Interdomain Ingress Traffic Engineering Through Optimized AS-Path Prepending

Ruomei Gao, Constantinos Dovrolis, and Ellen W. Zegura

College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332[⋆]
{gaorm, dovrolis, ewz}@cc.gatech.edu

Abstract. In INterdomain Ingress Traffic Engineering (INITE), a "target" Autonomous System (AS) aims to control the ingress link at which the traffic of one or more upstream source networks enters that AS. In practice, ISPs often manipulate, mostly in a trial-and-error manner, the length of the AS-Path attribute of upstream routes through a simple technique known as prepending (or padding). In this paper, we focus on prepending and propose a polynomial-time algorithm (referred to as OPV) that determines the optimal padding for an advertised route at each ingress link of the target network. Specifically, given a set of "elephant" source networks and some maximum load constraints on the ingress links of the target AS, OPV determines the minimum padding at each ingress link so that the load constraints are met, when it is feasible to do so. OPV requires as input an AS-Path length estimate from each source network to each ingress link. We describe how to estimate this matrix, leveraging the BGP Looking Glass Servers. To deal with unavoidable inaccuracies in the AS-Path length estimates, and also to compensate for the generally unknown BGP tie-breaking process in upstream networks, we also develop a robust variation (RPV) of the OPV algorithm.

1 Introduction

Traffic Engineering (TE) is broadly divided into two types: intradomain and interdomain. In intradomain TE, the operator of an autonomous system (AS) controls the flow of traffic within that network by optimizing the link costs of the corresponding routing protocol (mostly OSPF or IS-IS) or through dynamic provisioning of virtual circuits (e.g., MPLS). For previous work in intradomain TE, we refer the reader to the survey [1] and the references therein. Intradomain TE assumes that the ingress and egress links of interdomain traffic flows are given as inputs in the form of a traffic matrix, and they cannot be manipulated.

Interdomain TE, on the other hand, aims to control exactly those ingress and egress flows. Let us consider a "target" AS, referred to as $\mathcal{T}$. $\mathcal{T}$ has a number of ingress links, receiving traffic from upstream ASes. If $\mathcal{T}$ is not a stub network, traffic that is destined to one of that network's customers eventually leaves $\mathcal{T}$ through an egress link. Controlling the egress link that the traffic will flow through is referred to as *Interdomain Egress*

[⋆] This work was supported by the NSF CAREER award ANIR-0347374 and by an equipment donation from Intel Corporation.

TE. On the other hand, controlling the ingress link through which the traffic of a source network will enter $\mathcal{T}$ is referred to as *INterdomain Ingress TE (INITE)*.

If one compares the maturity, in terms of both operations and research, between INITE and other types of TE, the former is much less deployed, understood, and trusted [2, 3]. The high-level reason is that INITE requires that the target network $\mathcal{T}$ has a way to influence BGP routing decisions in upstream ASes, without requiring the active cooperation of those networks. In more detail, to perform INITE the operator of $\mathcal{T}$ would face the following three major unknowns about the *upstream cloud*, i.e., the part of the Internet between a source network and the ingress links of $\mathcal{T}$. First, the *BGP policies*, expressed through BGP attributes (e.g., Local Preference) and upstream ingress/egress routing filters. Second, the actual AS-level topology and in particular, the *AS-Path lengths* from any upstream network to the ingress links of $\mathcal{T}$. Third, the *BGP tie-breaking behavior*, referring to the way a BGP peer selects the best route among a set of candidate routes with the same Local Preference and AS-Path length [4]. In addition to the previous unknowns, one has to include the more common challenges of any TE problem, including variations in the traffic loads, unexpected infrastructure events (e.g., router crashes), and so forth.

Given the previous difficulties, it is not surprising that some ISPs avoid INITE. One form of INITE that is, however, used by many ISPs is that of *AS-Path prepending*, or simply "prepending" (also known as *AS-Path padding*). Specifically, $\mathcal{T}$ can make a route less attractive to its upstream ASes by increasing the AS-Path length of that route by adding several instances of its own AS-Number to that attribute. A recent measurement study showed that 32% of the routes in the AT&T network have some form of prepending, with about 90% of the corresponding paths extended by 1-5 hops [3]. Prepending is often used for load balancing and for provisioning of backup routes. In the former, operators increase the degree of padding at an advertised route in a trial-and-error basis until the AS-Path is long enough to reduce the load of that ingress link by a certain amount. In the latter, the degree of padding is sufficiently large so that the route is not used unless if there is a failure in the primary route(s).

Our objective is to investigate the prepending technique more thoroughly, and understand its potential and limitations. The specific problem that we consider, in its basic form, is the following. Suppose that a target network $\mathcal{T}$ has N ingress links. For simplicity, we will assume that $\mathcal{T}$ is a multihomed stub network. The case in which $\mathcal{T}$ is a transit network, performing INITE for its customers, is considered in [5]. $\mathcal{T}$ receives most of its traffic from a set of M "elephant" sources, and an estimate of each source load s_i is given (i=1 $\ldots M$). $\mathcal{T}$ aims to impose a *maximum load* (maxload) constraint C_j at each ingress link j for the traffic that arrives from the M sources. To do so, $\mathcal{T}$ increases the AS-Path length of its advertised routes at ingress link j by prepending its own AS-Number $a_j \geq 0$ times. The main questions then are: *what is the value of a_j for each link j that will meet the given maxload constraints, and when is it infeasible to meet these constraints through prepending?* In particular, we are interested in the *optimal prepending*, i.e., the padding vector A that minimizes the sum $\sum a_j$. We refer to the previous as the *Constrained Optimal Prepending (COP)* problem. COP, despite its simple statement, captures an important objective of multihomed ASes, that of balancing the ingress load across a set of links, and it attempts to leverage a currently ad-hoc technique (prepending) in a more systematic methodology.

The first contribution of this paper is to develop a polynomial-time algorithm, referred to as *OPV* (for Optimal Padding Vector), which solves the COP problem. The algorithm is very simple: at each iteration, an overloaded ingress link is chosen and its padding is incremented. Then, given the new padding vector, the mapping from sources to ingress links is recomputed, and the algorithm moves to the next iteration. A key input to the OPV algorithm is the *AS-Path length matrix* P. Each element $P_{i,j}$ of this matrix is an estimate of the shortest AS-Path length from source network $\mathcal{S}_i$ to ingress link $\mathcal{L}_j$. The second contribution of the paper is to present four estimation techniques for P, leveraging the abundant Looking Glass Servers present in the Internet today, and to evaluate the accuracy of these techniques.

The errors of the AS-Path length estimation are not negligible however. To deal with errors in P, our third contribution is to develop the *Robust Padding Vector (RPV)* algorithm, a heuristic built on top of OPV. The objective of RPV is to determine a padding vector that will likely satisfy the given maxload constraints, even if the input matrix P has inaccurate elements and the BGP tie-breaking behavior is unknown in the upstream cloud. Simulation results show that, with the observed empirical error distribution in P, and the completely unknown tie-breaking behavior, RPV can still find a padding vector that satisfies the maxload constraints in more than 90% of the cases, as long as such a padding vector exists.

Related Work. A survey for interdomain TE is [2]. The feasibility of interdomain TE by a stub AS was examined in [6] and more recently in [7]. A simulation-based study of the effectiveness of AS-Path prepending for INITE has been published in [8]. The use of the BGP community attribute for INITE has been studied in [9]. The Internet Draft [10] proposed another BGP community-based solution. More recently, [11] proposed a tunneling-based mechanism for INITE, assuming the cooperation of the remote source networks. Instead of adding extensions to BGP, Agarwal et al. proposed an Overlay Policy Control Architecture (OPCA) [12]. Feamster et al. focused on egress interdomain TE [3]. Uhlig et al. also focused on egress interdomain TE in [13]. Bressoud and Rastogi examined the intradomain problem of choosing the optimal set of border routers for the advertisement of a set of routes from a transit provider [14].

Paper Structure. The COP problem is formally stated in §2, In §3, we propose and study the OPV algorithm, proving that it finds the optimal padding vector in polynomial time. In §4, we describe how to estimate the AS-Path length matrix and evaluate the accuracy of the proposed techniques. In §5, we present the RPV algorithm and evaluate its robustness through simulations. We conclude in §6.

2 Problem Statement and Formulation

Consider a target network $\mathcal{T}$ that aims to do INITE through prepending. INITE can be performed separately for each major customer of $\mathcal{T}$, if the latter is a transit network. Or, if $\mathcal{T}$ is a multihomed stub network, INITE can be performed for all the ingress traffic to $\mathcal{T}$. Suppose that $\mathcal{T}$ has N ingress links $\{\mathcal{L}_1, \ldots, \mathcal{L}_\mathcal{N}\}$, each of which advertises the

Table 1. Notation

$\mathcal{T}$	target network	M	number of sources
$\mathcal{S}$	source network vector	N	number of ingress links
$\mathcal{L}$	ingress link vector	$\mathcal{U}_i$	upstream tree for src i
S	source load vector	C	link maxload vector
P	AS-Path length matrix	Q	tie-breaking matrix
A	padding vector	$P(A)$	padded length matrix
$L(A)$	link assignment vector	$R(A)$	link load vector

Table 2. BGP route selection

1. Higher local preference
2. Shorter AS path
3. Lower origin type
4. Lower MED value
5. E-BGP over I-BGP routes
6. Lower IGP metric to next-hop
7. Lower BGP router ID

routes that originate at $\mathcal{T}$ to its upstream network through BGP. The ingress traffic for $\mathcal{T}$ may be produced by a potentially large number of upstream source networks. Instead of considering individual source networks, that may be impractical, we focus on *super-source ASes*. The latter may be an aggregation of several source networks that generate a relatively large portion of the total ingress traffic to $\mathcal{T}$. In the following, we refer to super-source ASes simply as "source networks". Suppose $\mathcal{T}$ has M source networks, and the average traffic loads from the M source networks are represented by the *source load vector* $S = \{s_1, \ldots s_M\}$.

The effectiveness of any AS-Path prepending technique, including ours, is limited by the use of local routing policies expressed through the Local Preference attribute. The reason is that the Local Preference attribute has a higher priority in the BGP path selection process than the AS-Path length (see Table 2). Consider a source network i, and suppose that i maintains a distinct route to each ingress link $\mathcal{L}_j$ of $\mathcal{T}$. This can be achieved if $\mathcal{T}$ advertises a unique *wayfinding prefix* at each link $\mathcal{L}_j$ (we return to this point in § 4). The selected routes from source i to the N ingress links of $\mathcal{T}$ form a directed acyclic graph (DAG) of AS links denoted by $\mathcal{U}_i$. In the following, we assume that the branching nodes of this DAG select their best routes to the N ingress links only based on the AS-Path length. In other words, the candidate routes for $\mathcal{T}$ at each branching node of $\mathcal{U}_i$ are assigned the same Local Preference value.

The BGP tie-breaking behavior is far from simple. As shown in Table 2, when two routes have the same Local Preference and AS-Path length, an ordered list of five other criteria is used to choose the best route. We do not attempt to estimate or infer all the parameters that affect those tie-breaking criteria; such a task would be extremely difficult and error prone. We assume, however, that the tie-breaking behavior is determined by a matrix Q. The i'th row of Q refers to the tree that connects the i'th source network to the N ingress links of $\mathcal{T}$. If two routes that originated from links $\mathcal{L}_j$ and $\mathcal{L}_k$ are received with equal AS-Path lengths by an AS in the upstream tree $\mathcal{U}_i$, then that AS will choose the route to link $\mathcal{L}_j$ if $Q_{i,j} < Q_{i,k}$; otherwise, it will choose the route to link $\mathcal{L}_k$. Different columns of the same row of Q must be different, while their absolute values do not matter. Notice that Q is just a model of the BGP tie-breaking behavior; it is not related to a real BGP attribute. Furthermore, Q represents a *globally consistent tie-breaking behavior*, i.e., we assume that a tie between two routes to $\mathcal{T}$ with the same AS-Path length is broken in the same way in any AS in an upstream tree $\mathcal{U}_i$.

Another key parameter is the *AS-Path length matrix* P. The element $P_{i,j}$ is the length of the shortest AS-Path from the source network $\mathcal{S}_i$ to the ingress link $\mathcal{L}_j$, where

$P_{i,j}$ is a positive integer. Any ties in the length of the shortest AS-Path are broken based on the Q matrix. Note that $P_{i,j}$ is the "original" AS-Path length, i.e., it should be measured before the target network applies any prepending. Estimation techniques for the matrix P are given in § 4. For now we assume that P is accurately known.

The ingress link $\mathcal{L}_j$ can increase the length of the AS-Path attribute by a_j, where a_j is a non-negative integer, through prepending. Given a padding vector A=$\{a_j, j = 1 \ldots N\}$, the "padded" AS-Path length of the route to link $\mathcal{L}_j$ is $P_{i,j}(A) = P_{i,j} + a_j$[1].

Based on the previous model, we can now show the following.

Lemma 1. *Given a padded AS-Path length matrix $P(A)$ and a tie-breaking matrix Q, the traffic of a source network $\mathcal{S}_i$ will enter the target network $\mathcal{T}$ through the ingress link $\mathcal{L}_j$, where*

$$j = \arg \min_{1 \leqslant l \leqslant N} P_{i,l}(A) \tag{1}$$

If there is a tie between links $\mathcal{L}_j$ and $\mathcal{L}_k$, then $Q_{i,j} < Q_{i,k}$.

Due to space constraints, the proofs are given in the extended version of this paper [5].

Based on the previous Lemma, we can now define the *link assignment vector* $L(A)$= $\{l_i(A), i = 1 \ldots M\}$, where $l_i(A)$ is the link $\mathcal{L}_j$ that source $\mathcal{S}_i$ selects based on Lemma 1. From the link assignment vector $L(A)$, the expected load at an ingress link $\mathcal{L}_j$ is

$$r_j(A) = \sum_{k=1 \ldots M : l_k(A) = \mathcal{L}_j} s_k \tag{2}$$

The vector $R(A)$=$\{r_j(A), j = 1 \ldots N\}$ is the *link load vector*. The type of INITE that we consider is based on a set of maximum load constraints for each ingress link. Specifically, let c_j be the maximum traffic load (maxload) allowed at link $\mathcal{L}_j$, with C=$\{c_j, j = 1 \ldots N\}$ being the corresponding *link maxload vector*. A link $\mathcal{L}_j$ is *overloaded* if $r_j(A) > c_j$. When none of the N ingress links is overloaded, we say that A is *acceptable* under C.

The COP problem can now be stated as follows. *Given an instance I=(S, C, P, Q), determined by the source load vector S, the link maxload vector C, the AS-Path length matrix P, and the tie-breaking matrix Q, does an acceptable padding vector A exist? When it is true, determine the* optimal padding vector A^**, such that across all acceptable padding vectors A* $\sum_{j=1}^{N} a_j^* \leq \sum_{j=1}^{N} a_j$. When there is an acceptable padding vector for a given instance I, we say that I is *feasible*; otherwise, I is *infeasible*.

3 Optimal Padding Vector Algorithm

The OPV algorithm (Algorithm 1) takes as input an instance I=(S, C, P, Q) of the COP problem, and examines the feasibility of a single padding vector $A^{(m)}$ in iteration m, starting from the zero padding vector. With each padding vector that is not acceptable, the algorithm identifies an overloaded link and then increments the corresponding

[1] Some ISPs have an agreement with their peers that they will announce the same AS-Path to a certain destination through all their peering links. If that is the case, $\mathcal{T}$ can group together all ingress links to each peer, and then apply the proposed algorithms at those link groups.

Algorithm 1.OPV (I=(S, C, P, Q))

```
1: Compute $\Phi$ from (3) and (4)
2: $A^{(0)} = [0, \ldots, 0]$;
3: for $m = 0$ to $\Phi$-1 do
4:     Compute $L(A^{(m)})$ from (1)
5:     Compute $R(A^{(m)})$ from (2)
6:     if $A^{(m)}$: acceptable (i.e., $\forall j, r_j(A^{(m)}) \leqslant c_j$) then
7:         return $A^{(m)}$
8:     end if
9:     $A^{(m+1)} = A^{(m)}$
10:    Identify an overloaded link $j$, i.e., $r_j(A^m) > c_j$
11:    $a_j^{(m+1)} = a_j^{(m)} + 1$
12: end for
13: return $I$: Infeasible
```

padding element. The exact selection of the overloaded link does not matter. The algorithm exits either when it has found an acceptable padding vector, or when it has completed Φ iterations, the upper bound of the number of iterations.

The bound Φ, which is proven to be a tight bound in Lemma 3 and Lemma 4, is computed as follows:

$$\phi_j = \max_{1 \leqslant s \leqslant M} \max_{1 \leqslant i \leqslant N} (P_{s,i} - P_{s,j}) \tag{3}$$

and

$$\Phi = N + \sum_{1 \leqslant j \leqslant N} \phi_j - \min_{1 \leqslant j \leqslant N} \phi_j \tag{4}$$

Note that Φ can be computed in polynomial time.

Later in this section, we show two important properties of OPV. First, when I is feasible, OPV reports the optimal padding vector A^*. Second, when I is infeasible, OPV exits after Φ iterations reporting that indeed, there is no acceptable padding vector.

The following lemmas prove that for a feasible instance: first, the optimal padding vector A^* has at least one zero element; second, when a padding element a_k reaches the value of the corresponding element in the optimal padding vector at iteration m, i.e., $a_k^{(m)} = a_k^*$, link k will not be overloaded in any subsequent iterations, and so a_k remains at a_k^*; third, each optimal padding element is upper bounded by a function of P that can be computed in polynomial time.

Lemma 2. *Suppose that the instance I is feasible, with optimal padding vector A^*. There exists a link j with $a_j^* = 0$.*

Lemma 3. *Suppose that the instance I is feasible, with an optimal padding vector A^*. If the OPV padding vector in iteration m is such that $a_j^{(m)} \leqslant a_j^*$ for all links j and there exists a link k such that $a_k^{(m)} = a_k^*$, then in all subsequent iterations n ($n > m$), $a_k^{(n)} = a_k^{(m)} = a_k^*$.*

Lemma 4. *Suppose that the instance I is feasible, with optimal padding vector A^*. Then, for any link j, the corresponding optimal padding is bounded by $a_j^* \leqslant \phi_j + 1$, where ϕ_j is given by (3).*

The following theorems give the main result for the OPV algorithm in the case of feasible instances and infeasible instances, respectively.

Theorem 1. *Suppose that the instance I is feasible, with optimal padding vector A^*. The OPV algorithm returns the vector A^* as its final outcome in polynomial time.*

Theorem 2. *If instance I is infeasible, then the OPV algorithm detects that there is no acceptable padding vector in polynomial time.*

4 Estimation of Input Parameters

In this section, we focus on the two key unknown inputs of the OPV algorithm, namely the set of super-source networks and the corresponding source load vector S, and the length estimation matrix P. As mentioned in §2, we do not attempt to estimate the tie-breaking matrix Q; the algorithm of the following section determines a robust padding vector independent of Q. Also, the maxload vector C is supposed to be known, given that it is chosen by the target network operator.

4.1 Selection of Super-Sources

We assume that the ingress routers of $\mathcal{T}$ collect statistics (with Cisco's NetFlow for instance) of the arriving traffic, aggregated by source networks. Since the ingress traffic may originate from many source networks, keeping track of the traffic by individual source networks may not be feasible at $\mathcal{T}$. We reduce the number of sources that need to be monitored by first aggregating several source networks into a super-source, and then only collecting volume statistics for large super-sources. The aggregation of source networks is based on the knowledge of the AS-level topology, which can be constructed with multiple vantage points [15]. For example, NetFlow data may show that several major sources of traffic for $\mathcal{T}$ belong to three stub ASes that are all single-homed customers of a tier-2 provider AS-X. In that case, AS-X can be considered as a super-source $\mathcal{S}_i$. The corresponding source load element s_i would be estimated as the sum of the load estimates of the actual sources, measured with NetFlow. To deal with the variability and unpredictability in the traffic volume of each source network, the capacity of an ingress link can be increased so that it has a sufficient headroom.

4.2 Estimation of AS-Path Length Matrix

To estimate $P_{i,j}$, $\mathcal{T}$ has to originate a route for a *wayfinding prefix* at each ingress link $\mathcal{L}_j$. A wayfinding prefix can be just the IP address of $\mathcal{L}_j$. If that is an unacceptably long prefix, the wayfinding prefix can cover instead a small part of the target network's address space. To estimate $P_{i,j}$, the operator of $\mathcal{T}$ can use one of the four following techniques, in the given order (see Figure 1). The first three techniques rely on *Looking*

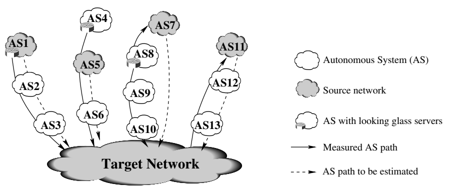


Fig. 1. Four estimation techniques for the AS-Path length

Glass Servers (LGS). LGS's are abundant in the Internet today, providing a "peek" inside a network's BGP routing tables. They are mostly used for problem diagnosis and monitoring of interdomain routing. The publicly available LGS's that are listed in *www.traceroute.org* include 273 servers that in some cases provide access to multiple routers within an AS. It is likely however that major ISPs have private access to even more LGS's in remote networks, to accommodate synergistic problem diagnosis.

- **LGS in $\mathcal{S}$:** The source network $\mathcal{S}$ may include an LGS. This ideal case leads to the most accurate estimation. In the example of Figure 1, AS1 is a source network that deploys an LGS. $\mathcal{T}$ can just query that LGS for the AS-Path to wayfinding prefixes.
- **LGS in a customer of $\mathcal{S}$:** Here, a customer of $\mathcal{S}$ deploys an LGS. In that case, $\mathcal{T}$ can query that LGS for each wayfinding prefix, and then remove from the returned AS-Paths the part that includes that customer AS. In Figure 1, AS4 is a customer of the source network AS5. AS4 deploys an LGS.
- **LGS in the path from $\mathcal{T}$ to $\mathcal{S}$:** Another possibility is that $\mathcal{T}$ can locate an LGS in a network in the reverse path, from $\mathcal{T}$ to $\mathcal{S}$. In Figure 1, AS7 is a source network and AS8 is a network deploying an LGS in the reverse path. In that case, $\mathcal{T}$ can estimate the AS-Path length $P_{i,j}$ as the sum of the AS-Path lengths from the LGS to $\mathcal{L}_j$ and from the LGS to $\mathcal{S}$. This technique assumes that the AS-Paths from the LGS to $\mathcal{S}$ and from $\mathcal{S}$ to the LGS have the same length.
- **Reverse path estimation:** When none of the previous techniques is applicable, $\mathcal{T}$ can just estimate the length of the AS-Path from $\mathcal{S}$ to $\mathcal{L}_j$ based on the length of the reverse path from $\mathcal{L}_j$ to $\mathcal{S}$, which is of course directly available at the border router at $\mathcal{L}_j$. The problem with this technique is that it relies on the symmetry of the forward and reverse AS-Paths, which is often not the case in the Internet. On the positive side, the technique still produces a correct estimate if the two paths have the same length, even if those paths are different.

Note that we cannot use the AS-level traceroute tool [16, 17] in the estimation of the P matrix, because that technique cannot report the degree of padding in an AS path.

AS-Path length distribution and estimation errors: The first two of the four previous techniques would give no estimation errors, because the corresponding LGS's report directly the AS-Path from the source to the ingress links of the target network. The third and fourth techniques, on the other hand, depend on the symmetry assumption, and they can suffer from estimation errors. To quantify these errors, we set up an experiment using 79 of the publicly available LGS's listed at *www.traceroute.org*. In this experiment, we estimate the AS-Path length from every LGS to each of the 78 other LGS's using the previous two techniques: "LGS in the reverse path" (when applicable), and "reverse path estimation" (always applicable). Then, we compare each AS-Path length estimate with the length of the actual AS-Path, as that is reported in the remote LGS.

The unconditional estimation error for the previous two estimation techniques is as follows: 60-65% of the estimates have zero error, 85-90% have an error of less than ± 1 hop, and 95% have an error of less than ± 2 hops. More importantly, we calculate the conditional probability of an estimation error e, given the estimated AS-Path length $\hat{p}$. The estimation error e is measured as $\hat{p} - p$, where p is the actual AS-Path length. The conditional probability that an estimate is error-free if $\hat{p}$ is 3 or 4 hops is 78% and 70%, respectively, which is higher than the corresponding unconditional accuracy. Similarly, the conditional probability that an estimate has an error of less than ± 1 hop if $\hat{p}$ is 5 or 6 hops is 88% and 86%, respectively. We also use the LGS servers to measure the AS path length distribution. About 90-95% of the paths are up to 6 AS hops, including any prepending, while 60% of the paths are either 4 or 5 hops. We use this AS-Path length distribution, as well as the previous estimation error conditional probabilities, in the robustness study of the next section.

5 Robust Padding Vector Algorithm

In this section, we first describe an algorithm that aims to determine a robust padding vector in the presence of these errors, and then evaluate the effectiveness of that algorithm with simulations.

5.1 Robust Padding Vector Algorithm

Recall that an instance I of the COP problem is defined by the set (S, C, P, Q). The OPT algorithm of §3 was developed to find an optimal padding vector A^* for I. In practice, we do not have a way to estimate Q, and P may include estimation errors. To deal with these two issues, we develop the *RPV (Robust Padding Vector)* algorithm. The basic idea in RPV is the following. Suppose that P_0, our original estimate of P, together with an arbitrary tie-breaking matrix Q and the given S and C vectors, form the instance I_0 that we start from. The unknown *actual instance* I_a, based on the correct AS-Path length matrix and the real tie-breaking behavior[2], belongs in the space $\mathcal{I}$ of all possible instances that can be generated by applying a given error model to P_0, and by considering all possible tie-breaking behaviors.

First, RPV generates a subset $\mathcal{I}_X$ of $\mathcal{I}$ that consists of X feasible instances. For each instance I_i in $\mathcal{I}_X$, the corresponding optimal padding vector (computed with OPV) is A_i. The set of padding vectors $\mathcal{A}$=$\{A_i, i = 1 \ldots X\}$ is our candidates for the desired robust solution. To generate padding vectors that differ significantly, we create the subset $\mathcal{I}_X$ by applying the AS-Path length estimation error model only to the largest sources of I_0. Errors that correspond to small sources (relative to the rest of the sources) may not have an impact on the resulting padding vector. Second, RPV generates a large

[2] We still assume that the real tie-breaking behavior can be captured by a matrix such as Q.

subset $\mathcal{I}_Y$ of $\mathcal{I}$ that includes Y instances. The fraction of feasible instances in $\mathcal{I}_Y$ is the *feasibility index* of $\mathcal{I}_Y$. If Y is sufficiently large, the feasibility index estimates the probability that the actual instance I_a is feasible. Third, for each candidate padding vector A_i in $\mathcal{A}$, RPV measures the fraction of *feasible* instances in $\mathcal{I}_Y$ for which A_i is acceptable. That fraction is the *robustness index* of A_i. If Y is large, the robustness index estimates the probability that the corresponding padding vector A_i is acceptable for the actual instance I_a, conditioned on the fact that the latter is feasible.

Eventually, RPV reports the padding vector $\tilde{A}$ with the maximum robustness index. The reason is that that vector maximizes the likelihood that it will be acceptable for I_a. The higher Y is, the more samples we collect from the space of possible instances, and so the more reliable our robustness index will be. The robustness, on the other hand, can be increased if we increase X.

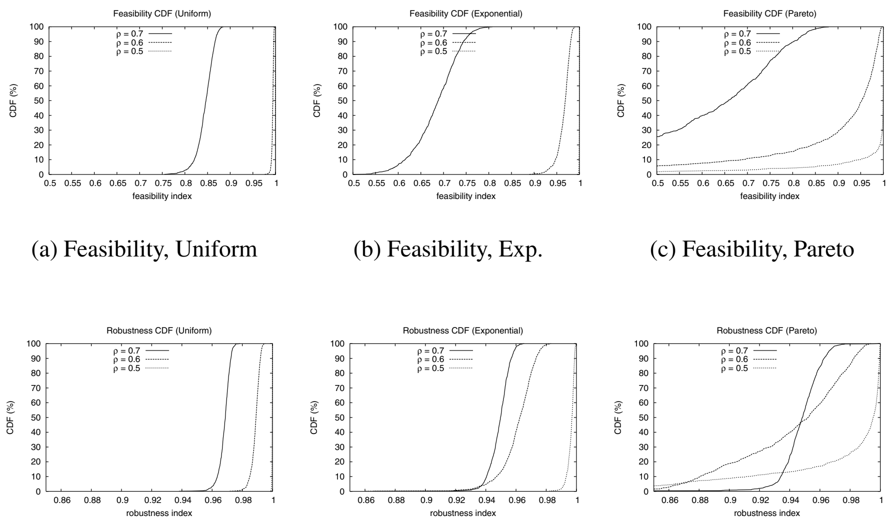
5.2 Robustness Evaluation

We evaluate the robustness of the padding vector that RPV reports using simulations. In the following, we consider a target network with N=5 ingress links that have the same maxload constraint (i.e., C_j=C for all j). The number of source networks is M=100. The source load distribution is the same for all sources, and it follows one of the following models: Uniform, Exponential, and Pareto (shape parameter: 1.7). The mean source load $\bar{s}$ is the same in all distributions. The AS-Path length matrix P_0 is generated based on the LGS empirical distribution mentioned in §4. The error model applied to P_0 is based on the conditional estimation errors, which are collected through the measurement described in §4, with given AS-Path lengths. The RPV algorithm uses X=100 and Y=10000 instances. $\mathcal{I}_X$ is formed by applying errors to the set of sources that generate, in total, at least 80% of the aggregate load.

The feasibility index depends on the relation between the aggregate offered load $M\bar{s}$ and the aggregate maxload constraint NC. Given the source load vector S, we define our *load metric*, $\rho = \frac{\sum_{i=1}^{M} s_i}{NC}$, which represents well the tightness of the given resource allocation problem for the homogeneous maxload constraints that we consider, and of course it should be less than 100% for any instance to be feasible. To achieve a certain load ρ given an instance with a source load vector S, we calculate the required C.

Figure 2 shows CDFs for the feasibility index and the robustness index of $\tilde{A}$ for each of the previous three load distribution models, and for three load conditions (ρ=0.5, 0.6, and 0.7). Each CDF resulted from 1000 executions of the RPV algorithm with a different original instance I_0. The CDF curves that are not visible in the graphs are equal to 100% for all instances. Note that even with the Uniform distribution for S, which is the most likely of the three to produce feasible instances, the feasibility index drops below 90% when ρ is 0.7 (Fig.2(a)). For the heavy-tailed Pareto sources, the feasibility is often below 90% even when ρ=0.6 (Fig.2(c)). The feasibility could be even lower if we had simulated non-homogeneous maxload constraints across different links.

Figure 2 shows that the robustness index of $\tilde{A}$ is larger than 90% for all three load conditions, with both the Uniform and Exponential distributions. With the Pareto distribution, on the other hand, the robustness can be lower than 90% in 10-20% of the cases, but rarely lower than 80% (Fig.2(f)). Note that a higher load ρ does not necessarily mean a lower robustness index. The reason is that the latter is conditioned on



(a) Feasibility, Uniform (b) Feasibility, Exp. (c) Feasibility, Pareto

(d) Robustness, Uniform (e) Robustness, Exp. (f) Robustness, Pareto

Fig. 2. Feasibility and robustness for three source load distributions and load conditions. Note that the CDF curves that are not visible are equal to 100% for all instances

feasible instances only. It can happen that even though the feasibility index is low, a large number of padding vectors are acceptable for the few feasible instances.

The overall conclusion from these results is that RPV can produce a robust padding vector $\tilde{A}$, in the sense that that vector will be acceptable for the actual instance I_a with a probability of more than 80-90%, if the latter is feasible. For I_a to be feasible with a high probability however, say more than 90%, the load metric ρ should be below 50-70%, depending on the distribution of S, at least for the homogeneous maxload constraints that we simulated here.

6 Conclusions

INITE is challenging mostly because it requires that an AS is able to affect BGP routing decisions in remote ASes. ISPs have been using AS-Path prepending to control the flow of ingress traffic. Prepending is widely viewed, however, as an ad-hoc technique, and it has not received much attention in the related literature. In this work, we took a first step of exploring the use of prepending in a more algorithmic framework, and its potential and limitations. Our main contribution is a polynomial-time algorithm (OPV) that can determine the optimal padding vector given the maximum load constraints of ingress links. Even though OPV relies on accurate parameters that can be only roughly estimated in practice, it is still important because it provides the best-case scenario for the effectiveness of prepending if all the required information is available. On a more practical level, the contribution of this paper is to describe how to apply prepending in a robust manner, considering that the AS-Path length information may be subject to estimation errors, and the tie-breaking behavior is unknown. Interestingly, our simulations show that it is possible to determine an acceptable padding vector even in that case, as long as the maximum load constraints are not too tight.

References

1. Fortz, B., Rexford, J., Thorup, M.: Traffic engineering with traditional IP routing protocols. IEEE Communications Magazine (2002)
2. Quoitin, B., Uhlig, S., Pelsser, C., Swinnen, L., Bonaventure, O.: Interdomain traffic engineering with BGP. IEEE Communications Magazine (2003)
3. Feamster, N., Borkenhagen, J., Rexford, J.: Guidelines for interdomain traffic engineering. In: ACM SIGCOMM Computer Communications Review. (2003)
4. Stewart, J.W.: BGP4: Interdomain routing in the Internet. Addison Wesley Longman, Inc. (1999)
5. Gao, R., Dovrolis, C., Zegura, E.: Interdomain ingress traffic engineering through optimized AS-path prepending. Technical Report GIT-CC-05-02, College of Computing, Georgia Tech (2005)
6. Uhlig, S., Bonaventure, O.: Implications of interdomain traffic characteristics on traffic engineering. European Transactions on Telecommunications (2002)
7. Uhlig, S., Magnin, V., Bonaventure, O., Rapier, C., Deri, L.: Implications of the topological properties of Internet traffic on traffic engineering. In: 19th ACM Symposium on Applied Computing, Special Track on Computer Networks. (2004)
8. Quoitin, B., Pelsser, C., Bonaventure, O., Uhlig, S.: A performance evaluation of bgp-based traffic engineering. International Journal of Network Management (Wiley) (2004)
9. Quoitin, B., Tandel, S., Uhlig, S., Bonaventure, O.: Using redistribution communities for interdomain traffic engineering. Computer Communications Journal (2004) 355–363
10. Agarwal, S., Griffin, T.G.: BGP proxy community community. http://www.ietf.org/internet-drafts/draft-agarwal-bgp-proxy-community-00.txt (2004)
11. Quoitin, B., Bonaventure, O.: A cooperative approach to interdomain traffic engineering. In: 1st Conference on Next Generation Internet Networks Traffic Engineering. (2005)
12. Agarwal, S., Chuah, C.N., Katz, R.H.: OPCA: Robust interdomain policy routing and traffic control. In: The 6th IEEE Conference on Open Architectures and Network Programming. (2003)
13. Uhlig, S., Bonaventure, O., Quoitin, B.: Interdomain traffic engineering with minimal BGP configurations. In: 18th International Teletraffic Congress. (2003)
14. Bressoud, T., Rastogi, R., Smith, M.: Optimal configuration for BGP route selection. In: IEEE INFOCOM. (2003)
15. Subramanian, L., Agarwal, S., Rexford, J., Katz, R.H.: Characterizing the Internet hierarchy from multiple vantage points. In: IEEE INFOCOM. (2002)
16. Mao, Z.M., Rexford, J., Wang, J., Katz, R.H.: Towards an accurate as-level traceroute tool. In: ACM SIGCOMM. (2003)
17. Mao, Z.M., Johnson, D., Rexford, J., Wang, J., Katz, R.H.: Scalable and accurate identification of as-level forwarding paths. In: IEEE INFOCOM. (2004)