# Designing BGP-based outbound traffic engineering techniques for stub ASes

Steve Uhlig
Computer Science and Engineering Department
Université catholique de Louvain, Belgium
suh@info.ucl.ac.be

Olivier Bonaventure
Computer Science and Engineering Department
Université catholique de Louvain, Belgium
Bonaventure@info.ucl.ac.be

## ABSTRACT
Today, most multi-connected autonomous systems (AS) need to control the flow of their interdomain traffic for both performance and economical reasons. This is usually done by manually tweaking the BGP configurations of the routers on an error-prone trial-and-error basis. In this paper, we demonstrate that designing systematic BGP-based traffic engineering techniques for stub ASes are possible. Our approach to solve this traffic engineering problem is to allow the network operator to define objective functions on the interdomain traffic. Those objective functions are used by an optimization box placed inside the AS that controls the interdomain traffic by tuning the iBGP messages distributed inside the AS. We show that the utilization of an efficient evolutionary algorithm allows to both optimize the objective function and limit the number of iBGP messages. By keeping a lifetime on the tweaked routes, we also show that providing stability to the interdomain path followed by the traffic is possible. We evaluate the performance of solution based on traffic traces from two stub ASes of different sizes. Our simulations show that the interdomain traffic can be efficiently engineered by using not more than a few iBGP advertisements per minute.

Our contribution in this paper is to demonstrate that by carefully thinking the design of the interdomain traffic engineering technique, stub ASes can engineer their outbound traffic over relatively short timescales, by exclusively tweaking their BGP routes, and with a minimal burden on BGP. Systematic BGP-based traffic engineering for stub ASes is thus possible at a very limited cost in terms of iBGP messages.

## Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing Protocols*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management, Network Monitoring*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks

## Keywords
BGP, interdomain traffic engineering, multiple-objectives optimization

## 1. INTRODUCTION
The current state-of-the-art in interdomain traffic engineering is primitive [6]. Traffic engineering techniques today mainly target transit ISPs [6]. The main problem tackled up to now in the traffic engineering literature is the one of optimizing the traffic *inside* large transit ASes. Two main techniques exist for that. The first technique relies on a full mesh of tunnels established inside the AS and traffic engineering is implemented by changing these tunnels [7, 5, 66]. The other way to traffic engineering the traffic *inside* an AS is to tune the IGP weights [27, 65, 26]. At the interdomain level, the situation is not better. Operators change their routing policies and the BGP attributes of the routes manually without a proper understanding of such changes on the flow of the traffic. Many problems arise due to misconfigurations in the routers [37]. The current practice in BGP-based traffic engineering is often "trial-and-error" [24], i.e. an operator changes the BGP attributes of some routes that were observed to carry a large amount of traffic and observes the effect on the interdomain traffic. For large transit ISPs, interdomain traffic engineering is a complex problem even for outbound traffic due to interactions between BGP and the IGP [1]. In the case of stub ASes on the other hand, the reason for the absence of a proper engineering of BGP is mainly a lack of understanding of the working of BGP and its effect on the traffic. The aim of this paper is to demonstrate that properly tuning BGP to engineer the outbound traffic without having to destabilize the BGP routing of a stub AS is possible, even on relatively short timescales of a few minutes.

### 1.1 Related work
Recently, commercial BGP-based solutions commonly referred as *route optimization* techniques [47, 18, 58, 40] have been deployed. These solutions work on relatively short timescales and target multi-connected networks. Their principle is to find, based on active and passive measurements, the "best" route to reach a destination or/and to be reached by an external hosts [4, 33]. These techniques work on timescales in the order of a round-trip time or more and adapt the traffic flow to the current conditions of the network, i.e. they try to find the best upstream provider to send or receive traffic. Route optimization techniques seem to target paths performance rather than obtaining a particular distribution of the traffic among the egress links [39], although they seem to be able to deal with load balancing the outbound traffic. It is unclear how load-balancing is performed, whether these techniques try to minimize the impact

on BGP and if so how it is done. Still, two types of route optimization techniques seem to be used [39]: DNS-based [47, 40] ones and BGP-based [18, 58] ones. DNS-based techniques [47, 40] rely on smart NAT coupled with very small DNS TTL's to choose the best-performing upstream provider for both outbound and inbound connections. NAT-based solutions however are not scalable to large networks such as ISPs, with many hosts behind the NAT. BGP-based techniques [18, 58] mainly deal with outbound traffic and seem to be better suited to optimize the trade-off between paths performance and the cost of the traffic [39]. It is however not clear how BGP-based route optimization techniques deal with the tweaking of the BGP routes, nor what burden they require on the access routers in terms of the BGP updates. However, as these solutions mainly aim at adapting the best BGP route choice to the real-time performance of the interdomain paths, they incur the risk of requiring many BGP route changes depending on the network conditions. Similarly to route optimization techniques, [28] studied the trade-off between traffic cost and performance for multihoming. The off-line and on-line algorithms proposed in [28] optimize either traffic cost or performance or both, but without caring about the cost in terms of BGP changes.

Note that the objectives of route-optimization techniques are much centered on user's perspective of network performance. In this paper, we rather focus on stub ASes who care more about aspects related to the distribution of the traffic over egress points than end-to-end properties. As route optimization techniques are rather black-box techniques over which the network manager as limited control, these solutions target smaller networks that have not enough experience with BGP to deploy themselves their BGP-based traffic engineering techniques. Also, evaluating BGP-based traffic engineering techniques that care about the end-to-end properties of the traffic would need to actually implement the technique and monitor the end-to-end path properties. We leave this aspect as further work.

On router vendors' side, several features exist in BGP implementations to engineering the traffic. Cisco's BGP multipath [55] allows to load-balance outgoing traffic when several equal cost (only the router-id differs) eBGP routes are learned from a neighboring AS. Per-packet or per-destination load balancing is then performed by the router among the multiple paths. Unequal cost load-balancing is also possible by relying on the extended community attribute [51]. However, Cisco's BGP load-sharing techniques [56] are limited to multiple links on a single router and do not work well with multiple routers. In addition, these techniques are rather limited in their scope: outgoing load-balancing (packet-based or destination-based) and static incoming traffic distribution based on static relative available link bandwidth. The same features are also available on Juniper routers [41].

Very few papers dealt with designing techniques for BGP-based traffic engineering [67, 61]. [67] proposed a random search algorithm, called recursive random search, to deal with the problem of finding the right value of the parameters for the configuration of large networks. The principle of [67] is to rely on random search and restart the search to converge towards the optimal parameter's values. The problem tackled in [61] consisted in modifying the LOCAL-PREF attribute of the BGP routes in an off-line manner (once every day) to balance (or minimize a cost function) the outbound traffic among several BGP neighbors of a stub AS. The purpose of [61] was to try to minimize the router's configuration changes to be performed for traffic engineering purposes by working on "BGP filters" defined at different topological aggregation levels, i.e. regular expressions defined on the AS path of the BGP routes. [61] showed that the reduction of the number of BGP routes that need to be tweaked is limited while working on BGP filters instead of BGP routes increases the search space by a large factor. Note that the technique proposed in [67] could be used for the problem of this paper. Recursive random search however was designed to work on the paremeter's space, it focusses on the values of the parameters to be changed to the network. Our technique is different in that the value of the parameters (BGP route attributes) to be changed are fixed in advance and the traffic engineering problem as we define it concerns the choice of which BGP routes should be tweaked to optimize an objective function defined on the outbound traffic of a stub AS. Furthermore, it is unclear how recursive random search is able to cope with several traffic engineering objectives. Our technique on the other hand has been designed to deal with multiple-objective problems, our algorithm is able to find a whole front of non-dominated solutions.

## 1.2 Problem statement

Assume that a multi-connected stub AS receives a full BGP routing table on each of its border BGP routers. To control the flow of its outbound traffic, this stub AS can rely on the information found in these BGP routing tables. BGP is a path vector routing protocol. Each BGP router sends BGP advertisements to its peers. A route advertisement sent by an AS through BGP means that the AS advertising the route agrees to forward IP packets to the destinations corresponding to this route. In addition, the AS PATH attribute [53] contained in this route advertisement tells through which ASes the IP packets will transit to reach their destination.

Upon receiving a BGP route, an AS may modify the value of some of the BGP attributes of this route through BGP filters [30]. For instance, it is possible to apply a BGP filter that changes the value of the LOCAL-PREF attribute of some route to prefer the route toward some destination received through one particular BGP peer over other routes for the same destination received through other BGP peers.

Even if a relatively small number of popular destinations receive most of the traffic [49, 62], this *small* number of destinations to be taken into account can grow as large as hundreds of destination prefixes whose routes need to be tweaked [46, 24]. The combinatorial nature of the interdomain traffic engineering problem makes it difficult in practice for even a limited number of destinations [61]. When the number of choices in the BGP routes that can be tweaked grows, finding the best BGP routes to be tweaked to optimize the traffic becomes more difficult. Having much choice in the prefixes that can be tweaked does not make the problem easier. More detailed treatments of interdomain traffic engineering with BGP can be found in [46, 24]. We do not deal in this paper with engineering the inbound traffic received by stub ASes. The main reason concerns the fact that modifying the flow of the inbound traffic with BGP requires changing how remote ASes choose their best BGP route to reach the local AS. BGP-based outbound traffic engineering for stubs only changes the traffic pattern, it should not change the BGP advertisements outside of the local domain. Inbound traffic engineering for stubs on the other hand requires to announce the tweaked routes and might affect a large fraction of the Internet. Inbound interdomain traffic engineering should thus be more of a collaborative nature [2] than selfish BGP tweaking as for outbound interdomain traffic engineering.

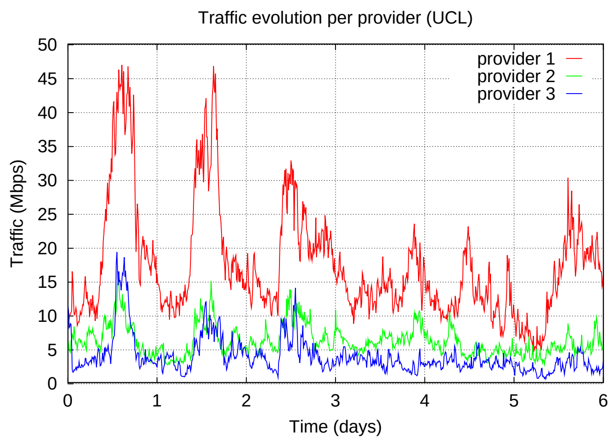

**Figure 1: Example time evolution of interdomain traffic for a campus network.**

Another aspect of the interdomain traffic engineering problem considered in this paper concerns the dynamics of the interdomain traffic. Figure 1 shows an example of how the outbound traffic of a stub AS for a description of this might be split among its three providers. The purpose of Figure 1 is to illustrate the variability of the traffic of a stub AS as well as to show the uneven way in which default BGP routing distributes the total traffic over several providers. On this figure, we took the whole outgoing traffic of the Université catholique de Louvain (see section 3.2) and BGP routing tables from Oregon route-views [38]. We simulated the presence of three would-be providers that the university could have. We show the amount of traffic that would be sent if the BGP tables over the six days were those the university used, using 10 minutes bins. Traffic variability is important at these timescales and random traffic surges happen on the three providers. The main issue appears for the first would-be provider because the ROUTER-ID of its routes are the lowest. The BGP decision process hence systematically favors the routes learned from the first provider. If the local AS does not rely on the LOCAL-PREF attribute to differentiate the routes of its providers, it is the AS path length that differentiates the BGP routes among the providers during the BGP decision process. Therefore, the first provider carries the traffic for all routes excepted for those that have a shorter AS path length through the other two providers. Simulations have shown that this situation is actually quite common with stub ASes [9].

While transit ASes that care about both the distribution of their traffic inside their AS and among their interdomain links, stub ASes mainly need to select among the available access links which one to use to reach some destination prefix. The main traffic engineering problem of stub ASes is to be able to manage how to choose the access link to reach some destination prefix, to react on traffic disruptions or on random traffic surges that happen on their access links. The goal of the traffic engineering technique presented in this paper is to track an optimal distribution of the outbound interdomain traffic on timescales of several minutes. Eventhough engineering the interdomain traffic on such short timescales as minutes might not make sense from an interdomain traffic engineering viewpoint, our purpose in relying on such a fine time-granularity is to demonstrate the feasibility of working on timescales at which BGP convergence takes places [29, 35] to tweak the BGP routes while keeping the burden on BGP very small. The problem we address would thus allow stub ASes to have a more reactive BGP routing under changes of the traffic pattern or of routing failure, only by tweaking the BGP routes. In this paper we chose to work with time intervals of 10 minutes. Although this choice might seem questionable, we feel that in practice this choice is an operational problem. For our algorithm to work, the choice of the time interval is an issue. We think that in practice the most crucial problem will be to obtain the traffic statistics on time to compute how to tweak the BGP routes. The choice of the time interval depends on how frequently the network operator is willing to change its BGP routes, and how adaptive he wants such a kind of traffic engineering to be. In this paper, we address the question of the feasibility of optimizing traffic engineering objectives for stubs without creating a burden on BGP.

The traffic objectives to be optimized by stub ASes might differ depending on their size and their main business. In this paper, we use two illustrative objectives. The first is to evenly distribute the total traffic over the available providers, specified as $min(max(tr_i))$, $i = 1, ..., n$, $n$ being the number of available providers and $tr_i$ denotes the amount of traffic sent to provider $i$. Because traffic balancing might be too simple an objective, we also evaluate the optimization of a second objective of minimizing the cost of the traffic, specified as $min \sum_{i=1}^{n} c_i \times tr_i$ where $c_i$ denotes the cost of one unit of traffic sent to provider $i$. The second traffic objective models the volume-based billing performed by transit ISPs.

Contrary to *route optimization* techniques that might require an arbitrarily number of iBGP changes to adapt the the real-time performance of the paths used for each destination prefix, our technique aims at providing stability to the path followed by the traffic and to minimize the number of BGP path changes that the traffic will have to undergo. In this paper, we do not take into account performance metrics as the end-to-end delay or available bandwidth of the flows as done by route optimization techniques. For these reasons, our technique is very different in scope to *route optimization* techniques as our purpose is mainly to prevent changes in the choice of the best BGP route as much as possible when performing interdomain traffic engineering. We do not discuss real-time objectives like using the smallest RTT interdomain route or the one having largest available bandwidth, as these objectives require real-time monitoring of the paths having most of the traffic. As the quality of the interdomain paths might change drastically over small time periods of minutes, adapting to the real-time conditions along the paths could require to tweak a large number of BGP routes. Such a kind of traffic engineering is performance-centered, while the traffic engineering we focus on in this paper is rather of trying to improve the flow of the interdomain traffic by changing the best BGP route at the timescales at which BGP reacts to topological changes.

Note that in this paper we chose to limit the burden on BGP by trying to minimize the number of iBGP messages generated by the traffic engineering scheme. The choice of the iBGP metric to measure the burden on BGP is driven by the assumption that stub ASes will not have to advertize eBGP messages when performing outbound traffic engineering with BGP. Contrary to transit ASes who might care about the number of eBGP messages to their customer ASes, we assume that traffic engineering in stub ASes mainly creates iBGP messages. In practice however, it is unclear how the burden on routing created by traffic engineering should be minimized. The algorithm we propose in this paper uses this iBGP messages metric as a built-in dimension of the search. Changing this metric will require significant, although not unworkable, changes to the search algorithm.

## 1.3 Context

The ASes we mainly focus on in this paper are stub ASes that constitute the majority of the ASes in the Internet [54]. At the date of June 19, 2004 more than 86% (15141 over 17443) of the ASes were considered as stub ASes by APNIC's BGP routing table [52]. Among these stub ASes, we focus on ASes connected to the Internet through several different providers. Because stub ASes do not offer transit service, they do not advertise via eBGP the routes they learn from their peers. In the case of a stub AS, there is no interaction between the local BGP route tweaking performed on the eBGP routes learned from the peers and the rest of the Internet.

Figure 2 plots the distribution of the number of providers stub ASes have according to the BGP tables gathered from several vantage points on December 12, 2003 [54]. On Figure 2, we only consider the ASes from the gathered BGP routing tables that were not classified as provider for any of their peering according to the heuristic proposed in [54]. According to the data of [54], 14287 ASes were stub ASes. Among these stubs ASes, 5671 (40%) are single-homed and 6748 (47%) dual-homed. The average number of providers (distinct ASes) per stub is 1.87. Multi-homing is not only used for backup purposes, but a trend toward more multi-homing has been recorded during the last few years [2]. Although the AS-level topology inferred from [54] is not representative of the AS-level Internet, it is likely to be representative of customer-provider relationships as these peering relationships are not hidden by BGP as are peer-peer peerings [15]. Because [54] relied on BGP routing tables, the AS-level topology seen by [54] does not see some private peer-peer peerings. ASes that are multi-connected for other

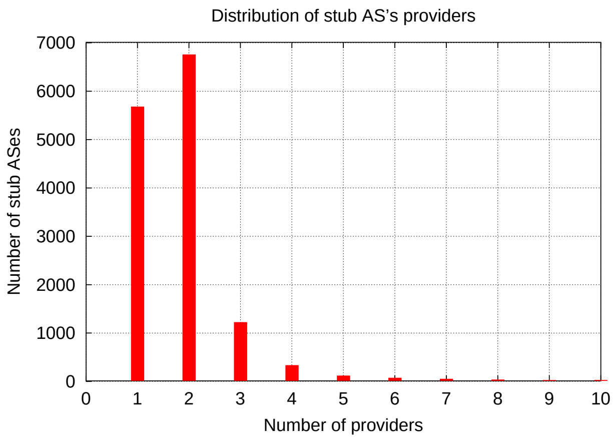

**Figure 2: Number of providers for stub ASes.**

reasons than solely backup purposes need to distribute their interdomain traffic among several interdomain links. Almost all ASes have to pay large ISPs to get a global connectivity on the Internet. The cost of the traffic on these links can become significant. These ASes would find interest in redistributing some traffic from more expensive providers towards less expensive ones or simply better balancing their traffic among the various available access links. We do not expect that the number of providers per stub to grow much in the future since [3] has shown the limited value in terms of performance metrics of going beyond three providers.

The remainder of this paper is structured as follows. In section 2 we discuss the issues raised by short-timescales outbound interdomain traffic engineering. In section 3 we provide simulations to evaluate the feasibility of short-timescales outbound traffic engineering with BGP. Section 4 evaluates the contributions of this paper. Finally, section 5 summarizes the content of this paper and discusses the various paths this work has opened for further research.

## 2. COMPLEXITY OF INTERDOMAIN TRAFFIC ENGINEERING

The problem we tackle in this paper consists in controlling the distribution of the interdomain traffic in time by tweaking the BGP routes. This problem is discrete and the control of the traffic is indirect, by tweaking the BGP routes. Furthermore, tracking the traffic dynamics makes the problem even harder. Finally, as we assume that stub AS are likely to want to optimize several objectives at the same time, we place the context of this paper within the framework of multiple-objectives optimization.

In this paper, we rely on an evolutionary algorithm (EA) to perform such on-line traffic engineering. The choice of evolutionary algorithms lies in their ability to solve complex problems and particularly multi-objective optimization problems [17, 19]. Because our problem is a dynamic multiple-objectives one, a population-based heuristic makes the search easier [13].

The principles of our scheme are as follows. For each time interval (typically a few minutes), the current choice of the best BGP routes drives the distribution of the outbound interdomain traffic. During each time interval, the algorithm must find a non-dominated front of solutions (with respect to the objectives defined in section 1.2) that will be used for the next short-term time interval. What we call a *solution* is a set of best BGP routes that differ from those best routes found by BGP. There are three main issues to be tackled during the design of the algorithm: 1) tracking the traffic distribution for the next time interval, 2) finding non-dominated solutions and 3) choosing the next solution among the non-dominated ones.

For the third issue, the choice of the solution to use during the next time interval among the possible non-dominated solutions is an issue whenever large trade-offs exist between the number of BGP advertisements and the value of the traffic objective. The results of the next sections will show that even if these two objectives are conflicting, permitting more and more BGP advertisements provides a decreasing marginal gain in the traffic objective. Henceforth, very few BGP advertisements are required in practice and network operators will only have to choose the upper bound on the number of BGP advertisements allowed.

**Pareto-optimality** The concept of Pareto-optimality is related to the set of solutions whose components cannot be improved in terms of one objective without getting worse in at least one of the other objectives. More formally, a multiple-objectives search space is partially ordered in the sense that two solutions are related to each other in two possible ways: either one dominates or neither dominates. Consider the multiple-objectives minimization problem:

$$\begin{array}{ll} Minimize & y = f(x) = (f_1(x), ..., f_n(x)) \\ where & x = (x_1, ..., x_m) \in X \\ & y = (y_1, ..., y_n) \in Y \end{array}$$

$x$ is called the decision vector, $y$ the objective vector, $X$ the parameter space and $Y$ the objective space. In the context of this paper, the parameter space is the attributes of the BGP routes and the objective space is the values of the traffic objective.

A decision vector $x_1 \in X$ is said to dominate another decision vector $x_2 \in X$ ($x_1 \succ x_2$), iff
$\forall i \in 1,..,n : f_i(x_1) \geq f_i(x_2) \quad \wedge$
$\exists j \in 1,..,n : f_j(x_1) > f_j(x_2)$.

Let us define now what a Pareto-optimal decision vector is: a decision vector $x$ is said Pareto-optimal *iff* $x$ is non-dominated regarding $X$, i.e. $\nexists x' \in X : x' \succ x$.

A Pareto-optimal decision vector cannot be improved in any objective without degrading at least one of the other objectives. These are global optimal points. In this paper, we are not interested in global optima but optimal points in some neighborhood for some of the objectives. More precisely, we aim at finding the Pareto-optimal points with respect to the traffic objective and the number of BGP route changes, but within a distance of one BGP route tweaked inside the AS. Hence we do not search for globally Pareto-optimal decision vectors but locally Pareto-optimal decision vectors:

*Consider a set of decision vectors* $X' \subseteq X$.
1. The set X' is denoted as a local Pareto-optimal set *iff*
$\forall x' \in X' : \nexists x \in X : x \succ x' \wedge ||x - x'|| < \epsilon \wedge ||f(x) - f(x')|| < \delta$
where $||.||$ denotes a distance metric, $\epsilon > 0$ and $\delta > 0$.
2. The set $X'$ is called a global Pareto-optimal set *iff*
$\forall x' \in X' : \nexists x \in X : x \succ x'$.

In this paper, we often use the term *front* (non-dominated front or Pareto-optimal front) to mean *a set of solutions*. Because in this paper we rely on heuristics, we have no guarantee about the optimality of the solutions found. These solutions can thus only be said non-dominated, not Pareto-optimal. The reader may regard the terms non-dominated and Pareto-optimal as roughly equivalent although this is not true from an optimization viewpoint.

## 2.1 Tracking interdomain traffic

An issue to be tackled by the traffic engineering scheme is the tracking of the amount of traffic that will be sent towards each destination prefix during the next time interval. Network traffic is known to be bursty (see [42] and references therein), so that predicting the amount that will be sent through any provider could be critical to be able to find a good solution. In this section, we show that the complexity of the predictor used does not fundamentally improve the prediction error. Henceforth, the choice of the best performing predictor is not critical in the traffic engineering context. In addition, as the focus of the paper is not to find out which predictor works best over some traffic type, we demonstrate our point on a single traffic trace. The current section is thus mainly illustrative and is not generalizable to other traffic traces. Note that traffic provisioning schemes like [22, 34] are intended to limit resource over-provisioning, they are not aimed at tracking the traffic dynamics. For the algorithm to work in most practical situations, not having to rely on a particular model for the data is desirable, since different stubs may have very different user profiles and thus different traffic types.

Several works have dealt with the predictability of network traffic [50, 44]. [50] studied the multi-step ahead predictability of network traffic relying on the ARMA and MMPP models, to evaluate the possibility of multi-step ahead prediction for traffic control. The main result of [50] was that both smoothing and traffic aggregation help the prediction. [44] studied the multiscale predictability of network traffic, for one step ahead prediction. [44] confirmed the results of [50] concerning the benefits of smoothing. However, [44] also found by studying many different traffic traces that the predictability of network traffic highly depends on the considered trace. [44, 50] both evaluated the performance of predictability based on the ratio of mean squared error to variance. The results of [44] indicated that the performance of different predictors does not improve much with their increasing complexity. Relying on the last value of the time series or a simple moving average is as efficient as AR, ARIMA, and ARFIMA models [12].

Since smoothing might provide some benefits while complex predictors do not, we compare in this paper the performance of adaptive exponential smoothing with a simple predictor: the previous value. The purpose of this section is not to find *the* optimal predictor for our particular trace, but to determine whether prediction is practically useful at all in the context of interdomain traffic engineering.

[14] compared several versions of adaptive exponential moving averages for bandwidth prediction. The predictor the authors of [14] found ideal for Internet traffic is an adaptive EMA algorithm: the low pass EMA (LpEMA). The basic idea is to modify the classical EMA formula

$$e_i = (1 - \alpha) \times e_{i-1} + \alpha \times tr_i \tag{1}$$

where $e_i$ is the estimate at time $i$, $\alpha$ the weight of the moving average and $tr_i$ the traffic at time $i$. Instead of a fixed weight $\alpha$, the weight is made adaptive and computed as follows:

$$\alpha_i = \alpha_{max} \times \frac{1}{1 + \frac{|m_i|}{m_{norm}}} \tag{2}$$

where $\alpha_{max}$ is the maximum weight, $m_i$ is the gradient of the traffic at time $i$ ($\frac{tr_i - tr_{i-1}}{t_i - t_{i-1}}$) and $m_{norm}$ a normalizing gradient. In this paper, we computed $m_{norm}$ as the mean gradient over a time window of one hour (six 10 minutes intervals). We chose to rely on a time granularity of 10 minutes in this paper.

Let us have a closer look at the working of the LpEMA predictor. First, compute $m_{norm}$ as the mean gradient (in absolute value) over the last 6 time intervals. The adaptive EMA weight, $\alpha_i$, is computed according to Formula 2. The name Low pass EMA is due to the behavior of $\alpha_i$ that reduces the impact of large changes in the traffic by smoothing out the time-series, having as reference the mean gradient as an indication of the mean variability. If on the other hand a relatively limited change in the time-series occur during the current time interval, then the value of $\alpha_i$ comes closer to $\alpha_{max}$ and the EMA better tracks the short-term changes. The rationale behind this adaptive behavior is that large changes in the traffic should not be followed unless they are longer-term trends. The greater adaptation of the EMA whenever the shifts in the traffic correspond more closely to the average behavior of the signal (mean gradient) are meant to follow these short-term variations that better match the dynamics of the traffic. The main reason for the smoothing behavior of the LpEMA concerns the burstiness of Internet traffic, as tracking large shifts in the traffic is useless for prediction purposes. Unless these changes are longer-term trends, adapting the prediction based on traffic shifts will merely destabilize the prediction while not help in predicting such random traffic surges.

To assess the benefit of the LpEMA predictor to track the traffic of each provider, Figure 3 provides both the mean and the standard deviation of the prediction error for different values of $\alpha_{max}$ and each provider. On Figure 3, the prediction error has been computed for each time interval as the relative error between the predictor and the true value of the traffic. Each graph of Figure 3 plots the mean and standard deviation of this prediction error over the 6 days of the trace. Both the mean and the standard deviation of the error for the LpEMA predictor exhibits a first decreasing part but increase after some value of $\alpha_{max}$. The "last value" predictor has a lower mean error than the LpEMA one (no matter the value of $\alpha_{max}$) for the three providers. The standard deviation of the "last value" predictor is not always lower than the one of the LpEMA predictor. Only for provider 2 has the "last value" predictor a smaller mean and standard deviation than those of the LpEMA predictor for all values of $\alpha_{max}$. The behavior of the LpEMA predictor can be explained in the following way. Increasing the value of $\alpha_{max}$ first allows the predictor to adapt to the natural variability of the traffic. Then, as the value of $\alpha_{max}$ increases beyond some value, both the mean and the standard deviation of the error increase. This is due to a more adaptive behavior of the LpEMA predictor for increasing values of $\alpha_{max}$.

**The results of this section indicate that there is no benefit at relying on adaptive EMA predictors since their mean error is always larger than the one of the last value. Adaptive EMA schemes also require some tuning of their parameters, whose ideal values seem to depend on the particular traffic trace [44]. The last value predictor already constitutes an accurate predictor whose performance is comparable to an adaptive exponential moving average, both in its average and standard deviation. The simulations of section 3.3 will confirm that the last value predictor performs better than adaptive EMA when used by our traffic engineering technique.**

## 2.2 Searching the Pareto front

Once the method used to predict the traffic distribution during the next time interval has been chosen, we need to choose how to search for the best BGP route changes. For each time interval, the algorithm must find which BGP routes to tweak to optimize the traffic objective. At each time interval, the goal is to find a non-dominated front representing the trade-off between the number of BGP routes to be tweaked and the value of the traffic objective. The main issue with these two objectives is their conflicting nature: the larger the number of BGP routes tweaked, the better the expected value of the traffic objective. In the case where one knows the traffic demand, increasing the number of BGP routes tweaked will always allow to improve the traffic objective. When the traffic pattern changes with time however, having to find how to tweak the BGP routes under constraint that a set of BGP routes have been tweaked during the last time interval might lead to problematic situations. If some traffic from a previously heavily loaded provider has been moved to another (previously lightly loaded) provider, it could be necessary to undo the concerned traffic move to improve the traffic objective if more traffic has now carried over the previously lightly loaded provider.

We have several options concerning how the search can take place. We can start at each time interval with the previously used set of BGP route changes and search for the Pareto-optimal front for the next time interval. Another option is to start with the default BGP routing (no BGP route tweaked by the optimization) and search for the Pareto-optimal front from scratch during each time interval.

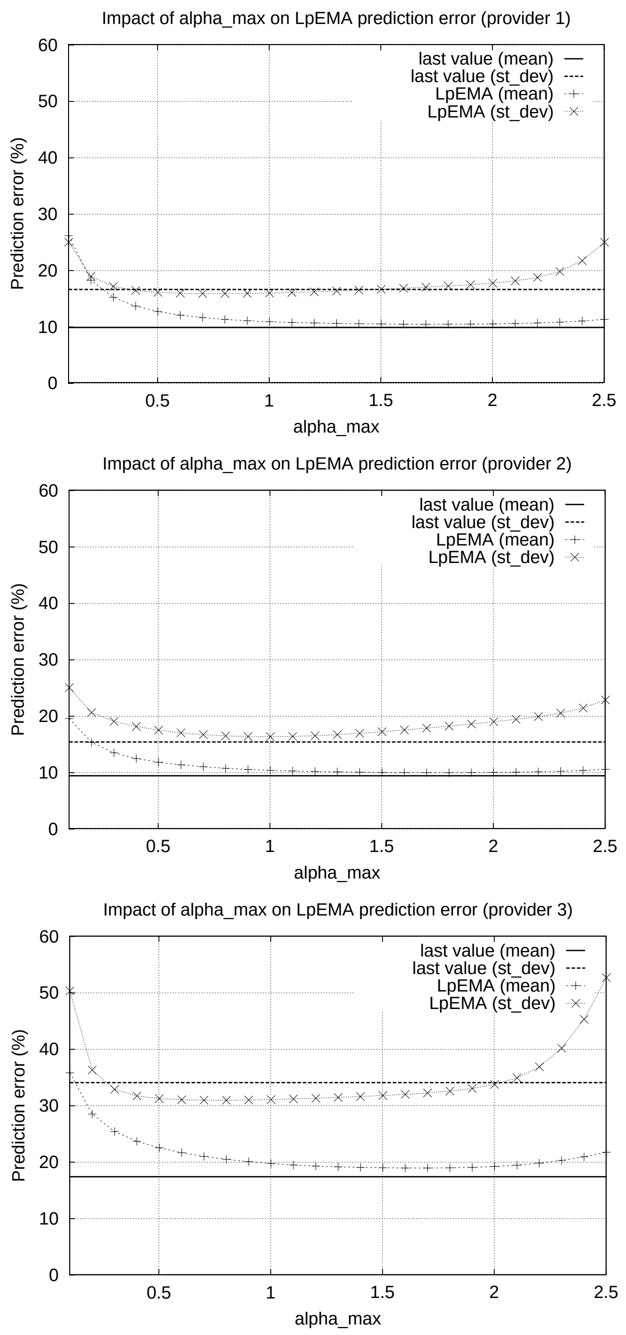

**Figure 3: Effect of $\alpha_{max}$ value on prediction error: provider 1 (top), provider 2 (middle) and provider 3 (bottom).**

Finally, we can also start with the whole non-dominated front found during the last time interval in search for the new Pareto-optimal front.

The first option has the drawback of requiring to potentially undo BGP route changes that are not suitable anymore due to changes in the traffic demand. By having to start the search with a set of BGP changes, there is a risk of having to undo some of these changes that were good during the previous time interval but need to be removed for the current time interval to be able to find a good solution. To illustrate what happens on the objective plane, Figure 4

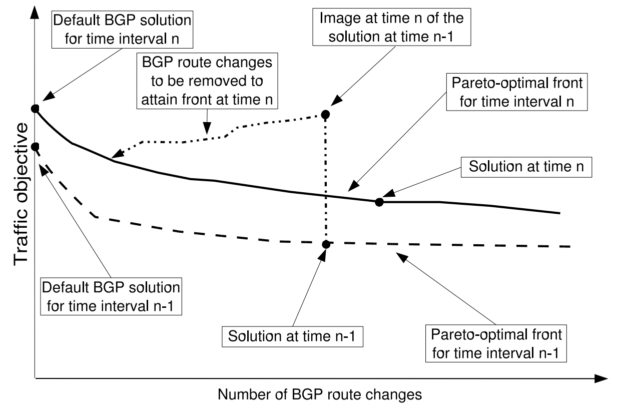
Default BGP solution for time interval n
Image at time n of the solution at time n-1
BGP route changes to be removed to attain front at time n
Pareto-optimal front for time interval n
Solution at time n
Traffic objective
Default BGP solution for time interval n-1
Solution at time n-1
Pareto-optimal front for time interval n-1
Number of BGP route changes

**Figure 4: Search of the Pareto front from one time interval to another.**

shows the two Pareto-optimal fronts for two consecutive time intervals, $n-1$ and $n$. Time is not explicitly represented on Figure 4, but each Pareto-optimal front represents a different time interval. The x-axis of Figure 4 represents the number of BGP routes to be tweaked by the optimization and the y-axis gives the value of the traffic objective of the best possible solution requiring a given number of BGP routes tweaked. Figure 4 shows the two Pareto-optimal fronts for the two consecutive time intervals, containing both the default BGP solution (0 BGP route change) and the set of BGP route changes chosen at each time interval. Suppose that we have to start the search with the solution chosen at time $n-1$, containing a few BGP route changes. To find the Pareto-optimal front for time interval $n$, we need to undo some tweaking we did during the last time interval $n-1$. On Figure 4, we show the value of the traffic objective at time $n$ of the solution used at time $n-1$. Because the traffic pattern changes, a good solution during the last time interval might have a relatively large value of the traffic objective during the next time interval.

Figure 4 for instance shows that the search path between the solution at time $n-1$ and the Pareto-optimal front at time $n$ requires undoing some tweaking of the BGP routes. The effect of undoing this tweaking is to improve the traffic objective, but undoing the tweaking of some BGP routes might worsen the traffic objective as well in practice.

In the case of the second option, we start from the default BGP routing, and build the Pareto-optimal front from scratch at each time interval. This solution has the advantage of limiting the size of the search space because the algorithm can iterate by only tweaking additional BGP routes to find the Pareto-optimal front. The drawback of this method is that one does not leverage the knowledge of potentially good solutions found during previous time intervals.

The last option consists in building the next Pareto-optimal front by starting from the last front. This solution has the same potential drawback as the first option, in that the size of the search space is very large because the search must allow to do and to undo the tweaking of BGP routes. Actually, the most annoying issue concerns having to work with a set of potential solutions from which we start the search. This requires that the search effort be distributed over the different points of the front. Some of the points of the old Pareto-optimal front however will not easily join the new Pareto-optimal front, so that valuable computational time may be wasted.

Experience with the search heuristics indicated that allowing to undo BGP route changes or starting with a whole front of solutions when searching for the next front had problems to sample the Pareto-optimal front, without providing a gain in terms of the solutions found by the heuristic. In the remainder of the paper, we do not allow to undo BGP route changes and always start with a single solution at each time interval in search of the Pareto-optimal front. In practice anyway, one starts with a given state of the best BGP routes and needs to find which routes should be changed for the next time interval. Initializing the search at some time interval with a front is not useful as one wants to find the set of changes to be applied to the BGP routes with respect to the current configuration of the routers, not to start from an arbitrary configuration that does not match the actual one.

It must be noted that heuristics only provide a non-dominated front. To have guarantees about the optimality of these heuristics, further work needs to be done to obtain the actual Pareto-optimal front.

## 2.3 Search algorithm

The previous section discussed the choice of the search heuristic. In this section, we sketch the working of the heuristic. A more detailed description of this search algorithm is given in Chapter 6 of [59].

In the previous discussion, we limited the traffic objective to be a single-value function. In practice, one of the interesting features of our algorithm is that it has been designed so as to be able to work with several traffic objectives at the same time. The principle is to rely on a random search to find improvements in any traffic objective by tweaking additional BGP routes and maintaining a non-dominated front of the improved solutions throughout the search. At each time interval, we run the algorithm a number of times bounded by the maximal number of BGP tweaks to be added during the current time interval. In the remainder of this section we focus on the working of a single iteration of the heuristic that tries to improve the current non-dominated front by adding one BGP tweak. An iteration where a population is improved by trying an additional change to the individuals of the population is called a "generation" in the evolutionary algorithms jargon as it generates a new population from the old one. We kept the name "generation" in the description below to be consistent with the evolutionary algorithms literature.

Depending on the relationships between the traffic objectives which might be conflicting, harmonious or neutral [43], the search on the non-dominated front should have to be different. Recall that we do not know beforehand the relationship between the traffic objectives. This means that our search method must be as lightly biased as possible towards any of the traffic objectives to sample in the best possible manner the search space. Because sampling the whole search space would make the search space grow very large, we decided that the heuristic would iterate over the BGP routing changes by trying to add one BGP routing change at each generation of the algorithm. Doing this puts additional pressure on the population by forcing improvements in the traffic engineering objectives to have as few BGP route changes as possible early on during the optimization. This does not guarantee that the solutions found will contain the smallest set of BGP tweakings to optimize the traffic objectives. However, as shown in [60, 59] optimizing the cost of the traffic of a stub AS on daily timescales while balancing the traffic over the available providers over timescales of minutes seems to be conflicting objectives. This suggests that in practice if traffic engineering needs care about several objectives the optimization problem is likely to be a multiple-objective one where the core of the problem will be to correctly sample a non-dominated front to find a good trade-off between the optimized objectives. The aspect of whether the algorithm finds *the* optimal set of BGP tweaks is hence likely to be of limited practical importance. Note that there is no known way of sampling a Pareto-optimal front without incurring the cost of having to explore a large number of potential solutions. As many relevant multiple-objectives problems are NP-hard, this problem is unlikely to be solved in the near future.

```
accepted = 0
iter = 0
while ((accepted < MAXPOP) AND (iter == MAXITER)){
 foreach individual $k$ {
    // Try a random BGP route change
    BGP_change.prefix = rand_int_uniform(1,NUM_PREFIXES)
    BGP_change.exit = rand_int_uniform(1,NUM_EXIT_POINTS)
    // If effect of BGP change is improvement accept it
    if (improved(k,BGP_change)){
        accept(k,BGP_change)
        // Update counter for accepted improved individuals
        accepted++
    } // end if
} // end foreach individual
// update iteration counter
iter++
} // end while
```

**Figure 5: Pseudo-code of search procedure for a single generation.**

Figure 5 provides a pseudo-code description of the search procedure for a single generation. The principle of the search for each time interval is as follows. At the first generation, we start with a population of individuals initialized at the default solution found by BGP routing (taking into account the BGP filters that are used by the border routers of the AS). Hence at generation zero all individuals have the same values of the traffic objectives and contain no BGP routing change (default best BGP routes). At each generation, we use a random local search aimed at improving the current population by applying an additional BGP routing change (a $<prefix, exitpoint>$ pair). Each individual of the population is non-dominated with respect to the other members of the population for what concerns the traffic objectives. In addition, the current population is always made of individuals having the same number of BGP routing changes. At each generation, we parse the whole population and for each individual we try to apply an additional randomly chosen BGP routing change. Whenever a BGP routing change provides improvement with respect to at least one of the traffic objectives, we accept this improved individual and put it in the set of accepted individuals. We iterate this procedure until we find a target number of improved individuals or stop when we have performed a target number of tries (the variable ITER). Note that the pseudo-code given at Figure 5 concerns only one generation, and that the purpose of variable ITER is not to count the generations but to ensure that the search will not loop indefinitely during the current generation.

### 2.3.1 *Sampling the non-dominated front*

Up to now, we described the procedure to search for BGP routings changes that improve the individuals of the previous population with respect to any of the traffic objectives. These improved individuals however are not non-dominated. Some of them can be dominated since we did not check for non-domination when accepting an improved individual. Improvement was sufficient to accept an individual. The next step just after the random search at each generation is to check for non-domination on this population of improved individuals to obtain a non-dominated front. For that purpose, we rely on the fast non-domination check procedure introduced in [20]. This procedure has time complexity $0(MN^2)$ where $M$ is the number of objectives and $N$ the size of the population. We do not describe this procedure in details but refer to [20] for the original idea and to [19] for an thorough explanation. Let us only mention the main points here. Let $P$ denote the set of non-dominated individuals found so far at the current generation. $P$ is initialized with anyone of the individuals among the accepted ones. Then try to add individuals from the set of accepted ones one at a time in the following way:

- temporarily add individual $k$ to $P$
- compare $k$ with all other individuals $p$ of $P$:
  - if $k$ dominates any individual $p$, delete $p$ from $P$
  - else if $k$ is dominated by other members of $P$ remove $k$ from $P$

This procedure ensures that only non-dominated individuals are left in $P$. The number of domination checks is in the order of $0(N^2)$ while for each domination check $M$ comparisons are necessary (one for each objective). The time complexity is thus $0(MN^2)$.

Having found the non-dominated front for a given number of BGP routing changes (the current population), we are left with selecting the individuals of the population for the next generation. Actually, the number of non-dominated individuals from the set of improved ones is due to be smaller than the size of the population we use during the search process (MAXPOP). To constitute the population for the next generation, we have to decide how many individuals in the next population each non-dominated solution will produce. Because non-dominated individuals are not comparable between one another, we must choose a criterion that will produce MAXPOP individuals from the set of non-dominated ones. On the one hand, we would like to include at least every non-dominated individual in the population. On the other hand, depending on the way the accepted solutions are spread over the non-dominated front, we must sample differently different regions of the front for a given number of BGP routing changes. This notion of sampling the non-dominated front is close to an idea of distance between neighboring individuals in the objective space. Maintaining diversity on the non-dominated front requires that individuals whose neighbors are farther apart be preferred over non-dominated individuals whose neighbors are close. The rationale behind this is that less crowded regions should require more individuals to be correctly explored than regions having more non-dominated individuals. The computation of the crowding distance for each individual is done according to [19] pp. 248. First the non-dominated individuals are sorted according to each objective. Then the individuals having the smallest and largest value for any objective are given a crowding distance $d^m$ of $\infty$ to ensure that they will be selected in the population. For each objective $m$, the crowding distance of any individual $i$, $1 \leq i \leq (|P| - 2)$, is given by

$$d_i^m = \left| \frac{f_{i+1}^m - f_{i-1}^m}{f_{max}^m - f_{min}^m} \right| \qquad (3)$$

where $f_i^m$ denotes the value of individual $i$ for objective $m$, $f_{max}^m$ (respectively $f_{min}^m$) denotes the maximum (respectively minimum) of the objective value $m$ among individuals of the set $P$ of non-dominated individuals. The global crowding distance for all objectives is the sum of the crowding distance for all objectives. For our two objectives, this crowding distance represents half the perimeter of the box in which individual $i$ is enclosed by its direct neighbors in the objective space.

The performance of the search algorithm is sufficient to be used as an on-line traffic engineering scheme. Finding the set of BGP routes to be tweaked during the next time interval requires $n$ iterations of the algorithm. This set contains at most $n$ BGP routes to tweak. To find the non-dominated population with an additional BGP route change, the algorithm has linear complexity with the population size and the non-domination check quadratic time with respect to the number of accepted individuals. Although our search technique does not ensure that we find the optimum, the low time complexity ensures that a non-dominated solution will be found within a short amount of time. Finding the optimal set of BGP routes to change at a given time interval would be far more complex (exponential time complexity) as the problem is NP-hard in the strong sense [61]. The current prototype Perl version of the algorithm takes in the order of one second per iteration on a P4 2.4 GHz. A C version would probably be at least an order of magnitude faster. All Perl scripts used for the simulations of section 3 are available at `http://www.info.ucl.ac.be/~suh/`. Note that it is unclear whether recursive random search [67] could be used to solve the same problem since it is not population-based and would hence have problems to sample the non-dominated front.

## 3. PERFORMANCE EVALUATION

This section studies the performance of the our interdomain traffic engineering solution introduced in section 2. Section 3.1 first presents the typical context in which our solution is to be used in practice. Section 3.2 presents the traffic traces used for the simulations of the paper. Section 3.3 then discusses the impact of the traffic variability on the quality of the solutions found. Section 3.4 proposes a modification to the basic traffic engineering solution that reduces the number of required iBGP advertisements. Section 3.5 then goes on to show that relying on the MED attribute of the BGP routes performs almost as well as relying on the LOCAL-PREF attribute. Finally, section 3.6 evaluates the performance of our solution for the second traffic objective introduced in section 1.2.

### 3.1 Scenario

Figure 6 illustrates the typical scenario of a multi-homed stub. It consists of a stub AS connected to several providers. The local stub AS shares one or more physical links with each provider. We assume that for each interdomain link, there is a corresponding eBGP session between the local egress router and the remote AS. A TE-route reflector [11, 23] centralizes the traffic statistics [36, 16, 57] as well as the BGP routes from the border routers of the domain. Two solutions can be envisioned to allow the TE-route reflector to know about all external routes known by the border routers. The first consists in establishing eBGP multihop sessions between the TE-route reflector and the border routers of the neighboring ASes. In that case, there must be one eBGP multihop session for each eBGP session between the border routers of the domain and the neighboring ASes for the TE-route reflector to know exactly the same BGP routes as all the border routers of the domain. Another solution would be to force the border routers of the domain to advertise all the routes they have learned from the neighboring ASes to the TE-route reflector, for instance by relying on [64]. Note that a solution like [64] will force each border router to advertise all its BGP routes present in the Adj-RIB-in's, not just one or two routes. The TE-route reflector is actually not used as a route-reflector by the AS but serves the sole purpose of learning the BGP routes and advertise the tweaked routes for traffic engineering purposes. Having all the known external BGP routes and the traffic statistics learned from the border routers of the domain, the TE-route reflector runs our search algorithm that computes which BGP routes should be tweaked and advertised to each border router to optimize the outbound traffic. To prevent routing loops from occurring inside the domain, the TE-route reflector must also know the IGP topology as well as the local policies applied by the border routers and compute the paths used by each border router to reach a destination. To deal with failures of the TE-route reflector, one might use two redundant TE-route reflectors, one announcing tweaked routes with a LOCAL-PREF value of $x$, and the other with a LOCAL-PREF value of $x - 1$.

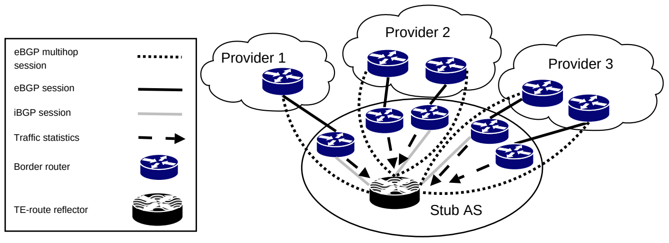

**Figure 6: Typical scenario for stub ASes.**

To perform its traffic engineering function, the TE-route reflector must send iBGP updates to the border routers of the domain. These iBGP updates are of two types. Whenever the TE-route reflector chooses to tweak a BGP route, i.e. to change the preferred route to reach a destination prefix, it must advertise to each border router which route is to be preferred. For example, if the TE-route reflector on Figure 6 decides that all routers of the domain must send their traffic towards prefix A.B.C.D/E through the border router connected to provider 1, then the TE-route reflector will send an iBGP update message to each router to indicate that the best route it knows to reach prefix A.B.C.D/E is to be preferred, for instance by putting a higher LOCAL-PREF value than all other routes the other border routers know to reach prefix A.B.C.D/E. Note that if all border routers have the same route to reach prefix A.B.C.D/E at the same time, no routing loop will occur inside the domain due to the outbound traffic engineering. The other type of iBGP update message sent by the TE-route reflector to border routers of the domain are those for the previously tweaked routes that are not to be tweaked during the next time interval anymore. In that case, the TE-route reflector will just resend to all border routers of the domain the best route computed by BGP among those known by the TE-route reflector. This iBGP update message only contains the BGP route which would be advertised by the TE-route reflector if it were acting as a traditional route reflector. If the best route chosen by the TE-route reflector stops from being reachable, then the TE-route reflector will behave like a normal route reflector and it will find another route to reach the destination prefix and advertise it to all the border routers.

### 3.2 Data

The first traffic trace used in our simulations is a six days trace of all the outbound traffic from the Université catholique de Louvain starting from March 19, 2003 00:00 CET. During this six days period, the university sent 1.7 terabytes of traffic or an average of 27 Mbps. This six days long trace was cut into time intervals of ten minutes. This trace was collected with nProbe [21] on a link just behind the access router of the university. No packet loss was reported by the system for the whole duration of the trace. For each ten minutes interval, we recorded the amount of bytes sent to each BGP prefix. In the remainder of this paper, we call this trace the *UCL trace*.

The second traffic trace is an 18 days long trace of all the outbound traffic from BELNET, a gigapop stub with 5 Gbps of capacity with its providers, starting from October 30, 2003 12:00 CET. The trace was collected using the NetFlow sampling available on the access routers of BELNET with a 1/4000 packet sampling rate. Figure 7 shows the evolution of the outbound traffic of BELNET during the 18 days of the trace. Since packet sampling was used, we multiplied the byte count per second of the sampled trace by 4000 to obtain the expected total traffic. The purpose of the trace from BELNET is to evaluate how larger stub ASes could behave under short-timescales BGP-based interdomain traffic engineering. Several traffic disruptions appear on the graph of Figure 7, they are only due to maintenance operations on the Netflow collector. Although no actual traffic disruption occurred on the access routers of BELNET during the duration of the trace, these periods can be considered as shutdown periods for the traffic engineering technique. In the remainder of this paper, we call this trace the *BELNET trace*.

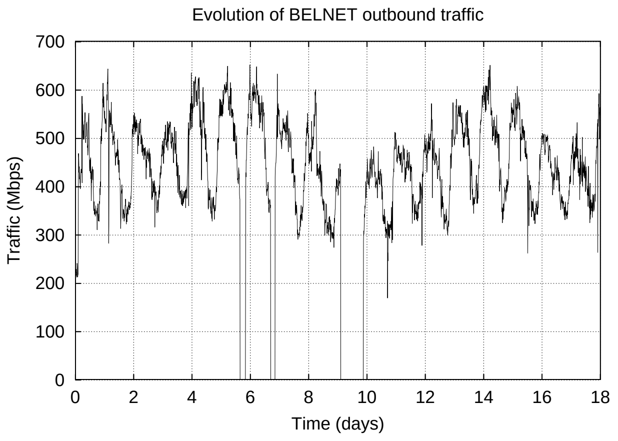

**Figure 7: Evolution of BELNET outbound traffic.**

In addition, we gathered routing information bases from Oregon Route views [38] dating from April 2, 2003 (for the UCL trace) and October 30, 2003 (for the BELNET trace) to simulate the BGP routing tables of the would-be providers to which our stubs would be connected. This particular choice of the would-be providers will merely impact the default distribution of the traffic among providers, not the performance of the optimization. As the purpose of this paper is not to provide guarantees about the working of our solution for any set of providers a stub AS could have, we do not deal with this aspect in the remainder of this paper. Note however that [10] studied how sensitive the BGP decision process was to the choice of the set of providers of a stub AS. [10] showed that when choosing a random pair of providers among those present in Oregon route-views, 60% of the routes had the same AS path length. This is an indication that the default distribution of the traffic for multi-homed stubs will depend largely on which tie-breaking rules of the BGP decision process are used to choose the best route, not on the particular providers of the AS.

While it is known that BGP routes dynamics can be important for some prefixes, it has however been shown in [49, 62, 25] that BGP routes can be considered as stable over relatively long periods for traffic destinations, so we use one BGP routing table per provider for the whole duration of each trace. One could also replay the eBGP updates received from would-be providers by using a tool like CBGP [45]. Note that in this paper we consider the stability over time of the amount of traffic per prefix, unlike [49] that considered the stability of the BGP routes over time, without respect to the time evolution of the traffic volume per BGP route. As the purpose of this paper is not to reproduce the exact routing of some network but to show the feasibility of short-timescales outbound interdomain traffic engineering for stubs, we did not consider this aspect as we do in [63]. For the UCL trace, the setting of the simulation was of 3 providers (AS1239, AS7018 and AS1668). For the BELNET trace, 3 providers have also been used (AS1239, AS3356 and AS1668). The impact of the choice of the particular BGP routing table used to simulate a would-be provider is limited, merely influencing the traffic imbalance under default BGP routing. Studying the impact of the choice of the particular provider on the default traffic distribution found by BGP is irrelevant in the context of this paper since its purpose is to demonstrate the feasibility of short-timescales outbound traffic engineering with BGP, not to quantify the expected gain of such traffic engineering.

## 3.3 Impact of traffic uncertainty

In this section, we study the effect of the impact of the predictors aimed at tracking the traffic dynamics on the quality of the achieved traffic balance. We also compare the results of the algorithm with and without the uncertainty of the traffic demands to understand to what extent the traffic dynamics prevents the algorithm to approach the optimal traffic balance.

Figure 8 plots the average traffic imbalance as a function of the number of iBGP updates allowed per 10 minutes interval, for the UCL trace. We call *imbalance ratio* the maximum amount of traffic sent through any of the three providers divided by the ideal traffic balance (total traffic divided by the number of providers). To compute the average, we took the whole non-dominated front for each time interval, and averaged the *imbalance ratio* for the six days of the UCL trace. Because we performed a non-domination sorting of the solutions based on their objective value and their number of BGP route changes, there is not always a non-dominated solution for a given number of BGP route changes for some time interval. This happens because a solution with fewer BGP route changes performed better those found with more BGP routing changes due to a changing traffic demand.

On Figure 8, we provide the results for four different states of uncertainty about the traffic demand. Each graph of Figure 8 provides for a given predictor the average gain in traffic imbalance together with the standard deviation around the average, as a function of the number of iBGP messages per time interval. For each graph of Figure 8, we provide for a given value of the number of iBGP messages the average gain in traffic imbalance as well as the average minus the standard deviation and the average plus the standard deviation. The top left graph of of Figure 8 corresponds to

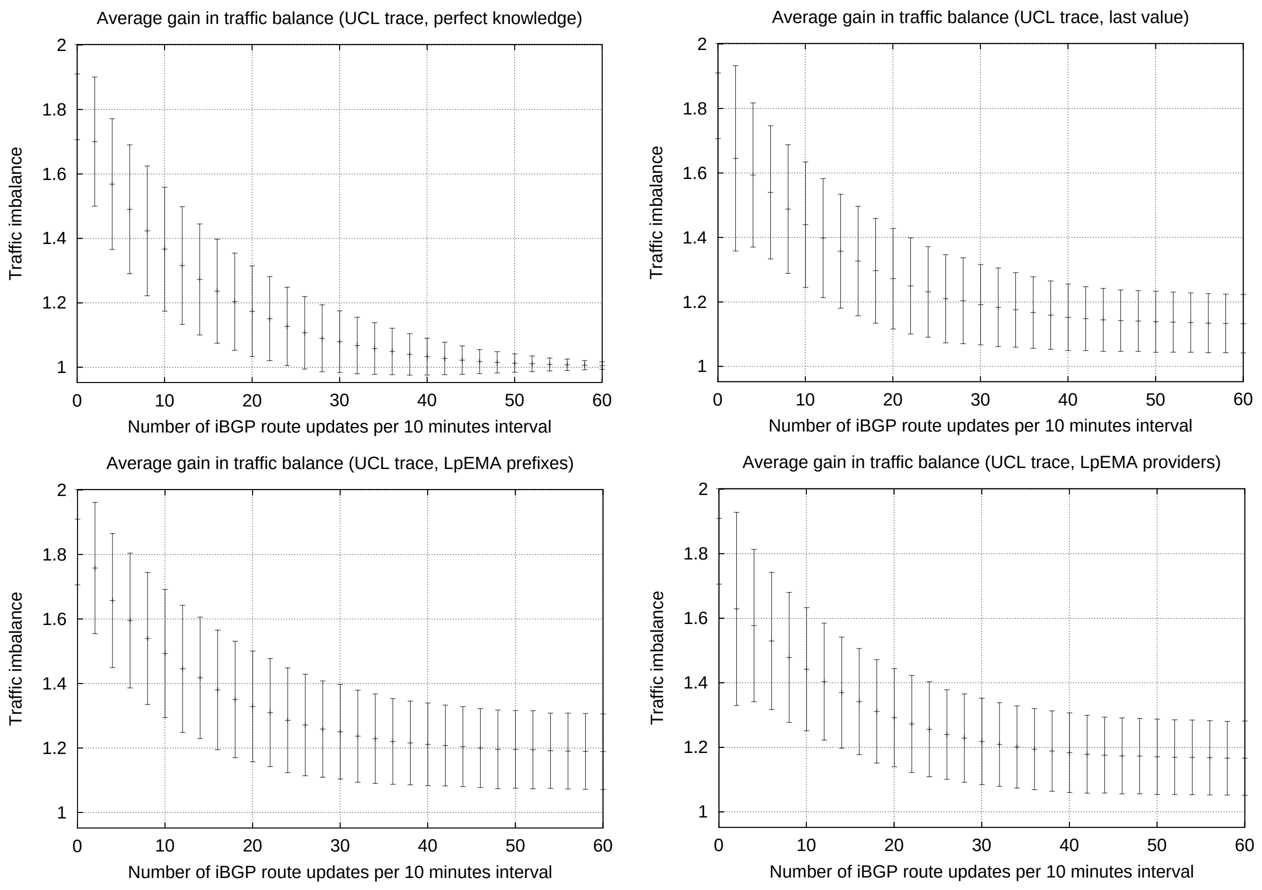

**Figure 8: Gain in traffic balance as a function of the number of iBGP updates.**

the case where the exact amount of traffic that will be sent towards each destination prefix is known. This situation is the one under perfect prediction of the traffic. The top right graph of Figure 8 corresponds to the traffic engineering scheme where the optimizer used the last value of the amount of traffic sent towards any destination prefix as the prediction for the current time interval. The two bottom graph of Figure 8 correspond to a particular EMA, the LpEMA scheme proposed in [14] to predict aggregated Internet traffic. The name LpEMA comes from low-pass EMA, to indicate that this is a smoothed adaptive EMA predictor. The bottom left (right) of Figure 8 corresponds to the LpEMA predictor where the traffic prediction is performed independently for each destination prefix (provider). One can notice that except for the perfect knowledge where the standard deviation decreases for larger numbers of iBGP messages, all prediction schemes have a comparable and quite large standard deviation. The decreasing standard deviation for the perfect knowledge is due to the fact that when increasing the number of iBGP messages, the optimizer achieves a smaller traffic imbalance without suffering from any uncertainty concerning the traffic. This standard deviation for the perfect knowledge hence represents mainly the traffic variability over time, not the uncertainty of the prediction. For the other three prediction schemes on the other hand, their increased standard deviation reflects the uncertainty of the traffic prediction. The standard deviation for the three predictors is similar.

The four graphs of Figure 8 start with the same value of the average traffic imbalance of about 1.9. The default route choice by the BGP decision process chose to send on average 1.9 times more traffic on one of the three providers than the average traffic over the three providers. With 10 iBGP updates, the average traffic imbalance lies below 1.5 for all traffic predictors. With 20, 30, 40, 50 and 60 iBGP updates, the average traffic imbalance is respectively below 1.33, 1.25, 1.21, 1.20 and 1.19. So for all practical purposes, relying on more than 40 iBGP updates provides a very limited improvement for the traffic balancing objective. The difference in the average traffic imbalance between the "last value" predictor and the prefix-based LpEMA is of about 0.05. This section confirms the results of section 2.1 showing that the LpEMA predictor was not better than the last value. Even under perfect knowledge of the future traffic demand, the improvement provided by more than 40 iBGP updates is of at most 0.03. Relying on many iBGP updates to try to improve the traffic objective will not provide a significant gain.

**The number of iBGP updates required for the optimization is small compared to the level of the "BGP noise". By BGP noise, we mean the iBGP updates that are routinely exchanged between BGP routers. A recent study at a large ISP [1] reports a BGP noise of more than one hundred iBGP updates per minute.**

While Figure 8 showed the average and standard deviation of the imbalance for the three traffic predictors, a single statistic cannot render the variation of the traffic imbalance over time. Figure 9 thus shows the cut of the non-dominated front for the perfect knowledge of the traffic demand, for each time interval and 0, 10 and 20 BGP

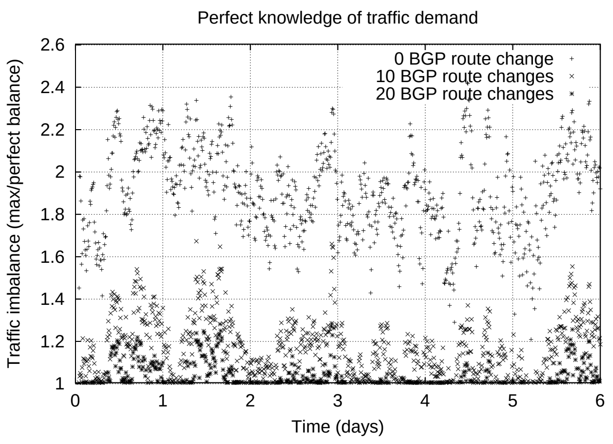

**Figure 9: Evolution over time of traffic imbalance under perfect knowledge of traffic demand.**

route changes. The sole purpose of Figure 9 is to illustrate the variability of the traffic imbalance over time. The points for 0 BGP route change on Figure 9 correspond to an *imbalance ratio* varying between 1.2 to 2.4. Hence without relying on outbound interdomain traffic engineering, our stub with its three providers would have a maximum load on any of its providers that varies by a factor of two over time. Although our traffic trace and the would-be providers we chosen might not be representative of a typical multihomed stub, it is likely that a stub that does rely on default BGP routing experiences important fluctuations in the load of its outbound traffic among its Internet access links. Tweaking BGP might thus prevent some stub ASes from having to over-provision their access links, without having to tweak a lot of BGP routes. Relying on 10 BGP route changes reduces this imbalance to less than 1.6 while 20 BGP route changes to less than 1.3. The time evolution of the traffic imbalance shows a broadly periodic behavior, with a larger imbalance during the busy hours of the day. The larger traffic imbalance during the busy hours of the day requires more BGP route changes to approach a given value of the traffic imbalance. This phenomenon is connected to the number of BGP prefixes having a large fraction of the traffic that also follows the time of the day pattern [62]. Influencing a given percentage of the total traffic during the busy hours thus requires to influence more prefixes than during the other parts of the day, at least on the considered traces.

*Under perfect knowledge of the traffic demand*, a limited number of BGP route changes could allow to better balance the traffic. In practice, the next traffic demand has to be predicted. The most simple way of predicting the amount of traffic seen for some prefix during the next time interval is to rely on the current amount of traffic seen. The top right graph of Figure 9 shows the evolution of the *imbalance ratio* for the "last value" prediction scheme, also for 0, 10 and 20 BGP route changes. The traffic objective value of each solution depends both on how well the current solution will work during the next time interval. The points for 10 and 20 BGP route changes on the other hand exhibit values of the *imbalance ratio* a little poorer than those under perfect knowledge of the traffic demand, with most solutions with 10 BGP route changes below an *imbalance ratio* of 1.5 and 1.4 for 20 BGP route changes. The time evolution of the *imbalance ratio* on the bottom graphs of Figure 9 for the two LpEMA predictors confirm the poorer average *imbalance ratio* shown on Figure 8. The two LpEMA predictors provide poorer results than the "last value" predictor, consistently over time and not only on average as shown by Figure 8.

**Complex traffic predictors as LpEMA do not provide any advantage in terms of the *imbalance ratio* for interdomain traffic engineering purposes. For practical purposes, relying on the simplest predictor −the current traffic demand− to engineer the interdomain traffic seems the best choice.**

## 3.4 Tabu prefixes

In the previous section, we saw that relying on complex predictors was useless for traffic engineering purposes. From now on, we only rely on the "last value" predictor. Furthermore, we saw that a limited number of iBGP updates were enough to obtain a satisfactory traffic balance. An issue with the short-timescales traffic engineering we did not mention yet is that some BGP route might be tweaked back and forth during consecutive time intervals. Traffic flows could have to switch from one provider to another. We do not wish that interdomain traffic engineering changes the path followed by the traffic too often. If the traffic engineering technique decides to change the provider used to carry the traffic towards some destination prefix, it would be desirable to stick to this choice for a sufficiently long time to limit the effect on the instability of the topological distribution of the traffic [62].

For that purpose, we added a *tabu list* that maintains the set of BGP prefixes for which the traffic engineering technique modified the route attributes during the last $x$ time intervals. BGP prefixes cannot be tweaked by the algorithm as long as they are present in the *tabu list*, these are "tabu prefixes". The traffic for the BGP prefixes present in the *tabu list* cannot be moved to another provider by the algorithm during $x$ time intervals, starting from the time interval during which this BGP route change was applied. The *tabu list* contains the BGP routes changed by the algorithm during the last $x$ time intervals, hence BGP prefixes and the associated preferred provider. If the provider chosen as the best route by the traffic engineering scheme fails, the BGP next hop of the tweaked route will become unreachable and the traffic will be automatically switched to another provider as during normal BGP routing.

To evaluate the practical interest of the *tabu list* method, two questions must be addressed:

1. How many new BGP route changes should be accepted per time interval?
2. For how long should a BGP route change be present in the tabu list?

Figure 10 plots the average traffic imbalance for several values of the number of the iBGP updates per time interval (10, 20, 30 and 40), using various lifetimes of the *tabu list* entries. The purpose of Figure 10 is to show that the average traffic imbalance achieved by the *tabu list* method mainly depends on the total number of BGP routes tweaked by the technique during any time interval. We insist that what we call the number of BGP routes tweaked during any time interval is different from the number of iBGP messages sent during any time interval. This is so because iBGP messages are sent during some time interval only for the BGP routes that will be tweaked during the next time interval and are not yet in the *tabu list*. Hence the total number of BGP routes tweaked during any time interval is equal to the number of entries of the *tabu list* plus the BGP routes tweaked corresponding to the iBGP messages sent.

On the x-axis of Figure 10, we show the the total number of iBGP routes that have been tweaked by the traffic engineering technique for any time interval. This total number of BGP routes that have been tweaked is not the same as the number of iBGP updates because with the *tabu list* method, we have no iBGP message for the entries of the *tabu list* that stay the same between two consecutive time intervals. With the *tabu list* method, there are iBGP updates only for the BGP routes whose attributes are reset to the default value (entries of the *tabu list* that expire) as well as iBGP updates for the BGP routes that are tweaked by the traffic engineering technique (entries that enter the *tabu list*).

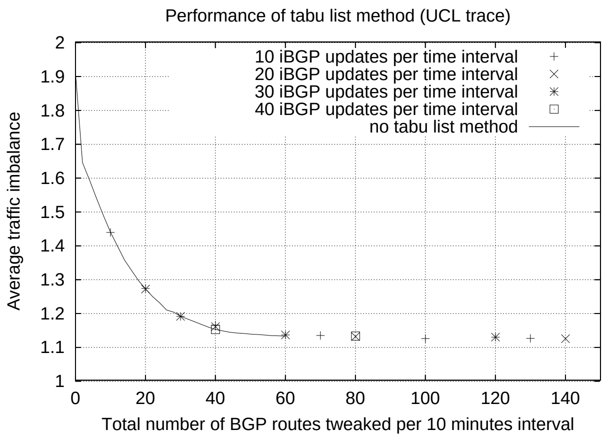

**Figure 10: Insensitivity of *tabu list* method to number of iBGP messages sent.**

Let $m$ represent the number of time intervals that an entry of the *tabu list* remains in the *tabu list* and let $n$ represent the number of new *tabu list* entries accepted per time interval. Each time interval, there are $2n$ iBGP updates, $n$ for the new *tabu list* entries (newly tweaked routes) and $n$ for the old *tabu list* entries that expire. An entry of the *tabu list* expires when it has been present in the *tabu list* for more than $m$ time intervals. During each time interval, there are $n \times m$ *tabu list* entries that have not yet expired, as well as $n$ new ones selected to enter the *tabu list*, hence a total of $n \times (m+1)$ BGP routes that are tweaked.

Figure 10 shows that only the total number of BGP routes tweaked during any time interval drives the achieved average *imbalance ratio*. The number of BGP routes that are currently being tweaked by the traffic engineering technique corresponds to the number of active entries in the *tabu list*. On the UCL trace, there is ample choice for the number of iBGP updates allowed as well as the number of *tabu list* entries present during each time interval. By changing the size of the *tabu list*, the network operator can choose the acceptable burden in terms of iBGP messages as well as how close he wants to get from the optimal traffic balance achievable in practice. The two questions above are thus interrelated and only the total number of modifications with respect to the default BGP routes seems to be relevant for what concerns the value of the traffic objective achieved. On Figure 10, we also reproduce the results obtained without relying on a *tabu list* ("last value" curve of Figure 8), for up to 60 BGP route changes. The *tabu list* results are seen to match those not relying on a *tabu list*, with the points for the *tabu list* results closely following the "no tabu list" curve. This indicates that, if the number of iBGP messages to be sent per time interval is fixed, the value of the traffic objective largely depends on the number of *tabu list* entries.

The benefits of the method are obvious when comparisons are made with respect to the number of iBGP updates as this technique has been especially designed to limit the number of iBGP messages. Figure 11 provides, for the BELNET trace, the *imbalance ratio* as a function of the number of iBGP updates per time interval with the *tabu list* method. The comparison made on Figure 11 is intentionally unfair towards the "no tabu list" method since the *tabu list* method "cheats" by not requiring to send iBGP messages for BGP routes that stay in the *tabu list* during consecutive time intervals. The continuous curve on Figure 11 labeled "no tabu list" gives the improvement in traffic balance when the optimization method does not take into account the BGP routes that have been tweaked by the algorithm in the past. The continuous curve hence provides an upper bound on the number of iBGP updates per time interval to achieve a given traffic balance. The other three curves on Figure 11, labeled "tabu entry lifetime = x min", correspond to the result of the *tabu list* method where the entries lifetime is of $x$ minutes ($x = 10$, 20, 30 and 60). The "no tabu list" method for the BELNET trace starts (default BGP routing) with an *imbalance ratio* of less than 1.5 and the improvement in the *imbalance ratio* decreases slowly, to stick to a traffic imbalance of about 1.06 for more than 30 iBGP updates per time interval. The curves corresponding to the *tabu list* method on the other hand start from a lower *imbalance ratio* thanks to their larger number of total BGP routes tweaked during any given time interval. With only 10 iBGP updates per time interval, the *tabu list* method is able to achieve an *imbalance ratio* a little above 1.06 for a *tabu list* entries lifetime of 30 and 60 minutes.

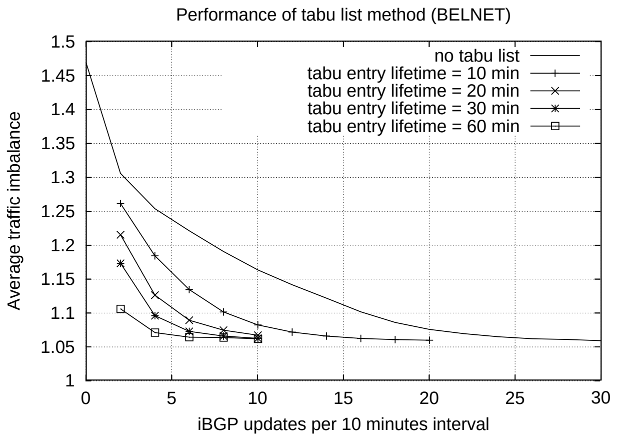

**Figure 11: Reduction of the number of iBGP messages with the *tabu list* method.**

**The tabu list method has the advantage of allowing to reduce the burden on the number of iBGP updates to be sent during each time interval, without impacting on the quality of the traffic engineering. Maintaining a list of tabu BGP routes is thus extremely interesting from a practical viewpoint since it performs better in terms of the *imbalance ratio* and the burden on BGP, while providing stability to the best BGP route choice.**

## 3.5 MED tweaking

Up to now, the tweaking of the BGP routes was implemented by means of the LOCAL-PREF attribute to ensure that the optimization will override all other BGP tweaking. This way of tweaking the BGP routes could make the BGP decision process choose a route with a longer AS path in order to optimize the traffic balance, potentially leading to a worse end-to-end quality of the routes. [31] has however shown that path length and performace were not correlated. If the on-line optimization is required not to override other kinds of traffic engineering (e.g. MED or IGP cost), then ideally one would prefer that the optimization tweaks a BGP attribute that is evaluated by the BGP decision process just before the last rule (lowest ROUTER-ID [53]).

In the case of non-transit ASes, the optimization could rely on the MED attribute. According to the BGP Internet draft [48], the MED attribute cannot be compared between routes learned from different ASes, the route with the lowest MED being best for the BGP decision process. Allowing to always compare the MED attribute among BGP routes is allowed on many routers, see [30] p. 166. Although there is no reason a priori to compare the MED attribute of routes learned from different neighboring ASes, stub ASes could rely on the MED attribute or another attribute having a meaning local to the AS to perform traffic engineering. Only limited modifications to BGP would be needed.

It must be noted that in practice relying on MED to tweak the BGP routes might lead to oscillations problems [8]. The presence of route-reflectors, as in the scenario proposed in section 3.1, makes relying on MED potentially dangerous. The problems caused by MED would not arise if some BGP attribute used after the IGP cost rule of the decision process was available. If such traffic engineering techniques as proposed in this paper become widely used by stub ASes, adding a new rule to the decision process after the IGP cost one to allow interdomain traffic engineering to prefer equal-cost BGP routes could be envisioned. The question of whether one should change the decision process in the whole Internet for a limited number of stub ASes that perform BGP-based traffic engineering is outside the scope of this paper.

The reader might ask why MED should be used at all to perform traffic engineering. As LOCAL-PREF is already used to enforce policies, using it for traffic engineering purposes will force ISPs to choose one range of values of LOCAL-PREF for policy routing and another for traffic engineering. This could lead in messing up the two types of BGP tweakings. Furthermore, using LOCAL-PREF for traffic engineering should not override LOCAL-PREF used to enforce routing policies, hence using another attribute seems to us a better solution from a configuration viewpoint.

In this section, we allow the BGP instance running on the TE-route reflector of the local AS to set the MED attribute of the routes received from the different neighboring ASes to prefer one route over another. The working of the optimization technique is similar to the one presented in section 3.4, except that the preferred route for the optimization algorithm gets a lower MED value compared to the other routes towards a given prefix instead of having a higher value of the LOCAL-PREF attribute. The main difference with the algorithm of section 3.4 is that only routes having the same LOCAL-PREF value and AS path length will have their MED value compared. The optimization technique has thus a more limited choice in the prefixes that can be used to balance the traffic, compared to the LOCAL-PREF way.

Figure 12 presents the simulation results of the *tabu list* method with the MED attribute for UCL. Figure 12 shows the average *imbalance ratio* as a function of the number of iBGP updates per 10 minutes interval. Figure 12 is the counterpart of Figure 11. Figure 12 shows that the average *imbalance ratio* of the *tabu list* method with MED is quite comparable to the one with the LOCAL-PREF attribute, only slightly worse. The two continuous curves on Figure 12 give the simulation results for the "no tabu list" method, with LOCAL-PREF and MED. The dotted lines of Figure 12 on the other hand provide the simulation results for the *tabu list* method by tweaking MED. Increasing the size of the *tabu list* (by increasing the entries lifetime) provides a significant reduction of the number of iBGP updates to be announced.

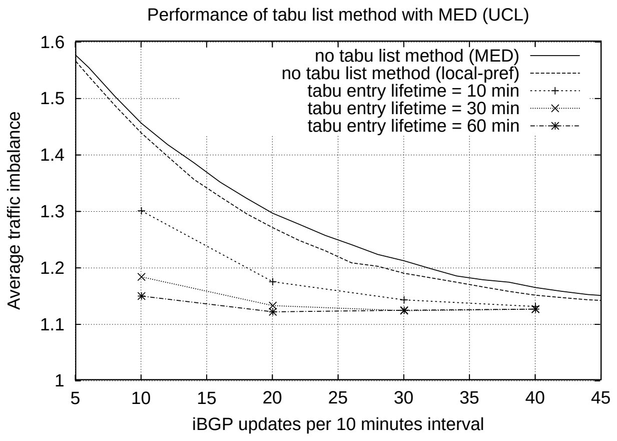

**Figure 12: Using MED to tweak the BGP routes.**

**Relying on the MED attribute instead of the LOCAL-PREF attribute to tweak the BGP routes of stub ASes does not significantly worsens the achieved average *imbalance ratio*. It only limits the choice of the BGP routes that can be tweaked. Tweaking the BGP routes without over-riding the policy routing performed by stub ASes hence still leaves room for the *tabu list* technique to work properly.**

## 3.6 Cost-weighted traffic objective

The previous sections have provided simulations with a relatively simple traffic objective. In this section, we provide simulation results with the cost-weighted traffic objective introduced in section 1.2 for the traffic of BELNET. This objective is related to the cost of the interdomain traffic, it is thus particularly relevant in practice. The cost-weighted traffic objective is defined as *min* $\sum_{i=1}^{n} c_i \times tr_i$ where $c_i > 0 \;\; \forall i$, denotes the cost of one unit of traffic sent to provider $i$. We used the following costs: $c_1 = 1.5, c_2 = 1.25$ and $c_3 = 1$.

We left the costs $c_i$ fixed in the following simulations but nothing in our scheme prevents the costs from varying with time or depending on other parameters. For instance, a larger cost can be attributed to the traffic sent to a provider during the busy hours or the cost could depend on the relative load of the access links. The sole constraint on the objective function is that the impact on the objective function of a change in the best BGP route for some destination prefix must be known for the search algorithm to be able to improve it. However, no assumption on the look of the objective function (convexity, piecewise linearity,...) is made by the algorithm.

Figure 13 presents the results of the interdomain traffic engineering when using the LOCAL-PREF attribute and for the traffic of BELNET. The y-axis of Figure 13 represents the average normalized cost, i.e. the value of the traffic objective divided by the optimal cost. The x-axis represents the number of iBGP updates to be made per 10 minutes time interval, for up to 40 iBGP updates. The top curve on Figure 13 provides the result of the simulation without use of the *tabu list* method. The three bottom curves show the results using the *tabu list* method and with *tabu list* entries lifetimes of 30 minutes, 1 hour and 2 hours. As shown by the slower decrease rate of the top curve of Figure 13, the cost-weighted traffic objective is more difficult to optimize. More iBGP updates are required to approach the optimal cost compared to the traffic balancing objective of the previous sections. The initial value of the cost-weighted traffic objective found by BGP is 1.33. Without the tabu list method, 20 iBGP updates per 10 minutes interval yield an average normalized cost of 1.2, 100 iBGP advertisements an average normalized cost of 1.1 (not shown on Figure 13), and 180 iBGP updates an average normalized cost of 1.06 (not shown on Figure 13).

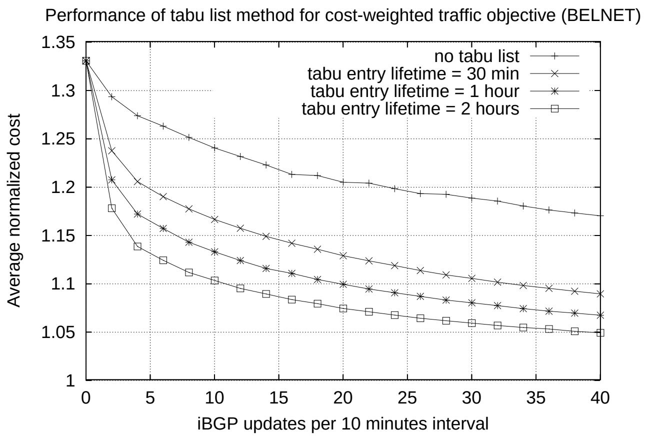

**Figure 13: Cost-weighted traffic objective.**

**As for the traffic balancing objective, the *tabu list* method allows in the case of the cost-weighted traffic objective to drastically reduce the number of iBGP updates required to achieve a given gain in terms of the traffic objective. The larger the lifetime of the tabu list entries, the smaller the number of BGP advertisements required to reach a given average normalized cost.**

## 4. EVALUATION

Although we limited the scope of this paper to traffic balancing and the cost-weighted traffic objectives, the results for other traffic objectives will be similar qualitatively. The main difference will be in the absolute value of the average optimization gain, that are context dependent anyway. Both the initial traffic objective value found by the default BGP traffic distribution as well as the particular traffic objective to be optimized drive the actual optimization gain to be achieved in practice. [61] relied on other traffic objectives and the look of the Pareto-front has in every case been found to be similar to the one found on Figure 8. The reason for the convexity of the traffic objective value for increasing BGP route changes allowed merely lies in the decreasing amount of traffic sent towards the largest BGP routes (when ordered by decreasing byte count). Because of the implicit constraint of the algorithm to try to minimize the number of BGP route changes, the algorithm often chooses the BGP routes for which there is the largest amount of traffic that provide the largest gain in the traffic objective. This is however only true for the first few BGP routes tweaked. When approaching the optimum of the traffic objective on the other hand, improving the traffic objective requires a non-trivial choice of the set of BGP routes to tweak.

In this paper we mostly relied on the LOCAL-PREF attribute to tweak the BGP routes. As explained in the previous paragraph, this is likely to interfere with IGP-based traffic engineering techniques [26]. Our solution can also work on the MED attribute of the BGP routes, as shown in section 3.5. Working on the MED attribute would allow a transit provider to tweak the BGP routes that have the same value of the LOCAL-PREF attribute and the same AS path length. Since the MED attribute can be reset when advertising the best route to a peer, it would only affect the choice of which BGP route among those having the same AS path length will be advertised. If other peer ASes do not filter the BGP routes based on the AS numbers of the AS path, then their choice of the best BGP route should not be influenced too much by the MED tweaking.

The simulations of section 3 relied on traffic objectives that were functions of the total traffic sent through each provider. In the case of objective functions that depend on end-to-end properties of the path followed by the IP packets, a BGP-based solution might not be desirable. The possibility of specifying objectives that are functions of the BGP routes does not imply that optimizing these objectives should be done through BGP. For instance, "route optimization" [4, 33] techniques seem to choose the best BGP route towards a particular destination according to the measured "quality" of the end-to-end path between the local AS and the destination. If the end-to-end properties of the routes change too fast compared to the timescale at which the BGP advertisements are made by the interdomain traffic engineering technique (about a few minutes) then it probably means that BGP should not be used to perform such traffic engineering. What we mean is that traffic engineering techniques that require too many BGP advertisements per minute should probably be implemented by means of other means than the interdomain routing protocol because BGP has been designed as a reachability protocol [32], not a traffic engineering tool. Our technique could however be useful for route optimization techniques to limit their burden on BGP. As our technique is able to work with several objectives at the same time, our technique could be integrated in route optimization techniques so that both performance aspects and the impact on BGP are optimized.

The main result of this paper is to show that tracking the dynamics of the outbound interdomain traffic of a stub AS is possible with a limited burden on BGP. We showed in section 3.4 that with 5 BGP route changes every 10 minutes interval, a stub AS could balance its outbound traffic within 6 % of the perfect traffic balance on average. However, it is not clear whether working on timescales as small as minutes is practically relevant for interdomain traffic engineering purposes. The technique we proposed in this paper can work on any timescale larger than minutes. In this paper we relied on a very small time granularity of 10 minutes so as to demonstrate that even on such a small timescale BGP-based traffic engineering is possible in stub ASes at a very small cost in terms of the number of iBGP updates required.

## 5. CONCLUSION

In today's Internet, for both cost and performance reasons, Internet Service Providers often need to engineer their interdomain traffic. This interdomain traffic engineering is often done by manually tweaking the configurations of the BGP routers on an error-prone trial-and-error basis.

In this paper, we have proposed a systematic approach to solve the important operational problem of designing an interdomain traffic engineering technique to optimize the outbound traffic of stub ASes. Our approach allows the network operator to define objective functions on the interdomain traffic. Those objective functions are used by an optimization box placed inside the AS that controls the traffic. For this, it relies on the BGP routes and the traffic statistics received from the border routers of the AS. Based on the traffic statistics from the last period, the optimization box uses an efficient evolutionary algorithm to select the required iBGP changes to optimize the traffic during the next period. Those iBGP changes are then distributed inside the AS.

We have then applied our solution to the design of traffic engineering techniques to optimize the outbound traffic of stub ASes over timescales of minutes. By way of simulations with two different objective functions, we showed that the outgoing traffic can be efficiently engineered on a 10 minutes timescale by advertising not more than a few iBGP messages per minute per border router. This is lower than the BGP noise in the Internet. Our Perl prototype implementation can be used in real-time since it requires only a few seconds of CPU time on a P4 2.4 GHz to select the required iBGP changes for each 10 minutes period. All Perl scripts used in this paper are available at http://www.info.ucl.ac.be/ suh/.

## 6. ACKNOWLEDGMENTS

The authors wish to thank BELNET, the Belgian research network, and particularly Jan Torreele and Pascal Panneels, for providing the BGP session and the traffic trace. The authors also acknowledge the insightful comments of the anonymous reviewers. This work was partially supported by the Walloon Government (DGTRE) within the TOTEM project (http://totem.info.ucl.ac.be).

## 7. REFERENCES

[1] S. Agarwal, C. Chuah, S. Bhattacharyya, and C. Diot. The Impact of BGP Dynamics on Intra-Domain Traffic. In *Proc. of ACM SIGMETRICS*, June 2004.

[2] S. Agarwal, C. Chuah, and R. Katz. Opca: Robust interdomain policy routing and traffic control. In *IEEE Openarch*, 2003.

[3] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A measurement-based analysis of multihoming. In *Proceedings of ACM SIGCOMM 2003*, August 2003.

[4] D. Allen. NPN: Multihoming and route optimization: Finding the best way home. *Network Magazine*, February 2002.

[5] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter. RATES: A server for MPLS traffic engineering. *IEEE Network Magazine, pages 34–41*, March/April 2000.

[6] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. Internet Engineering Task Force, RFC3272, May 2002.

[7] D. Awduche, J. Malcom, B. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. Internet RFC 2702, September 1999.

[8] A. Basu, C. Ong, A. Rasala, F. Shepherd, and G. Wilfong. Route oscillations in i-BGP with route reflection. In *Proceedings of ACM SIGCOMM 2002*, August 2002.

[9] O. Bonaventure, B. Quoitin, and S. Uhlig. Beyond interdomain reachability. Position paper at the Workshop on Internet Routing Evolution and Design (WIRED), October 2003.

[10] O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin (Eds.), A. Azcorra, M. Bagnulo, P. Flegkas, A. Garcia-Martinez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, S. Tandel, and S. Uhlig. Internet Traffic Engineering. Chapter of COST263 final report, LNCS 2856, Springer-Verlag, September 2003.

[11] O. Bonaventure, S. Uhlig, and B. Quoitin. The case for more versatile BGP Route Reflectors. Internet draft, draft-bonaventure-bgp-route-reflectors-00.txt, work in progress, July 2004.

[12] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice-Hall, 1994.

[13] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.

[14] L. Burgstahler and M. Neubauer. New Modifications of the Exponential Moving Average Algorithm for Bandwidth Estimation. In *Proc. of the 15th ITC Specialist Seminar*, July 2002.

[15] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Towards Capturing Representative AS-Level Internet Topologies. *Computer Networks Journal, Elsevier*, 44(6):737–755, April 2004.

[16] B. Claise. Packet Sampling (PSAMP) Protocol Specifications. Internet draft, draft-ietf-psamp-protocol-01.txt, work in progress, February 2004.

[17] C. A. Coello Coello. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32(2):109–143, 2000.

[18] INTERNAP NETWORK SERVICES CORP. Internap Flow Control Platform. `http://www.internap.com/`.

[19] K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Wiley Interscience series in systems and optimization, 2001.

[20] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proc. of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer-Verlag (LNCS 1917).

[21] L. Deri. nProbe: an Open Source NetFlow probe for Gigabit Networks. In *Proc. of Terena TNC 2003*, May 2003.

[22] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan, and J. van der Merwe. Resource management with hoses: point-to-cloud services for virtual private networks. *IEEE/ACM Transactions on Networking*, 10(5):679–692, 2002.

[23] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. van der Merwe. The case for separating routing from routers. In *Proc. SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, August 2004.

[24] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for interdomain traffic engineering. *ACM SIGCOMM Comput. Commun. Rev.*, 33(5):19–30, 2003.

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

[26] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, October 2002.

[27] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM2000*, March 2000.

[28] D. Goldenberg, L. Qiu, H. Xie, Y. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In *Proceedings of ACM SIGCOMM 2004*, August 2004.

[29] T. Griffin and G. Wilfong. An analysis of BGP convergence properties. In *Proc. of ACM SIGCOMM'99*, September 1999.

[30] B. Halabi. *Internet Routing Architectures (2$^{nd}$ edition)*. Cisco Press, 2000.

[31] B. Huffaker, M. Fomenkov, D. Plummer, D. Moore, and K. Claffy. Distance Metrics in the Internet. In *Proc. of IEEE International Telecommunications Symposium (ITS)*, September 2002.

[32] J. Johnson. BGP is a reachability protocol. NANOG25 meeting, Toronto, Canada. June 2002. Available at `http://www.nanog.org/mtg-0206/ppt/jerm2/index.html`.

[33] J. Johnson. Intelligent route control improves BGP. *Network World*, February 2002.

[34] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener. Algorithms for provisioning virtual private networks in the hose model. *IEEE/ACM Transactions on Networking*, 10(4):565–578, 2002.

[35] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. An experimental study of Internet routing convergence. In *SIGCOMM 2000*, August 2000.

[36] S. Leinen. Evaluation of candidate protocols for IP flow information export (IPFIX). Internet draft, draft-leinen-ipfix-eval-contrib-02, work in progress, January 2004.

[37] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP Misconfigurations. In *Proc. ACM SIGCOMM '02*, September 2002.

[38] D. Meyer. University of Oregon Route Views Project. Available at `http://antc.uoregon.edu/route-views/`.

[39] P. Morrissey. Mapping out the best route. Network Computing, `http://www.nwc.com/showArticle.jhtml?articleID=16401572`, December 2003.

[40] F5 NETWORKS. Big-IP Link Controller. `http://www.f5.com`.

[41] Juniper Networks. Junos software release 5.6 : New features list. `http://www.juniper.net/products/ip_infrastructure/junos/105012.html`.

[42] K. Park and W. Willinger (editors). *Self-Similar Network Traffic and Performance Evaluation*. Wiley-Interscience, 2000.

[43] R. Purshouse and P. Fleming. Conflict, Harmony, and Independence: Relationships in Multi-criterion Optimisation. In *Proc. of the Second International Conference on Multi-Criterion Optimization (EMO2003), Portugal*, pages 16–30, April 2003.

[44] Y. Qiao, J. Skicewicz, and P. Dinda. Multiscale predictability of network traffic. Technical Report NWU-CS-02-13, Nothwestern University, October 2002.

[45] B. Quoitin. C-BGP, an efficient BGP simulator. `http://cbgp.info.ucl.ac.be/`, September 2003.

[46] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain traffic engineering with BGP. *IEEE Communications Magazine*, May 2003.

[47] RADWARE. Linkproof. `http://www.radware.com/`.

[48] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). Internet draft, draft-ietf-idr-bgp4-24.txt, work in progress, November 2003.

[49] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP Routing Stability of Popular Destinations. In *Proc. of the second ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[50] A. Sang and S. Li. A Predictability Analysis of Network Traffic. In *Proc. of IEEE INFOCOM 2000*, 2000.

[51] S. Sangli, D. Tappan, and Y. Rekhter. BGP Extended Communities Attribute. Internet draft, draft-ietf-idr-bgp-ext-communities-06.txt, work in progress, August 2003.

[52] P. Smith. Weekly routing table report. Weekly reports from APNIC's router in Japan sent to `bgp-stats@lists.apnic.net`.

[53] J. Stewart. *BGP4 : interdomain routing in the Internet*. Addison Wesley, 1999.

[54] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *INFOCOM 2002*, June 2002.

[55] Cisco Systems. BGP Case Studies Section 1. `http://www.cisco.com/warp/public/459/13.html`.

[56] Cisco Systems. Sample Configurations for Load Sharing with BGP in Single and Multihomed Environments. `http://www.cisco.com/warp/public/459/40.html`.

[57] Cisco Systems. NetFlow services and applications. White papern, `http://www.cisco.com/warp/public/732/netflow`, 1999.

[58] ROUTESCIENCE TECHNOLOGIES. PathControl. `http://www.routescience.com/`.

[59] S. Uhlig. Implications of the traffic characteristics on interdomain traffic engineering. PhD Thesis, Computer Science and Engineering Department, Université catholique de Louvain, March 2004.

[60] S. Uhlig. A multiple-objectives evolutionary perspective to interdomain traffic engineering in the internet. In *Workshop on Nature Inspired Approaches to Networks and Telecommunications (NIANT) in PPSN04, Birmingham, UK*, September 2004.

[61] S. Uhlig, O. Bonaventure, and B. Quoitin. Interdomain Traffic Engineering with minimal BGP Configurations. In *Proc. of ITC-18*, September 2003.

[62] S. Uhlig, V. Magnin, O. Bonaventure, C. Rapier, and L. Deri. Implications of the Topological Properties of Internet Traffic on Traffic Engineering. In *Proc. of the 19th ACM Symposium on Applied Computing*, March 2004.

[63] S. Uhlig and B. Quoitin. BGP-based interdomain traffic engineering for transit ASes. Under submission, `http://cbgp.info.ucl.ac.be/apps.html#section_transit_te`.

[64] D. Walton, D. Cook, A. Retana, and J. Scudder. Advertisement of multiple paths in BGP. Internet draft, draft-walton-bgp-add-paths-01.txt, work in progress, November 2002.

[65] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *INFOCOM2001*, April 2001.

[66] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic engineering with MPLS in the Internet. *IEEE Network Magazine*, March 2000.

[67] T. Ye and S. Kalyanaraman. A recursive random search algorithm for large-scale network parameter configuration. In *Proc. of ACM SIGMETRICS'03*, 2003.