Tweak-it: BGP-based Interdomain Traffic Engineering for transit ASs

Steve Uhlig and Bruno Quoitin
Department of Computing Sciences and Engineering
Université catholique de Louvain, Louvain-la-Neuve, Belgium
{suh,bqu}@info.ucl.ac.be

Abstract— **Today, engineering interdomain traffic in large transit ASs is a difficult task due the opacity of BGP and the interactions between the BGP decision process and IGP routing. In this paper we propose *Tweak-it*, a tool that, based on the steady-state view of BGP routing inside the AS and the traffic demands of the AS, computes the BGP updates to be sent to the ingress routers of a transit AS to traffic engineer its interdomain traffic over time.**

***Tweak-it* is based on two components: 1) a scalable BGP simulator (CBGP) that computes the steady-state behavior of BGP routing and 2) a multiple-objectives evolutionary heuristic that can deal with multiple conflicting objectives as they can occur in real networks. *Tweak-it* takes the intradomain configuration (IGP weights and topology), BGP messages received from peers, BGP routing policies, and traffic demands. By keeping the state of the routing inside the AS up-to-date and based on the traffic demands, the heuristics computes how to engineer the traffic of the AS while trying to minimize the number of BGP tweakings required.**

Keywords— **traffic management, interdomain traffic engineering, BGP, multiple-objectives evolutionary optimization.**

I. Introduction

Most of the current effort in traffic engineering has been spent on intradomain aspects [1], [2]. Even though techniques exist to optimize the IGP weights so as to optimize the intradomain routing inside large providers' networks [2], most providers today still rely on manual tuning to appropriately set their IGP weights. At the interdomain level, the current state-of-the-art has not gone beyond general guidelines for interdomain traffic engineering [3]. Recent efforts at the interdomain level still concern computing the state of the BGP routes inside an autonomous system (AS) [4] and checking that BGP is properly configured inside an AS [5]. Operational practices include changing routing policies and the BGP attributes of the routes manually without a thorough understanding of such changes on the flow of the traffic. These practices sometimes lead to routers misconfigurations [6] and the associated unexpected routing problems.

Interdomain traffic engineering in transit providers is a non-trivial task [3]. As transit providers typically carry large amounts of traffic, one of the important traffic engineering objectives they want to consider is to minimize the time IP packets travel across their network. The idea of minimizing the time packets travel inside the AS is sometimes called hot-potato routing [7]. Hot-potato routing consists in choosing the exit point in the network that is closest to the ingress point where the IP packet was received, for instance in terms of the IGP cost. At the same time, transit ISPs also want to prevent high loads on their links, so as to minimize delay due to the transit in their network, to minimize the cost of their peerings, etc. With uneven traffic distribution between ingress-egress pairs and changing traffic demands [8], interdomain traffic engineering may require using different exit points than those driven by hot-potato routing practices. Interdomain traffic engineering for transit ASs thus consists in practice in dealing with several potentially conflicting objectives.

One of the current issues to properly evaluate interdomain traffic engineering is to reproduce the state of the BGP routing inside a large network [4]. As a limited fraction of the prefixes carry the vast majority of the interdomain traffic [9], [10], [11] and these popular prefixes have stable BGP routes [11], reproducing in minutest details the routing inside the AS is not necessary for traffic engineering purposes [4]. Furthermore, BGP routes impacted by IGP routing instabilities or instable BGP routes should preferably not be considered when doing interdomain traffic engineering. To prevent traffic engineering to act on unstable routes, one could think of maintaining per-route statistics and only allow the tweaking to be performed on routes that did not change for the last couple of hours.

In this paper, we propose *Tweak-it*, a tool that combines a multiple-objectives evolutionary heuristic with a BGP simulator to compute how to tweak the BGP routing inside a transit AS so as to engineer the distribution of the interdomain traffic. As practical traffic engineering objectives can be conflicting, there is no single optimal solution, but rather a front of Pareto-optimal solutions. This motivates our choice of our heuristic, a multiple-objective evolutionary algorithm that allows to go beyond the limitations of traditional optimization approaches that have difficulties to sample a Pareto-optimal front in a single run [12]. To efficiently compute the state of the routing inside a large transit AS, we developed CBGP, an open source BGP simulator that computes the steady-state solution found by BGP routing [13]. All the code used in this paper is open source and freely available at [14]. Synthetic traffic, configuration files and simulation outputs for different versions of the tool can be also found at [14].

The remainder of this paper is structured as follows. Section II first explains the complexity of BGP-based traffic engineering in transit ASs. Section III presents the related work. Section IV presents the architecture of *Tweak-it*. Section V presents simulation results on the GEANT network. Section VI then concludes and discusses the further work.

II. BGP-based traffic engineering by transit ASs

To explain the complexity of BGP-based traffic engineering, let us consider the example of Figure 1 (left diagram). In this figure, we show a transit AS that wants to tweak its traffic to-

0-7803-8901-8/05/$20.00 (c) 2005 IEEE

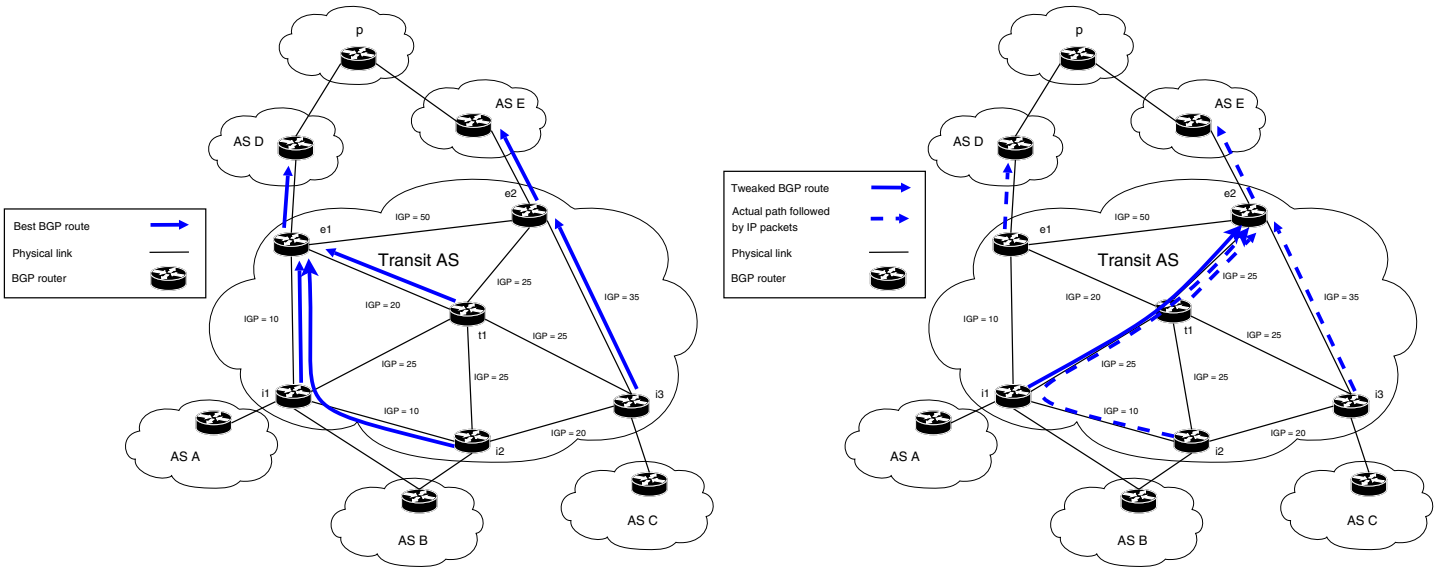

Fig. 1. Example of interdomain traffic engineering: best BGP routes chosen by ingress routers (left) and "dependent routes" for the considered BGP tweaking (right).

wards prefix p. This prefix can be reached through AS D and AS E. The transit AS has three ingress routers. Ingress $i1$ has a BGP peering with AS A and AS B, ingress $i2$ with AS B, and ingress $i3$ with AS C. The best BGP route chosen by each ingress router to reach prefix p is shown on Figure 1 (left diagram). Assuming that the AS path lengths of the two routes received by the ingress routers from egresses $e1$ and $e2$ are the same, and that a full mesh of iBGP sessions is used between the BGP routers of the transit AS, then each ingress chooses as its best BGP route the one whose IGP cost to reach the exit point is smallest. Ingress router $i1$ thus reaches p through $e1$ (IGP cost = 10), ingress $i2$ through $i1$ and $e1$ (IGP cost = 20), and ingress $i3$ through $e2$ (IGP cost = 35).

Now suppose that the transit AS wants to relieve egress $e1$ from the traffic sent by AS A towards prefix p. To do that, $i1$ chooses as its best BGP route towards p the one it learned from egress $e2$. This BGP route through egress $e2$ crosses router $t1$ and exits the AS at $e2$. If ingress $i1$ changes its best route to reach prefix p through egress $e2$, this does not ensure that its traffic sent towards prefix p will exit through egress $e2$. Indeed, router $t1$ on the shortest IGP path towards $e2$ has as best BGP route to reach p the one learned through egress $e1$ having a smaller IGP cost (20) than the one learned through $e2$. Tweaking BGP routes inside an ingress router does not guarantee that the exit point indicated in the chosen BGP route will actually be used by the traffic towards the destination prefix. To prevent routers on the path towards the new exit point to forward the traffic over another path than the one $i1$ expects, one can ensure that all routers on the path chosen by $i1$ forward traffic towards p over the same path as the one of $i1$.

Once it is guaranteed that all routers on the shortest IGP path between itself and $e2$ forward traffic towards p over the same path, one issue remains. Although all routers on the shortest IGP path between $i1$ and $e2$ forward their traffic towards p through exit point $e2$, other routers of the AS can have their traffic towards p *deflected* by the routers on the shortest IGP path between $i1$ and $e2$. The right diagram on Figure 1 shows how the traffic from all routers of our transit AS might be deflected due to the tweaking of $i1$ towards p and the resolution of the deflection problem on the path between $i1$ and $e2$. On the right diagram of Figure 1, we see that $i3$, $e1$ and $e2$ are not affected by the tweaking. $I2$ on the other hand might naively think that its traffic exits at $e1$. $I2$'s traffic will actually be deflected by $i1$ on the path between $i2$ and $e1$. All routers whose shortest IGP path to reach the exit point of their best BGP route crosses the shortest IGP path between $i1$ and $e2$ will have their traffic exiting the AS at $e2$. All these routers must therefore choose as their best route the one announced by $e2$ to ensure that they will announce to their external peers the route whose path is actually followed by the IP packets. In the remainder of this paper, we call *dependent routes* all the routes that must be changed in the AS in addition to the one of the tweaked ingress. This simple example illustrates that BGP-based traffic engineering requires great care and that doing it manually could lead to serious forwarding problems if one does not check for consistency in the paths chosen by the different routers of the AS.

In the preceding section, we did not define precisely what we meant by a tweaking. What we call in this paper a tweaking is an <i, p, e> triple corresponding to ingress i forcing its traffic towards prefix p to exit the AS at egress e. A tweaking is an abstract term connected to traffic engineering whose practical implementation will differ depending on the technical solution. We choose to rely on this particular definition of a tweaking mostly because it is the finest interdomain traffic engineering change one can do with BGP inside an AS. For instance, one could choose to work on coarser tweakings as the <ingress POP, prefix, egress POP> triple. In situations where the routing configuration inside the AS is more complex (BGP route reflectors, IGP hierarchies), checking the consistency of routing will be more involved that in the example provided of this section.

For what concerns the actual implementation of tweakings, two main techniques can be envisioned. The first has already

been explained before. It is purely BGP-based and requires changing the best BGP route of the tweaked ingress as well as the *dependent routes*. For this solution to work, all BGP routers must know the IGP path from each ingress towards each egress point to ensure that no forwarding loop occurs and that routing is consistent inside the AS. Many large transit providers have full IP networks and hence would rely on purely BGP-based traffic engineering.

Another way to implement the tweaking is through a tunnel (IP [15] or MPLS [16]) that would be established between the tweaked ingress and the new exit point. The advantage of a tunnel-based solution is that other routers of the AS do not have to change their best route to ensure consistency of the forwarding paths as only the traffic received by the tweaked ingress will be concerned (not the one of other ingress routers). In addition, tunnels will not have to be established on a per-prefix fashion, but only on a per-<ingress,egress> pair basis. We do not consider tunnel-based solutions in this paper due to space limitations. However, the simulations presented in section V have also been performed for the tunnel-based solution and can be found in [14].

III. Related work

As explained in [3], the state of interdomain traffic engineering is primitive. Few tools exist to predict the best route chosen by BGP to reach an external destination in large ISP networks [13], [4] due to the opacity and complexity of BGP and its interactions with the IGP routing. [13], [4] that constitute the state-of-the-art in BGP route prediction do not deal with the transients of the BGP routing. By transients of the BGP routing, we mean the dynamics of the BGP state machine. Few studies have been devoted to evaluating the feasibility and practical challenges of engineering the interdomain traffic by tweaking BGP [17], [18], [19], [3], [20]. [17] dealt with the problem of the optimal and off-line selection of the border routers for the advertisement of network prefixes so as to minimize the IGP cost of the traffic across a transit's network while maintaining the egress bandwidth constraints at the border routers. [18] proposed a random search algorithm to deal with the problem of finding the right value of the parameters for the configuration of large-scale networks. [19] tackled the problem of 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. Finally, [20] studied the problem of designing BGP-based interdomain traffic engineering techniques to engineer the outbound traffic of stub ASs. [20] showed that dynamically engineering the outgoing traffic of stub ASs over timescales of a few minutes could be done with a limited number of iBGP update messages.

In essence, *Tweak-it* is very similar in purpose to the intelligent route reflectors of [21] or the RCP platform in a single AS of [22], where the objective is to engineer the traffic of an AS while limiting as much as possible the burden in terms of the BGP messages required. To our knowledge, this paper is the first to propose a tool that allows to engineer the interdomain traffic of transit ASs by dealing with the real BGP routes received by a transit AS and the dynamics of the traffic demand over time. Most papers in the literature either assume that BGP routing is static or that the traffic demand does not change with time.

IV. Tweak-it

The main component of *Tweak-it* is a Perl script whose function is to keep an up-to-date state of the routing inside the transit AS, and based on this routing information to compute how to tweak the BGP routes inside the AS. The main external component of *Tweak-it* is CBGP [13], a scalable BGP simulator aimed at reproducing the steady-state policy routing made by BGP. By steady-state routing, we mean the solution at which BGP is due to converge independently from the dynamics of the BGP protocol state machine at each router. The steady-state solution found by BGP routing as computed by CBGP corresponds to the current state-of-the-art in BGP route prediction [4].

Figure 2 sketches the architecture of *Tweak-it*. The central component ("main script" in Figure 2) is the Perl script that manages the different inputs, communicates with CBGP and computes the tweaking of the BGP routes. The Perl script receives as input the BGP routing tables (RIB) and BGP updates received from the external peers of the AS, as well as the traffic statistics from all ingress routers. The script also needs the internal topology, IGP weights, and BGP routing policies enforced by each BGP router of the AS. With this information, the script builds the CBGP configuration file it will inject into CBGP ("initialization" in Figure 2). Then, the RIBs of the border routers having peerings with other ASs are injected into CBGP to populate the BGP routing tables of all BGP routers inside the AS. This finishes the "initialization phase" of the script.

The goal of the second part of the main script ("main loop" in Figure 2) is to compute the tweakings needed for the traffic engineering using the multiple-objectives evolutionary heuristic described in section IV-B. In this main loop, the Perl script first injects into CBGP all required changes for the routing to be up-to-date, then retrieves the state of the BGP routing from each router of the AS. Once the script has all the routing information it needs, it runs the multiple-objectives evolutionary heuristic described in section IV-B. Only the BGP routes towards popular destination prefixes are maintained in CBGP, typically a few hundreds or thousands, instead of the 140,000 routes present in current BGP routing tables. [9], [10], [11] have shown that most of the traffic in the Internet is sent to a limited fraction of the destination prefixes. [11] has also shown that BGP routes for popular destinations are stable.

A. CBGP simulator design overview

In order to model how BGP chooses its best route to reach a destination and to compute for each router in a domain the next-hop that would have been selected by BGP to reach each destination prefix, we have designed and implemented a new kind of BGP simulator. This simulator models the complete decision and filtering processes of BGP without reproducing the time-consuming packet exchanges that occur between simulated routers in packet-level simulators such as SSFNet [23] or J-Sim [24]. The purpose of the simulator that we have designed is to compute the BGP routes that routers know once the BGP routing has converged [25]. If the order in which the BGP messages are transmitted matters to determine the solution to which BGP would converge [26], then there is no guarantee that CBGP will

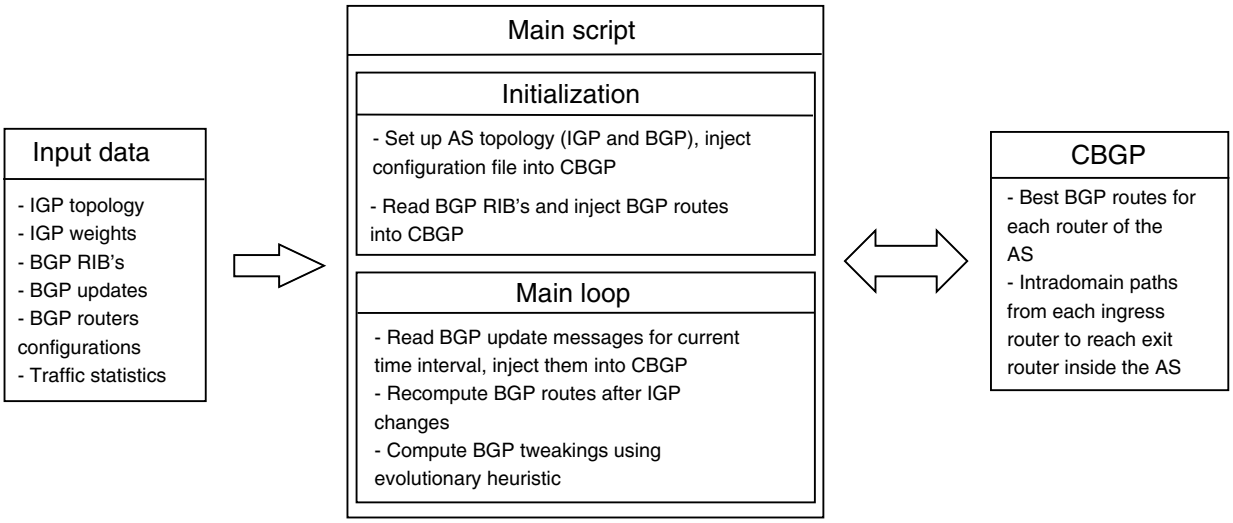

Fig. 2. Interaction between components of Tweak-it.

find the same solution as the actual BGP would. The aim of CBGP is however not to reproduce the routing plane of real networks but to provide a scalable way to compute the solution found by BGP routing in large networks.

To do this, CBGP accurately reproduces how BGP works. CBGP reproduces the propagation of UPDATE and WITHDRAW messages between routers; how these messages are filtered on input before being stored in adjacent routing information bases (Adj-RIB-ins); it reproduces the BGP decision process [27] that selects for each destination prefix a single route among the routes available in the Adj-RIB-ins; it stores the selected routes in a local routing information base (Loc-RIB); the routes present in the Loc-RIB are then subjected to the output filters and stored in the adjacent routing information bases (Adj-RIB-outs) before being propagated to other BGP routers in the simulation.

The simulator uses a network wide CISCO-like configuration file. This configuration file describes the network topology as a graph of nodes and edges. A node in the graph represents a router identified by a unique IP address (typically a router's loopback address) and an edge represents a link. Each link possesses its own IGP cost. The configuration file also describes the BGP routers. For each BGP router, the configuration file specifies the domain to which it belongs, identified by its AS number, the sessions it has with neighboring routers and the policies that the router will apply on routes received from- and exported to its neighbors [28], [29]. The `router-id` of the router is its IP address. The configuration file also specifies which prefixes each router originates. Finally, the configuration file describes static routing information that might be required by certain routers to establish sessions with their neighbors.

This configuration file is the input data required to run the simulation. Once the simulator parses this configuration file, its first operation is to build an efficient representation of the network in memory. That is, it creates a graph with all the nodes and links specified in the configuration file in a manner that makes possible a fast retrieval of whatever node (router) or edge (link). For each router, it creates a routing table that will contain the static routes, the IGP routes and the best BGP routes. Then, the simulator populates the routing table of each router with the static routes described in the configuration file and IGP routes computed thanks to the knowledge of the IGP costs of all the links. The simulator also creates for each BGP router a local routing information base (Loc-RIB) and a pair of adjacent RIBs (Adj-RIB-in and Adj-RIB-out) for each of its neighbors, one to store the routes received from the neighbor, the other to store the routes advertised to the neighbor. Finally, the simulator builds for each router a representation of all its filters that it can access in an efficient manner.

The second operation that the simulator begins when the simulation setup is terminated, is the propagation of route advertisements. The simulator starts with an arbitrary BGP router, looks up the prefixes it must advertise, builds the corresponding UPDATE messages and sends them to the router's neighbors according to the output filters. For each sent message, the simulator looks up in the router's routing table to find the link along which the message must be forwarded to reach the next hop. The message is forwarded on a hop-by-hop basis until it reaches its final destination. The generated BGP messages are pushed in a single global linear first-in first-out queue that keeps the message ordering. In the real world, BGP messages are exchanged over TCP connections. In this simulator, we assume that there is no packet loss. The simulator does this for all the BGP routers.

Another way to insert routes into the simulation is by feeding real BGP messages in the widely used MRT format [30]. These messages are "sent" to a simulated BGP router by the way of a virtual peer. A virtual peer is not a router that exists in the simulation, but it is the representation of an external router. When a MRT message is fed to a BGP router, it appears to come from a peer. It is thus filtered and, if accepted, inserted as-is in the Adj-RIB-in that corresponds to the virtual peer. Eventually, the message is taken into account during the decision process. This feature makes possible the simulation of a real ISP by building a topology with only the ISP's routers, by collecting the real BGP messages received from the ISP's neighbors and by inserting them into the simulation. The simulator can thus react to exter-

nal route changes and predict the new choices that are made by BGP in the ISP's network.

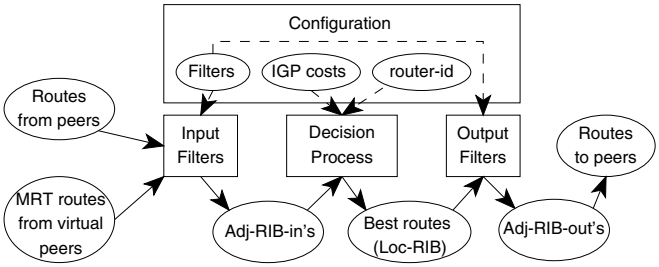

Fig. 3. A single BGP router in a CBGP simulation.

Once all the prefixes have been advertised, the simulation starts. The simulator pops the first message from the queue, and wakes up the router corresponding to the current hop of the message. If this router is not the final destination of the message, the router looks up in its routing table along which link the message should be forwarded. Otherwise, the router processes the BGP message. If the BGP message is a WITHDRAW, the router removes from the corresponding Adj-RIB-in the route towards the withdrawn prefix, and runs the decision process. If the BGP message is an UPDATE, the router checks if the route it contains is accepted by its input filters. If so, the route is stored in the Adj-RIB-in and the decision process is run for the prefix of the route. The decision process retrieves from the Adj-RIB-ins all the reachable routes for the considered prefix, compares them according to its rules and selects a single best route for this prefix. The simulator does this until the message queue is empty, which means that the simulation has converged.

When the simulation has converged, the simulator provides convenient ways to analyze the results. A first possibility is to have a look at a single or all the routers's routing tables, Loc-RIBs or Adj-RIBs. Another possibility is to trace the route followed by packets that travel from one router to another router. The simulator provides the sequence of IP addresses of the routers that have been used to reach the destination router.

B. Multiple-objectives evolutionary heuristic

The task of the evolutionary algorithm is to find out which tweakings to use to engineer the interdomain traffic of the AS. The heuristic can deal with any set of objectives that can be computed based on the amount of traffic for each <ingress,prefix> pair. Any function that can be computed can be used as an objective of the algorithm. The core of the heuristic is a multiple-objectives evolutionary algorithm [31]. This heuristic relies on a population of individuals. Each individual is a potential solution, i.e. a set of tweakings to be applied on the BGP routers of the AS to engineer the traffic for a particular time interval.

For each time interval, the algorithm uses the latest state of the BGP routes known by ingress routers as well as the traffic statistics from the previous time interval to compute the tweakings be applied during the next time interval. This state of the BGP routes is obtained by querying the CBGP simulator. The current implementation of our Perl script does not send the iBGP messages to perform the tweaking as it has no BGP session with real routers. However, our solution can be used in practice to send iBGP messages to the BGP routers of an AS to engineer its interdomain traffic. Our solution could be run either on a route reflector [32] that would also centralize the traffic statistics, or on a stand-alone BGP router having iBGP sessions with all the BGP routers of the AS (or only with the route reflector of each POP in the case of a hierarchy of route reflectors).

We do not describe the core of the evolutionary optimization for lack of space but refer to [33] for a detailed description. For a given time interval, the algorithm consists of a loop where each iteration adds to each individual (a set of BGP tweakings) of the population a randomly chosen tweaking among the possible ones. The heuristic is an evolutionary algorithm where competition among the individuals of the population (the set of individuals) drives which set of tweakings will be chosen so as to sample the trade-off between the traffic engineering objectives. Evolutionary algorithms have been extensively used to sample Pareto-optimal fronts in a single run [12], where traditional optimization methods often fail.

The algorithm accepts from one iteration to another the solutions that are non-dominated in terms of the traffic engineering objectives. A solution in a set of solutions is said non-dominated if no other solution of the considered set is better in all objectives at the same time. By keeping only the non-dominated solutions, the search spends its time on trying to find better solutions in terms of all objectives at the same time. This ensures that at each time interval, the algorithm selects one non-dominated solution among the possible trade-offs between the traffic engineering objectives.

First, before searching for good tweakings, the routing and traffic statistics must be kept up-to-date. Before starting the optimization, the script checks that both the IGP routing (physical links, links IGP weights, routers reachability) and the BGP routing (next hop reachability, best route choice by each ingress router) are up-to-date by interacting with CBGP, and loads the traffic statistics concerning the traffic sent by each ingress point towards each destination prefix collected during the last time interval.

The evolutionary optimization then starts with a loop that iterates as many times as the maximum number of tweakings we allow to be applied during the next time interval. The purpose of this loop is to guide the local search in selecting an additional random tweaking for any individual. Making the number of the tweakings a dimension of the search increases the pressure to find as early as possible the tweakings that improve the most the objectives in terms of the traffic. If the population is unable to find improvement at some iteration, the population is re-initialized with the last non-dominated set of solutions found.

During every iteration of its loop, the evolutionary algorithm performs the following steps. The first step consists of a random local search that applies to the individuals a random tweaking (i.e. <ingress,prefix> pair) among the possible ones. As the number of possible tweakings grows with the number of considered prefixes times the number of ingresses, allowing to try them all makes the size of the search space very large while for practical purposes only those having a large amount of traffic are of interest. The Perl script takes as parameter this percentage of the total traffic that the user wants to consider in the optimization.

The number of destination prefixes having been reduced, the

algorithm sorts the <ingress,prefix> pairs by decreasing byte count and further reduces the search space by considering only the largest pairs. Tweaking pairs having too small an amount of traffic is worthless for traffic engineering purposes. The main script takes this number of pairs as parameter. Even after removing most of the irrelevant prefixes and <ingress,prefix> pairs, there are still a lot of possible <ingress,prefix> pairs. We used 500 pairs in the simulations provided in [14].

It should be noted that any <ingress,prefix> pair cannot be tried for a given individual. As an individual may already contain tweakings, the algorithm checks before trying a new tweaking that its <ingress,prefix> pair is not already part of the considered individual, as this would contradict a previously chosen tweaking.

Then, the algorithm tries to apply the tweaking by computing the new values of the traffic objectives for every reachable exit point. For each reachable exit point, it applies the tweaking and checks whether this new individual is non-dominated, in which case the individual is placed in the set of accepted solutions for this iteration. Since the purpose of any multiple-objectives optimization search algorithm is to sample the trade-off between the objectives, it is necessary to allow *any* non-dominated individual to be accepted as a potential solution. It will be up to the selection procedure (see below) to decide how to sample the set of non-dominated solutions found during the current iteration.

The last step within an iteration of the evolutionary algorithm is to select among the accepted individuals, those that will constitute the population for the next generation. This step is crucial for the multiple-objectives evolutionary optimization to work, as among non-dominated solutions, one must choose those that will be selected to make the population during the next generation and in which proportion of the total population. Since non-dominated solutions form a front that samples the trade-off between the two traffic objectives, the aim of the selection procedure is to pick the individuals of the next population according to how the non-dominated solutions sample this front. The idea is that the more points are concentrated in some area of the front, the smaller the effort should be placed on further sampling this area. This is the logic behind crowding-based selection schemes, which we do not further describe but refer to [31].

Finally, after the maximal number of iterations allowed, the algorithm must choose among the individuals forming the non-dominated front during the last iteration, the solution that will actually be implemented in the network during the next time interval. As these non-dominated solutions form a non-dominated front, the choice of the solution to apply is quite arbitrary. In our simulations of [14] for instance, we chose the solution having the smallest IGP cost on the non-dominated front. Any other solution will do, the choice depending on the relative importance given to the different traffic engineering objectives.

V. SIMULATIONS

In this section, we illustrate the use of *Tweak-it* in the context of the GEANT network. We rely on the GEANT network topology available on DANTE's website at `http://www.dante.net`. GEANT is a multi-gigabit pan-European data communications network, reserved specifically for research and education use. The GEANT network is operated by DANTE. GEANT has interconnections to two general Internet peers, which we call AS X and AS Y. GEANT has peerings with AS X at four different points and at two different points with AS Y, one GEANT egress router having both a peering with AS X and AS Y.

The scenario of our simulations consists in synthetic traffic to be received at each ingress point of the GEANT network towards destination prefixes reachable through the general Internet peers of the network. We used the IGP topology of GEANT, as well as the BGP data received from its two general Internet peers. We chose in this BGP data a continuous one month period starting at 9:41 AM on February 8, 2004. This one month of BGP updates contains 35,563,511 BGP messages, about 823 eBGP messages per minute on average. We used a time granularity of one hour, i.e. for every hour we injected the BGP messages into CBGP, loaded the traffic demand of the previous hour, and ran the algorithm described in section IV-B.

As we did not dispose of the GEANT traffic at the time of the writing of this paper, we generated artificial traffic as could be seen in a transit network. We randomly selected 1000 destination prefixes to be considered as the most popular destinations for the GEANT network, and generated traffic from each ingress router of GEANT towards each of the 1000 destination prefixes according to a Weibull distribution of shape 0.15 [34]. The choice of a Weibull distribution is motivated by the observation that most of the traffic in the Internet is exchanged with a limited fraction of the reachable prefixes [9], [10], [11]. This distribution captures quite well this property while still a significant percentage of the mass of the distribution lies in the non-tail part of the distribution. After having generated the total traffic to be sent from each ingress towards the destination prefixes, we simulated a periodic evolution of the traffic for each <ingress,prefix> pair over the one month period. For the sake of simplicity, we sampled a sinusoidal signal with a period corresponding to one day and with a time granularity of one hour. The coefficients were normalized as to sum to one, and the total traffic for the <ingress,prefix> pair was multiplied by the normalizing coefficients. This provided a regularly-varying traffic for each pair. Then, to reflect the fact that an ingress point of a transit AS can in practice be located anywhere in the world, we added a random phase (between 0 and 23 hours) to the sinusoidal signal to shift the busy hours period differently for each ingress point. Although the generated traffic might not reflect the dynamics of the traffic seen by a transit AS, the main properties of interdomain traffic are reproduced by this synthetic traffic. Both the Perl script used to generate the synthetic traffic as well as the files containing the produced traffic during our simulations are freely available at [14].

In this section, the heuristic optimizes the way traffic is balanced over the external peerings of the AS while minimizing the IGP cost for the whole traffic[1], in as few tweakings as possible.

Simulation results for one version of the algorithm are shown on Figure 4. Four different flavors of the algorithm are provided in [14] as well as the corresponding simulation results similar to

[1]Minimize $\sum_{\forall(i,p)} traf(i,p) \times IGP\_cost(i,e)$ where $traf(i,p)$ represents the amount of traffic received by ingress router i having as destination external prefix p and $IGP\_cost(i,e)$ represents the IGP cost for ingress i to reach egress router e

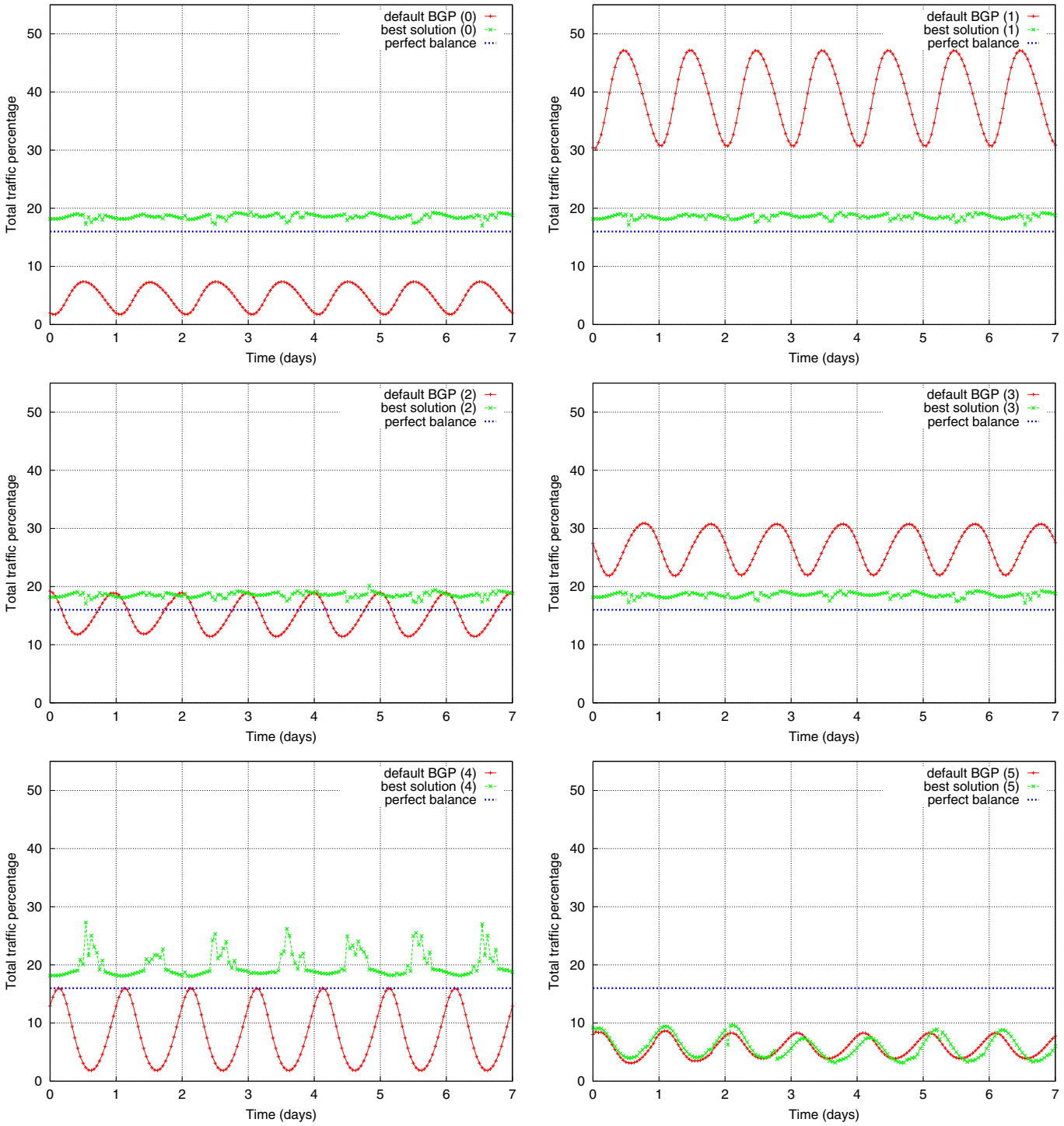

Fig. 4. Evolution of traffic per external peering.

those presented in this section. For these simulations, we limited the number of allowed tweakings found by the evolutionary algorithm to 50, for each time interval. Figure 4 shows for each external peering (randomly assigned an identifier between 0 and 5), the evolution of the percentage of the total traffic during each time interval over a time period of one week. Each graph of Figure 4 displays the evolution of the traffic percentage if the choice of the best egress point is left to the BGP decision process ("default BGP") with the traffic percentage when tweakings are made by the evolutionary algorithm ("best solution"). Each graph also shows the ideal traffic balance per peering of one sixth of the total traffic ("perfect balance").

The first thing that appears on Figure 4 is the periodic evolution of the traffic percentage for each peering. This traffic variability is created by the periodic traffic per destination prefix coupled with the random phase per ingress router. Second, even after applying tweakings, peering 5 (bottom right on Figure 4) is not well balanced. The reason is that both peerings 1

and 5 are attached to the same egress router. As tweakings have been defined in terms of the egress router, not the external peering, tweakings can act on the total traffic sent to egress routers. To improve the balance among egress routers the algorithm decreases the share of peering 1 without being able to increase the share of peering 5. If peering 1 and 5 were on different routers or if tweakings were defined in terms of external peerings, then all peerings would have been evenly balanced. Even with the previous issue, the improvement in balance provided by *Tweak-it* is significant while the cost in terms of tweakings is limited, of about 40 per hour [14].

VI. Conclusion and further work

In this paper we proposed *Tweak-it*, an open source tool for interdomain traffic engineering in large ISP networks. *Tweak-it* consists of a multiple-objectives evolutionary heuristic coupled with an efficient BGP simulator. *Tweak-it* takes the intradomain configuration (IGP weights and topology), BGP routes received from peers, BGP routing policies, and traffic demand.

We described the architecture of *Tweak-it*, namely the interaction of an evolutionary heuristic and our efficient BGP simulator. We applied the tool on the GEANT network to show that *Tweak-it* can be useful to find out how to tweak the BGP inside a transit AS to engineer the distribution of the traffic inside its network. The genericity of the evolutionary algorithm as well as the scalability of the CBGP simulator make of *Tweak-it* a elegant way to investigate interdomain traffic engineering problems.

The current version of our solution replays the real BGP updates received from peers, as well as synthetic traffic traces generated by our means. By the time of the writing of this paper, we have already implemented a module that imports the IGP routing messages (for ISIS) and recomputes the IGP weights and topology consistently with the IGP routing information in the same manner as for the BGP updates.

We are currently collaborating with a large commercial transit provider and extending our tool to tackle operational problems for this provider's network. We will apply the tool to scenarios with larger transit ISP topologies and different traffic engineering objectives. The tool will be integrated in the TOTEM traffic engineering toolbox (`http://totem.info.ucl.ac.be`).

VII. Acknowledgments

The authors would like to thank Tim Griffin from Intel Research Cambridge and Nicolas Simar from DANTE for providing the BGP data from the GEANT network. Steve Uhlig is funded by the FNRS (Fonds National de la Recherche Scientifique, Belgium). This work was partially supported by the Walloon Government (DGTRE) within the TOTEM project (`http://totem.info.ucl.ac.be`).

References

[1] 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.

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

[3] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *Comput. Commun. Rev.*, vol. 33, no. 5, pp. 19–30, October 2003.

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

[5] Nick Feamster and Hari Balakrishnan, "Detecting BGP Configuration Faults with Static Analysis," in *Proc. 2nd Symposium on Networked Systems Design and Implementation*, Boston, MA, May 2005.

[6] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP Misconfigurations," in *Proc. ACM SIGCOMM*, September 2002.

[7] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of hot-potato routing in IP networks," in *Proc. of ACM SIGMETRICS*, June 2004.

[8] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: methodology and experience," in *Proc. of ACM SIGCOMM*, September 2000.

[9] W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *IEEE Global Internet Symposium*, December 1999.

[10] S. Uhlig and O. Bonaventure, "Implications of Interdomain Traffic Characteristics on Traffic Engineering," *European Transactions on Telecommunications*, January 2002.

[11] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP Routing Stability of Popular Destinations," in *Proc. of ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[12] C. A. Coello Coello, "An updated survey of GA-based multiobjective optimization techniques," *ACM Computing Surveys*, vol. 32, no. 2, pp. 109–143, 2000.

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

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

[15] J. Lau, M. Townsley, and I. Goyret, "Layer Two Tunneling Protocol (Version 3)," Internet Draft, draft-ietf-l2tpext-l2tp-base-14, work in progress, June 2004.

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

[17] T. Bressoud and R. Rastogi, "Optimal configuration for BGP route selection," in *Proc. of IEEE INFOCOM'03*, April 2003.

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

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

[20] S. Uhlig and O. Bonaventure, "Designing BGP-based outbound traffic engineering techniques for stub ASes," *Comput. Commun. Rev.*, vol. 34, no. 5, 2004.

[21] 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.

[22] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *FDNA'04 workshop*, August 2004.

[23] B. J. Premore, "SSF Implementations of BGP-4," available from `http://www.cs.dartmouth.edu/~beej/bgp/`, 2001.

[24] H. Tyan, *Design, realization and evaluation of a component-based compositional software architecture for network simulation*, Ph.D. thesis, Ohio State University, 2002.

[25] T. Griffin, F. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, 2002.

[26] T. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proc. of ACM SIGCOMM*, September 1999.

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

[28] B. Halabi, *Internet Routing Architectures (2nd edition)*, Cisco Press, 2000.

[29] I. van Beijnum, *Building Reliable Networks with the Border Gateway Protocol*, September 2002.

[30] Merit Network, "MRT: multi-threaded routing toolkit," `http://www.mrtd.net`.

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

[32] T. Bates, R. Chandra, and E. Chen, "BGP Route Reflection - An Alternative to Full Mesh IBGP," Internet Engineering Task Force, RFC2796, April 2000.

[33] S. Uhlig, "A multiple-objectives evolutionary perspective to interdomain traffic engineering in the internet," in *Workshop on Nature Inspired Approaches to Networks and Telecommunications*, September 2004.

[34] M. Evans, N. Hastings, and B. Peacock, *Statistical distributions*, Wiley-InterScience, 2000.