where MED is only compared across routes from the same AS.

to dictate preferred exit points on routes advertised at multiple exit points, but it prevents a router from forming a consistent ordering of preferences over routes. Route reflectors were introduced to allow an iBGP topology to scale, but they do so in a way that prevents routers from having full visibility of eBGP-learned routes. In this section, we explore possible solutions to the problems introduced by MED and route reflection.

6.1 MED-ication for Late-Exit Semantics

The MED attribute causes problems because it is not comparable across routes from different neighboring ASes, which prevents a router from producing a consistent total ordering over all possible routes. Also, note that in networks without route reflection, inconsistent preferences between pairs of routes is based on the router ID attribute, an arbitrary tiebreak that carries no meaningful semantics (as in Figure 4, for example).

Before we consider solutions to the problems introduced by MED, it is worth noting that *MED, as it operates today, does not satisfy late-exit semantics when used with route reflection*. Consider the example shown in Figure 12. A neighboring AS sending routes a and b with MED values 10 and 20, respectively, expects that the AS shown would always prefer route a over route b, as long as both existed, causing router X to perform cold-potato routing (i.e., send its traffic via route b via router Y). Unfortunately, the AS shown will *not* do so: RR prefers route c, so router X will never learn route b, and it will continue to forward packets via route a. In other words, setting MEDs may have no effect whatsoever on route selection!

Recognizing the root causes of the problems with MED allowed us to make the following realization: if MED values are *remapped* into an explicit ranking (i.e., 1st, 2nd, etc.), rather than arbitrary values, then MED *can* be compared across all routes with impunity. Comparing an exit-rank across all routes can sometimes result in different outcomes than those of BGP today, but in many cases the differences do not affect the important semantics of BGP. For example, consider Figure 4, but where the MED attribute is compared across all routes. Suppose that AS 2's MED values of 10 and 20 are remapped to 1 and 2, and that the highest MED value of any eBGP-learned route, 2, is added to the MED value on every route learned via iBGP (this transformation guarantees that eBGP-learned routes are still preferred over iBGP-learned routes). In this case, routers X and Y would ultimately select routes c and d, respectively, as opposed to a and d in BGP today. Although X selects c instead of a, its preference between these two routes was based on the arbitrary router ID tiebreak; therefore, having router X select c instead does not destroy any meaningful semantics.

The type of remapping we have described would preserve late-exit semantics (in fact, as opposed to the way MED works today, it would actually *respect* late-exit semantics), but implementing an exit-rank requires visibility into the set of available routes that is not available today. Unfortunately, MED values are typically based on dynamic values (e.g., IGP path costs across the network), so an AS that sends MED cannot simply configure a static ranking. Given today's architectures, neither the sending nor receiving AS could perform a remapping of MED values into an exit-rank, since no single router learns the complete set of routes advertised from a neighboring AS. Performing such a remapping would require either the sending or receiving AS to have complete visibility over all routes being sent or received for a destination. On the other hand, the Routing Control Platform (RCP) [13] or similar recently proposed architectures [14] can perform such a remapping, since RCP has full visibility of routes sent from a neighboring AS (as well as full control over the routes that it sends to a neighboring AS).

This modification would eliminate intra-AS oscillation; furthermore, it facilitates modeling BGP route selection: the algorithm from Figure 3 would be sufficient to compute the outcome of BGP route selection; this algorithm is significantly more efficient and less complex than the algorithm in Figure 13.

6.2 Scalability without Route Reflection

Route reflectors allow iBGP topologies to scale to large number of routers because they obviate the need to have a "full mesh" topology with $O(|R|^2)$ sessions. Unfortunately, they restrict route visibility because they only send a single best route, from all of the routes they have learned. In this paper, we have explained how this restriction complicates the modeling of BGP path selection; previous work has also noted that it can cause persistent oscillation and forwarding loops [7, 12].

To remedy the problems with persistent oscillation, Basu *et al.* proposed that route reflectors forward *all routes* that are equally good up to and including the MED comparison. It turns out that this modification correctly emulates a full mesh iBGP topology; thus, it is possible to model the outcome of their modified protocol with the algorithm from Figure 6. Unfortunately, this proposal requires modifications to the routers, since each router readvertises multiple routes instead of a single best route. Additionally, because each router readvertises multiple routes to its neighboring routers, every router must select routes using a *consistent* selection criterion. Otherwise, given multiple routes, some router along the path to an exit point might select a different route, causing a deflection. This restriction precludes certain policies and configurations (e.g., a router may not manipulate attributes on a route learned via iBGP).

Architectures such as RCP propose separating route computation from the routers and placing this functionality in a system that computes routes on behalf of all of the routers within an AS [13]. Rather than returning only a single best route to all of its clients (as a route reflector does), RCP advertises to each router *the route that it would have selected in a full mesh iBGP configuration*. This architecture allows the network to scale in the same way that route reflectors do, but it provides some important additional advantages. First, because the RCP explicitly assigns routes to all routers in the network, it can *guarantee* that the assignments are free from deflections and forwarding loops. Second, RCP allows for a more scalable network design. Because RCP is not on the forwarding path, it does not have to make the same routing decisions as its clients (as route reflectors do today). As a result, unlike route reflectors, RCP servers can be replicated at arbitrary places in the IGP topology.

7. Related Work

Previous work presented an IGP emulator that helps network operators optimize link weights for intradomain traffic engineering [15], but this emulator does not model changes to BGP routing policies or the effects of iBGP on path selection. There has also

been much focus on modeling BGP *convergence* [9, 16, 11], but this is the first paper to model BGP *route selection*.

Recent work proposes efficient techniques for large-scale parameter optimization for various network protocols, including the tuning of the local preference attribute in BGP [17]. This work is complementary to ours—the proposed search techniques could use our emulator as the "inner loop". These techniques currently use simulators such as SSFNet [2], but they only depend on the outcome of BGP path selection (not on dynamics) and would likely benefit from having an efficient, accurate emulator as an inner loop.

The BGP model in this paper applies several previous theoretical results in new ways. The constraints for iBGP configuration that we present in Section 2 are motivated by previously-derived sufficient conditions for iBGP to guarantee that the routing protocols converge to a stable assignment [7, 18]. This work specified these conditions to ensure correct routing behavior, but these constraints are also required to *model* BGP routing. The route prediction algorithm in Section 4.2 also uses results from previous work. We applied a constructive proof regarding stable inter-AS policy configurations [11] to iBGP configuration and used this proof as the basis for the third phase of the algorithm.

In previous work, we explored practical traffic engineering techniques in BGP; we assumed the existence of a BGP emulator for testing traffic engineering techniques offline [1]. We previously presented a model that *accurately* and *efficiently* predicts the outcome of the BGP route selection process in a single AS using only a snapshot of the network configuration and the eBGP-learned routes from neighboring domains, without simulating protocol dynamics [4]. We implemented an emulator based on this model to demonstrate that our algorithm is accurate and efficient enough to be used in practice for many network engineering tasks. This paper extends that work by: designing an algorithm that models BGP path selection in networks that use *both* MEDs and route reflection; formally presenting the proofs, algorithms, and running time complexity for various cases of network configuration; formalizing the complexity introduced by MED and route reflection; and proposing protocol improvements that achieve the goals of MED and route reflection reduce modeling complexity, and prevent undesirable side effects (e.g., oscillations).

8. Conclusion

To perform everyday network engineering tasks effectively, efficiently, and with minimal unnecessary changes to the live network, operators need a way to model how a routing protocol configuration will behave before deploying that configuration. The model we have presented is a necessary step for advancing the state of the art of network engineering. We believe that our model and BGP emulation tool present several immediate possibilities for future work. First, network-wide BGP route prediction could be combined with traffic measurements to help network operators select BGP configuration changes that achieve various traffic engineering tasks. Second, the emulator could be combined with higher-level mechanisms that spot misconfiguration or check that other constraints, such as robustness, are satisfied [19].

Finally, we note that modeling BGP routing is much more difficult than it should be. In the future, we hope that routing protocol designers will consider ease of modeling as a design goal; as we describe in Section 6, some of these simplifications that aid protocol modeling also fix problems with protocol *operation*. Routing protocols that are easy to model and reason about will make everyday network engineering tasks more tractable.

Acknowledgments

We thank Ramesh Johari and Renata Teixeira for helpful comments and suggestions.

9. References

[1] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 33, October 2003.

[2] "SSFNet." `http://www.ssfnet.org/`, 2003.

[3] "How BGP Routers Use the Multi-Exit Discriminator for Best Path Selection." `http://www.cisco.com/warp/public/459/37.html`.

[4] N. Feamster, J. Winick, and J. Rexford, "A model of BGP routing for network engineering," in *Proc. ACM SIGMETRICS*, June 2004.

[5] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of Internet stability and wide-area network failures," in *Proc. Fault Tolerant Computing Symposium*, June 1999.

[6] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *Proc. Internet Measurement Workshop*, November 2002.

[7] T. G. Griffin and G. Wilfong, "On the correctness of IBGP configuration," in *Proc. ACM SIGCOMM*, August 2002.

[8] C. Villamizar, R. Chandra, and R. Govindan, "BGP Route Flap Damping." Request for Comments 2439, November 1998.

[9] T. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Networking*, vol. 10, pp. 232–243, April 2002.

[10] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Trans. Networking*, vol. 9, June 2001.

[11] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Trans. Networking*, vol. 9, pp. 681–692, December 2001.

[12] A. Basu, A. Rasala, C.-H. L. Ong, F. B. Shepherd, and G. Wilfong, "Route oscillations in I-BGP with route reflection," in *Proc. ACM SIGCOMM*, August 2002.

[13] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merve, "The case for separating routing from routers," *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2004.

[14] O. Bonaventure, S. Uhlig, and B. Quoitin, "The case for more versatile BGP route reflectors." Internet Draft draft-bonaventure-bgp-route-reflectors-00.txt, July 2004.

[15] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks," *IEEE Network Magazine*, pp. 11–19, March 2000.

[16] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," *IEEE/ACM Trans. Networking*, vol. 9, pp. 293–306, June 2001.

[17] T. Ye, H. T. Kaur, and S. Kalyanaraman, "A recursive random search algorithm for large-scale network parameter configuration," in *Proc. ACM SIGMETRICS*, June 2003.

[18] T. G. Griffin and G. Wilfong, "Analysis of the MED oscillation problem in BGP," in *Proc. International Conference on Network Protocols*, November 2002.

[19] N. Feamster and H. Balakrishnan, "Verifying the correctness of wide-area Internet routing," Tech. Rep. MIT-LCS-TR-948, Massachusetts Institute of Technology, May 2004.

APPENDIX

A. Prototype Implementation

In this section, we describe a prototype that incorporates the model of BGP path selection described in this paper. Our current prototype separates the computation of egress routers for a given destination from the assignment of other routers to those egress routers. *This separation of functionality requires that* $b_r \in \gamma(E)$*, which does not hold when* both *MED and route reflection are used (as shown in Section 5.3).* We refer readers to our previous work, which discusses the design, implementation, and evaluation of the prototype implementation in more detail [4].

A.1 Design Overview

We now highlight the high-level design of the prototype, shown in Figure 14. We briefly describe: the necessary inputs for driving the prototype, the decomposition of functionality into three distinct modules and the relationships of those modules to the algorithms described in this paper, and optimizations that reduce computational complexity.

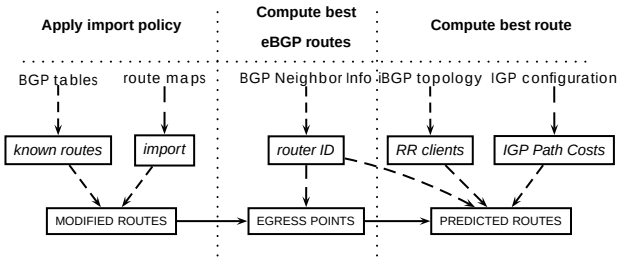

Figure 14: Data flow in the prototype. Fonts specify raw inputs**,** *input tables***, and** DERIVED TABLES**. In practice, operators might collect raw inputs once a day.**

Input data. The prototype uses three inputs:

- *BGP routing tables:* The BGP tables for the eBGP-speaking routers provide the first stage of the algorithm with a snapshot of the routes advertised by neighboring ASes. We ignore the router's current view of the best route and the current setting of the local preference attribute, since these relate to the existing network configuration rather than the scenarios we might want to emulate.
- *Router configuration files:* The configuration files are used to (1) determine the import policies (i.e., route maps) for each eBGP session, (2) determine the iBGP signaling graph, and (3) compute the IGP path costs between each pair of routers. The import policies are used to manipulate attributes of the eBGP routes in the first stage of the algorithm, and the iBGP and IGP information are needed for the third stage.
- *BGP neighbor information:* Because the BGP decision process depends on the router ID associated with the BGP session announcing the route, our algorithms require knowing the router ID associated with each BGP session. The second stage uses the router IDs of the *eBGP* sessions, while the third stage uses the router IDs for the *iBGP* sessions.

We emphasize several points with regard to the input data. First, a network operator can capture all of the necessary data with telnet or ssh access to each router. Second, many aspects of the input data do not change very often; as such, the prototype is useful even if all of the input data is collected infrequently (e.g., once a day). Finally, because certain inputs can be approximated (e.g., router ID is typically the loopback IP address of the router), the prototype can be effective even with limited input.

Prototype operation. The prototype uses a database back-end, which provides efficient access to small subsets of the configuration data and routes and also stores intermediate results, which allow us to validate each part of the algorithm separately. Figure 14 summarizes how the prototype uses the inputs and intermediate results to generate a table of predicted routes. The three modules shown in Figure 2 correspond to the first two stages from Section 2.2; assuming that $b_r \in \gamma(E)$ allows us to break the second stage into two simpler modules. The prototype performs three operations:

Applying import policy to eBGP-learned routes: This operation corresponds to the first step described in Section 2.2. Each row of the *import* table specifies how a particular set of rows in the *known routes* table should be modified; the prototype performs the actual modifications on the MODIFIED ROUTES table. For each row in the *import* table, the first operation applies the policy by (1) finding the appropriate routes by selecting the set of routes learned at the corresponding router on that BGP session that match the specified AS path regular expression and (2) setting the other attributes (e.g., local preference) according to the values specified in that row of the *import* table.

Computing the egress routers for a destination: This operation generates the set of the set of best eBGP-learned routes B using the technique from Section 4.2, corresponding to the first half of stage 2 in Section 2.2. This part of the algorithm performs "select" statements on the MODIFIED ROUTES table to successively refine the set of candidate routes. The *router ID* table contains the router ID for every BGP session at each router, which is needed for step 7 of the decision process. As the method from Section 3 marks "best" routes, these routes are inserted into the EGRESS POINTS table for use by the third operation.

Computing the predicted routes: This operation uses the iBGP signaling graph, IGP path costs, and technique from Section 5.2 to determine the best BGP route for each prefix at each router. The module uses the iBGP signaling graph to determine which routes are advertised to each router, the IGP path costs between each pair of routers to determine the closest eBGP-speaking router to each ingress router (used in step 6 of the decision process), and the router ID of each iBGP session (step 7) to determine the egress router that each ingress router will select. The *RR clients* table represents the iBGP signaling graph and *IGP path costs* represents the shortest IGP path between each pair of routers in the AS. Each row of *RR clients* specifies a route reflector client for a particular cluster; this provides the partial ordering needed by the algorithm. When applying the IGP tiebreaking step at an ingress router, *IGP path costs* is used to determine the egress router with the shortest IGP path.

Optimizations. To ensure that the prototype operates on reasonable timescales, even with a large number of routes and eBGP sessions, we made the following optmizations: (1) as many routes have the same AS path attribute, store the AS paths in a separate table to accelerate lookups based on AS path regular expressions; (2) as many prefixes are advertised in exactly the same manner (i.e., at the same set of exit routers and with the same attributes), execute route computation only once for each *group of prefixes*; and (3) upon an incremental policy change, only recompute the routes for prefixes affected by that change.

A.2 Evaluation

The analysis focuses on a snapshot of the network state from early morning (EST) on February 4, 2003. We validate the prediction algorithm for the 91,554 prefixes whose eBGP routes are

learned at peering points to other large providers, since we have routing tables from all of these locations; we excluded prefixes that were learned at other routers. (Recall that the prediction algorithm relies on knowing all of the potential egress routers where routes to a prefix are learned.) The initial BGP routing data consists of 1,620,061 eBGP-learned routes with 43,434 distinct AS paths. We apply the tool to these inputs and check whether the emulator produces the same answers that the operational routers selected. In addition to collecting BGP routing tables from the peering routers (where the eBGP routes are learned), we also collect BGP tables from several route reflectors and access routers to verify the results.

Performance evaluation. We ran the prototype on a Sun Fire 15000 with 192 GB of RAM and 48 900 MHz Ultrasparc-III Copper processors. Because this is a time-shared machine, we ran each of our experiments several times to ensure the accuracy of our measurements. Except where noted, the prototype used only two 900 MHz processors (one for the database process and one for the emulator itself); the combined memory footprint of the database process and the emulator never exceeded 50 MB. Because the emulator did not use more resources than a standard PC, the results of our evaluation should reasonably reflect the emulator's performance on commodity hardware.

While our evaluation is preliminary, our performance tests demonstrate that the prototype can operate on timescales that could allow an operator to use a BGP prototype based on our algorithms in a practical setting. Our evaluation demonstrates the following points:

- The prototype computes the best routes for *one prefix* throughout a large tier-1 ISP network in about one second. Although predicting the best route for *all* prefixes at all routers in such a network takes several hours, this type of computation does not need to be performed all that frequently in practice.
- Exploiting commonalities among route advertisements to eliminate redundant computation reduces the running time of the prototype by approximately 50%.
- Evaluating the effects of an incremental change to router configuration typically takes only a few seconds.

Validation. To verify that the prototype produces correct answers, we perform validation using complete routing protocol implementations on production routers in a large operational network. We performed independent validation for each of the three modules, as well as an end-to-end validation to study the effect of error propagation on the best routes ultimately predicted by the prototype. We summarize the results of the end-to-end evaluation here.

We compared the prototype's computation with the same four routing tables used for the validation of the third module, with the exception that the input included the errors from the first two modules. At these four routers, the prototype predicted the correct routes for more than 99% of all prefixes, as summarized in Table 4. Prediction errors are infrequent and result mainly the dynamics of

Router	*# Predictions*	*Total Errors*
RR1	89,343	554 (0.620%)
RR2	88,647	394 (0.444%)
AR1	88,649	391 (0.441%)
AR2	76,733	511 (0.666%)

Table 4: Summary of errors for end-to-end validation.

the inputs. Since most prefixes whose routes change frequently do not receive much traffic [6], these inconsistencies would be permissible for most traffic engineering tasks.